Explainability and Robustness of Deep Visual Classification Models

Jindong Gu

Munich 2022

Explainability and Robustness of Deep Visual Classification Models

Jindong Gu

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Jindong Gu
aus Anhui, China

München, den 18. 04. 2022

Erstgutachter: Prof. Dr. Volker Tresp

Zweitgutachter: Prof. Dr. Nuno Vasconcelos

Drittgutachter: Prof. Dr. Dacheng Tao

Tag der mündlichen Prüfung: 06. 07. 2022

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Gu, Jindong
Name, Vorname

München, Mar. 13. 2022
Ort, Datum

Gu, Jindong
Unterschrift Doktorand/in

Formular 3.2

Contents

Abstract

Deep learning has revolutionized AI and deep neural networks, in particular, have been hugely successful in a wide range of applications. Deep neural network architectures with different inductive biases have been proposed in different communities. In the computer vision community, Convolutional Neural Networks (CNNs), first proposed in the 1980's, have become the standard visual classification model. Recently, as alternatives to CNNs, Capsule Networks (CapsNets) and Vision Transformers (ViTs) have been proposed. CapsNets, which were inspired by the information processing of the human brain, are considered to have more inductive bias than CNNs, whereas ViTs are considered to have less inductive bias than CNNs. All three classification models have received great attention since they can serve as backbones for various downstream tasks, e.g. object detection and semantic segmentation. However, these models are far from being perfect.

As pointed out by the community, there are two weaknesses in standard Deep Neural Networks (DNNs). One of the limitations of DNNs is lack of explainability. Even though they can achieve or surpass human expert performance in the image classification task, the DNN-based decisions are difficult to understand. In many real-world applications, however, individual decisions need to be explained. The other limitation of DNNs is adversarial vulnerability. Concretely, the small and imperceptible perturbations of inputs can mislead DNNs. The vulnerability of deep neural networks poses challenges to current visual classification models. The potential threats thereof can lead to unacceptable consequences. Besides, studying model adversarial vulnerability can lead to a better understanding of the underlying models.

Our research aims to address the two limitations of DNNs. Specifically, we focuses on deep visual classification models, especially the core building parts of each classification model, e.g. dynamic routing in CapsNets and self-attention module in ViTs.

We argue that both the lack of explainability and adversarial vulnerability can be attributed to the difference in the visual features used by visual recognition models and the human visual system to recognize objects. Namely, the visual clues used by standard CNNs are different from the ones used by our visual system. The differences make the interpretation of classifications difficult. Similarly, the differences also leave attackers the chance to manipulate decisions with quasi-imperceptible input perturbations.

We have analyzed if the brain-inspired Capsule Network (CapsNet) performs more robustly than the CNNs. Our investigation on CapsNet shows CapsNets with more inductive

bias do not perform better than CNNs. The dynamic routing therein can even harm the robustness, in contrast to the common belief. Compared to CNNs and CapsNets, Vision Transformers (ViTs) are considered to have less inductive bias in its architecture. Given the patch-wise input image representation of ViT, we dissect ViT with adversarial patch attack methods. We find that vision transformers are more robust to naturally corrupted patches than CNNs, whereas they are more vulnerable to adversarial patches. Specifically, the attention module can effectively ignore natural corrupted patches. However, when attacked by an adversary, it can be easily fooled.

Overall, our work provides a detailed analysis of CNNs, CapsNet, and ViTs in terms of explainability and robustness. The contribution of this thesis will facilitate the application of existing popular deep visual classification models and inspires the development of more intelligent classifiers in the future.

Zusammenfassung

Deep Learning hat die Künstliche Intelligenz revolutioniert, und insbesondere sind tiefe neuronale Netze in einer Vielzahl von Anwendungen sehr erfolgreich. Architekturen der tiefen neuronalen Netze mit unterschiedlichen induktiven Verzerrungen werden in verschiedenen Arbeiten vorgestellt. In der Community des Computersehens sind Convolutional Neural Networks (CNNs), die erstmals in den 1980er Jahren vorgestellt wurden, zum Standardmodell für die visuelle Klassifikation geworden. Als Alternativen zu CNNs werden kürzlich Capsule Networks (CapsNets) und Vision Transformers (ViTs) vorgestellt. CapsNets, die von der Informationsverarbeitung des menschlichen Gehirns begeistert wurden, gelten als mehr induktiv verzerrt als CNNs, während ViTs als weniger induktiv verzerrt als CNNs angesehen werden. Alle drei Klassifikationsmodelle erfahren viel Aufmerksamkeit, da sie als Backbone für verschiedene nachgelagerte Aufgaben dienen können, z.B., Objekterkennung und Semantische Segmentierung. Allerdings sind diese Modelle weit davon entfernt, perfekt zu sein.

Wie die Gemeinschaft darauf hingewiesen hat, gibt es zwei Schwachstellen in standardmäßigen Deep Neural Networks (DNNs). Eine der Einschränkungen von DNNs ist die mangelnde Erklärbarkeit. Obwohl sie die Leistung menschlicher Experten bei der Bildklassifizierungsaufgabe erreichen oder übertreffen können, sind die DNN-basierten Entscheidungen schwer zu verstehen. In vielen echten Anwendungen müssen jedoch einzelne Entscheidungen erklärt werden. Die andere Einschränkung von DNNs ist die gegnerische Verletzlichkeit. Nämlich können die kleinen und nicht wahrnehmbaren Störungen der Eingaben DNNs irreführen. Die Verletzlichkeit der DNNs stellt aktuelle visuelle Klassifizierungsmodelle vor Herausforderungen. Die potenziellen Bedrohungen davon können zu inakzeptablen Konsequenzen führen. Außerdem kann die Forschung von gegnerische Verletzlichkeit der Modelle zu einem besseren Verständnis der zugrunde liegenden Modelle führen.

Unsere Forschung zielt darauf ab, die beiden Einschränkungen von DNNs anzugehen. Nämlich konzentrieren wir uns auf tiefe visuelle Klassifikationsmodelle, insbesondere auf

die Kernbestandteile jedes Klassifikationsmodells, z. dynamisches Routing von CapsNets und Selbstaufmerksamkeitsmodul von ViTs.

Wir argumentieren, dass sowohl der Mangel an Erklärbarkeit als auch die Verletzlichkeit von DNNs auf den Unterschied in den visuellen Merkmalen zurückzuführen sind, die von visuellen Erkennungsmodellen und dem menschlichen visuellen System zur Erkennung von Objekten verwendet werden. Die visuellen Hinweise, die von standardmäßigen CNNs verwendet werden, unterscheiden sich nämlich von denen, die von unserem visuellen System verwendet werden. Die Unterschiede erschweren die Interpretation der Klassifikationen. Ebenso lassen die Unterschiede Angreifern auch die Möglichkeit, Entscheidungen mit quasi unmerklichen Eingabestörungen zu manipulieren.

Wir haben analysiert, ob das vom Gehirn begeisterte Capsule Network (CapsNet) eine robustere Leistung als die CNNs erbringt. Unsere Forschung zu CapsNet zeigt, dass CapsNets mit stärker induktiver Verzerrung nicht besser als CNNs verhalten. Entgegen der landläufigen Meinung kann das darin enthaltene dynamische Routing sogar der Robustheit schaden. Im Vergleich zu CNNs und CapsNets wird davon ausgegangen, dass Vision Transformers (ViTs) eine weniger induktive Verzerrung in ihrer Architektur aufweisen. Angesichts der patchweisen Eingabebilddarstellung von ViT analysieren wir ViT mit gegnerischen Patch-Angriffsmethoden. Wir stellen fest, dass Vision Transformer gegenüber natürlich beschädigten Patches robuster als CNNs sind, während sie verletzlicher für gegnerische Patches sind. Insbesondere kann das Selbstaufmerksamkeitsmodul natürlich beschädigte Patches effektiv ignorieren. Wenn es jedoch von einem Gegner angegriffen wird, kann es leicht getäuscht werden.

Insgesamt liefert unsere Arbeit eine detaillierte Analyse von CNNs, CapsNet und ViTs in Bezug auf Erklärbarkeit und Robustheit. Der Beitrag dieser Arbeit wird die Anwendung bestehender populärer tiefer visueller Klassifikationsmodelle erleichtern und die Entwicklung intelligenterer Klassifikatoren in der Zukunft anregen.

Acknowledgement

This dissertation summarizes my Ph.D. study at Ludwig Maximilian University of Munich. It would not be possible without the support of many people.

First, I would like to thank my Ph.D. supervisor, Prof. Dr. Volker Tresp. During my Ph.D. study, he is always available to help. His team has provided me with the best possible platform to conduct my research. After both of us had defined the general goals of my dissertation, he has given me great trust and freedom to pursue various ideas. He is very patient to help me to be a world-class researcher. I really appreciate his efforts in improving my research skills including critical thinking and scientific writing.

I spent the first part of my Ph.D. in Siemens Technology. I would like to thank many people therein, including but not limited to Dr. Daniela Ölke, Dr. Yinchong Yang, Dr. Nikou Günnemann, Dr. Rui Zhao, Dr. Zhiliang Wu and Dr. Sebastian Mittelstädt. Dr. Daniela Ölke prepares me to start my Ph.D. study and improve my skills to present my research. Dr. Yinchong Yang and Dr. Nikou Günnemann help me to deeply understand my research topics. Besides, I have wonderful collaborations with Zhiliang. He often has a deep understanding of the underlying question. I learn a lot from our collaborations. Apart from them, I have also received a lot of help and advice from many other colleagues from Volker's team. I would like to express my thanks to them, including but not limited to Zhen Han, Sahand Sharifzadeh, Yushan Liu, Yao Zhang, Gengyuan Zhang, Hang Li, Dr. Yunpu Ma, and Dr.Marcel Hildebrandt.

During my Ph.D. study, I am very fortunate to have the chance to intern at different places, which greatly broaden my horizon. I would like to first thank Dr. Wei Liu who provides me with a research intern position in Tencent AI Lab. During this internship, I have learned a lot from Prof. Dr. Baoyuan Wu and Yonlong Tian. I also spent a wonderful time with Dr. Han Hu in Microsoft Research. As my mentor, he shows me how to go for high-quality research work, which the community really cares about. I would also like to give my thanks to Dr. Yao Qin. I learn how to conduct neat research from our previous

collaborations. She encourages me to think in a creative way and in a neat way. I am looking forward to working closely together with her in the incoming research internship at Google Brain.

Besides, I am very fortunate to have a visiting study in Torr Vision Group, at University of Oxford. Great thanks to Prof. Dr. Philip Torr for providing me this opportunity as well as an oxford visiting grant. During this visit, I have a wonderful experience collaborating with many members of TVG. Especially, Dr. Hengshuang Zhao and Dr. Adel Bibi have always pursued a deep understanding of the research topic from different perspectives, which inspires me a lot. I would also like to thank them as well as other team members from TVG for the inspiring talks.

My sincere gratitude also goes to all members of my dissertation committee. Especially, I am very honored to have Prof. Dr. Dacheng Tao and Prof. Dr. Nuno Vasconcelos in my dissertation committee. I thank them for their agreement to be the examiners of my doctoral thesis. I appreciate their time to evaluate this thesis as well as their feedback.

Last but not least, I would like to thank my parents for being supportive throughout my study overseas.

List of Publications and Declaration of Authorship

- Gu, Jindong, Yinchong Yang, and Volker Tresp. "Understanding Individual Decisions of CNNs via Contrastive Backpropagation."Proceedings of the Asian Conference on Computer Vision (ACCV). Springer, Cham, 2018. DOI: 10.1007/978-3-030-20893-6_8.

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-authors, Yinchong Yang, and Volker Tresp. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 2.

- Gu, Jindong. "Interpretable Graph Capsule Networks for Object Recognition." Proceedings of the AAAI Conference on Artificial Intelligence (AAAI). Vol. 35. No. 2. P. 1469-1477. 2021.

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the draft of the manuscript and did all of the corrections.

  This published work serves as Chapter 3.

- Gu, Jindong, and Volker Tresp. "Improving the Robustness of Capsule Networks to Image Affine Transformations." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (pp. 7285-7293). 2020. DOI: 10.1109/CVPR42600.2020.00731

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-author, Volker Tresp.

  This published work serves as Chapter 4.

- Gu, Jindong, Volker Tresp, and Han Hu. "Capsule Network is Not More Robust than Convolutional Network." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (pp. 14309-14317) 2021. DOI: 10.1109/CVPR46437.2021.01408

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-authors, Volker Tresp, and Han Hu. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 5.

- Gu, Jindong, Baoyuan Wu, and Volker Tresp. "Effective and Efficient Vote Attack on Capsule Networks." International Conference on Learning Representations (ICLR) , 2021.

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-authors, Baoyuan Wu, and Volker Tresp. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 6.

- Gu, Jindong, Volker Tresp and Qin Yao. "Are Vision Transformers robust to Patch-wise Perturbation? " European Conference on Computer Vision (ECCV), 2022.

  I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I regularly discussed this work with the co-authors, Qin Yao, and Volker Tresp. All co-authors contributed to improving the manuscript.

  This published work serves as Chapter 7.

Other Publications

- Gu, Jindong, and Volker Tresp. "Saliency Methods for Explaining Adversarial Attacks" NeurIPS workshop, 2019.
- Gu, Jindong, Zhiliang Wu, and Volker Tresp. "Introspective Learning by Distilling Knowledge from Online Self-explanation." Proceedings of the Asian Conference on Computer Vision (ACCV). Springer, Cham, 2020. DOI: 10.1007/978-3-030-69538-5_3
- Gu, Jindong, and Volker Tresp. "Search for Better Students to Learn Distilled Knowledge." European Conference on Artificial Intelligence (ECAI), 2020. DOI: 10.3233/FAIA200214
- Gu, Jindong, Wei Liu, and Yonglong Tian. "Simple Distillation Baselines for Improving Small Self-supervised Models." ICCV workshop, 2021.
- Gu, Jindong, Hengshuang Zhao, Volker Tresp, Phillip Torr. "SegPGD: An Effective and Efficient Attack for Evaluating and Boosting Segmentation Robustness", European Conference on Computer Vision (ECCV), 2022
- Wu, Boxi*, Jindong Gu*, Zhifeng Li, Deng Cai, Xiaofei He, Wei Liu. "Towards Efficient Adversarial Training on Vision Transformers", European Conference on Computer Vision (ECCV), 2022
- Liu Xinwei, Jian Liu, Yang Bai, Jindong Gu, Tao Chen, Xiaojun Jia, Xiaochun Cao. "Watermark Vaccine: Adversarial Attacks to Prevent Watermark Removal", European Conference on Computer Vision (ECCV), 2022

Chapter 1

Introduction

1.1 Motivation

Artificial intelligence changes our daily lives in many perspectives. The recent advances of artificial intelligence are mainly powered by Deep Learning method [69]. As a revolutionary technique, Deep Learning methods are also embraced by other disciplines, *e.g.* bioscience and astronomy. As a representative model in the framework of deep learning, deep neural networks (DNNs) dominate the community due to their powerful expressiveness. However, two limitations of deep neural networks prevent their wide application in safety-critical domains, *e.g.* the medical domain and autonomous driving system.

One of the limitations of deep neural networks is their lack of explainability. Even though the DNN-based intelligent system can achieve or surpass human expert performance on some tasks, it is not clear how the system reaches its decisions. For example, Deep convolutional neural networks (DCNNs) achieve start-of-the-art performance on many tasks, such as visual object recognition [115, 49, 121, 57]. However, since they lack transparency, they are considered as "black box" solutions. In real-world applications, however, individual decisions need to be explained to gain the trust of the users. *e.g.*, autonomous driving systems should reassure passengers by giving explanations when braking the car abruptly [65, 66]. Decisions made by deep models are also required to be verified in the medical domain. Mistakes of unverified models could have an unexpected impact on humans or lead to unfair decisions [79, 46]. Besides, AI applications must comply with related legislation, *e.g.*, the right to explanation in GDPR of the European Union [108].

The other limitation of deep neural networks is limited generalization robustness. When deep neural networks are deployed in real-world applications, the input can deviate from

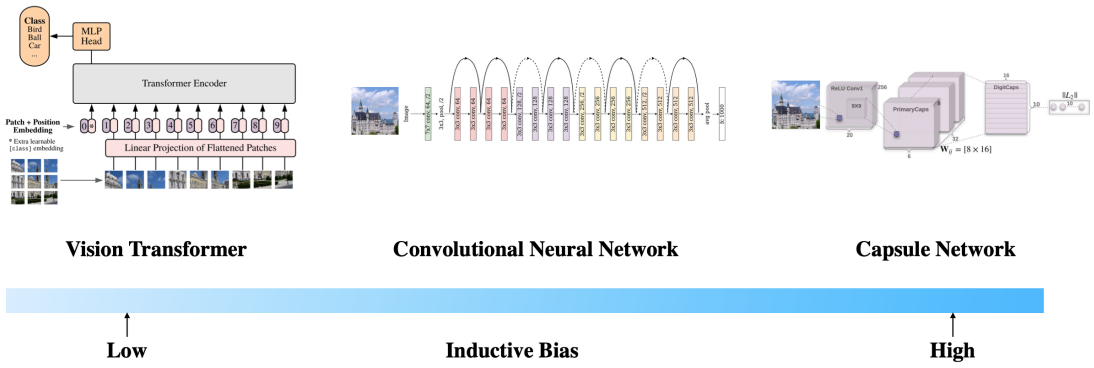


Figure 1.1: The overview of deep visual classification model architectures. This figure is based on the figures in [25, 49, 102]

the training data distribution. The inference on the input with overlapped patterns [102], affine-transformed pattern [102, 37], and natural corruption [52] can result in unexpected results. Besides the robustness to out-of-distribution data, the low robustness to artificial perturbation also raises great concern in the community. Concretely, the small and imperceptible artificial perturbations of inputs can mislead DNN-based intelligent systems. For example, given an image correctly classified by a deep convolutional neural network, a hardly human-perceptible artificial perturbation can cause the convolutional neural network to misclassify the image when added to it. The vulnerability of Deep Learning poses challenges to current intelligent systems. The adversarial images on CNNs can pose potential threats to security-sensitive CNN-based applications, *e.g.*, face verification [112] and autonomous driving [26]. The potential threats thereof can lead to unacceptable consequences. Besides, the existence of adversarial images demonstrates that the object recognition process in CNNs is dramatically different from that in human brains. Hence, the study of adversarial examples on deep neural networks can also lead to a better understanding of the underlying object recognition models.

Since [68] proposed the AlexNet, deep neural networks have revolutionized the computer vision community. In the image classification task, the classification model consists of two parts, *i.e.*, feature extractor and classifier. The modules that extract features from input images are also adopted as feature extractor (dubbed *backbone*) in downstream tasks, *e.g.*, object detection [29, 48] and semantic segmentation [83, 144, 15]. The improvement of the classification models often also benefits the downstream tasks due to the improved backbone. In this thesis, we focus on deep visual classification models from the perspectives of explainability and robustness.

As one of the representatives of deep visual classification models, convolutional neural networks have dominated the computer vision community in the last decade [68]. However, CNNs suffer from many limitations, *e.g.*, only local information aggregation at lower layers and the broken equivariance. Recently, the community has been attempting to propose new models to overcome the limitations. Two among them have received great attention from the community. The one is Capsule Networks (CapsNet) which is inspired by the information processing in the human brain [102]. Compared to CNNs, CapsNet is more inductively-biased where the partial information processing in the human brain is integrated into the model, *e.g.*, the transformation process. The other is Vision Transformer(ViT) [25]. Given the success of Transformer in natural language processing (NLP), the work [25] generalizes Transformer architectures to image classification task by representing the input image as a sequence of image patches. Compared to CNNs, ViTs are less inductive-biased where information aggregation is also possible at lower layers. Convolutional Neural Networks, Capsule Networks, and Vision Transformers raise great attention in the community. Hence, in this work, we mainly focus on the three deep visual classification models.

In the rest of this chapter, we first introduce background knowledge about CNNs, CapsNets, and ViTs in Section 1.2. Then, in Section 1.3, we present a summary of the explainability of deep visual classifications and describe our contributions to the explainability of deep visual classification models. Last, in Section 1.4, we show the categorization of the robustness of deep visual classifications and describe our contributions to the robustness of deep visual classification models.

Contributions. In this dissertation, our contributions can be summarized from two perspectives. From the perspective of explainability, we first present a novel method, called CLRP, to explain CNN-based image classifications in Chapter 2. Then, in Chapter 3, we present our interpretable capsule networks whose predictions can be explained with built-in modules. Last, we show our understanding of ViT-based image classifications in Chapter 7. From the perspective of robustness, our contributions mainly focus on the role the model architecture plays in terms of both natural robustness and adversarial robustness. We present our findings and improvements of Capsule Networks' natural robustness to non-additive perturbation in Chapters 4 and 5, and further propose our adversary Vote Attack method to show the vulnerability of CapsNets in Chapter 6. Besides, we introduce our understanding of the robustness of ViT-based classifications to patch-wise perturbations in Chapter 7.

1.2 Background Knowledge

1.2.1 Convolutional Neural Networks

To recognize the patterns of the images, many operations have been proposed, e.g., Scale-Invariant Feature Transform (SIFT) [84], Histogram of Oriented Gradients(HOG) [22], and Convolution. Especially, the convolutional operation dominates the community in the last decade as an image feature extraction operation.

Formally, convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other. In the domain of computer vision, the discrete variant of convolution is adopted since the images are saved as discrete signals. Concretely, given an image $\boldsymbol{X} \in \mathbb{R}^{(C \times H \times W)}$ and a convolution kernel $\boldsymbol{k} \in \mathbb{R}^{(C \times P \times Q)}$, the feature map $\boldsymbol{H} \in \mathbb{R}^{(H' \times W')}$ extracted by the convolution kernel is computed as

$$\boldsymbol{H}_{(i,\ j)} = \sum_{c=1}^{C} \sum_{p=1}^{P} \sum_{q=1}^{Q} \boldsymbol{X}_{(c,\ i+p-1,\ j+q-1)}\ \boldsymbol{k}_{(c,\ p,\ q)}, \tag{1.1}$$

where (i, j) is the index of elements in the feature map $\boldsymbol{H}$, C is the number of channels of input images and (P, Q) are the size of the feature map. A single kernel corresponds to a single feature map. Multiple kernels are often applied to extract multiple feature maps.

Besides, the pooling (subsampling) operation is applied to the feature maps extracted by convolution operation to aggregate the visual information. In the pooling operation, the mean operation or the max operation is often applied. The pooling operation with size (s, s) can be expressed as

$$\boldsymbol{H}'_{(i,\ j)} = \max_{p=1}^{P} \boldsymbol{H}_{(i,\ j)}. \tag{1.2}$$

Convolution can be further applied to the pooled feature maps. The convolutional and pooling operations are applied alternatively on the image to obtain the final feature maps.

The features $\boldsymbol{H}^{L}_{(i,\ j)}$ extracted by a list of convolutional operations and pooling operations are taken as the final image representation. A single or multiple fully connected layers (i.e. a MLP module) is used as classifier that maps the features into the ground-truth class.

$$\boldsymbol{Z} = MLP(\boldsymbol{H}^{L}_{(i,\ j)}) \tag{1.3}$$

The output probabilities can be obtained by applying softmax function on the logits $\boldsymbol{Z}$. The predicted class is defined as $argmax(\boldsymbol{Z}_i)$.

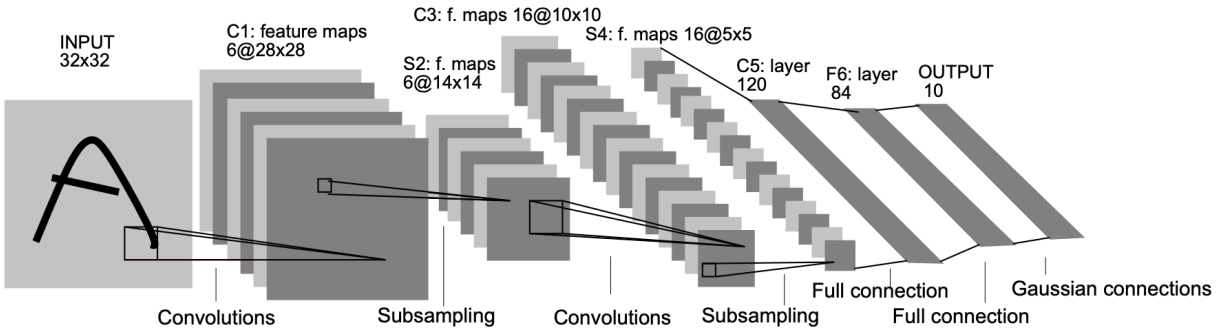


Figure 1.2: The overview of LeNet-5 architecture [70].

The work [70] proposes Convolution Neural Network (CNN) in the end-to-end learning framework to recognize hand-written digits. Therein, LeNet-5 is the classic instance of convolution neural networks, which is visualized in Fig. 1.2. The proposed LeNet-5 starts with two convolutional layers, and each is followed by a pooling layer. Then, a three-layer MLP module maps the feature to the logits.

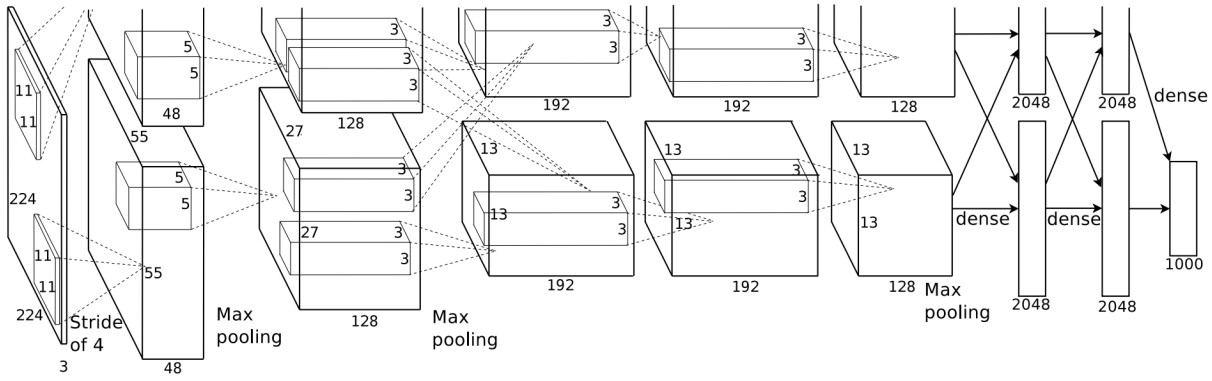


Figure 1.3: The overview of AlexNet architecture [68].

Given the limited computational resource, the architecture and the corresponding training strategy proposed in [70] does not scale well to the large-scale dataset. With the advance of the computational power, the work [68] proposes AlexNet, which achieves impressive accuracy on ImageNet-1k dataset. AlexNet consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. In terms of model architecture, AlexNet is deeper and wider than LeNet-5. From the perspective training strategy, to make AlexNet work well, the work [68] proposes non-saturating neurons, i.e., Rectified Linear Units (ReLUs) to activate the neurons and

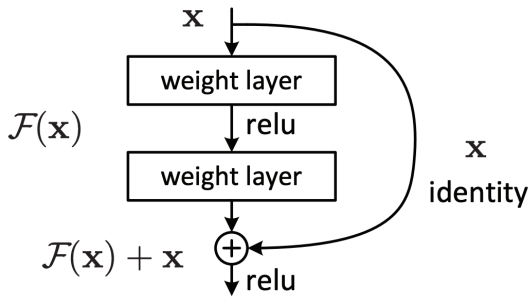


Figure 1.4: The overview of Residual block with skip connection [49].

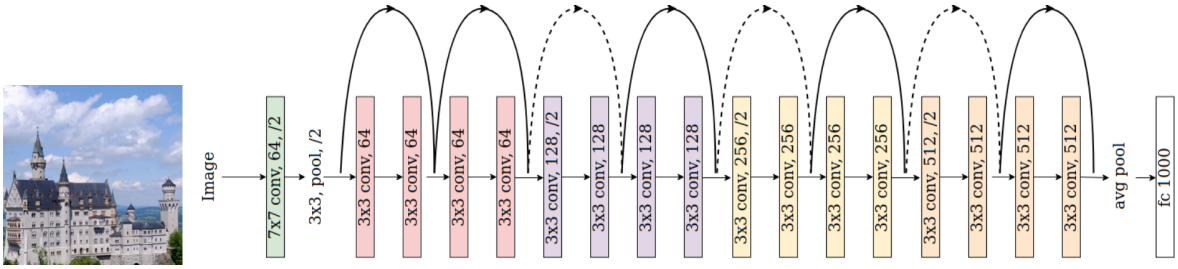


Figure 1.5: The overview of ResNet architecture [49].

employs dropout method to regularize the training process. Especially, they propose a GPU-specific implementation of GPU operation to make the training process feasible.

One intuitive way to improve AlexNet is to build deeper layers. However, the AlexNet with deeper layers does not converge well during training due to the gradient vanishing problem. Namely, the gradients become zeros or close to zeros when propagating from the output layer to low layers. Due to the gradient vanishing problem, the parameter update of low layers is challenging. To overcome the challenges, the work [49] proposes skip-connection, which can propagate the gradients from deep layers to low layers directly by skipping some intermediate layers.

The block with such a skip connection is called residual block. A popular residual block is shown in Fig. 1.4. As an instance, the work [49] proposes ResNet which consists of a list of residual blocks. When equipped with skip connections, ResNets with even more than 100 layers can converge well. ResNets still dominate the computer vision community. We show the ResNet18 in Fig. 1.5 as an example where 18 layers are built into the ResNet to extract features.

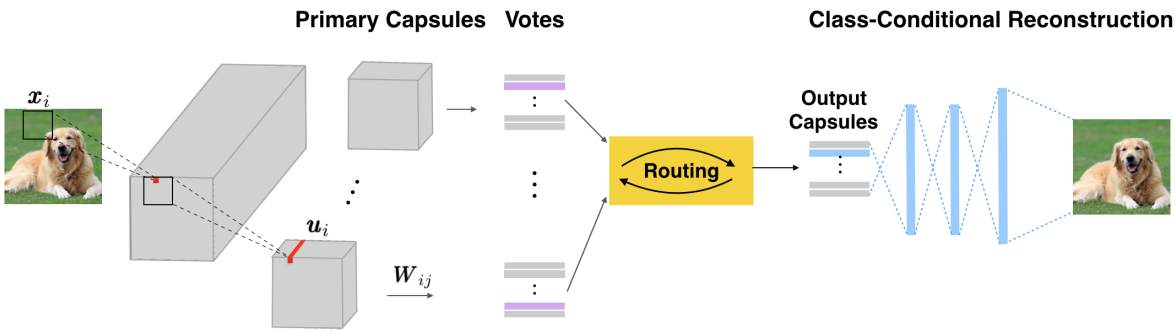


Figure 1.6: The overview of CapsNet architectures. The CapsNet architecture consists of four components, such as primary capsule extraction, voting, routing, and class-conditional reconstruction. The primary capsule extraction module first maps the raw input features to low-level capsules. The voting process transforms low-level capsules to make votes with a transformation matrix. Then, the routing module identifies the weight of each vote and computes the final high-level capsules. In the last part, the reconstruction subnetwork reconstructs input images from capsules to regularize the learning process.

Convolutional Network Follow-Ups: The CNN-based deep visual classifier has already surpassed human-level performance in the image classification task [63]. In the last years, the architectures of convolutional neural networks have still been improved from different perspectives. On the one hand, the more advanced architectures have been proposed to further push the state-of-the-art performance [121, 57, 21]. On the other hand, the efficiency of architecture has received great attention since real-world CNN-based applications often require less memory consumption and computational cost. The efficiency of architecture has been addressed from different perspective, e.g., light-weight architecture design [56, 143], architecture pruning [71, 47, 45, 87], and distilling knowledge from large architectures to small architectures [53, 99, 38]. More recently, many researchers focus on neural architecture search where the architectures are searched automatically from a predefined search space [148, 77, 78]. The found architecture can surpass the manually designed ones.

1.2.2 Capsule Networks

Inspired by the information process in the human brain, Hinton proposes Capsule Networks (CapsNet) [102]. Different from CNNs, CapsNets represent a visual entity with a vector instead of a single scale value, called Capsule. CapsNets [102] encode visual entities with

capsules. Each capsule is represented by an activity vector (e.g., the activation of a group of neurons), and elements of each vector encode the properties of the corresponding entity. The length of the activation vector indicates the confidence of the entity's existence. The output classes are represented as high-level capsules.

The most popular version of Capsule Networks is Dynamic Routing Capsule Networks (DR-CaosNet). We introduce the architecture details of DR-CapsNet as follows. As shown in Fig. 1.6, CapsNet starts with one (or more) convolutional layer(s) that convert the raw pixel intensities $\boldsymbol{X}$ into low-level visual entities $\boldsymbol{u}_i$. Concretely, CapsNet extracts feature maps of shape (C', H', W') from input image $\boldsymbol{X} \in \mathbb{R}^{(C \times H \times W)}$ with two standard convolutional layers where C', H', W' are the number of channels, the height, and the width of the feature maps, respectively. The extracted feature maps are reformulated as primary capsules $(C'/D_{in}, H', W', D_{in})$ where D_{in} is the dimensions of the primary capsules. There are $N = C'/D_{in} * H' * W'$ primary capsules all together. Each capsule $\boldsymbol{u}_i$, a D_{in}-dimensional vector, consists of D_{in} units across D_{in} feature maps at the same location. For example, the red bar marked with $\boldsymbol{u}_i$ in Fig. 1.6 is a low-level capsule.

In the voting process, each primary capsule is transformed to make a vote with a transformation matrix $\boldsymbol{W}_{ij} \in \mathbb{R}^{(D_{in} \times N * D_{out})}$ in, where N is the number of output classes and D_{out} is the dimensions of output capsules. The vote from the i-th low-level capsules to the j-th high-level capsules is

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij} \boldsymbol{u}_i. \tag{1.4}$$

Then, a routing module is applied to identify weight for each vote. Given all N votes $\hat{\boldsymbol{u}}_{j|i}$ of the L-th layer with N capsules, M high-level capsule $\boldsymbol{s}_j$ of the $(L+1)$-th layer with M capsules, the routing process is

$$\boldsymbol{s}_j = \sum_{i}^{N} c_{ij} \hat{\boldsymbol{u}}_{j|i} \tag{1.5}$$

where c_{ij} is a coupling coefficient that models the degree with which $\hat{\boldsymbol{u}}_{j|i}$ is able to predict $\boldsymbol{s}_j$. The capsule $\boldsymbol{s}_j$ is shrunk to a length in [0, 1) by a non-linear squashing function $g(\cdot)$, which is defined as

$$\boldsymbol{v}_j = g(\boldsymbol{s}_j) = \frac{\|\boldsymbol{s}_j\|^2}{1 + \|\boldsymbol{s}_j\|^2} \frac{\boldsymbol{s}_j}{\|\boldsymbol{s}_j\|}. \tag{1.6}$$

By doing the squashing operation, the length of the vector is mapped to [0, 1) that represents the confidence of the high-level entity's existence. In DR-CapsNet, the high-level capsules correspond to output classes, and its length means the output probability.

Note that the coupling coefficients $\{c_{ij}\}$ in Equation 1.5 are computed by an iterative routing procedure. They are updated so that high agreement ($a_{ij} = \boldsymbol{v}_j^T \hat{\boldsymbol{u}}_{j|i}$) corresponds to a high value of c_{ij}.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{1.7}$$

where initial logits b_{ik} are the log prior probabilities and updated with $b_{ik} = b_{ik} + a_{ij}$ in each routing iteration. The coupling coefficients between a i-th capsule of the L-th layer and all capsules of the $(L+1)$-th layer sum to 1, i.e., $\sum_{j=1}^{M} c_{ij} = 1$. The steps in Equations 1.9, 1.5, 1.6, and 1.7 are repeated K times in the routing process, where $\boldsymbol{s}_j$ and c_{ij} depend on each other.

The length of the final output capsule $\boldsymbol{v}_j$ corresponds to the output probability of the j-th class. Different from CNNs where cross-entropy loss is often applied to compute classification loss. In DR-CapsNet, the margin loss function is applied to compute the classification loss

$$\begin{aligned} L_k =& T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 \\ &+ \lambda(1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2 \end{aligned} \tag{1.8}$$

where $T_k = 1$ if the object of the k-th class is present in the input. As in [102], the hyper-parameters are often empirically set as $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$.

A reconstruction sub-network reconstructs the input image from all N output capsules with a masking mechanism. The ones corresponding to the non-ground-truth classes are masked with zeros before being transferred to the reconstruction sub-network. Due to the masking mechanism, only the capsule of the ground-truth class is visible for the reconstruction. Hence, the reconstruction process is called class-conditional reconstruction. The reconstruction loss is computed as a regularization term in the loss function.

Capsule Network Follow-Ups: Many routing mechanisms have been proposed to improve the performance of CapsNet, such as Expectation-Maximization Routing [54], Self-Routing [43], Variational Bayes Routing [96], Straight-Through Attentive Routing [2], and Inverted Dot-Product Attention routing [126]. An alternative to the routing mechanism to aggregate information is proposed in work [34] where they replace the dynamic routing with a multi-head attention-based graph pooling approach. To reduce the parameters of CapsNet, a matrix or a tensor is used to represent an entity instead of a vector [54, 95]. The size of the learnable transformation matrix can also be reduced by the matrix/tensor representations. Besides, the work [37] proposes to share a transformation matrix to reduce the

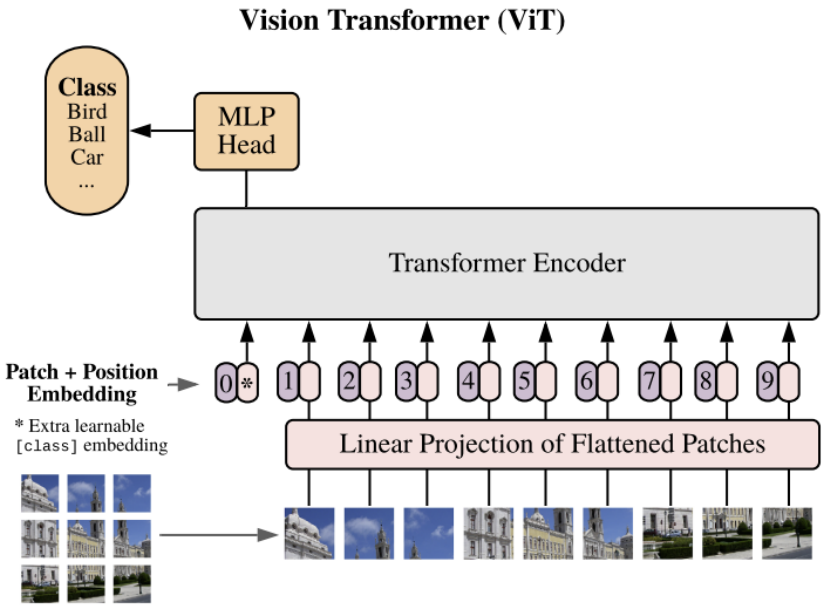


Figure 1.7: The overview of Vision Transformer Architectures. The figure is taken from [25].

network parameters. Another way to improve CapsNet is to integrate advanced modules of ConvNet into CapsNet, e.g., skip connections [49, 95] and dense connections [57, 92].

1.2.3 Vision Transformers

Transformers with self-attention-based architectures have become the model of choice in natural language processing (NLP) [127]. Inspired by the success of Transformers in NLP community, the work [25] proposes Vision Transformer(ViT) where they replace the convolutions entirely with self-attention layers and achieve remarkable performance in the image classification task. As a promising alternative to CNNs, Vision Transformer raises the great attention of our community.

Different from CNNs, ViT represents an input image as a sequence of image patches. Then, the list of self-attention modules are applied to the sequence of image patches sequentially. We now introduce the details of the primary Vision Transformer architecture in [25]. As shown in Fig. 1.7, the input image $\boldsymbol{X} \in \mathbb{R}^{(C \times H \times W)}$ is split into image patches $\{\boldsymbol{x}_i \in \mathbb{R}^{P \times P \times C} | i \in (1, 2, 3, ..., H/P \times W/P)\}$ where P is the patch size. The embedding of each patch is extracted from the raw image patch with linear projection parameters $\boldsymbol{W}^0 \in \mathbb{R}^{(HW/P^2 \times D_p)}$. Before the application of self-attention module, the position informa-

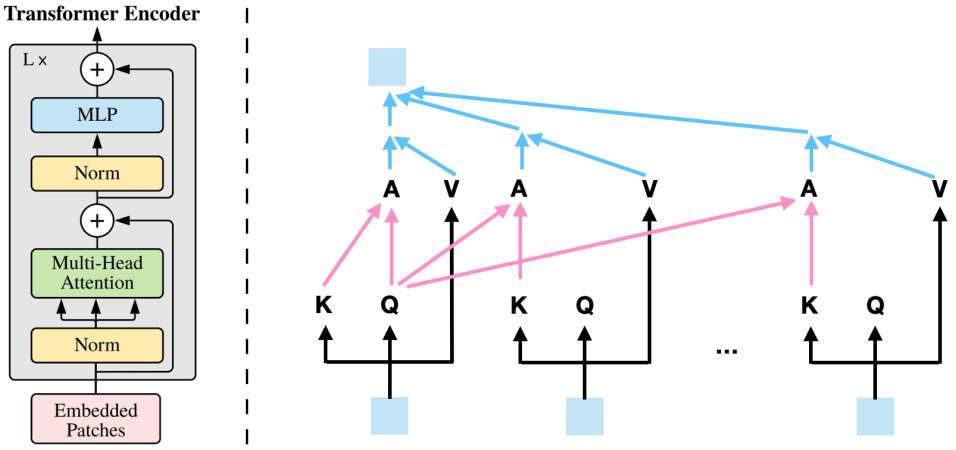


Figure 1.8: The overview of Transformer Encoder.

tion of image patches is also integrated into the patch embedding. The embedding of the patch $\boldsymbol{x}_i$ is described as

$$\boldsymbol{E}_i^0 = \boldsymbol{x}_i \cdot \boldsymbol{W}^0 + \boldsymbol{PE}_i, \tag{1.9}$$

where $\boldsymbol{PE}_i$ is the position embedding of the image patch $\{\boldsymbol{x}_i$, which encodes the patch position information in the input images. The position embedding $\boldsymbol{PE}_i$ could be manually designed or learnable. In ViT, the learnable version is adopted.

A learnable class-token embedding $\boldsymbol{E}_0^0$ is added into the list of patch embeddings. The class embedding in the last layer is taken as the image embedding for classification. We now introduce the transformer encoder where the list of blocks is applied to transform the input embeddings. As shown in Fig. 1.8, each block consists of two main modules, namely, a multi-head self-attention module to model the inter-patch relationship and an MLP module to project each patch respectively.

When the self-attention module with a single head in $l+1$-th layer is applied to input patches $\{\boldsymbol{E}_i^l \in \mathbb{R}^{D_p} | i \in (0, 1, 2, 3, ..., H/P \times W/P)\}$ in the l-th layer, the output embedding of the patch $\boldsymbol{E}_i^l$ is

$$\begin{aligned}
\boldsymbol{K}_i^{l+1} &= \boldsymbol{W}_k^{l+1} \cdot \boldsymbol{E}_i^l, \\
\boldsymbol{Q}_i^{l+1} &= \boldsymbol{W}_q^{l+1} \cdot \boldsymbol{E}_i^l, \\
\boldsymbol{V}_i^{l+1} &= \boldsymbol{W}_v^{l+1} \cdot \boldsymbol{E}_i^l, \\
\boldsymbol{A}_i^{l+1} = Softmax(\boldsymbol{Q}_i^{l+1} \cdot \boldsymbol{K}_0^{l+1}, &\quad \boldsymbol{Q}_i^{l+1} \cdot \boldsymbol{K}_1^{l+1}, \quad ..., \quad \boldsymbol{Q}_i^{l+1} \cdot \boldsymbol{K}_{H/P \times W/P+1}^{l+1},), \\
\boldsymbol{E}_i^{l+1} &= \sum_{j=1}^{H/P \times W/P+1} \boldsymbol{A}_{ij}^{l+1} \cdot \boldsymbol{V}_j.
\end{aligned} \tag{1.10}$$

In this equation, the key, query, and value of patch embedding is computed first. The attention of $\boldsymbol{E}_i^{l+1}$ to all patches in l-th layer is obtained with the query of i-th patch and all keys. The output embedding $\boldsymbol{E}_i^{l+1}$ is the weighted sum of all values of patches. The output embeddings of different heads are concatenated as the final embedding. Then, an MLP module with two MLP layers is applied to project the final embedding of each patch into a new feature space. The final embedding of the class-token patch is taken as the image representation to classify the image. A linear classifier maps the features to output space.

Vision Transformer Follow-Ups: Since the ViT was proposed, many new vision transformer architectures have been proposed [124, 44, 81]. A hybrid architecture that consists of both convolutional layers and self-attention blocks has also been explored [33, 136]. Besides, the pure patch-based architecture without attention mechanism has also been proposed [123]. By the time this thesis is written, the arm-race between ResNet and Vision Transformers is still going on [82]. Recently, many researchers employ the Transformer architecture as a uniform architecture that model both images and texts [93, 129].

Approach	Description
Saliency Maps	Identifying the relevance of each input pixel to the output class [114, 7, 109, 113, 120, 116, 41, 118, 20, 107, 97, 147, 28].
Counterfactual Explanation	Identifies how the given input could change such that the classifier would output a different specified class [12, 32].
Explanatory Sentences	Generating natural language sentences that describe the class-discriminative pixels [50, 51].
Supporting Training Images	Identifying training images most responsible for a given prediction [67].
Built-in Explanation	Generating Explanations with built-in modules (in explainable classifier) for a given prediction [67].
Disentangled Representations	Identifying the human-interpretable properties of the recognized object in the input image [106, 102, 61, 146].

Table 1.1: Summarization of different approaches for explaining image classifications.

1.3 Explanability of Deep Visual Classifications

1.3.1 Introduction

Deep Neural Networks (DNNs) have shown impressive performance in high-dimensional input data. Especially, the performance of DNNs can even surpass human-level performance in the image classification task. The traditional machine learning methods classify images with hand-crafted images, while DNNs make predictions based on the features learned automatically from data with an optimization algorithm. Hence, it is challenging to understand the classification decisions made by DNNs. In recent years, many directions have been explored to explain individual image classifications. We summarize and roughly categorize them in Table 1.1. We introduce each approach as follows.

Saliency Maps, as intuitive explanations, have received great attention in the community. The saliency map is a heat map, each element of which indicates the importance of the pixel in the corresponding position. The saliency map is expected to have recognizable patterns like the objects in the input image. The primary work [114] takes the vanilla gradient of the loss with respect to the input as the saliency map. However, the gradients are noisy and the pattern therein is barely recognizable. To improve the saliency map, many

methods have been proposed [114, 117, 7, 109, 113, 120, 116, 41, 118, 35, 36]. The primary method and the improved variants are model-aware, which leverage the parameters and the gradients of neural networks to compute saliency maps. Besides the model-aware methods, the model-agnostic saliency methods are also preferred in many scenarios. For example, they are able to explain any classifiers; the explanations produced from two or more different types of models are comparable; an ensemble model can be explained without requiring knowledge of model components. There are two types of model-agnostic saliency methods. The one is to build an explanation generation model, e.g. a neural network with U-net architecture [100, 20, 107]. The other is to approximate the local decision boundary of the underlying model with an explainable model, e.g., linear classifier [97]. The explanation generated from the explainable surrogate model can be used to explain individual decisions.

Counterfactual Explanation describes what changes to the situation would have resulted in arriving at the alternative decision. In the case of image classification, Counterfactual Explanation is the counterfactual image, which indicates that the output will become the target class if the input image is replaced with the counterfactual image. The work [12] creates a counterfactual image with a conditional generative model, which generates part of the pre-defined image region conditional on the rest of the image. The desired property of the generated image is to most change the classifier's decision. Another work [32] formulates the generation of the counterfactual image as an image editing problem. Their method performs well even in the fine-grained classifications.

Natural language, as a natural interface, has also been explored to explain the visual classifications. The works [50, 51] build modules to generate natural language sentences to explain the decisions where the sentences describe the class-discriminative features. The explanatory sentences are different from the caption/description generated by multi-model models. The contemporary vision-language models describe image content but fail to tell class-discriminative features which justify visual predictions.

Another way to explain visual classifications is to identify the training points most responsible for a given prediction. To trace a model's prediction back to its training data, the work [67] leverages influence functions, i.e., a classic technique from robust statistics. Given a classification, they can be the most responsible training image that supports the predictions. The created explanation can tell where the local decision boundary of the model came from at a specific data point.

The approaches introduced above are post-hoc. Namely, the explanations are created for off-shelf models without intervening in their training process. An alternative to post-

hoc explanation methods is to integrate dedicated modules into the model to be trained, e.g. attention mechanism [34], explanation module [20] and prototype module [14]. In the inference stage, the modules can be used to create explanations directly. The created explanations are dubbed built-in explanations, which are more efficient and easy to create.

The image representations learned by DNNs are often distributed, which makes the classification decision less explanation. It is difficult to interpretable the decision process inside the model. One way to mitigate this problem is to constrain the model to learn disentangled representations where each element of representation corresponds to a human-understandable concept [106, 102, 61, 146].

In this subsection, we have introduced the popular methods applied to explain individual classification decisions. In the rest of this section, we present our contributions towards understanding the classifications. Specifically, we briefly introduce our works on the topic of explaining classification decisions made by Convolutional Networks, Capsule Networks, and Vision Transformers.

1.3.2 Explainability of Convolutional Neural Network-based Classification

A large number of saliency methods have been proposed to better understand individual decisions of deep convolutional neural networks. As one of the representatives, the Layer-wise Relevance Propagation (LRP) approach is able to create pixel-wise explanatory saliency maps. LRP method has also been widely applied to many tasks in different domains, e.g., in medical domain [140] and in NLP [5].

The explanations generated by LRP are known to be pixel-wise and instance-specific. However, the discriminativeness of the explanations has not been evaluated yet. Ideally, the visualized objects in the explanation should correspond to the class that the class-specific neuron represents. Namely, the explanations should be class-discriminative.

Our work [41] evaluates the discriminativeness of the explanations generated by LRP. Concretely, we evaluate the explanations generated by LRP on the off-the-shelf models, e.g., VGG16 [115] pre-trained on the ImageNet dataset [23]. The results are shown in Fig. 1.9. For each test image, we create four saliency maps as explanations. The first three explanation maps are generated for top-3 predictions, respectively. The fourth one is created for randomly chosen 10 classes from the top-100 predicted classes (which ensure that the score to be propagated is positive). The white text in each explanation map indicates the

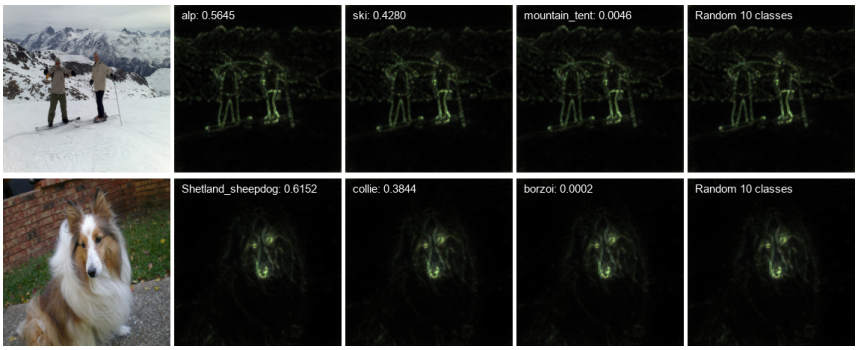


Figure 1.9: The explanations generated by LRP on VGG16 Network. The images from validation datasets of ImageNet are classified using the off-the-shelf models pre-trained on the ImageNet. The classifications of the images are explained by the LRP approach. For each image, we generate four explanations that correspond to the top-3 predicted classes and a randomly chosen multiple-classes. The explanations are not class-discriminative.

class the output neuron represents and the corresponding classification probability. The generated explanations are instance-specific, but not class-discriminative. In other words, they are independent of class information. The explanations for different target classes, even randomly chosen classes, are almost identical.

Based on LRP, our work [41] proposes Contrastive Layer-wise Relevance Propagation (CLRP), which is capable of producing instance-specific, class-discriminative, pixel-wise explanations. Before introducing our CLRP, we first discuss the conservative property in the LRP. In a DNN, given the input $\boldsymbol{X} = \{x_1, x_2, x_3, \cdots, x_n\}$, the output $\boldsymbol{Y} = \{y_1, y_2, y_3, \cdots, y_m\}$, the score S_{y_j} (activation value) of the neuron y_j before softmax layer, the LRP generate an explanation for the class y_j by redistributing the score S_{y_j} layer-wise back to the input space. The assigned relevance values of the input neurons are $\boldsymbol{R} = \{r_1, r_2, r_3, \cdots, r_n\}$. The conservative property is defined as follows: The generated saliency map is conservative if the sum of assigned relevance values of the input is equal to the score of the class-specific neuron, $\sum_{i=1}^{n} r_i = S_{y_j}$.

The overview of the CLRP are shown in Fig. 1.10. We first describe the LRP as follows. The j-th class-specific neuron y_j is connected to input variables by the weights $\boldsymbol{W}$ of layers between them. The neuron y_j models a visual concept O. For an input example $\boldsymbol{X}$, the LRP maps the score S_{y_j} of the neuron back into the input space to get relevance vector $\boldsymbol{R} = f_{LRP}(\boldsymbol{X}, \boldsymbol{W}, S_{y_j})$. In our contrastive LRP, we construct a dual virtual concept $\overline{O}$, which models the opposite visual concept to the concept O. For instance, the concept O

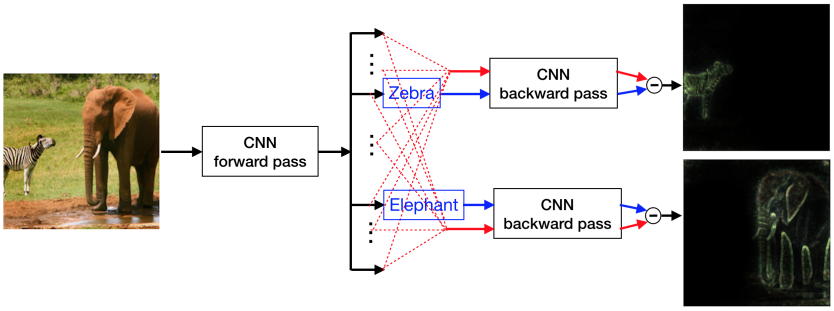


Figure 1.10: The figure shows an overview of our CLRP. For each predicted class, the approach generates a class-discriminative explanation by comparing two signals. The blue line means the signal that the predicted class represents. The red line models a dual concept opposite to the predicted class. The final explanation is the difference between the two saliency maps that the two signal generate.

models the **zebra**, and the constructed dual concept $\overline{O}$ models the **non-zebra**. One way to model the $\overline{O}$ is to select all classes except for the target class representing O, i.e. the dashed red lines in Fig. 1.10 are connected to all classes except for the target class **zebra**. Next, the score S_{y_j} of target class is uniformly redistributted to other classes. Given the same input example $\boldsymbol{X}$, the LRP generates an explanation $\boldsymbol{R}_{dual} = f_{LRP}(\boldsymbol{X}, \overline{\boldsymbol{W}}, S_{y_j})$ for the dual concept. The Contrastive LRP is defined as follows:

$$\boldsymbol{R}_{CLRP} = \max(\boldsymbol{0}, (\boldsymbol{R} - \boldsymbol{R}_{dual})) \tag{1.11}$$

where the function $\max(\boldsymbol{0}, \boldsymbol{X})$ means replacing the negative elements of $\boldsymbol{X}$ with zeros. The difference between the two saliency maps cancels the common parts. Without the dominant common parts, the non-zero elements in $\boldsymbol{R}_{CLRP}$ are the most relevant pixels.

Besides the qualitative evaluation, we also evaluate the explanations quantitatively with a Pointing Game and an ablation study. Both qualitative and quantitative evaluations show that the CLRP generates better explanations than the LRP.

1.3.3 Explainability of Capsule Network-based Classification

Capsule Networks, as alternatives to Convolutional Neural Networks, have been proposed to recognize objects from images. The current literature demonstrates many advantages of CapsNets over CNNs. However, how to create explanations for individual classifications of CapsNets has not been well explored.

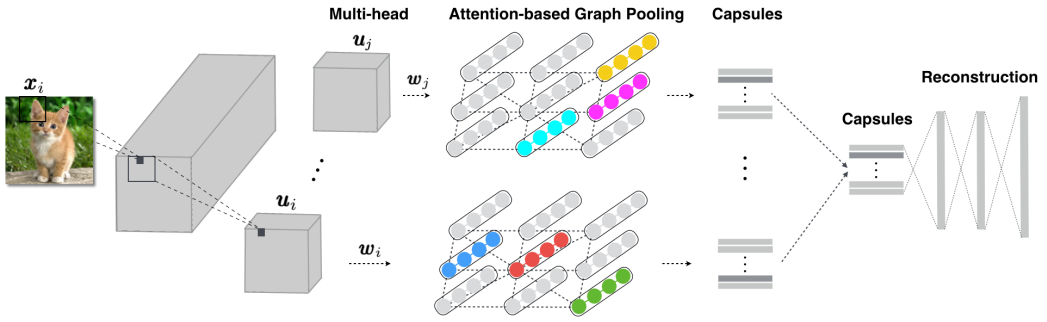


Figure 1.11: The illustration of GraCapsNets: The extracted primary capsules are transformed and modeled as multiple graphs. The pooling result on each graph (head) corresponds to one vote. The votes on multiple graphs (heads) are averaged to generate the final prediction.

The widely used saliency methods are mainly proposed for explaining CNN-based classifications; they create saliency map explanations by combining activation values and the corresponding gradients, e.g., Grad-CAM. They combine activation values and the received gradients in specific layers, e.g., deep convolutional layers. In CapsNets, instead of deep convolutional layers, an iterative routing mechanism is applied to extract high-level visual concepts. Hence, these saliency methods cannot be trivially applied to CapsNets. Besides, the routing mechanism makes it more challenging to identify interpretable input features relevant to a classification.

To overcome the lack of interpretability, we can either propose new post-hoc interpretation methods for CapsNets or modify the model to have build-in explanations. In our published work [34], we explore the latter. Specifically, we propose interpretable Graph Capsule Networks (GraCapsNets), where we replace the routing part with a multi-head attention-based Graph Pooling approach. Our GraCapsNet includes an attention-based pooling module, with which individual classification explanations can be created effectively and efficiently.

As introduced in Background Section, CapsNets start with convolutional layers that convert the input pixel intensities $\mathbf{X}$ into primary capsules $\mathbf{u}_i$ (i.e., low-level visual entities). Each $\mathbf{u}_i$ is transformed to vote for high-level capsules $\hat{\mathbf{u}}_{j|i}$ with learned transformation matrices. Then, a routing process is used to identify the coupling coefficients c_{ij}, which describe how to weight votes from primary capsules. Finally, a squashing function is applied to the identified high-level capsules $\mathbf{s}_j$ so that the lengths of them correspond to the confidence of the class's existence.

Different routing mechanisms differ only in how to identify c_{ij}. Routing processes describe one way to aggregate information from primary capsules into high-level ones. In our GraCapsNets, we implement the information aggregation by multi-head graph pooling processes. In CapsNets, the primary capsules represent object parts, e.g., the eyes and nose of a cat. In our GraCapsNets, we explicitly model the relationship between the primary capsules (i.e., part-part relationship) with graphs. Then, the followed graph pooling operations pool relevant object parts from the graphs to make a classification vote. Since the graph pooling operation reveals which input features are pooled as relevant ones, we can easily create explanations to explain the classification decisions.

The overview of our GraCapsNets is illustrated in Fig. 1.11. In GraCapsNet, the primary capsules $\mathbf{u}_i$ are transformed into a feature space. All transformed capsules $\mathbf{u}'_i$ are modeled as multiple graphs. Each graph corresponds to one head, the pooling result on which corresponds to one vote. The votes on multiple heads are averaged as the final prediction.

The transformed capsules $\mathbf{u}'_i$ can be modeled as multiple graphs. A graph consists of a set of nodes and a set of edges. As shown in Fig. 1.11, the primary capsules are reshaped from L groups of feature maps. Each group consists of C feature maps of the size $K \times K$. Correspondingly, the transformed capsules $\mathbf{u}'_i$ where $i \in \{1, 2, ...K^2\}$ form a single graph with K^2 nodes. Each node corresponds to one transformed capsule $\mathbf{u}'_i$, and the activation vector of $\mathbf{u}'_i$ is taken as features of the corresponding node. The graph edge can be represented by an adjacency matrix, where different priors can be modeled. The spatial relationship between primary capsules is modeled in our work.

Given node features $\mathbf{X}^l \in \mathbb{R}^{(K^2 \times D_{out})}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{(K^2 \times K^2)}$ in the l-th head of GraCapsNet. We first compute the attention of the head as $\mathbf{Att}^l = \text{softmax}(\mathbf{A}\mathbf{X}^l\mathbf{W})$ where $\mathbf{W} \in \mathbb{R}^{D_{out} \times M}$ are learnable parameters. D_{out} is the dimension of the node features and M is the number of output classes. The output is of the shape $(K^2 \times M)$. In our GraCapsNet for object recognition, $\mathbf{Att}^l$ corresponds to the visual attention of the heads. The graph pooling output $\mathbf{S}^l \in \mathbb{R}^{(M \times D_{out})}$ of the head is computed as $\mathbf{S}^l = (\mathbf{Att}^l)^T\mathbf{X}^l$. The final predictions of GraCapsNets are based on all L heads with outputs $\mathbf{S}^l$ where $l \in \{1, 2, ..., L\}$. The output capsules are $\mathbf{V} = \text{squash}(\frac{1}{L}\sum_{l=1}^{L}\mathbf{S}^l)$.

In our GraCapsNet, we can use visual attention as built-in explanation to explain the predictions of GraCapsNets. The averaged attenion over l heads is

$$\mathbf{E} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{Att}^l \tag{1.12}$$

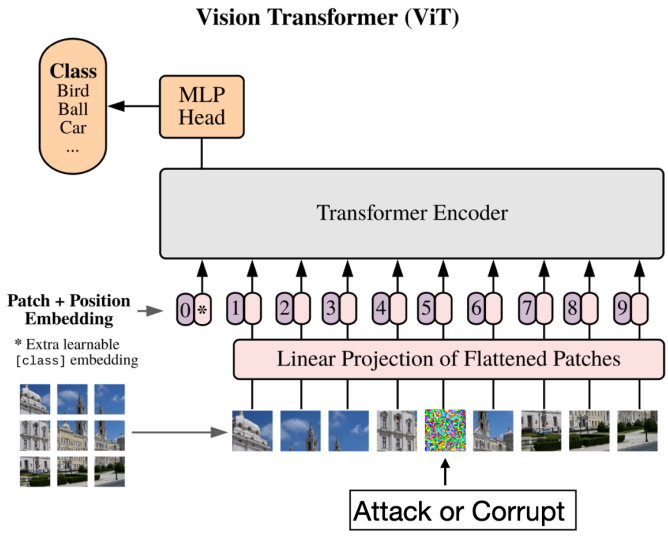


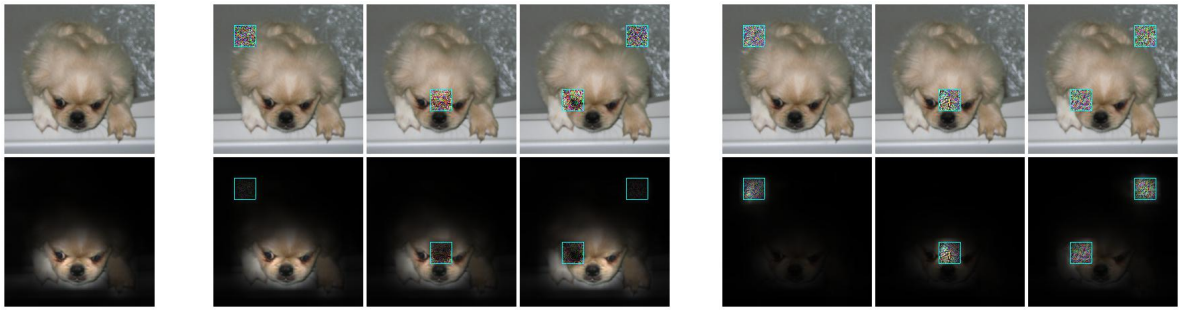
Figure 1.12: Adversarial Patch Attack or Natural Patch Corruption on Vision Transformer.

where $\mathbf{Att}^l$ corresponds to the attention of the l-th head. The created explanations $\mathbf{E}$ are of the shape $(K^2 \times M)$. Given the predicted class, the $K \times K$ attention map indicates which pixels of the input image support the prediction.

The explanations for individual classifications of GraCapsNets can be created in an effective and efficient way. Surprisingly, without a routing mechanism, our GraCapsNets can achieve better classification performance and better adversarial robustness, and still keep other advantages of CapsNets, namely, disentangled representations and affine transformation robustness.

1.3.4 Explainability of Vision Transformer-based Classification

The recent advances in Vision Transformer (ViT) have demonstrated its impressive performance in image classification [25, 124], which makes it a promising alternative to Convolutional Neural Network (CNN). Unlike CNNs, ViT represents an input image as a sequence of image patches. Then, a self-attention mechanism is applied to aggregate information from all patches. The attention can be used to create saliency maps to explain ViT-based classification decisions, e.g. with Rollout Attention method [1]. The patch-wise input image representation in ViT makes the following question interesting: How does the attention of ViT change when individual input image patches are perturbed with natural corruptions or adversarial perturbations? For example, Fig. 1.12 illustrates the case where a single patch of the input is perturbed or attacked.



(a) Clean Image (b) with Naturally Corrupted Patch (c) with Adversarial Patch

Figure 1.13: Images with patch-wise perturbations (top) and their corresponding attention maps (bottom). The attention mechanism in ViT can effectively ignore the naturally corrupted patches to maintain a correct prediction, whereas it is forced to focus on the adversarial patches to make a mistake. The images with corrupted patches are all correctly classified. The images with adversary patches in subfigure 1.13c are misclassified as *dragonfly*, *axolotl*, and *lampshade*, respectively.

In our work [40], we study the robustness of vision transformers to patch-wise perturbations. Surprisingly, we find that vision transformers are more robust to naturally corrupted patches than CNNs, whereas they are more vulnerable to adversarial patches. Furthermore, we conduct extensive qualitative and quantitative experiments to understand the classification under patch perturbations.

We have revealed that ViT's stronger robustness to natural corrupted patches and higher vulnerability against adversarial patches are both caused by the attention mechanism. Specifically, the attention model can help improve the robustness of vision transformers by effectively ignoring natural corrupted patches. However, when vision transformers are attacked by an adversary, the attention mechanism can be easily fooled to focus more on the adversarially perturbed patches and cause a mistake.

Digging down further, we find the reason behind this is that the self-attention mechanism of ViT can effectively ignore the natural patch corruption, while it's also easy to manipulate the self-attention mechanism to focus on an adversarial patch. This is well supported by rollout attention visualization [1] on ViT. As shown in Fig. 1.13 (a), ViT successfully attends to the class-relevant features on the clean image, i.e., the head of the dog. When one or more patches are perturbed with natural corruptions, shown in Fig. 1.13 (b), ViT can effectively ignore the corrupted patches and still focus on the main foreground to make a correct prediction. In Fig. 1.13 (b), the attention weights on the positions of

naturally corrupted patches are much smaller even when the patches appear in the foreground. In contrast, when the patches are perturbed with adversarial perturbations by an adversary, shown in Fig. 1.13 (c), ViT is successfully fooled to make a wrong prediction because the attention of ViT is misled to focus on the adversarial patches instead.

In our work [40], we provide our understanding of the attention changes of ViT when individual input image patches are perturbed with natural corruptions or adversarial perturbations.

Natural Robustness	Additive	Natural Corruption	Robustness to the noisy images that are added with various noise [72, 52], such as, white noise, blur, weather, and digital categories.
	Non-Additive	Affine Transformation	Robustness to the images that are affine-transformed from standard ones [13, 102, 37].
Adversarial Robustness	Additive	Dense Attack	Robustness to the images where all pixels can be changed under a certain constraint [30, 85].
		Sparse Attack	Robustness to the images where only a few pixels of each image can be manipulated [90].
		Patch Attack	Robustness to the perturbed images where only a single patch (a specific region) of each image can be manipulated [10, 62].
	Non-Additive	Transformation -Based Attack	Robustness to adversarial images that is created by delicated affine transformations [135].
		Sementic Attack	Robustness to semantic adversarial images that is created by image synthesis [55].

Table 1.2: Categorization of Robustness in Image Classification Task.

1.4 Robustness of Deep Visual Classification Models

1.4.1 Introduction

In this thesis, we mainly consider two types of robustness, namely, natural robustness and adversarial robustness. When an image is captured, different corruption can happen, e.g., the existence of white noise, the effect of weather, the compression in the digitalization process, and random affine transformation. The robustness to these images with natural corruption is denoted as natural robust. Adversarial robustness describes the robustness of models to adversarial images, which is created by an adversary. Both natural robustness and adversarial robustness are critical in some safety-critical domains. We summarize and categorize the robustness in Tab. 1.2.

Besides the type of attacks in Tab. 1.2, adversarial attacks can be categorized into targeted and untargeted ones. The goal of targeted attacks is to mislead the model to a specific target class, while the goal of untargeted ones is to fool the model to make wrong predictions.

In terms of the availability of the target models, adversarial attacks can also be categorized into white-box and black-box attacks. The white-box attacks assume that the

adversary has all access to target models including model parameters, model architectures, and even defense methods. In contrast, in the setting of black-box attacks, the adversary can only obtain the output of the target model. The black-box attacks have also received great attention since it is realistic in real-world applications.

The implementation of white-box attacks is relatively cheap where they create adversarial examples with the gradients of the self-defined objective function with respect to inputs. However, the implementation of black-box attacks can be computationally expensive given the limited available information. One way to created adversarial examples in a black-box fashion is to leverage their transferability [80, 138, 24, 149, 42, 134, 58, 59, 76, 130], namely, the adversarial examples created on one model can also fool another. The adversary first trains a surrogate model on the same training data as the one used for the target model and creates adversarial examples on the surrogate model to fool the target model, which is called transfer-based black-box attack. However, the transfer-based black-box attacks require access to the training data of the target model. To overcome the limitation, the query-based black-box attacks have been proposed where the attacks are based on the outputs obtained by querying the target models directly [16, 17, 8, 17, 3].

In addition, based on the constraints on the adversarial images, the generated adversarial perturbations can be quasi-imperceptible or unbounded. The popular metric of to measure the distance between clean images and adversarial image is ℓ_p norm [90], such as, ℓ_1, ℓ_2 and ℓ_∞. However, the metric is not perfectly aligned with human perception. The more advanced metric has also been explored in the community, e.g., Wasserstein distance [133].

Given the potential threats posed by adversarial attacks, many defense strategies have been proposed to build adversarially robust models. One of the most effective defense methods is adversarial training, which creates adversarial examples and adds them to the training dataset in each training iteration. Besides, the pre-processing methods have been explored to purify adversarial examples [31, 125, 11, 110, 141, 142, 4, 64, 132, 98, 128, 74, 131, 119]. However, some of the defense strategies have broken again in later publications [6]. Some defense methods provide certified robustness to break arm-race between adversary and defense [94, 60, 19, 75, 105, 103, 86, 104]. Even many methods have been published to address, the accuracy of the model under attacks is still much lower than the accuracy on clean images, especially on the large dataset [137]. In addition to building robust model, another way to address the threats is to detect adversarial examples first [139, 27, 89, 73, 145, 101, 18].

In this subsection, we categorize the robustness of image classifications. Our contributions of this thesis mainly focus on the role the model architecture plays in terms of both natural robustness and adversarial robustness. In the rest of this section, we present our contributions towards the robustness of image classification models, such as Capsule Networks and Vision Transformers.

1.4.2 Robustness of Capsule Network-based Classification

Human visual recognition is quite insensitive to affine transformations. For example, entities in an image, and a rotated version of the entities in the image, can both be recognized by the human visual system, as long as the rotation is not too large. Convolutional Neural Networks (CNNs), the currently leading approach to image analysis, achieve affine robustness by training on a large amount of data that contain different transformations of target objects. Given limited training data, a common issue in many real-world tasks, the robustness of CNNs to novel affine transformations is limited [102].

With the goal of learning image features that are more aligned with human perception, Capsule Networks (CapsNets) have recently been proposed [102]. Our work [37] first investigates the effectiveness of components that make CapsNets robust to input affine transformations, with a focus on the routing algorithm. However, recent work [88] shows that all routing algorithms proposed so far perform even worse than a uniform/random routing procedure.

From both numerical analysis and empirical experiments, our investigation reveals that the dynamic routing procedure contributes neither to the generalization ability nor to the affine robustness of CapsNets. Therefore, it is infeasible to improve the affine robustness by modifying the routing procedure. Instead, we investigate the limitations of the CapsNet architectures and propose a simple solution. Namely, we propose to apply an identical transformation function for all primary capsules and replace the routing with a simple averaging procedure.

Besides the high affine transformation robustness, CapsNets also demonstrate other advantages, such as the ability to recognize overlapping digits and the semantic representation compactness. In recent years, It has been suggested that CapsNets have the potential to surpass the dominant convolutional networks in these aspects [102, 54, 95, 37]. However, there lack of comprehensive comparisons to support this assumption, and even for some reported improvements, there are no solid ablation studies to figure out which ones of the components in CapsNets are, in fact, effective.

In our work [39], we first carefully examine the major differences in design between the capsule networks and the common convolutional networks adopted for image classification. The difference can be summarized as *a non-shared transformation module, a dynamic routing layer to automatically group input capsules to produce output capsules, a squashing function, a marginal classification loss, and a class-conditional reconstruction sub-network with a reconstruction loss.*

Unlike previous studies [102, 54] which usually take CapsNet as a whole to test its robustness, our work [39] instead tries to study the effects of each of the above components in their effectiveness on robustness. We consider the three different aspects, such as the robustness to affine transformations, the ability to recognize overlapping digits, and the semantic representation compactness.

Our investigations reveal that some widely believed benefits of Capsule networks could be wrong:

1. The dynamic routing actually may harm the robustness to input affine transformation, in contrast to the common belief;

2. The high performance of CapsNets to recognize overlapping digits can be mainly attributed to the extra modeling capacity brought by the transformation matrices.

3. Some components of CapsNets are indeed beneficial for learning semantic representations, e.g., the conditional reconstruction and the squashing function, but they are mainly auxiliary components and can be applied beyond CapsNets.

In addition to these findings, we also enhance common ConvNets by the useful components of CapsNet, and achieve greater robustness. Our investigation shows that Capsule Network is not more robust than Convolutional Network.

1.4.3 Robustness of Vision Transformer-based Classification

CapsNets with brain-inspired architectures have more inductive bias than CNNs. Different from CapsNet, Vision Transformer (ViT) [25] has less architecture bias than CNNs. ViT processes the input image as a sequence of image patches. Then, a self-attention mechanism is applied to aggregate information from all patches. Existing works have shown that ViTs are more robust than CNNs when the whole input image is perturbed with natural corruptions or adversarial perturbations [9, 111, 91]. Given the patch-based architecture of ViT, our work studies the robustness of ViT to patch-based perturbation.

Two typical types of perturbations are considered to compare the robustness between ViTs and CNN (e.g., ResNets [49]). One is natural corruptions [52], which is to test models' robustness under distributional shift. The other is adversarial perturbations [122, 31], which are created by an adversary to specifically fool a model to make a wrong prediction.

We reveal that ViT does not always perform more robustly than ResNet. When individual image patches are naturally corrupted, ViT performs more robustly than ResNet. However, when input image patch(s) are adversarially attacked, ViT shows a higher vulnerability. Digging down further, we find the reason behind this is that the self-attention mechanism of ViT can effectively ignore the natural patch corruption, while it's also easy to manipulate the self-attention mechanism to focus on an adversarial patch.

Based on the patch-based architectural structure of vision transformers, we further investigate the sensitivity of ViT against patch positions and patch alignment of adversarial patches. First, we discover that ViT is insensitive to different patch positions, while ResNet shows high vulnerability on the central area of input images and much less on corners. We attribute this to the architecture bias of ResNet where pixels in the center can affect more neurons than the ones in corners. In contrast, each patch within ViT can equally interact with other patches regardless of its position. Further, we find that for ViT, the adversarial perturbation designed to attack one particular position can successfully transfer to other positions of the same image as long as they are aligned with input patches. In contrast, the ones on ResNet hardly do.

To summarise, in our work [40], we compare ViT and CNNs in terms of the robustness to natural patch corruptions or adversarial patch attacks.

Bibliography

[1] S. Abnar und W. Zuidema: *Quantifying attention flow in transformers*, In *Annual Meeting of the Association for Computational Linguistics (ACL)*. (2020).

[2] K. Ahmed und L. Torresani: *STAR-caps: Capsule networks with straight-through attentive routing*, In *Advances in Neural Information Processing Systems*. (2019).

[3] M. Andriushchenko, F. Croce, N. Flammarion und M. Hein: *Square attack: a query-efficient black-box adversarial attack via random search*, In *ECCV*. (2020).

[4] M. Andriushchenko und N. Flammarion, *arXiv preprint arXiv:2007.02617* (2020).

[5] L. Arras, G. Montavon, K.R. Müller und W. Samek, *arXiv preprint arXiv:1706.07206* (2017).

[6] A. Athalye, N. Carlini und D. Wagner: *Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples*, In *International conference on machine learning*. PMLR (2018) Seiten 274–283.

[7] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.R. Müller und W. Samek: *On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation*, In *PloS one*. (2015).

[8] A.N. Bhagoji, W. He, B. Li und D. Song: *Practical black-box attacks on deep neural networks using efficient query mechanisms*, In *ECCV*. (2018).

[9] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner und A. Veit, *arXiv:2103.14586* (2021).

[10] T.B. Brown, D. Mané, A. Roy, M. Abadi und J. Gilmer, *arXiv preprint arXiv:1712.09665* (2017).

[11] N. Carlini und D. Wagner: *Towards evaluating the robustness of neural networks*, In *2017 ieee symposium on security and privacy (sp)*. IEEE (2017) Seiten 39–57.

[12] C.H. Chang, E. Creager, A. Goldenberg und D. Duvenaud: *Explaining Image Classifiers by Counterfactual Generation*, In *ICLR*. (2019).

[13] S. Chang, J. Yang, S. Park und N. Kwak: *Broadcasting convolutional network for visual relational reasoning*, In *Proceedings of the European Conference on Computer Vision (ECCV)*. (2018) Seiten 754–769.

[14] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin und J.K. Su, *Advances in neural information processing systems* **32** (2019).

[15] L.C. Chen, G. Papandreou, F. Schroff und H. Adam: *Rethinking atrous convolution for semantic image segmentation*, In *arXiv:1706.05587*. (2017).

[16] P.Y. Chen, H. Zhang, Y. Sharma, J. Yi und C.J. Hsieh: *Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models*, In *Proceedings of the 10th ACM workshop on artificial intelligence and security*. (2017).

[17] M. Cheng, T. Le, P.Y. Chen, J. Yi, H. Zhang und C.J. Hsieh, *arXiv preprint arXiv:1807.04457* (2018).

[18] G. Cohen, G. Sapiro und R. Giryes: *Detecting adversarial samples using influence functions and nearest neighbors*, In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. (2020) Seiten 14 453–14 462.

[19] J. Cohen, E. Rosenfeld und Z. Kolter: *Certified adversarial robustness via randomized smoothing*, In *International Conference on Machine Learning*. PMLR (2019) Seiten 1310–1320.

[20] P. Dabkowski und Y. Gal: *Real time image saliency for black box classifiers*, In *Advances in Neural Information Processing Systems*. (2017) Seiten 6967–6976.

[21] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu und Y. Wei: *Deformable convolutional networks*, In *Proceedings of the IEEE International Conference on Computer Vision*. (2017) Seiten 764–773.

[22] N. Dalal und B. Triggs: *Histograms of oriented gradients for human detection*, In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Band 1. Ieee (2005) Seiten 886–893.

[23] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li und L. Fei-Fei: *Imagenet: A large-scale hierarchical image database*, In *2009 IEEE conference on computer vision and pattern recognition*. Ieee (2009) Seiten 248–255.

[24] Y. Dong, T. Pang, H. Su und J. Zhu: *Evading defenses to transferable adversarial examples by translation-invariant attacks*, In *CVPR*. (2019).

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly et al. : *An image is worth 16x16 words: Transformers for image recognition at scale*, In *arXiv:2010.11929*. (2020).

[26] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno und D. Song: *Robust physical-world attacks on deep learning visual classification*, In *Proceedings of the IEEE conference on computer vision and pattern recognition*. (2018) Seiten 1625–1634.

[27] R. Feinman, R.R. Curtin, S. Shintre und A.B. Gardner, *arXiv preprint arXiv:1703.00410* (2017).

[28] R.C. Fong und A. Vedaldi: *Interpretable explanations of black boxes by meaningful perturbation*, In *Proceedings of the IEEE International Conference on Computer Vision*. (2017) Seiten 3429–3437.

[29] R. Girshick, J. Donahue, T. Darrell und J. Malik: *Rich feature hierarchies for accurate object detection and semantic segmentation*, In *CVPR*. (2014) Seiten 580–587.

[30] I. Goodfellow, J. Shlens und C. Szegedy: *Explaining and harnessing adversarial examples*, In *ICLR*. (2014).

[31] I.J. Goodfellow, J. Shlens und C. Szegedy, *arXiv preprint arXiv:1412.6572* (2014).

[32] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh und S. Lee: *Counterfactual Visual Explanations*, In *ICML*. (2019).

[33] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou und M. Douze: *LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference*, In *arXiv:2104.01136*. (2021).

[34] J. Gu: *Interpretable Graph Capsule Networks for Object Recognition*, In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. (2020).

[35] J. Gu und V. Tresp: *Saliency methods for explaining adversarial attacks*, In *arXiv preprint arXiv:1908.08413*. (2019).

[36] J. Gu und V. Tresp: *Semantics for Global and Local Interpretation of Deep Neural Networks*, In *arXiv preprint arXiv:1910.09085*. (2019).

[37] J. Gu und V. Tresp: *Improving the Robustness of Capsule Networks to Image Affine Transformations*, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (2020) Seiten 7285–7293.

[38] J. Gu und V. Tresp: *Search for better students to learn distilled knowledge*, In *arXiv preprint arXiv:2001.11612*. (2020).

[39] J. Gu, V. Tresp und H. Hu: *Capsule network is not more robust than convolutional network*, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. (2021) Seiten 14 309–14 317.

[40] J. Gu, V. Tresp und Y. Qin: *Are Vision Transformers Robust to Patch Perturbations?*, In *arXiv preprint arXiv:2111.10659*. (2021).

[41] J. Gu, Y. Yang und V. Tresp: *Understanding Individual Decisions of CNNs via Contrastive Backpropagation*, In *Asian Conference on Computer Vision*. (2018).

[42] Y. Guo, Q. Li und H. Chen: *Backpropagating linearly improves transferability of adversarial examples*, In *NeurIPS*. (2020).

[43] T. Hahn, M. Pyeon und G. Kim: *Self-Routing Capsule Networks*, In *Advances in Neural Information Processing Systems (NeurIPS)*. (2019) Seiten 7658–7667.

[44] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu und Y. Wang: *Transformer in transformer*, In *arXiv:2103.00112*. (2021).

[45] S. Han, J. Pool, J. Tran und W. Dally: *Learning both weights and connections for efficient neural network*, In *NeurIPS*. (2015) Seiten 1135–1143.

[46] T.B. Hashimoto, M. Srivastava, H. Namkoong und P. Liang: *Fairness Without Demographics in Repeated Loss Minimization*, In *ICML*. (2018).

[47] B. Hassibi und D.G. Stork: *Second order derivatives for network pruning: Optimal brain surgeon*, In *NeurIPS*. (1993).

[48] K. He, G. Gkioxari, P. Dollár und R. Girshick: *Mask r-cnn*, In *International Conference on Computer Vision (ICCV)*. (2017) Seiten 2961–2969.

[49] K. He, X. Zhang, S. Ren und J. Sun: *Deep residual learning for image recognition*, In *Proceedings of the IEEE International Conference on Computer Vision*. (2016) Seiten 770–778.

[50] L.A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele und T. Darrell: *Generating visual explanations*, In *European Conference on Computer Vision (ECCV)*. Springer (2016) Seiten 3–19.

[51] L.A. Hendricks, R. Hu, T. Darrell und Z. Akata: *Grounding visual explanations*, In *European Conference on Computer Vision (ECCV)*. Springer (2018) Seiten 269–286.

[52] D. Hendrycks und T. Dietterich: *Benchmarking neural network robustness to common corruptions and perturbations*, In *International Conference on Learning Representations (ICLR)*. (2019).

[53] G. Hinton, O. Vinyals und J. Dean: *Distilling the Knowledge in a Neural Network*, In *stat*, Band 1050. (2015) Seite 9.

[54] G.E. Hinton, S. Sabour und N. Frosst: *Matrix capsules with EM routing*, In *International conference on learning representations (ICLR)*. (2018).

[55] H. Hosseini und R. Poovendran: *Semantic adversarial examples*, In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. (2018) Seiten 1614–1619.

[56] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto und H. Adam: *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, In *arXiv preprint arXiv:1704.04861*. (2017).

[57] G. Huang, Z. Liu, L. Van Der Maaten und K.Q. Weinberger: *Densely connected convolutional networks*, In *Proceedings of the IEEE International Conference on Computer Vision*. (2017) Seiten 4700–4708.

[58] Q. Huang, I. Katsman, H. He, Z. Gu, S. Belongie und S.N. Lim: *Enhancing adversarial example transferability with an intermediate level attack*, In *International Conference on Computer Vision (ICCV)*. (2019).

[59] N. Inkawhich, K.J. Liang, B. Wang, M. Inkawhich, L. Carin und Y. Chen: *Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability*, In *NeurIPS*. (2020).

[60] J. Jia, X. Cao, B. Wang und N.Z. Gong, *arXiv preprint arXiv:1912.09899* (2019).

[61] D. Jung, J. Lee, J. Yi und S. Yoon: *iCaps: An Interpretable Classifier via Disentangled Capsule Networks*, In *arXiv preprint arXiv:2008.08756*. (2020).

[62] D. Karmon, D. Zoran und Y. Goldberg: *Lavan: Localized and visible adversarial noise*, In *International Conference on Machine Learning*. PMLR (2018) Seiten 2507–2515.

[63] A. Khan, A. Sohail, U. Zahoora und A.S. Qureshi: *A survey of the recent architectures of deep convolutional neural networks*, In *Artificial intelligence review*. (2020).

[64] H. Kim, W. Lee und J. Lee, *arXiv preprint arXiv:2010.01799* (2020).

[65] J. Kim und J.F. Canny: *Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention.*, In *International Conference on Computer Vision (ICCV)*. (2017) Seiten 2961–2969.

[66] J. Kim, A. Rohrbach, T. Darrell, J. Canny, Z. Akata et al. : *Textual explanations for self-driving vehicles*, In *ECCV*. Springer (2018) Seiten 577–593.

[67] P.W. Koh und P. Liang: *Understanding black-box predictions via influence functions*, In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org (2017) Seiten 1885–1894.

[68] A. Krizhevsky, I. Sutskever und G.E. Hinton: *Imagenet classification with deep convolutional neural networks*, In *Advances in neural information processing systems*. (2012).

[69] Y. LeCun, Y. Bengio und G. Hinton: *Deep learning*, In *nature*. (2015).

[70] Y. LeCun, L. Bottou, Y. Bengio und P. Haffner: *Gradient-based learning applied to document recognition*, In *Proceedings of the IEEE*. (1998).

[71] Y. LeCun, J.S. Denker und S.A. Solla: *Optimal brain damage*, In *NIPS*. (1990) Seiten 598–605.

[72] J. Lee, T. Won, T.K. Lee, H. Lee, G. Gu und K. Hong, *arXiv preprint arXiv:2001.06268* (2020).

[73] K. Lee, K. Lee, H. Lee und J. Shin, *Advances in neural information processing systems* **31** (2018).

[74] S. Lee, H. Lee und S. Yoon: *Adversarial vertex mixup: Toward better adversarially robust generalization*, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. (2020) Seiten 272–281.

[75] B. Li, C. Chen, W. Wang und L. Carin, *Advances in neural information processing systems* **32** (2019).

[76] Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang und A. Yuille: *Learning transferable adversarial examples via ghost networks*, In *AAAI*. (2020).

[77] H. Liu, K. Simonyan, O. Vinyals, C. Fernando und K. Kavukcuoglu: *Hierarchical Representations for Efficient Architecture Search*, In *ICLR*. (2018).

[78] H. Liu, K. Simonyan und Y. Yang: *DARTS: Differentiable Architecture Search*, In *ICLR*. (2019).

[79] L.T. Liu, S. Dean, E. Rolf, M. Simchowitz und M. Hardt: *Delayed impact of fair machine learning*, In *ICML*. (2018).

[80] Y. Liu, X. Chen, C. Liu und D. Song, *arXiv preprint arXiv:1611.02770* (2016).

[81] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin und B. Guo: *Swin transformer: Hierarchical vision transformer using shifted windows*, In *arXiv:2103.14030*. (2021).

[82] Z. Liu, H. Mao, C.Y. Wu, C. Feichtenhofer, T. Darrell und S. Xie: *A ConvNet for the 2020s*, In *arXiv preprint arXiv:2201.03545*. (2022).

[83] J. Long, E. Shelhamer und T. Darrell: *Fully convolutional networks for semantic segmentation*, In *Proceedings of the IEEE International Conference on Computer Vision*. (2015) Seiten 3431–3440.

[84] D.G. Lowe: *Object recognition from local scale-invariant features*, In *Proceedings of the seventh IEEE international conference on computer vision*, Band 2. Ieee (1999) Seiten 1150–1157.

[85] A. Madry, A. Makelov, L. Schmidt, D. Tsipras und A. Vladu, *arXiv preprint arXiv:1706.06083* (2017).

[86] J. Mohapatra, C.Y. Ko, T.W. Weng, P.Y. Chen, S. Liu und L. Daniel, *Advances in Neural Information Processing Systems* **33** (2020), 4501.

[87] P. Molchanov, S. Tyree, T. Karras, T. Aila und J. Kautz: *Pruning Convolutional Neural Networks for Resource Efficient Inference*, In *ICLR*. (2017).

[88] I. Paik, T. Kwak und I. Kim: *Capsule networks need an improved routing algorithm*, In *Asian Conference on Machine Learning*. PMLR (2019) Seiten 489–502.

[89] T. Pang, C. Du, Y. Dong und J. Zhu, *Advances in Neural Information Processing Systems* **31** (2018).

[90] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik und A. Swami: *The limitations of deep learning in adversarial settings*, In *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE (2016) Seiten 372–387.

[91] S. Paul und P.Y. Chen, *arXiv:2105.07581* (2021).

[92] S.S.R. Phaye, A. Sikka, A. Dhall und D.R. Bathula: *Multi-level dense capsule networks*, In *Asian Conference on Computer Vision*. Springer (2018) Seiten 577–592.

[93] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark et al. : *Learning transferable visual models from natural language supervision*, In *International Conference on Machine Learning*. PMLR (2021) Seiten 8748–8763.

[94] A. Raghunathan, J. Steinhardt und P. Liang, *arXiv preprint arXiv:1801.09344* (2018).

[95] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne und R. Rodrigo: *DeepCaps: Going Deeper with Capsule Networks*, In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (2019) Seiten 10 725–10 733.

[96] F.D.S. Ribeiro, G. Leontidis und S.D. Kollias: *Capsule Routing via Variational Bayes*, In *AAAI*. (2019) Seiten 3749–3756.

[97] M.T. Ribeiro, S. Singh und C. Guestrin: *Why should i trust you?: Explaining the predictions of any classifier*, In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM (2016) Seiten 1135–1144.

[98] L. Rice, E. Wong und Z. Kolter: *Overfitting in adversarially robust deep learning*, In *International Conference on Machine Learning*. PMLR (2020) Seiten 8093–8104.

[99] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta und Y. Bengio: *FitNets: Hints for Thin Deep Nets*, In *ICLR*. (2015).

[100] O. Ronneberger, P. Fischer und T. Brox: *U-net: Convolutional networks for biomedical image segmentation*, In *International Conference on Medical image computing and computer-assisted intervention*. Springer (2015) Seiten 234–241.

[101] K. Roth, Y. Kilcher und T. Hofmann: *The odds are odd: A statistical test for detecting adversarial examples*, In *International Conference on Machine Learning*. PMLR (2019) Seiten 5498–5507.

[102] S. Sabour, N. Frosst und G.E. Hinton: *Dynamic routing between capsules*, In *Advances in neural information processing systems (NeurIPS)*. (2017) Seiten 3856–3866.

[103] H. Salman, J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck und G. Yang, *Advances in Neural Information Processing Systems* **32** (2019).

[104] H. Salman, M. Sun, G. Yang, A. Kapoor und J.Z. Kolter, *Advances in Neural Information Processing Systems* **33** (2020), 21945.

[105] H. Salman, G. Yang, H. Zhang, C.J. Hsieh und P. Zhang, *Advances in Neural Information Processing Systems* **32** (2019).

[106] M.H. Sarhan, A. Eslami, N. Navab und S. Albarqouni. In *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data*. Springer (2019), Seiten 37–44.

[107] P. Schwab und W. Karlen: *CXPlain: Causal explanations for model interpretation under uncertainty*, In *Advances in Neural Information Processing Systems*. (2019) Seiten 10 220–10 230.

[108] A. Selbst und J. Powles: *"Meaningful Information" and the Right to Explanation*, In *Conference on Fairness, Accountability and Transparency*. PMLR (2018) Seiten 48–48.

[109] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra et al. : *Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.*, In *International Conference on Computer Vision (ICCV)*. (2017) Seiten 618–626.

[110] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L.S. Davis, G. Taylor und T. Goldstein, *arXiv preprint arXiv:1904.12843* (2019).

[111] R. Shao, Z. Shi, J. Yi, P.Y. Chen und C.J. Hsieh, *arXiv:2103.15670* (2021).

[112] M. Sharif, S. Bhagavatula, L. Bauer und M.K. Reiter: *Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition*, In *Proceedings of the 2016 acm sigsac conference on computer and communications security*. (2016) Seiten 1528–1540.

[113] A. Shrikumar, P. Greenside und A. Kundaje: *Learning important features through propagating activation differences*, In *ICML-Volume 70*. JMLR. org (2017) Seiten 3145–3153.

[114] K. Simonyan, A. Vedaldi und A. Zisserman: *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*, In *ICLR*. (2013).

[115] K. Simonyan und A. Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition*, In *ICLR*. (2014).

[116] D. Smilkov, N. Thorat, B. Kim, F. Viégas und M. Wattenberg: *Smoothgrad: removing noise by adding noise*, In *arXiv preprint arXiv:1706.03825*. (2017).

[117] J.T. Springenberg, A. Dosovitskiy, T. Brox und M. Riedmiller: *Striving for simplicity: The all convolutional net*, In *ICLR*. (2014).

[118] S. Srinivas und F. Fleuret: *Full-gradient representation for neural network visualization*, In *Advances in Neural Information Processing Systems*. (2019) Seiten 4126–4135.

[119] G. Sriramanan, S. Addepalli, A. Baburaj et al. , *Advances in Neural Information Processing Systems* **34** (2021).

[120] M. Sundararajan, A. Taly und Q. Yan: *Axiomatic attribution for deep networks*, In *ICML-Volume 70*. JMLR. org (2017) Seiten 3319–3328.

[121] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens und Z. Wojna: *Rethinking the inception architecture for computer vision*, In *Proceedings of the IEEE International Conference on Computer Vision*. (2016) Seiten 2818–2826.

[122] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow und R. Fergus: *Intriguing properties of neural networks*, In *ICLR*. (2013).

[123] I.O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit et al. : *Mlp-mixer: An all-mlp architecture for vision*, In *Advances in Neural Information Processing Systems*, Band 34. (2021).

[124] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles und H. Jégou: *Training data-efficient image transformers & distillation through attention*, In *International Conference on Machine Learning*. PMLR (2021) Seiten 10 347–10 357.

[125] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh und P. McDaniel, *arXiv preprint arXiv:1705.07204* (2017).

[126] Y.H.H. Tsai, N. Srivastava, H. Goh und R. Salakhutdinov: *Capsules with Inverted Dot-Product Attention Routing*, In *International Conference on Learning Representations (ICLR)*. (2020).

[127] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser und I. Polosukhin: *Attention is all you need*, In *Advances in neural information processing systems*. (2017) Seiten 5998–6008.

[128] B. Vivek und R.V. Babu: *Single-step adversarial training with dropout scheduling*, In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE (2020) Seiten 947–956.

[129] W. Wang, H. Bao, L. Dong und F. Wei: *VLMo: Unified Vision-Language Pre-Training with Mixture-of-Modality-Experts*, In *arXiv preprint arXiv:2111.02358*. (2021).

[130] X. Wang, J. Ren, S. Lin, X. Zhu, Y. Wang und Q. Zhang: *A unified approach to interpreting and boosting adversarial transferability*, In *ICLR*. (2021).

[131] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou und Q. Gu, *arXiv preprint arXiv:2112.08304* (2021).

[132] E. Wong, L. Rice und J.Z. Kolter, *arXiv preprint arXiv:2001.03994* (2020).

[133] E. Wong, F. Schmidt und Z. Kolter: *Wasserstein adversarial examples via projected sinkhorn iterations*, In *International Conference on Machine Learning*. PMLR (2019) Seiten 6808–6817.

[134] D. Wu, Y. Wang, S.T. Xia, J. Bailey und X. Ma: *Skip connections matter: On the transferability of adversarial examples generated with resnets*, In *ICLR*. (2020).

[135] C. Xiao, J.Y. Zhu, B. Li, W. He, M. Liu und D. Song, *arXiv preprint arXiv:1801.02612* (2018).

[136] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár und R. Girshick: *Early convolutions help transformers see better*, In *arXiv:2106.14881*. (2021).

[137] C. Xie, Y. Wu, L.v.d. Maaten, A.L. Yuille und K. He: *Feature denoising for improving adversarial robustness*, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. (2019) Seiten 501–509.

[138] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren und A.L. Yuille: *Improving transferability of adversarial examples with input diversity*, In *CVPR*. (2019).

[139] W. Xu, D. Evans und Y. Qi, *arXiv preprint arXiv:1704.01155* (2017).

[140] Y. Yang, V. Tresp, M. Wunderle und P.A. Fasching: *Explaining therapy predictions with layer-wise relevance propagation in neural networks*, In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE (2018) Seiten 152–162.

[141] D. Zhang, T. Zhang, Y. Lu, Z. Zhu und B. Dong, *NeurIPS* (2019).

[142] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui und M. Jordan: *Theoretically principled trade-off between robustness and accuracy*, In *International Conference on Machine Learning*. PMLR (2019) Seiten 7472–7482.

[143] X. Zhang, X. Zhou, M. Lin und J. Sun: *Shufflenet: An extremely efficient convolutional neural network for mobile devices*, In *Proceedings of the IEEE conference on computer vision and pattern recognition*. (2018) Seiten 6848–6856.

[144] H. Zhao, J. Shi, X. Qi, X. Wang und J. Jia: *Pyramid scene parsing network*, In *CVPR*. (2017).

[145] Z. Zheng und P. Hong, *Advances in Neural Information Processing Systems* **31** (2018).

[146] X. Zhu, C. Xu und D. Tao: *Where and what? examining interpretable disentangled representations*, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. (2021) Seiten 5861–5870.

[147] L.M. Zintgraf, T.S. Cohen, T. Adel und M. Welling: *Visualizing deep neural network decisions: Prediction difference analysis*, In *ICLR*. (2017).

[148] B. Zoph und Q.V. Le: *Neural Architecture Search with Reinforcement Learning*, In *ICLR*. (2017).

[149] J. Zou, Z. Pan, J. Qiu, X. Liu, T. Rui und W. Li: *Improving the transferability of adversarial examples with resized-diverse-inputs, diversity-ensemble and region fitting*, In *ECCV*. (2020).

Chapter 2

Understanding Decisions of CNNs via Contrastive Backpropagation

Understanding Individual Decisions of CNNs via Contrastive Backpropagation

Jindong Gu[1,2(✉)], Yinchong Yang[2(✉)], and Volker Tresp[1,2(✉)]

[1] The University of Munich, Munich, Germany
[2] Siemens AG, Corporate Technology, Munich, Germany
{jindong.gu,yinchong.yang,volker.tresp}@siemens.com

Abstract. A number of backpropagation-based approaches such as DeConvNets, vanilla Gradient Visualization and Guided Backpropagation have been proposed to better understand individual decisions of deep convolutional neural networks. The saliency maps produced by them are proven to be non-discriminative. Recently, the Layer-wise Relevance Propagation (LRP) approach was proposed to explain the classification decisions of rectifier neural networks. In this work, we evaluate the discriminativeness of the generated explanations and analyze the theoretical foundation of LRP, i.e. Deep Taylor Decomposition. The experiments and analysis conclude that the explanations generated by LRP are not class-discriminative. Based on LRP, we propose Contrastive Layer-wise Relevance Propagation (CLRP), which is capable of producing instance-specific, class-discriminative, pixel-wise explanations. In the experiments, we use the CLRP to explain the decisions and understand the difference between neurons in individual classification decisions. We also evaluate the explanations quantitatively with a Pointing Game and an ablation study. Both qualitative and quantitative evaluations show that the CLRP generates better explanations than the LRP.

Keywords: Explainable deep learning · LRP · Discriminative saliency maps

1 Introduction

Deep convolutional neural networks (DCNNs) achieve start-of-the-art performance on many tasks, such as visual object recognition [10,26,29], and object detection [7,18]. However, since they lack transparency, they are considered as "black box" solutions. Recently, research on explainable deep learning has received increased attention: Many approaches have been proposed to crack the "black box". Some of them aim to interpret the components of a deep-architecture model and understand the image representations extracted from

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-20893-6_8) contains supplementary material, which is available to authorized users.

C. V. Jawahar et al. (Eds.): ACCV 2018, LNCS 11363, pp. 119–134, 2019.
https://doi.org/10.1007/978-3-030-20893-6_8

deep convolutional architectures [5,12,14]. Examples are Activation Maximization [6,25], DeConvNets Visualization [32]. Others focus on explaining the individual classification decisions. Examples are Prediction Difference Analysis [21,24], Guided Backpropagation [25,27], Layer-wise Relevance Propagation (LRP) [3,15], Class Activation Mapping [23,36] and Local Interpretable Model-agnostic Explanations [19,20].

More concretely, the models in [17,36] were originally proposed to detect object only using category labels. They work by producing saliency maps of objects corresponding to the category labels. Their produced saliency maps can also explain the classification decisions to some degree. However, the approaches can only work on the model with a specific architecture. For instance, they might require a fully convolutional layer followed by a max-pooling layer, a global average pooling layer or an aggregation layer, before a final softmax output layer. The requirement is not held in most off-the-shelf models e.g., in [10,26]. The perturbation methods [19–21] require no specific architecture. For a single input image, however, they require many instances of forward inference to find the corresponding classification explanation, which is computationally expensive.

The backpropagation-based approaches [3,25,27] propagate a signal from the output neuron backward through the layers to the input space in a single pass, which is computationally efficient compared to the perturbation methods. They can also be applied to the off-the-shelf models. In this paper, we focus on the backpropagation approaches. The outputs of the backpropagation approaches are instance-specific because these approaches leverage the instance-specific structure information (ISSInfo). The ISSInfo, equivalent to *bottleneck* information in [13], consist of selected information extracted by the forward inference, i.e., the Pooling switches and ReLU masks. With the ISSInfo, the backpropagation approaches can generate instance-specific explanations. A note on terminology: although the terms "sensitivity map", "saliency map", "pixel attribution map" and "explanation heatmap" may have different meanings in different contexts, in this paper, we do not distinguish them and use the term "saliency map" and "explanation" interchangeably.

The primal backpropagation-based approaches, e.g., the vanilla Gradient Visualization [25] and the Guided Backpropagation [27] are proven to be inappropriate to study the neurons of networks because they produce non-discriminative saliency maps [13]. The saliency maps generated by them mainly depend on ISSInfo instead of the neuron-specific information. In other words, the generated saliency maps are not class-discriminative with respect to class-specific neurons in output layer. The saliency maps are selective of any recognizable foreground object in the image [13]. Furthermore, the approaches cannot be applied to understand neurons in intermediate layers of DCNNs, either. In [8,32], the differences between neurons of an intermediate layer are demonstrated by a large dataset. The neurons are often activated by certain specific patterns. However, the difference between single neurons in an individual classification decision has not been explored yet. In this paper, we will also shed new light on this topic.

The recently proposed Layer-wise Relevance Propagation (LRP) approach is proven to outperform the gradient-based approaches [15]. Apart from explaining image classifications [11,15], the LRP is also applied to explain the classifications and predictions in other tasks [2,28]. However, the explanations generated by the approach has not been fully verified. We summarise our three-fold contributions as follows:

1. We first evaluate the explanations generated by LRP for individual classification decisions. Then, we analyze the theoretical foundation of LRP, i.e., Deep Taylor Decomposition and shed new insight on LRP.
2. We propose Contrastive Layer-wise Relevance Propagation (CLRP). To generate class-discriminative explanations, we propose two ways to model the contrastive signal (i.e., an opposite visual concept). For individual classification decisions, we illustrate explanations of the decisions and the difference between neuron activations using the proposed approach.
3. We build a GPU implementation of LRP and CLRP using Pytorch Framework, which alleviates the inefficiency problem addressed in [24,34].

Related work is reviewed in the next section. Section 3 analyzes LRP theoretically and experimentally. In Sect. 4, the proposed approach CLRP is introduced. Section 5 shows experimental results to evaluate the CLRP qualitatively and quantitatively on two tasks, namely, explaining the image classification decisions and understanding the difference of neuron activations in single forward inference. The last section contains conclusions and discusses future work.

2 Related Work

The DeConvNets were originally proposed for unsupervised feature learning tasks [33]. Later they were applied to visualize units in convolutional networks [32]. The DeConvNets maps the feature activity to input space using ISSInfo and the weight parameters of the forward pass. [25] proposed identifying the vanilla gradients of the output with respect to input variables are their relevance. The work also showed its relation to the DeConvNets. They use the ISSInfo in the same way except for the handling of rectified linear units (ReLUs) activation function. The Guided Backpropagation [27] combine the two approaches to visualize the units in higher layers.

The paper [3] propose LRP to generate the explanations for classification decisions. The LRP propagates the class-specific score layer by layer until to input space. The different propagation rules are applied according to the domain of the activation values. [15] proved that the Taylor Expansions of the function at the different points result in the different propagation rules. Recently, one of the propagation rules in LRP, z-rule, has been proven to be equivalent to the vanilla gradients (saliency map in [25]) multiplied elementwise with the input [9]. The vanilla Gradient Visualization and the Guided Backpropagation are shown to be not class-discriminative in [13]. This paper rethinks the LRP and evaluates the explanations generated by the approach.

Existing work that is based on discriminative and pixel-wise explanations are [4,23,34]. The work Guided-CAM [23] combines the low-resolution map of CAM and the pixel-wise map of Guided Backpropagation to generate a pixel-wise and class-discriminative explanation. To localize the most relevant neurons in the network, a biologically inspired attention model is proposed in [31]. The work uses a top-down (from the output layer to the intermediate layers) Winner-Take-All process to generate binary attention maps. The work [34] formulate the top-down attention of a CNN classifier as a probabilistic Winner-Take-All process. The work also uses a contrastive top-down attention formulation to enhance the discriminativeness of the attention maps. Based on their work and the LRP, we propose Contrastive Layer-wise Relevance Propagation (CLRP) to produce class-discriminative and pixel-wise explanations. Another publication related to our approach is [4], which is able to produce class-discriminative attention maps. While the work [4] requires modifying the traditional CNNs by adding extra feedback layers and optimizing the layers during the backpropagation, our proposed methods can be applied to all exiting CNNs without any modification and further optimization.

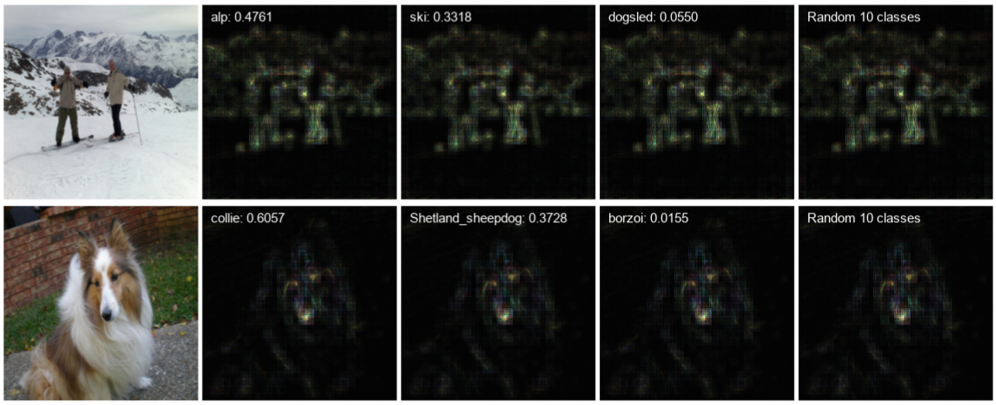
3 Rethinking Layer-Wise Relevance Propagation

Each neuron in DCNNs represents a nonliear function $X_i^{L+1} = \phi(\boldsymbol{X}^L \boldsymbol{W}_i^L + \boldsymbol{b}_i^L)$, where ϕ is an activation function and $\boldsymbol{b}_i^L$ is a bias for the neuron X_i^{L+1}. The inputs of the nonliear function corresponding to a neuron are the activation values of the previous layer $\boldsymbol{X}_i$ or the raw input of the network. The output of the function are the activation values of the neuron X_i^{L+1}. The whole network are composed of the nested nonlinear functions.

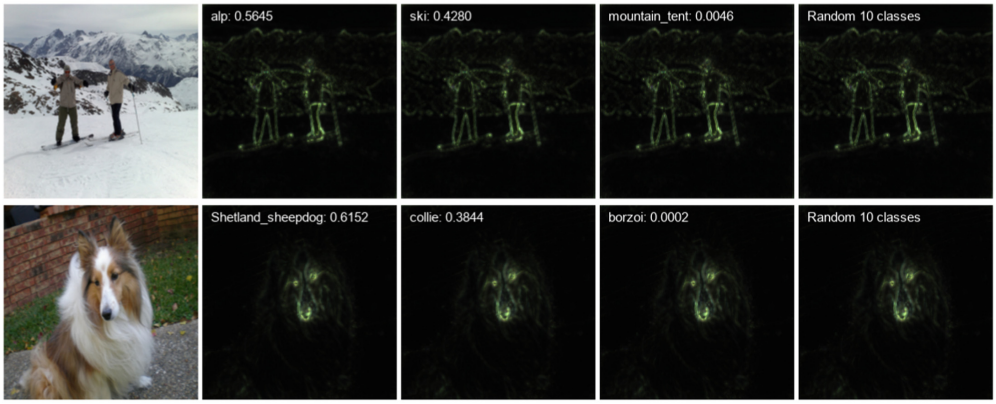
To identify the relevance of each input variables, the LRP propagates the activation value from a single class-specific neuron back into the input space, layer by layer. The logit before softmax normalization is taken, as explained in [3,25]. In each layer of the backward pass, given the relevance score $\boldsymbol{R}^{L+1}$ of the neurons $\boldsymbol{X}^{L+1}$, the relevance R_i^L of the neuron X_i^L are computed by redistributing the relevance score using local redistribution rules. The most often used rules are the z^+-rule and the z^β-rule, which are defined as follows:

$$
\begin{aligned}
z^+\text{-rule: } R_i^L &= \sum_j \frac{x_i w_{ij}^+}{\sum_{i'} x_{i'} w_{i'j}^+} R_j^{L+1} \\
z^\beta\text{-rule: } R_i^L &= \sum_j \frac{x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_{i'} x_{i'} w_{i'j} - l_{i'} w_{i'j}^+ - h_{i'} w_{i'j}^-} R_j^{L+1}
\end{aligned}
\tag{1}
$$

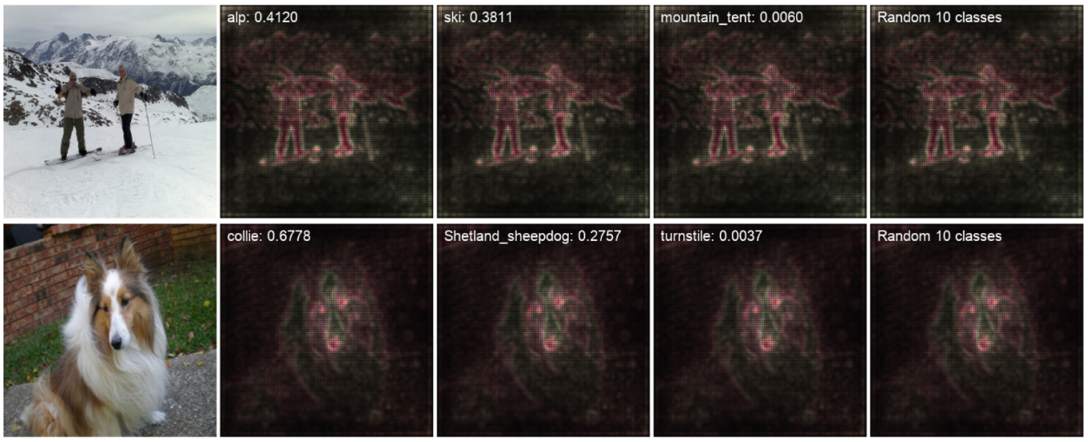
where w_{ij} connecting X_i^L and X_j^{L+1} is a parameter in L-th layer, $w_{ij}^+ = w_{ij} * 1_{w_{ij}>0}$ and $w_{ij}^- = w_{ij} * 1_{w_{ij}<0}$, and the interval $[l, h]$ is the domain of the activation value x_i.



(a) The explanations generated by LRP on AlexNet.



(b) The explanations generated by LRP on VGG16 Network.



(c) The explanations generated by LRP on GoogLeNet.

Fig. 1. The images from validation datasets of ImageNet are classified using the off-the-shelf models pre-trained on the ImageNet. The classifications of the images are explained by the LRP approach. For each image, we generate four explanations that correspond to the top-3 predicted classes and a randomly chosen multiple-classes.

3.1 Evaluation of the Explanations Generated by the LRP

The explanations generated by LRP are known to be instance-specific. However, the discriminativeness of the explanations has not been evaluated yet. Ideally, the visualized objects in the explanation should correspond to the class the class-specific neuron represents. We evaluate the explanations generated by LRP on the off-the-shelf models from *torchvision*, specifically, AlexNet [10], VGG16 [26] and GoogLeNet [29] pre-trained on the ImageNet dataset [22].

The experiment settings are similar to [15]. The z^{β}-rule is applied to the first convolution layer. For all higher convolutional layers and fully-connected layers, the z^{+}-rule is applied. In the MaxPooling layers, the relevance is only redistributed to the neuron with the maximal value inside the pooling region, while it is redistributed evenly to the corresponding neurons in the Average Pooling layers. The biases and normalization layers are bypassed in the relevance propagation pass.

The results are shown in Fig. 1. For each test image, we create four saliency maps as explanations. The first three explanation maps are generated for top-3 predictions, respectively. The fourth one is created for randomly chosen 10 classes from the top-100 predicted classes (which ensure that the score to be propagated is positive). The white text in each explanation map indicates the class the output neuron represents and the corresponding classification probability. The explanations generated by AlexNet are blurry due to incomplete learning (due to the limited expressive power). The explanations of VGG16 classifications are sharper than the ones created on GoogLenet. The reason is that VGG16 contains only MaxPooling layers and GoogLenet, by contrast, contains a few average pooling layers.

The generated explanations are instance-specific, but not class-discriminative. In other words, they are independent of class information. The explanations for different target classes, even randomly chosen classes, are almost identical. The conclusion is consistent with the one summarised in the paper [1,5], namely, almost all information about input image is contained in the pattern of non-zero pattern activations, not their precise values. The high similarity of those explanations resulted from the leverage of the same ISSInfo (see Sect. 3.2). In summary, the explanations are not class-discriminative. The generated maps recognize the same foreground objects instead of a class-discriminative one.

3.2 Theoretical Foundation: Deep Taylor Decomposition

Motivated by the divide-and-conquer paradigm, Deep Taylor Decomposition decomposes a deep neural network (i.e. the nested nonliear functions) iteratively [15]. The propagation rules of LRP are derivated from Deep Taylor Decomposition of rectifier neuron network. The function represented by a single neuron is $X_j^{L+1} = max(0, \boldsymbol{X}^L \boldsymbol{W}_j^L + b_j^{L+1})$. The relevance R_j^{L+1} of the neurons X_j^{L+1} is given. The Deep Taylor Decomposition assumes $R_j^{L+1} = max(0, \boldsymbol{X}^L \boldsymbol{W}_j^L + b_j^{L+1})$. The function is expanded with Taylor Series at a point $\boldsymbol{X}_i^r$ subjective to

$max(0, \boldsymbol{X}^{\boldsymbol{r}\,L}\boldsymbol{W}_j^L + b_j^{L+1}) = 0$. The LRP propagation rules are resulted from the first degree terms of the expansion.

One may hypothesize that the non-discriminativeness of LRP is caused by the first-order approximation error in Deep Taylor Decomposition. We proved that, under the given assumption, the same propagation rules are derived, even though all higher-order terms are taken into consideration (see the proof in the supplementary material). Furthermore, we found that the theoretical foundation provided by the Deep Taylor Decomposition is inappropriate. The assumption $R_j^{L+1} = max(0, \boldsymbol{X}^L\boldsymbol{W}_j^L + b_j^{L+1}) = X_j^{L+1}$ is not held at all the layers except for the last layer. The assumption indicates that the relevance value is equal to the activation value for all the neurons, which, we argue, is not true.

In our opinion, the explanations generated by the LRP result from the ISS-Info (ReLU masks and Pooling Switches). The activation values of neurons are required to create explanations using LRP. In the forward pass, the network output a vector $(y_1, y_2, \cdots, y_m)$. In the backward pass, the activation value of the class y_1 is layer-wise backpropagated into input space. In fully connected layers, only the activated neurons can receive the relevance according to any LRP propagation rule. In the Maxpooling layers, the backpropagation conducts an unpooling process, where only the neuron with maximal activations inside the corresponding pooling region can receive relevance. In the convolutional layer, only specific part of neurons R_{conv1} in feature map have non-zero relevance in the backward pass. The part of input pixels P_{input} live in the convolutional regions of those neurons (R_{conv1}). Only the pixels P_{input} will receive the propagated relevance. The pattern of the P_{input} is the explanation generated by LRP.

The backward pass for the class y_2 is similar to that of y_1. The neurons that receive non-zero relevance are the same as in case of y_1, even though their absolute values may be slightly different. Regardless of the class chosen for the backpropagation, the neurons of each layer that receive non-zero relevance stay always the same. In other words, the explanations generated by LRP are independent of the class category information, i.e., not class-discriminative.

In summary, in deep convolutional rectifier neuron network, the ReLU masks and Pooling Switches decide the pattern visualized in the explanation, which is independent of class information. That is the reason why the explanations generated by LRP on DCNNs are not class-discriminative. The analysis also explains the non-discriminative explanations generated by other backpropagation approaches, such as the DeConvNets Visualization [32], The vanilla Gradient Visualization [25] and the Guided Backpropagation [27].

4 Contrastive Layer-Wise Relevance Propagation

Before introducing our CLRP, we first discuss the conservative property in the LRP. In a DNN, given the input $\boldsymbol{X} = \{x_1, x_2, x_3, \cdots, x_n\}$, the output $\boldsymbol{Y} = \{y_1, y_2, y_3, \cdots, y_m\}$, the score S_{y_j} (activation value) of the neuron y_j before softmax layer, the LRP generate an explanation for the class y_j by redistributing the score S_{y_j} layer-wise back to the input space. The assigned relevance values

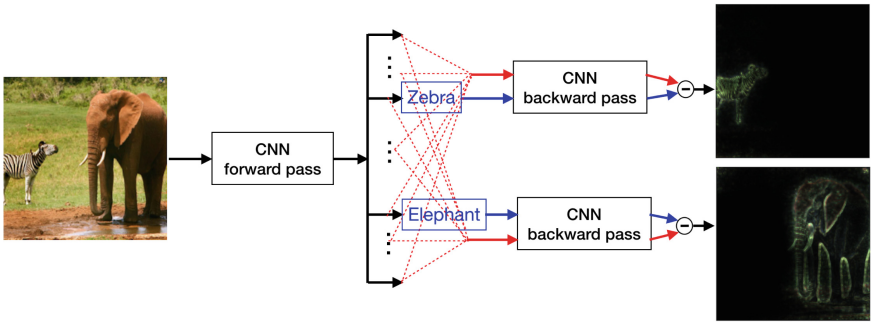


Fig. 2. The figure shows an overview of our CLRP. For each predicted class, the approach generates a class-discriminative explanation by comparing two signals. The blue line means the signal that the predicted class represents. The red line models a dual concept opposite to the predicted class. The final explanation is the difference between the two saliency maps that the two signal generate. (Color figure online)

of the input neurons are $\boldsymbol{R} = \{r_1, r_2, r_3, \cdots, r_n\}$. The conservative property is defined as follows:

Definition 1. *The generated saliency map is conservative if the sum of assigned relevance values of the input neurons is equal to the score of the class-specific neuron,* $\sum_{i=1}^{n} r_i = S_{y_j}$.

In this section, we consider redistributing the same score from different class-specific neurons respectively. The assigned relevance $\boldsymbol{R}$ are different due to different weight connections. However, the non-zero patterns of those relevance vectors are almost identical, which is why LRP generate almost the same explanations for different classes. The sum of each relevance vector is equal to the redistributed score according to the conservative property. The input variables that are discriminative to each target class are a subset of input neurons, i.e., $\boldsymbol{X}_{dis} \subset \boldsymbol{X}$. The challenge of producing the explanation is to identify the discriminative pixels $\boldsymbol{X}_{dis}$ for the corresponding class.

In the explanations of image classification, the pixels on salient edges always receive higher relevance value than other pixels including all or part of $\boldsymbol{X}_{dis}$. Those pixels with high relevance values are not necessary discriminative to the corresponding target class. We observe that $\boldsymbol{X}_{dis}$ receive higher relevance values than that of the same pixels in explanations for other classes. In other words, we can identify $\boldsymbol{X}_{dis}$ by comparing two explanations of two classes. One of the classes is the target class to be explained. The other class is selected as an auxiliary to identify $\boldsymbol{X}_{dis}$ of the target class. To identify $\boldsymbol{X}_{dis}$ more accurately, we construct a virtual class instead of selecting another class from the output layer. We propose two ways to construct the virtual class.

The overview of the CLRP are shown in Fig. 2. We describe the CLRP formally as follows. The $j-$th class-specific neuron y_j is connected to input variables by the weights $\boldsymbol{W} = \{\boldsymbol{W}^1, \boldsymbol{W}^2, \cdots, \boldsymbol{W}^{L-1}, \boldsymbol{W}_j^L\}$ of layers between them, where $\boldsymbol{W}^L$ means the weights connecting the $(L-1)-$th layer and the $L-$th layer, and

$\boldsymbol{W}_j^L$ means the weights connecting the $(L-1)-$th layer and the $j-$th neuron in the $L-$th layer. The neuron y_j models a visual concept O. For an input example $\boldsymbol{X}$, the LRP maps the score S_{y_j} of the neuron back into the input space to get relevance vector $\boldsymbol{R} = f_{LRP}(\boldsymbol{X}, \boldsymbol{W}, S_{y_j})$.

We construct a dual virtual concept $\overline{O}$, which models the opposite visual concept to the concept O. For instance, the concept O models the **zebra**, and the constructed dual concept $\overline{O}$ models the **non-zebra**. One way to model the $\overline{O}$ is to select all classes except for the target class representing O. The concept $\overline{O}$ is represented by the selected classes with weights $\overline{\boldsymbol{W}} = \{\boldsymbol{W}^1, \boldsymbol{W}^2, \cdots, \boldsymbol{W}^{L-1}, \boldsymbol{W}^L_{\{-j\}}\}$, where $\boldsymbol{W}_{\{-j\}}$ means the weights connected to the output layer excluding the $j-$th neuron. E.g. the dashed red lines in Fig. 2 are connected to all classes except for the target class **zebra**. Next, the score S_{y_j} of target class is uniformly redistributted to other classes. Given the same input example $\boldsymbol{X}$, the LRP generates an explanation $\boldsymbol{R}_{dual} = f_{LRP}(\boldsymbol{X}, \overline{\boldsymbol{W}}, S_{y_j})$ for the dual concept. The Contrastive Layer-wise Relevance Propagation is defined as follows:

$$\boldsymbol{R}_{CLRP} = \max(\boldsymbol{0}, (\boldsymbol{R} - \boldsymbol{R}_{dual})) \tag{2}$$

where the function $\max(\boldsymbol{0}, \boldsymbol{X})$ means replacing the negative elements of $\boldsymbol{X}$ with zeros. The difference between the two saliency maps cancels the common parts. Without the dominant common parts, the non-zero elements in $\boldsymbol{R}_{CLRP}$ are the most relevant pixels $\boldsymbol{X}_{dis}$. If the neuron y_j lives in an intermediate layer of a neural network, the constructed $\boldsymbol{R}_{CLRP}$ can be used to understand the role of the neuron.

Similar to [34], the other way to model the concept $\overline{O}$ is to negate the weights $\boldsymbol{W}_j^L$. The concept $\overline{O}$ can be represented by the weights $\overline{\boldsymbol{W}} = \{\boldsymbol{W}^1, \boldsymbol{W}^2, \cdots, \boldsymbol{W}^{L-1}, -1 * \boldsymbol{W}_j^L\}$. All the weights are same as in the concept O except that the weights of the last layer $\boldsymbol{W}_j^L$ are negated. In the experiments section, we call the first modeling method CLRP1 and the second one CLRP2. The contrastive formulation in [34] can be applied to other backpropagation approaches by normalizing and subtracting two generated saliency maps. However, the normalization strongly depends on the maximal value that could be caused by a noisy pixel. Based on the conservative property of LRP, the normalization is avoided in the proposed CLRP.

5 Experiments and Analysis

In this section, we conduct experiments to evaluate our proposed approach. The first experiment aims to generate class-discriminative explanations for individual classification decisions. The second experiment evaluates the generated explanations quantitatively on the ILSVRC2012 validation dataset. The discriminativeness of the generated explanations is evaluated via a Pointing Game and an ablation study. The last experiment aims to understand the difference between neurons in a single classification forward pass.

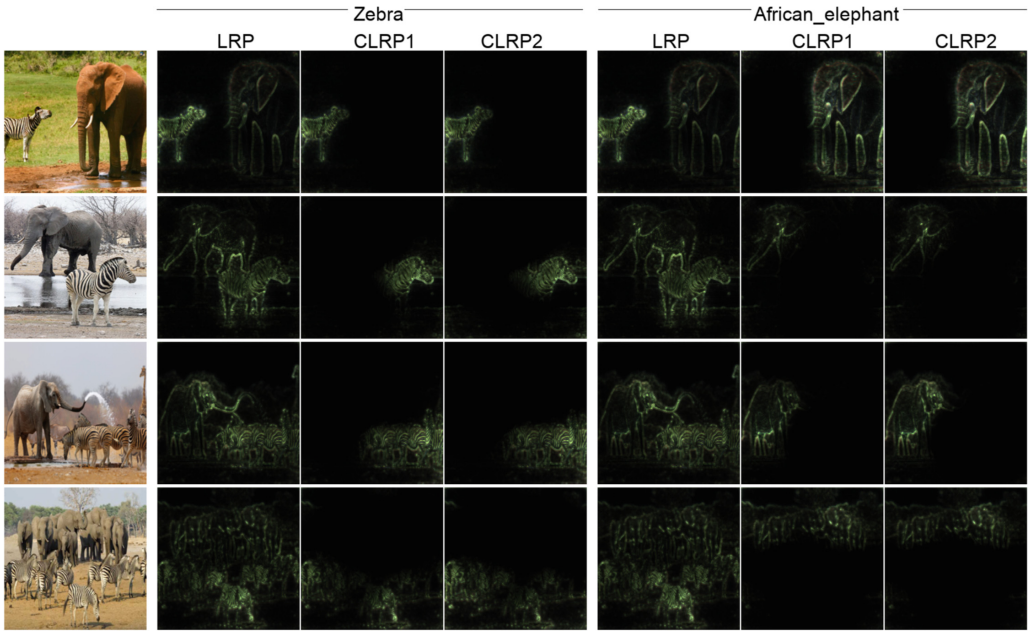


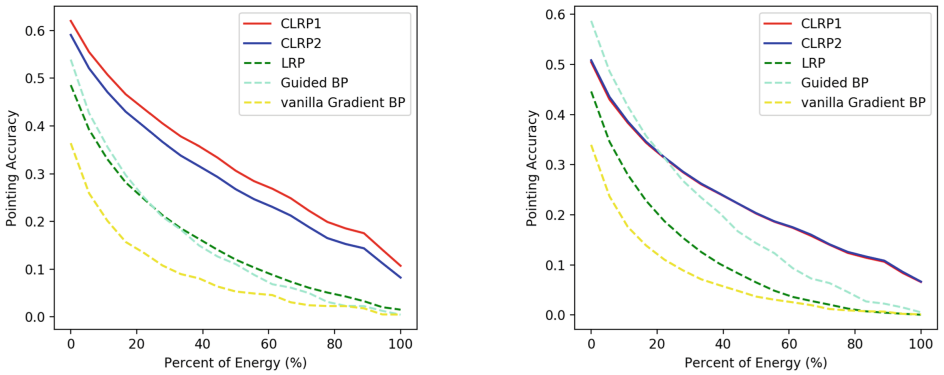
Fig. 3. The images of multiple objects are classified using VGG16 network pre-trained on ImageNet. The explanations for the two relevant classes are generated by LRP and CLRP. The CLRP generates class-discriminative explanations, while LRP generates almost same explanations.

5.1 Explaining Classification Decisions of DNNs

In this experiment, the LRP, the CLRP1 and the CLRP2 are applied to generate explanations for different classes. The experiments are conducted on a pre-trained VGG16 Network [26]. The propagation rules used in each layer are the same as in the Sect. 3.1. We classify the images of multiple objects. The explanations are generated for the two most relevant predicted classes, respectively. The Fig. 3 shows the explanations for the two classes (i.e., *Zebra* and *African_elephant*). The explanations generated by the LRP are same for the two classes. Each generated explanation visualizes both *Zebra* and *African_elephant*, which is not class-discriminative. By contrast, both CLRP1 and CLRP2 only identify the discriminative pixels related to the corresponding class. For the target class *Zebra*, only the pixels on the zebra object are visualized. Even for the complicated images where a zebra herd and an elephant herd co-exist, the CLRP methods are still able to find the class-discriminative pixels.

We evaluate the approach with a large number of images with multiple objects. The explanations generated by CLRP are always class-discriminative, but not necessarily semantically meaningful for every class. One of the reasons is that the VGG16 Network is not trained for multi-label classification. Other reasons could be the incomplete learning and bias in the training dataset [30].

The implementation of the LRP is not trivial. The one provided by their authors only supports CPU computation. For the VGG16 network, it takes the 30 s to generate one explanation on an Intel Xeon 2.90 GHz $\times$ 6 machine.



(a) Pointing Accuracy On the AlexNet (b) Pointing Accuracy On the VGG16

Fig. 4. The figure shows the localization ability of the saliency maps generated by the LRP, the CLRP1, the CLRP2, the vanilla Gradient Visualization and the Guided Backpropagation. On the pre-trained models, AlexNet and VGG16, the localization ability is evaluated at different thresholds. The x-axis corresponds to the threshold that keeps a certain percentage of energy left, and the y-axis corresponds to the pointing accuracy.

The computational expense makes the evaluation of LRP impossible on a large dataset [34]. We implement a GPU version of the LRP approach, which reduces the 30 s to 0.1824 s to generate one explanation on a single NVIDIA Telsa K80 GPU. The implementation alleviates the inefficiency problem addressed in [24,34] and makes the quantitative evaluation of LRP on a larget dataset possible.

5.2 Evaluating the Explanations

In this experiments, we quantitatively evaluate the generated explanations on the ILSVRC2012 validation dataset containing 50, 000 images. A Pointing Game and an ablation study are used to evaluate the proposed approach.

Pointing Game: To evaluate the discriminativeness of saliency maps, the paper [34] proposes a pointing game. The maximum point on the saliency map is extracted and evaluated. In case of images with a single object, a hit is counted if the maximum point lies in the bounding box of the target object, otherwise a miss is counted. The localization accuracy is measured by $Acc = \frac{\#Hits}{\#Hits+\#Misses}$. In case of ILSVRC2012 dataset, the naive pointing at the center of the image shows surprisingly high accuracy. Based on the reason, we extend the pointing game into a difficult setting. In the new setting, the first step is to preprocess the saliency map by simply thresholding so that the foreground area covers p percent energy out of the whole saliency map (where the energy is the sum of all pixel values in saliency map). A hit is counted if the remaining foreground area lies in the bounding box of the target object, otherwise a miss is counted.

The Fig. 4 show that the localization accuracy of different approaches in case of different thresholds. With more energy kept, the remained pixels are less

Table 1. Ablation study on ImagNet Validation dataset. The dropped activation values after the corresponding ablation are shown in the table.

	Random	vanilGrad [25]	GuidedBP [27]	**LRP** [3]	**CLRP1**	**CLRP2**
AlexNet	0.0766	0.1716	0.1843	0.1624	0.2093	0.2030
VGG16	0.0809	0.3760	0.4480	0.3713	0.3844	0.3913

likely to fall into the ground-truth bounding box, and the localization accuracy is low correspondingly. The CLRP1 and the CLRP2 show constantly much better pointing accuracy than that of the LRP. The positive results indicate that the pixels that the contrastive backpropagation cancels are on the cluttered background or non-target objects. The CLRP can focus on the class-discriminative part, which improves the LRP. The CLRP is also better than other primal backpropagation-based approaches. One exception is that the Guided Backpropagation shows a better localization accuracy in VGG16 network in case of high thresholds. In addition, the localization accuracy of the CLRP1 and the CLRP2 is similar in the deep VGG16 network, which indicates the equivalence of the two methods to model the opposite visual concept.

Ablation Study: In the Pointing Game above, we evaluate the discriminativeness of the explanations according to the localization ability. In this ablation study, we evaluate the discriminativeness from another perspective. We observe the changes of activation in case of ablating the found discriminative pixels. The activation value of the class-specific neuron will drop if the ablated pixels are discriminative to the corresponding class.

For an individual image classification decision, we first generate a saliency map for the ground-truth class. We identify the maximum point in the generated saliency map as the most discriminative position. Then, we ablate the pixel of the input image at the identified position with a 9×9 image patch. The pixel values of the image patch are the mean value of all the pixel values at the same position across the whole dataset. We classify the perturbated image and observe the activation value of the neuron corresponding to the ground-truth class. The dropped activation value is computed as the difference between the activations of the neuron before and after the perturbation. The dropped score is averaged on all the images in the dataset.

The experimental results of different approaches are shown in the Table 1. For the comparison, we also ablate the image with a randomly chosen position. The random ablation has hardly impact on the output. The saliency maps corresponding to all other approaches find the relevant pixel because the activations of the class-specific neurons dropped a lot after the corresponding ablation. In both networks, CLRP1 and CLRP2 show the better scores, which means the discriminativeness of explanations generated by CLRP is better than that of the LRP. Again, the Guided Backpropagation shows better score than CLRP. This ablation study only considers the discriminative of the pixel with maximal relevance value, which corresponds to a special case in the Pointing Game,

namely, only one pixel with maximal relevance is left after the thresholding. The two experiments show the consistent result that the Guided Backpropagation is better than LRP in the special case. We do not report the performance of the GoogLeNet in the experiments. Our approach shows that the zero-padding operations of convolutional layers have a big impact on the output of the GoogLeNet model in *torchvision* module of Pytorch. The impact leads to a problematic saliency map (see supplementary material).

5.3 Understanding the Difference Between Neurons

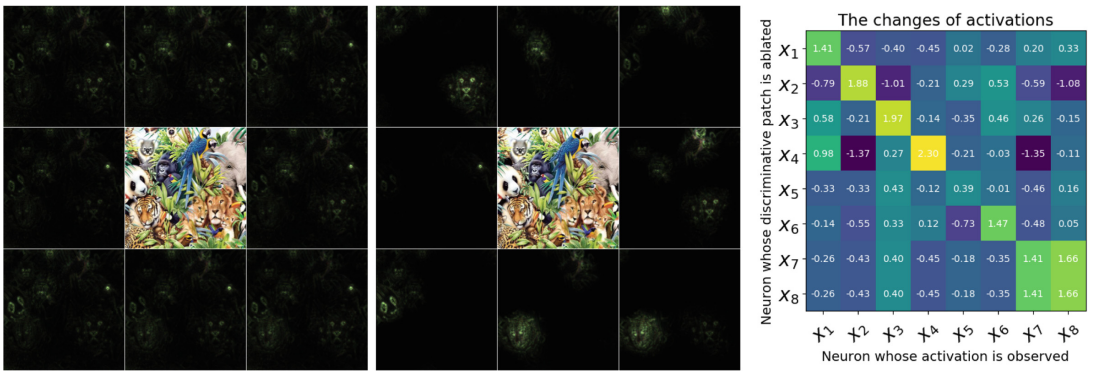
The neurons of DNNs have been studying with their activation values. The DeConvNets [32] visualize the patterns and collect the images that maximally activate the neurons, given an image set. The activation maximization method [6,16] aims to generate an image in input space that maximally activates a single neuron or a group of neurons. Furthermore, the work [8,35] understand the semantic concepts of the neurons with an annotated dataset. In this experiment, we aim to study the difference among neurons in a single classification decision.

The neurons of low layers may have different local receptive fields. The difference between them could be caused by the different input stimuli. We visualize high-level concepts learned by the neurons that have the same receptive fields, e.g., a single neuron in a fully connected layer. For a single test image, the LRP and the CLRP2 are applied to visualize the stimuli that activate a specific neuron. We do not use CLRP1 because the opposite visual concept cannot be modeled by the remaining neurons in the same layer.

In VGG16 network, we visualize 8 activated neurons x_{1-8} from the $fc1$ layer. The visualized maps are shown in Fig. 5. The image is classified as a *toyshop* by the VGG16 network. The receptive field (the input image) is shown in the center, and the 8 explanation maps are shown around it. While the LRP produces almost identical saliency map for the 8 neurons (in Fig. 5a), the CLRP2 gains a meaningful insight about their difference, which shows that different neurons focus on different parts of images. By comparison (see Fig. 5b), the neurons x_1, x_2, x_3 in the first row are activated more by the *lion*, the *gorilla*, and the *monkey* respectively. The neurons x_4, x_5 in the second row by the eye of the *elephant* and the *bird* respectively. The right-down one x_6 by the *panda*. The last two neurons x_7 and x_8 focus on the similar patterns (i.e., the *tiger*).

To our knowledge, there is no known work on the difference between neurons in an individual classification decision and also no evaluation metric. We evaluate the found difference by an ablation study. More concretely, we first find the discriminative patch for each neuron (e.g., x_{1-8}) using CLRP2. Then, we ablate the patch and observe the changes of neuron activations in the forward pass. The discriminative patch of a neuron is identified by the point with maximal value in its explanation map created by CLRP2. The 9×9 neighboring pixels around the maximum point are replaced with the values that are mean of pixel values in the same positions across the whole dataset.

The ablation study results are shown in the Fig. 5c. The positive value in the grid of the figure means the decreased activation value, and the negative



(a) Explanations by LRP (b) Explanations by CLRP (c) Ablation Study

Fig. 5. The figures show explanation maps of neurons in $fc1$ layers. The explanations generated by LRP are not discriminative. By contrast, the ones generated by CLRP explain the difference between the neurons.

ones mean the activations increase after the corresponding ablation. In case of the ablation corresponding to neuron x_i, we see that the activation of x_i is significantly dropped (could become not-activated). The maximal droped values of each row often occur on the diagonal axis. We also try with other ablation sizes and other neurons, which shows the similar results. The ablations for the last two neurons x_7 and x_8 are same because their explanation maps are similar. The changes of activations of all other neurons are also the same for the same ablation. We found that many activated neurons correspond to same explanation maps.

6 Conclusion

The explanations generated by LRP are evaluated. We find that the explanations are not class-discriminative. We discuss the theoretical foundation and provide our justification for the non-discriminativeness. To improve discriminativeness of the generated explanations, we propose the Contrastive Layer-wise Relevance Propagation. The qualitative and quantitative evaluations confirm that the CLRP is better than the LRP. We also use the CLRP to shed light on the role of neurons in DCNNs.

We propose two ways to model the opposite visual concept the class-specific neuron represents. However, there could be other more appropriate modeling methods. Even though our approach produces a pixel-wise explanation for the individual classification decisions, the explanations for similar classes are similar. The fine-grained discriminativeness are needed to explain the classifications of the intra-classes. We leave the further exploration in future work.

References

1. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 329–344. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10584-0_22
2. Arras, L., Montavon, G., Müller, K.R., Samek, W.: Explaining recurrent neural network predictions in sentiment analysis. arXiv preprint arXiv:1706.07206 (2017)
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE **10**(7), e0130140 (2015)
4. Cao, C., et al.: Look and think twice: capturing top-down visual attention with feedback convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2956–2964 (2015)
5. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4829–4837 (2016)
6. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Univ. Montreal **1341**(3), 1 (2009)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
8. Gonzalez-Garcia, A., Modolo, D., Ferrari, V.: Do semantic parts emerge in convolutional neural networks? Int. J. Comput. Vis. **126**(5), 476–494 (2018)
9. Kindermans, P.J., Schütt, K., Müller, K.R., Dähne, S.: Investigating the influence of noise and distractors on the interpretation of neural networks. arXiv preprint arXiv:1611.07270 (2016)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
11. Lapuschkin, S., Binder, A., Müller, K.R., Samek, W.: Understanding and comparing deep neural networks for age and gender classification. arXiv preprint arXiv:1708.07689 (2017)
12. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. CoRR abs/1412.0035 (2014)
13. Mahendran, A., Vedaldi, A.: Salient deconvolutional networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 120–135. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_8
14. Mahendran, A., Vedaldi, A.: Visualizing deep convolutional neural networks using natural pre-images. Int. J. Comput. Vis. **120**(3), 233–255 (2016)
15. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recogn. **65**, 211–222 (2017)
16. Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Advances in Neural Information Processing Systems, pp. 3387–3395 (2016)
17. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Is object localization for free?-weakly-supervised learning with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 685–694 (2015)
18. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)

19. Ribeiro, M.T., Singh, S., Guestrin, C.: Nothing else matters: model-agnostic explanations by identifying prediction invariance. arXiv preprint arXiv:1611.05817 (2016)
20. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. ACM (2016)
21. Robnik-Šikonja, M., Kononenko, I.: Explaining classifications for individual instances. IEEE Trans. Knowl. Data Eng. **20**(5), 589–600 (2008)
22. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. IJCV **115**, 211–252 (2015)
23. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. arXiv:1610.02391v3, vol. 7, no. 8 (2016)
24. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685 (2017)
25. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
26. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
27. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: the all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
28. Srinivasan, V., Lapuschkin, S., Hellge, C., Müller, K.R., Samek, W.: Interpretable human action recognition in compressed domain. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1692–1696. IEEE (2017)
29. Szegedy, C., et al.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9, June 2015
30. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1521–1528. IEEE (2011)
31. Tsotsos, J.K., Culhane, S.M., Wai, W.Y.K., Lai, Y., Davis, N., Nuflo, F.: Modeling visual attention via selective tuning. Artif. Intell. **78**(1–2), 507–545 (1995)
32. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
33. Zeiler, M.D., Krishnan, D., Taylor, G.W., Fergus, R.: Deconvolutional networks. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2528–2535. IEEE (2010)
34. Zhang, J., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down neural attention by excitation backprop. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 543–559. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_33
35. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Object detectors emerge in deep scene CNNs. arXiv preprint arXiv:1412.6856 (2014)
36. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2921–2929. IEEE (2016)

Chapter 3

Interpretable Graph Capsule Network

Interpretable Graph Capsule Networks for Object Recognition

Jindong Gu
University of Munich
Siemens AG, Corporate Technology
jindong.gu@outlook.com

Abstract

Capsule Networks, as alternatives to Convolutional Neural Networks, have been proposed to recognize objects from images. The current literature demonstrates many advantages of CapsNets over CNNs. However, how to create explanations for individual classifications of CapsNets has not been well explored. The widely used saliency methods are mainly proposed for explaining CNN-based classifications; they create saliency map explanations by combining activation values and the corresponding gradients, e.g., Grad-CAM. These saliency methods require a specific architecture of the underlying classifiers and cannot be trivially applied to CapsNets due to the iterative routing mechanism therein. To overcome the lack of interpretability, we can either propose new post-hoc interpretation methods for CapsNets or modifying the model to have build-in explanations. In this work, we explore the latter. Specifically, we propose interpretable Graph Capsule Networks (GraCapsNets), where we replace the routing part with a multi-head attention-based Graph Pooling approach. In the proposed model, individual classification explanations can be created effectively and efficiently. Our model also demonstrates some unexpected benefits, even though it replaces the fundamental part of CapsNets. Our GraCapsNets achieve better classification performance with fewer parameters and better adversarial robustness, when compared to CapsNets. Besides, GraCapsNets still keep other advantages of CapsNets, namely, disentangled representations and affine transformation robustness.

1 Introduction

In past years, Convolutional Neural Networks (CNNs) have become the standard model applied in object recognition. Our community has been pursuing more powerful CNN models with compact size (He et al. 2016). Besides, two weaknesses of CNNs have also been intensively investigated recently. Namely, 1) Adversarial Vulnerability (Szegedy et al. 2014): The predictions of CNNs can be misled by imperceptible perturbations of input images. 2) Lack of Interpretability (Simonyan, Vedaldi, and Zisserman 2013): The predictions of standard CNNs are based on highly entangled representations. The two weaknesses might be attributed to the fact that the representations learned by CNNs are not aligned to human perception.

Recently, Capsule Networks (CapsNets) (Sabour, Frosst, and Hinton 2017) have been proposed and received much attention since they can learn more human-aligned visual representations (Qin et al. 2020). The disentangled representations captured by CapsNets often correspond to human-understandable visual properties of input objects, e.g., rotations and translations. Recent work on CapsNets aims to propose more efficient routing algorithms (Hinton, Sabour, and Frosst 2018; Hahn, Pyeon, and Kim 2019; Zhang, Edraki, and Qi 2018; Tsai et al. 2020) and understand the contributions of the routing algorithms (Gu and Tresp 2020; Gu, Wu, and Tresp 2021).

However, how to explain individual classifications of CapsNets has been less explored. The state-of-the-art saliency methods are mainly proposed for CNNs, e.g., Grad-CAM (Selvaraju et al. 2017). They combine activation values and the received gradients in specific layers, e.g., deep convolutional layers. In CapsNets, instead of deep convolutional layers, an iterative routing mechanism is applied to extract high-level visual concepts. Hence, these saliency methods cannot be trivially applied to CapsNets. Besides, the routing mechanism makes it more challenging to identify interpretable input features relevant to a classification.

In this work, we propose interpretable Graph Capsule Networks (GraCapsNets). In CapsNets, the primary capsules represent object parts, e.g., eyes and nose of a cat. In our GraCapsNets, we explicitly model the relationship between the primary capsules (i.e., part-part relationship) with graphs. Then, the followed graph pooling operations pool relevant object parts from the graphs to make a classification vote. Since the graph pooling operation reveals which input features are pooled as relevant ones, we can easily create explanations to explain the classification decisions. Besides the interpretability, another motivation of GraCapsNets is that the explicit part-part relationship is also relevant for object recognition, e.g., spatial relationships.

The classic graph pooling algorithms are clustering-based, which requires high computational complexity. It is challenging to integrate these graph pooling algorithms into neural networks. Recent progress on graph pooling modules of Graph Neural Networks makes similar integrations possible. E.g., (Ying et al. 2018) proposed a differentiable graph pooling module, which can be integrated into various neural network architectures in an end-to-end fashion.

The capsule idea is also integrated into Graph Neural Networks for better graph classification (Verma and Zhang 2018; Xinyi and Chen 2019). They treat node feature vectors as primary capsules and aggregates information from the capsules via a routing mechanism. Different from their works, we integrate graph modeling into CapsNets for better object recognition. On the contrary, our GraCapsNets treat capsules as node feature vectors and represent them as graphs so that we can leverage graph structure information (e.g., the spatial part-part relationship between object parts).

Our main contribution of this work is to propose GraCapsNets, where we replace the fundamental routing part of CapsNets with multi-head attention-based Graph Pooling operations. On GraCapsNets, we can create explanations for individual classifications effectively and efficiently. Besides, our empirical experiments show that GraCapsNets achieve better performance with fewer parameters and also learn disentangled representations. GraCapsNets are also shown to be more robust to the primary white adversarial attacks than CNNs and various CapsNets.

2 Related Work

Routing Mechanism: The goal of routing processes in CapsNets is to identify the weights of predictions made by low-level capsules, called coupling coefficients (CCs) in (Sabour, Frosst, and Hinton 2017). Many routing mechanisms have been proposed to improve Dynamic Routing (Sabour, Frosst, and Hinton 2017); they differ from each other only in how to identify CCs.

Dynamic Routing (Sabour, Frosst, and Hinton 2017) identifies CCs with an iterative routing-by-agreement mechanism. EM Routing (Hinton, Sabour, and Frosst 2018) updates CCs iteratively using the Expectation-Maximization algorithm. (Chen and Crandall 2019) removes the computationally expensive routing iterations by predicting CCs directly. To improve the prediction of CCs further, Self-Routing (Hahn, Pyeon, and Kim 2019) predicts CCs using a subordinate routing network. However, (Gu and Tresp 2020) shows that similar performance can be achieved by simply averaging predictions of low-level capsules without learning CCs. In this work, we propose Graph Capsule Networks, where a multi-head attention-based graph pooling mechanism is used instead of routing.

Graph Pooling: Earlier works implement graph pooling with clustering-based graph coarsening algorithms, e.g., Graclus (Dhillon, Guan, and Kulis 2007), where the nodes with similar representations are clustered into one. In later works (Set2Set (Vinyals, Bengio, and Kudlur 2015) and SortPool (Zhang et al. 2018)), the graph features are also taken into consideration. However, they require the ordering of the nodes by a user-defined meaningful criterium. Recently, the seminal work (Ying et al. 2018) proposes a differentiable graph pooling module, which can be combined with various neural network architectures in an end-to-end fashion. For simplification of (Ying et al. 2018), top-K pooling (Gao and Ji 2019; Knyazev, Taylor, and Amer 2019) and self-attention pooling (Lee, Lee, and Kang 2019) have been proposed. Almost all the graph pooling strategies have been mainly used for graph classification. Based on the work (Ying et al. 2018), we propose multiple-heads attention-based graph pooling for object recognition.

Algorithm 1: Capsule Networks

Input: An image $\mathbf{X}$

Output: Class capsules $\mathbf{V} \in \mathbb{R}^{M \times D_{out}}$

1. Extract primary capsules $\mathbf{u}_i \in \mathbb{R}^{D_{in}}$ from input $\mathbf{X}$;
2. Transform each $\mathbf{u}_i$ into $\hat{\mathbf{u}}_{j|i} \in \mathbb{R}^{D_{out}}$;
3. Identify all c_{ij} with a routing process;
4. Compute $\mathbf{s}_j = \sum_{i=1}^{N} c_{ij} * \hat{\mathbf{u}}_{j|i}$;
5. Output capsules $\mathbf{v}_j = squash(\mathbf{s}_j)$

Adversarial Robustness: (Szegedy et al. 2014) shows that imperceptible image perturbations can mislead standard CNNs. Since then, many adversarial attack methods have been proposed, e.g., FGSM (Goodfellow, Shlens, and Szegedy 2015), C&W (Carlini and Wagner 2017). Meanwhile, the approaches to defend these attacks have also been widely investigated, e.g., Adversarial Training (Madry et al. 2017; Athalye, Carlini, and Wagner 2018), Certified Defenses (Wong and Kolter 2018; Cohen, Rosenfeld, and Kolter 2019). One way to tackle the adversarial vulnerability is to propose new models that learn more human perception-aligned feature representations, e.g., CapsNets (Sabour, Frosst, and Hinton 2017; Qin et al. 2020). Recent work (Hinton, Sabour, and Frosst 2018; Hahn, Pyeon, and Kim 2019) shows that CapsNets with their routing processes are more robust to white-box adversarial attacks.

Interpretability: A large number of interpretation methods have been proposed to understand individual classifications of CNNs. Especially, saliency maps created by post-hoc methods, as intuitive explanations, have received much attention. We categorize the methods into two categories. The first category is architecture-agnostic, such as, vanilla Gradients (Grad) (Simonyan, Vedaldi, and Zisserman 2013), Integrated Gradients (IG) (Sundararajan, Taly, and Yan 2017) as well as their smoothed versions (SG) (Smilkov et al. 2017). The second one requires specific layers or architecture of models, e.g., Guided Backpropagation (Springenberg et al. 2014; Gu and Tresp 2019), DeepLIFT (Shrikumar, Greenside, and Kundaje 2017), LRP (Bach et al. 2015; Gu, Yang, and Tresp 2018), Grad-CAM (Selvaraju et al. 2017). Only the architecture-agnostic methods can be trivially generalized to CapsNets due to the routing mechanism therein. In our GraCapsNets, the explanations can be created with attention in the graph pooling operations.

3 Graph Capsule Networks

We first briefly review CapsNets. As shown in Algorithm 1, CapsNets start with convolutional layers that convert the input pixel intensities $\mathbf{X}$ into primary capsules $\mathbf{u}_i$ (i.e., low-level visual entities). Each $\mathbf{u}_i$ is transformed to vote for high-level capsules $\hat{\mathbf{u}}_{j|i}$ with learned transformation matrices. Then, a routing process is used to identify the coupling coefficients c_{ij}, which describe how to weight votes from primary capsules. Finally, a squashing function is applied to the identified high-level capsules $\mathbf{s}_j$ so that the lengths of

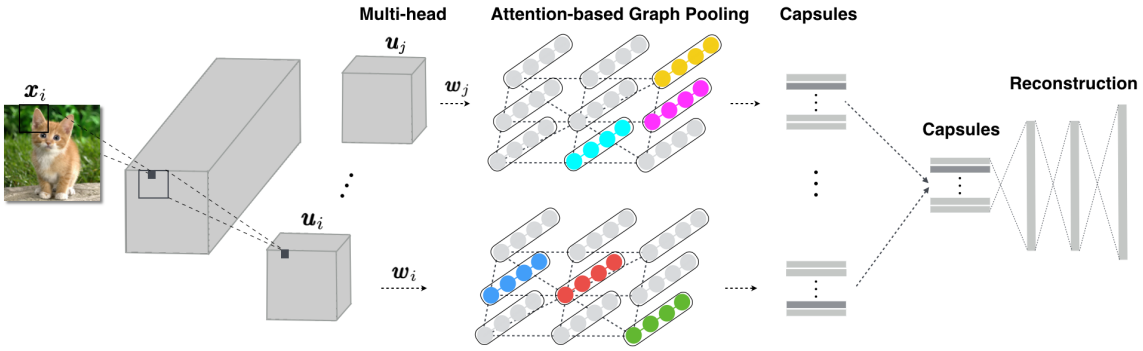


Figure 1: The illustration of GraCapsNets: The extracted primary capsules are transformed and modeled as multiple graphs. The pooling result on each graph (head) corresponds to one vote. The votes on multiple graphs (heads) are averaged to generate the final prediction.

Algorithm 2: Graph Capsule Networks

Input: An image $\mathbf{X}$

Output: Class capsules $\mathbf{V} \in \mathbb{R}^{M \times D_{out}}$

1. Extract primary capsules $\mathbf{u}_i \in \mathbb{R}^{D_{in}}$ from input $\mathbf{X}$;
2. Project each $\mathbf{u}_i$ into the feature space $\mathbf{u}'_i \in \mathbb{R}^{D_{out}}$;
3. Model all $\mathbf{u}'_i$ as multiple graphs;
4. Compute $\mathbf{s}_j \in \mathbb{R}^{D_{out}}$ with multi-head graph pooling;
5. Output capsules $\mathbf{v}_j = squash(\mathbf{s}_j)$

them correspond to the confidence of the class's existence. A reconstruction part works as regularization during training.

Different routing mechanisms differ only in the 3rd step, i.e., how to identify c_{ij}. Routing processes describe one way to aggregate information from primary capsules into high-level ones. In our GraCapsNets, we implement the information aggregation by multi-head graph pooling processes.

As shown in Algorithm 2, GraCapsNets differ from CapsNets in the steps of 2, 3, and 4. In GraCapsNet, the primary capsules $\mathbf{u}_i$ are transformed into a feature space. All transformed capsules $\mathbf{u}'_i$ are modeled as multiple graphs. Each graph corresponds to one head, the pooling result on which corresponds to one vote. The votes on multiple heads are averaged as the final prediction. The GraCapsNets is also illustrated in Figure 1.

In CapsNets, most of the parameters are from the transformation matrix $\mathbf{W}^t \in \mathbb{R}^{N \times D_{in} \times (M * D_{out})}$ where D_{in}, D_{out} are the dimensions of input primary capsules and output high-level capsules, N is the number of primary capsules, and M is the number of output classes. In GraCapsNets, the transformation matrix is $\mathbf{W}^t \in \mathbb{R}^{N \times D_{in} \times D_{out}}$ and the trainable parameters in the graph pooling layer is $\mathbf{W} \in \mathbb{R}^{D_{out} \times M}$. Hence, the parameters are reduced signigicantly.

3.1 Multiple Heads in GraCapsNets

We now introduce how to model all transformed capsules $\mathbf{u}'_i$ as multiple graphs. A graph consists of a set of nodes and a set of edges.

As shown in GraCapsNet in Figure 1, the primary capsules are reshaped from L groups of feature maps. Each group consists of C feature maps of the size $K \times K$. Correspondingly, the transformed capsules $\mathbf{u}'_i$ where $i \in \{1, 2, ...K^2\}$ form a single graph with K^2 nodes. Namely, the capsules of the same type (the ones on the same feature maps but different locations) are modeled in the same graph. Each node corresponds to one transformed capsule $\mathbf{u}'_i$, and the activation vector of $\mathbf{u}'_i$ is taken as features of the corresponding node.

The graph edge information can be represented by an adjacency matrix, in which different priors can be modeled, e.g., camera geometry (Khasanova 2019) and spatial relationships (Knyazev et al. 2019). In this work, we model the spatial relationship between primary capsules since they can be computed without supervision.

For the above graph with K^2 nodes, elements in the adjacency matrix $\mathbf{A} \in \mathbb{R}^{K^2 \times K^2}$ can be computed as

$$A_{ij} = e^{(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\sigma^2})} \tag{1}$$

where i, j are indice of nodes and $\mathbf{p}_i \in \mathbb{R}^2, \mathbf{p}_j \in \mathbb{R}^2$ are coordinates of the nodes, i.e. from $(1, 1)$ to (K, K). Similarly, we can build l graphs (heads) in total with the same adjcency matrix. They differ from each other in node features.

3.2 Graph Pooling in GraCapsNets

Given node features $\mathbf{X}^l \in \mathbb{R}^{(K^2 \times D_{out})}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{(K^2 \times K^2)}$ in the l-th head of GraCapsNet, we now describe how to make a vote for the final prediction by a attention-based graph pooling operation. We first compute the attention of the head as

$$\mathbf{Att}^l = \text{softmax}(\mathbf{A}\mathbf{X}^l\mathbf{W}) \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{D_{out} \times M}$ are learnable parameters. D_{out} is the dimension of the node features and M is the number of output classes. The output is of the shape $(K^2 \times M)$. In our GraCapsNet for object recognition, $\mathbf{Att}^l$ corresponds to the visual attention of the heads.

The visual attention describes how important each low-level visual entity is to an output class. We normalize attention output with softmax function in the first dimension, i.e., between low-level entities. Hence, the attention on a visual

Datasets	MNIST		Fashion MNIST		CIFAR10	
Model	#Para.(M)	Accuracy	#Para.(M)	Accuracy	#Para.(M)	Accuracy
CapsNets (Sabour, Frosst, and Hinton 2017)	6.54	99.41(± 0.08)	6.54	92.12(± 0.29)	7.66	74.64(± 1.02)
GraCapsNets	**1.18**	**99.50**(± 0.09)	**1.18**	**93.1**(± 0.09)	**2.90**	**82.21**(± 0.11)

Table 1: Compared to CapsNets, GraCapsNets achieve slightly better performance on grayscale image datasets and significantly better performance on CIFAR10 with fewer parameters.

entity could be nearly zero for all classes. Namely, a visual entity can abstain from voting. When some visual entities correspond to the noisy background of the input image, the noise can be filtered out by the corresponding abstentions.

The attention is used to pool nodes of the graph for output classes. The graph pooling output $\mathbf{S}^l \in \mathbb{R}^{(M \times D_{out})}$ of the head is computed as

$$\mathbf{S}^l = (\mathbf{Att}^l)^T \mathbf{X}^l. \tag{3}$$

The final predictions of GraCapsNets are based on all L heads with outputs $\mathbf{S}^l$ where $l \in \{1, 2, ..., L\}$. The output capsules are

$$\mathbf{V} = \text{squash}(\frac{1}{L}\sum_{l=1}^{L} \mathbf{S}^l) \tag{4}$$

Following CapsNets (Sabour, Frosst, and Hinton 2017), the squashing function is applied to each high-level capsule $\mathbf{s}_j \in \mathbb{R}^{D_{out}}$.

$$\text{squash}(\mathbf{s}_j) = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|} \tag{5}$$

and the loss function used to train our GraCapsNets is

$$\begin{aligned} L_k =& T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 \\ &+ \lambda(1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2 \end{aligned} \tag{6}$$

where $T_k = 1$ if the object of the k-th class is present. As in (Sabour, Frosst, and Hinton 2017), the hyper-parameters are often empirically set as $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$. The effectiveness of Graph Pooling as well as Multiple Heads is verified in the experimental section.

3.3 Interpretability in GraCapsNets

There is no interpretation method designed specifically for CapsNets. The existing ones were proposed for CNNs. Only the architecture-agnostic ones (Simonyan, Vedaldi, and Zisserman 2013; Sundararajan, Taly, and Yan 2017; Smilkov et al. 2017) can be trivially generalized to CapsNets, which only requires the gradients of the output with respect to the input.

In our GraCapsNet, we can use visual attention as built-in explanation to explain the predictions of GraCapsNets. The averaged attenion over l heads is

$$\mathbf{E} = \frac{1}{L}\sum_{l=1}^{L} \mathbf{Att}^l \tag{7}$$

where $\mathbf{Att}^l$ corresponds to the attention of the l-th head. The created explanations $\mathbf{E}$ are of the shape $(K^2 \times M)$. Given the predicted class, the $K \times K$ attention map indicates which pixels of the input image support the prediction.

4 Experiments

Many new versions of CapsNets have been proposed, and they report competitive classification performance. However, the advantages of CapsNets over CNNs are not only in performance but also in other properties, e.g., disentangled representations, adversarial robustness. Additionally, instead of pure convolutional layers, ResNet backbones(He et al. 2016) are often applied to extract primary capsules to achieve better performance.

Hence, in this work, we **comprehensively** evaluate our GraCapsNets from the four following aspects. All scores reported in this paper are averaged over 5 runs.

1. Classification Performance: Comparison of our GraCapsNets with original CapsNets built on two convolutional layers and the ones built on ResNet backbones.
2. Classification Interpretability: Comparison of explanations in Section 3.3 with the ones created by the architecture-agnostic saliency methods.
3. Adversarial Robustness: Comparison of GraCapsNets with various CapsNets and counter-part CNNs.
4. We show GraCapsNets also learn disentangled representations and achieve similar transformation robustness.

4.1 Classification Performance

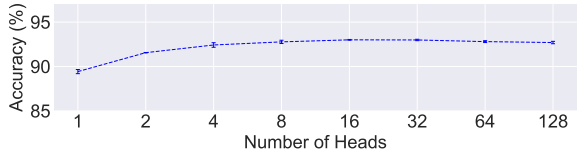
The datasets, MNIST (LeCun et al. 1998), F-MNIST (Xiao, Rasul, and Vollgraf 2017) and CIFAR10 (Krizhevsky et al. 2009), are used in this experiment. The data preprocessing, the arhictectures and the training procedure are set identically to (Sabour, Frosst, and Hinton 2017) Correspondingly, in GraCapsNets, 32 heads and $8D$ primary capsules are used. 3×3 kernels are used in Conv layers to obtain graphs with 144 nodes on MNIST, 196 nodes on CIFAR10.

Comparison with the original CapsNets The classification results are reported in Table 1. In grayscale images, GraCapsNets achieve slightly better performance with fewer parameters. In CIFAR10, our model outperforms the CapsNet by a large margin. The reason behind this is that our graph pooling process can better filter out the background noise. The pixel values of the background of grayscale images are often zeros, not noisy. Hence, our model performs much better on realistic datasets.

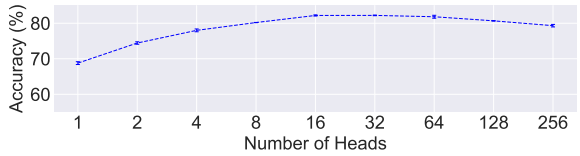
Ablation Study on Multiple Heads In this experiment, we set the number of feature maps fixed (e.g., 256 on F-MNIST). We train GraCapsNets with different number of heads 2^n where $n \in \{0, 1, ...7\}$. The corresponding dimensions of the primary capsules are 2^n where $n \in \{8, 7, ...1\}$. The performance is shown in Figure 2. The GraCapsNet with more heads achieves better performance in general.

Models	#Para.(M)	FLOPs(M)	CIFAR10	SVHN
Backbone + Avg	0.27	41.3	7.94(±0.21)	3.55(±0.11)
Backbone + FC	0.89	61.0	10.01(±0.99)	3.98(±0.15)
Dynamic Routing (Sabour, Frosst, and Hinton 2017)	5.81	73.5	8.46(±0.27)	3.49(±0.69)
EM Routing (Hinton, Sabour, and Frosst 2018)	0.91	76.6	10.25(±0.45)	3.85(±0.13)
Self-Routing (Hahn, Pyeon, and Kim 2019)	0.93	62.2	8.17(±0.18)	3.34(±0.08)
GraCapsNets	**0.28**	**59.6**	**7.99**(±0.13)	**2.98**(±0.09)

Table 2: Comparison to state-of-the-art CapsNets performance on the benchmark datasets.

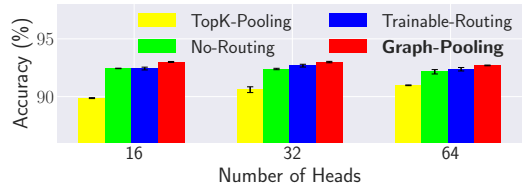


(a) F-MNIST Dataset

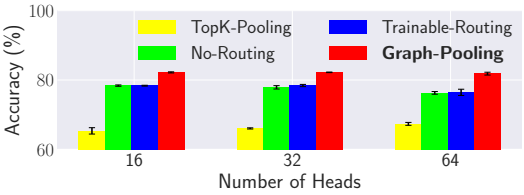


(b) CIFAR10 Dataset

Figure 2: Ablation study on multiple heads: Given fixed channels, the GraCapsNets with more heads perform better in general. The GraCapsNets with too many heads can degrade a little since the small primary capsules are not able to represent visual entities well.



(a) F-MNIST Dataset



(b) CIFAR10 Dataset

Figure 3: Ablation Study on Graph Pooling: GraCapsNets with graph modeling outperform others.

However, when too many heads are used, the performance decreases a little. In that case, the dimensions of the primary capsules are too small to capture the properties of low-level visual entities. Overall, our model is not very sensitive to the number of heads. When the number heads vary from 16 to 64, our models show similar performance with tiny variance.

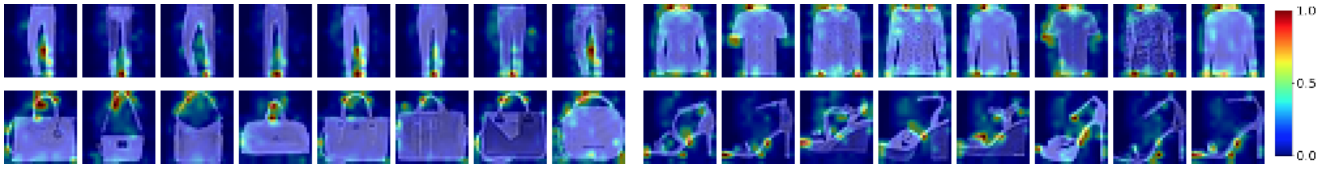
Ablation Study on Graph Pooling In GraCapsNets, we model the transformed capsules as multiple graphs. The spatial relationship between the transformed capsules is modeled in each graph. To investigate the effectiveness of the graph modeling, we compare GraCapsNets with closely related pooling operations as well as routing mechanisms.

Top-K graph pooling (Gao and Ji 2019; Knyazev, Taylor, and Amer 2019), simplified version of our graph pooling approach, projects node features into a feature space, and chooses the top-K ones to coarsen the graph, where the graph structure (spatial relationship) is not used. In addition, the trainable routing algorithm (Chen and Crandall 2019) predict directly which primary capsules should be routed to which output capsules. In No-routing algorithm (Gu and Tresp 2020), the transformed capsules are simply averaged to obtain output capsules. The two routing algorithms are strong baselines and leverage no graph information when aggregating information.

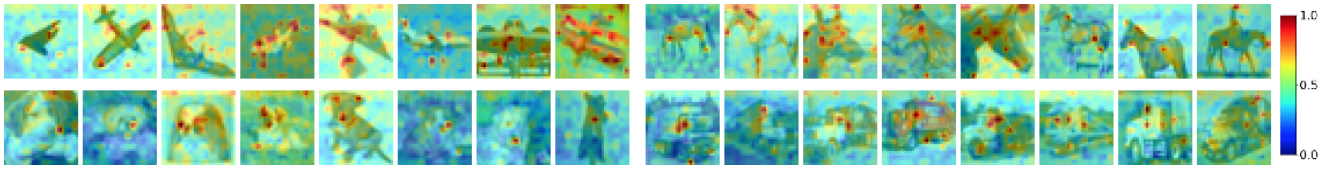
We report the performance of different graph pooling operations and routing algorithms in Figure 3. Our Graph-Pooling with different heads outperforms others on both datasets, which indicate the effectiveness of the part-part relationship modeled in our Graph-Pooling process.

Comparison with various CapsNets on ResNet Backbones The backbones are supposed to extract more accurate primary capsules. To compare with various CapsNets, we also build our GraCapsNets on their backbones. Following (Hahn, Pyeon, and Kim 2019), we apply Dynamic routing, EM-routing, Self-routing, and our Multi-head Graph Pooling on the ResNet20 (He et al. 2016) backbone. Two CNN baselines are Avg): the original ResNet20 and FC): directly followed by Conv + FC without pooling.

The performance is reported in Table 2. Our GraCapsNets outperform previous routing algorithm slightly, but with fewer parameters and less computational cost. Our GraCapsNets achieve better performance than similar-sized CNNs. The size of GraCapsNets is even comparable to the original ResNet20. Besides the popular routing mechanisms above, other new CapsNets architectures (Ahmed and Torresani 2019) and Routing mechanisms (Zhang, Edraki, and Qi 2018; Tsai et al. 2020) have also been recently proposed. They report scores on different backbones in different settings. Compared to scores reported in their papers, ours also achieves comparable performance with fewer parameters.



(a) Visual Attention as Explanations on F-MNIST Dataset.



(b) Visual Attention as Explanations on CIFAR10 Dataset.

Figure 4: Visual Attention in GraCapsNets: the models focus on discriminative input visual features, e.g., the handles of the handbags and the wings of the planes.

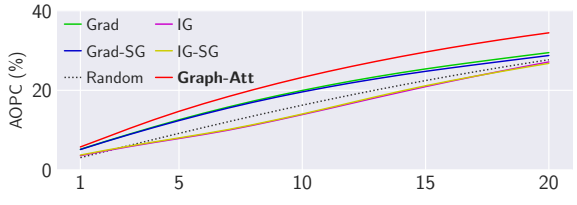
4.2 Classification Interpretability

The predictions of GraCapsNet can be easily explained with their visual attention. We visualize the attention in inferences and compare them with the explanations created by other appliable interpretation methods, namely, Grad (Simonyan, Vedaldi, and Zisserman 2013), IG (Sundararajan, Taly, and Yan 2017), Grad-SG and IG-SG (Smilkov et al. 2017). In this experiment, the settings of these methods follow Captum package (Kokhlikyan et al. 2019). Only GraCapsNets are used. We use the ones with basic architecture from Section 4.1.

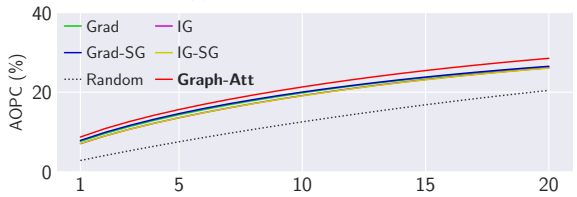
Qualitative Evaluation We make predictions with our GraCapsNets for some examples chosen randomly from test datasets. The visual attention is visualized on the original input in Figure 4. The color bars right indicate the importance of the input features, where blue corresponds to little relevance, dark red to high relevance.

For instance, in F-MNIST, the trouser legs and the gap between them are relevant for the recognition of the class *Trouser*, the handles is to *Bag*; In CIFAR10, the wings to *Plane*, and the heads (especially the noses) to *Dog*. Since the visual attention is more aligned with human-vision perception, the observations also explain why our models are more robust to adversarial examples. We also visualize explanations created by all baseline methods, which are less interpretable.

Quantitative Evaluation The quantitative evaluation of saliency map explanations is still an open research topic (Sturmfels, Lundberg, and Lee 2020). In this work, we quantitatively evaluate explanations with a widely used metric, i.e. Area Over the Perturbation Curve (AOPC) (Samek et al. 2017) $AOPC = \frac{1}{L+1}\langle \sum_{k=1}^{L} f(\mathbf{X}^{(0)}) - f(\mathbf{X}^{(k)}) \rangle_{p(\mathbf{X})}$, where L is the number of pixel deletion steps, $f(\cdot)$ is the model, $\mathbf{X}^{(K)}$ is the input image after k perturbation steps. The order of perturbation steps follow the relevance order of corresponding input pixels in explanations. In each perturbation step, the target pixel is replaced by a patch (5×5) with random values from $[0, 1]$. The higher the AOPC is, the more accurate the explanation are.



(a) On F-MNIST Dataset

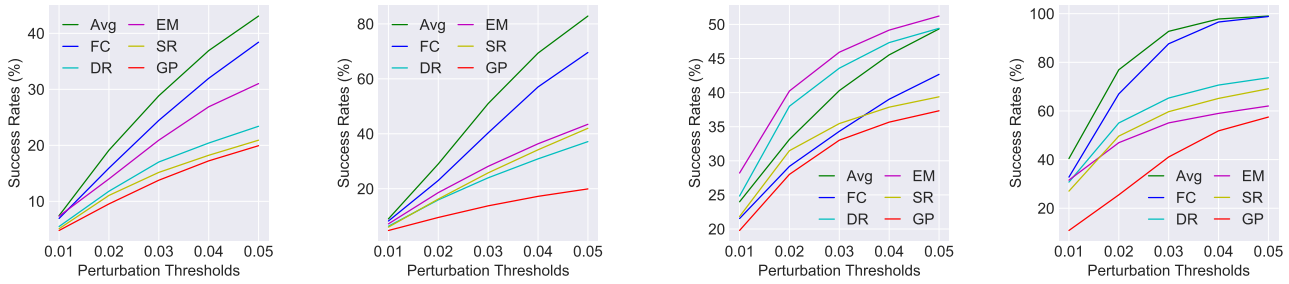


(b) On CIFAR10 Dataset

Figure 5: Quantitative evaluation of explanations with AOPC metric: Our Graph-Att performs the best.

The AOPC scores are shown in Figure 5. The difference between the baseline methods and their smoothed versions is small since our model is robust to input random perturbation noise. Our Graph-Att achieve better scores than other explanations. On F-MNIST dataset, IG is not better than Grad, even worse than Random. The existing advanced interpretation methods are not suitable for capsule-type networks. For more methods SquaredGrad and VarGrad (Adebayo et al. 2018), our methods are orthogonal to them and can also be combined with them.

Efficiency In GraCapsNets, the single explanation created by visual attention can be obtained in half forward pass without backpropagation. Grad requires a single forward and backward pass. IG interpolates examples between a baseline and inputs, which requires M(=50) times forward and backward passes. SG variants achieve smoothy explanation by adding different noise into inputs, which require N(=10) times more forward and backward passes, i.e., N*M(=500) for IG-SG. In summary, the explanations inside our GraCapsNets is better and require less computational cost.



(a) FGSM on SVHN (b) C&W on SVHN (c) FGSM on CIFAR10 (d) C&W on CIFAR10

Figure 6: On SVHN and CIFAR10, the attack methods attack our models (GP) with less success rate.

4.3 Adversarial Robustness

The work (Hahn, Pyeon, and Kim 2019) also claims that their routing mechanism is more robust to adversarial attacks. Follow their settings, we compare our model with routing algorithms in terms of the adversarial robustness.

In this experiment, we use the trained models in Section 4.1. FGSM (Goodfellow, Shlens, and Szegedy 2015) (a primary attack method) and C&W (Carlini and Wagner 2017) are applied to create adversarial examples. Their hyperparameter settings are default in Adversarial Robustness 360 Toolbox (Nicolae et al. 2018). The same settings are used to attack all models. Instead of choosing a single perturbation threshold, we use different thresholds, i.e., in the range $[0.01, 0.05]$ with the interval of 0.01.

Attack success rate is used to evaluate the model robustness. Only correctly classified samples are considered in this experiment. An untargeted attack is successful when the prediction is changed, and a targeted attack is successful if the input is misclassified into the target class.

Figure 6 shows the success rates of CNNs (Avg, FC), CapsNets (DR, EM, SR) and our GraCapsNets (GP) under untargeted setting. Overall, CapsNets with various routing algorithms more robust than CNNs. Especially, when the strong attack C&W is used under a large threshold of 0.05, all the predictions of CNNs can be misled by perturbations. The attack methods achieve less success rate on our models (GP). The experiments on the targeted setting also show similar results. In our models, the attention-based graph pooling process can filter out part of noisy input features, which makes successful attacks more difficult.

4.4 Disentangled Representations and Transformation Robustness

In CapsNets, the reconstruction net reconstructs the original inputs from the disentangled activity vectors of the output capsules. When elements of the vector representation are perturbated, the reconstructed images are also changed correspondingly. We also conduct the perturbation experiments on output capsules of GraCapsNet. Similarly, we tweak one dimension of capsule representation by intervals of 0.05 in the range $[-0.25, 0.25]$. The reconstructed images are visualized in Figure 7. We can observe that our GraCapsNet also captures disentangled representations. For instance, the property *Size* of the class *Bag* in F-MNIST.

Class	Property	Reconstructions by Perturbing Representations
0	Translation	
3	Rotation	
4	Style	
7	Thickness	

(a) MNIST Dataset

Class	Property	Reconstructions by Perturbing Representations
Trouser	Thickness	
Bag	Size	
Sandal	Translation	
Boot	Height	

(b) F-MNIST Dataset

Figure 7: Disentangled Individual Dimensions of Representations in GraCapsNets: By perturbing one dimension of an activity vector, the variations of an input image are reconstructed.

On the affine transformation benchmark task, where models are trained on the MNIST dataset and tested on the AffNIST dataset (novel affine transformed MNIST images), the CapsNets are shown to be more robust to input affine transformations than similar-sized CNNs (79% vs. 66%) (Sabour, Frosst, and Hinton 2017). Following their setting, we also test our GraCapsNet on this benchmark, the test performance on AffNIST dataset is slightly better (80.45%).

5 Conclusion

We propose an interpretable GraCapsNet. The explanations for individual classifications of GraCapsNets can be created in an effective and efficient way. Surprisingly, without a routing mechanism, our GraCapsNets can achieve better classification performance and better adversarial robustness, and still keep other advantages of CapsNets. This work also reveals that we cannot attribute the advantages of CapsNets to the routing mechanisms, even though they are fundamental parts of CapsNets.

References

Adebayo, J.; Gilmer, J.; Muelly, M.; Goodfellow, I.; Hardt, M.; and Kim, B. 2018. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 9505–9515.

Ahmed, K.; and Torresani, L. 2019. STAR-Caps: Capsule Networks with Straight-Through Attentive Routing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 9098–9107.

Athalye, A.; Carlini, N.; and Wagner, D. A. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *International Conference on Machine Learning (ICML)*.

Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.-R.; and Samek, W. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10(7): e0130140.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57.

Chen, Z.; and Crandall, D. 2019. Generalized capsule networks with trainable routing procedure. In *ICML Workshop*.

Cohen, J. M.; Rosenfeld, E.; and Kolter, J. Z. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *International Conference on Machine Learning (ICML)*.

Dhillon, I. S.; Guan, Y.; and Kulis, B. 2007. Weighted Graph Cuts without Eigenvectors A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 29.

Gao, H.; and Ji, S. 2019. Graph U-Nets. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2083–2092.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*.

Gu, J.; and Tresp, V. 2019. Saliency methods for explaining adversarial attacks. *arXiv preprint arXiv:1908.08413* .

Gu, J.; and Tresp, V. 2020. Improving the Robustness of Capsule Networks to Image Affine Transformations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gu, J.; Wu, B.; and Tresp, V. 2021. Effective and Efficient Vote Attack on Capsule Networks. In *International Conference on Learning Representations (ICLR)*.

Gu, J.; Yang, Y.; and Tresp, V. 2018. Understanding individual decisions of cnns via contrastive backpropagation. In *Asian Conference on Computer Vision (ACCV)*, 119–134. Springer.

Hahn, T.; Pyeon, M.; and Kim, G. 2019. Self-Routing Capsule Networks. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, 7658–7667.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hinton, G. E.; Sabour, S.; and Frosst, N. 2018. Matrix capsules with EM routing. In *International Conference on Learning Representations (ICLR)*.

Khasanova, R. 2019. Graph-based image representation learning. In *THESIS in EPFL*.

Knyazev, B.; Lin, X.; Amer, M. R.; and Taylor, G. W. 2019. Image Classification with Hierarchical Multigraph Networks. In *British Machine Vision Conference (BMVC)*.

Knyazev, B.; Taylor, G. W.; and Amer, M. 2019. Understanding Attention and Generalization in Graph Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, 4202–4212.

Kokhlikyan, N.; Miglani, V.; Martin, M.; Wang, E.; Reynolds, J.; Melnikov, A.; Lunova, N.; and Reblitz-Richardson, O. 2019. PyTorch Captum. https://github.com/pytorch/captum.

Krizhevsky, A.; et al. 2009. Learning multiple layers of features from tiny images. *Tech Report* .

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-attention graph pooling. In *International Conference on Machine Learning (ICML)*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*.

Nicolae, M.-I.; Sinn, M.; Tran, M. N.; Rawat, A.; Wistuba, M.; Zantedeschi, V.; Baracaldo, N.; Chen, B.; Ludwig, H.; Molloy, I. M.; et al. 2018. Adversarial Robustness Toolbox v0. 4.0. *arXiv preprint arXiv:1807.01069* .

Qin, Y.; Frosst, N.; Sabour, S.; Raffel, C.; Cottrell, G.; and Hinton, G. 2020. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. In *International Conference on Learning Representations (ICLR)*.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 3856–3866.

Samek, W.; Binder, A.; Montavon, G.; Lapuschkin, S.; and Müller, K.-R. 2017. Evaluating the Visualization of What a Deep Neural Network Has Learned. *IEEE Transactions on Neural Networks and Learning Systems* 28: 2660–2673.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D.; et al. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *ICCV*, 618–626.

Shrikumar, A.; Greenside, P.; and Kundaje, A. 2017. Learning Important Features Through Propagating Activation Differences. In *International Conference on Machine Learning (ICML)*.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *International Conference on Learning Representations (ICLR)*.

Smilkov, D.; Thorat, N.; Kim, B.; Viégas, F.; and Wattenberg, M. 2017. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* .

Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; and Riedmiller, M. 2014. Striving for simplicity: The all convolutional net. *International Conference on Learning Representations (ICLR)* .

Sturmfels, P.; Lundberg, S.; and Lee, S.-I. 2020. Visualizing the impact of feature attribution baselines. *Distill* 5(1): e22.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning (ICML)*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

Tsai, Y.-H. H.; Srivastava, N.; Goh, H.; and Salakhutdinov, R. 2020. Capsules with Inverted Dot-Product Attention Routing. In *International Conference on Learning Representations (ICLR)*.

Verma, S.; and Zhang, Z.-L. 2018. Graph capsule convolutional neural networks. *arXiv preprint arXiv:1805.08090* .

Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order Matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Wong, E.; and Kolter, J. Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* .

Xinyi, Z.; and Chen, L. 2019. Capsule Graph Neural Network. In *International Conference on Learning Representations (ICLR)*.

Ying, R.; You, J.; Morris, C.; Ren, X.; Hamilton, W. L.; and Leskovec, J. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang, L.; Edraki, M.; and Qi, G.-J. 2018. Cappronet: Deep feature learning via orthogonal projections onto capsule subspaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 5814–5823.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Chapter 4

Affine Transformation-Robust Capsule Networks

Improving the Robustness of Capsule Networks to Image Affine Transformations

Jindong Gu
University of Munich
Siemens AG, Corporate Technology
jindong.gu@siemens.com

Volker Tresp
University of Munich
Siemens AG, Corporate Technology
volker.tresp@siemens.com

Abstract

Convolutional neural networks (CNNs) achieve translational invariance by using pooling operations. However, the operations do not preserve the spatial relationships in the learned representations. Hence, CNNs cannot extrapolate to various geometric transformations of inputs. Recently, Capsule Networks (CapsNets) have been proposed to tackle this problem. In CapsNets, each entity is represented by a vector and routed to high-level entity representations by a dynamic routing algorithm. CapsNets have been shown to be more robust than CNNs to affine transformations of inputs. However, there is still a huge gap between their performance on transformed inputs compared to untransformed versions. In this work, we first revisit the routing procedure by (un)rolling its forward and backward passes. Our investigation reveals that the routing procedure contributes neither to the generalization ability nor to the affine robustness of the CapsNets. Furthermore, we explore the limitations of capsule transformations and propose affine CapsNets (Aff-CapsNets), which are more robust to affine transformations. On our benchmark task, where models are trained on the MNIST dataset and tested on the AffNIST dataset, our Aff-CapsNets improve the benchmark performance by a large margin (from 79% to 93.21%), without using any routing mechanism.

1. Introduction

Human visual recognition is quite insensitive to affine transformations. For example, entities in an image, and a rotated version of the entities in the image, can both be recognized by the human visual system, as long as the rotation is not too large. Convolutional Neural Networks (CNNs), the currently leading approach to image analysis, achieve affine robustness by training on a large amount of data that contain different transformations of target objects. Given limited training data, a common issue in many real-world tasks, the robustness of CNNs to novel affine transformations is limited [23].

With the goal of learning image features that are more aligned with human perception, Capsule Networks (CapsNets) have recently been proposed [23]. The proposed CapsNets differ from CNNs mainly in two aspects: first, they represent each entity by an activation vector, the magnitude of which represents the probability of its existence in the image; second, they assign low-level entity representations to high-level ones using an iterative routing mechanism (a dynamic routing procedure). Hereby, CapsNets aim to keep two important features: equivariance of output-pose vectors and invariance of output activations. The general assumption is that the disentanglement of variation factors makes CapsNets more robust than CNNs to affine transformations.

The currently used benchmark task to evaluate the affine robustness of a model is to train the model on the standard MNIST dataset and test it on the AffNIST[1] dataset. CapsNets achieve 79% accuracy on AffNIST, while CNNs with similar network size only achieve 66% [23]. Although CapsNets have demonstrated their superiority on this task, there is still a huge performance gap since CapsNets achieve more than 99% on the untransformed MNIST test dataset.

In our paper, we first investigate the effectiveness of components that make CapsNets robust to input affine transformations, with a focus on the routing algorithm. Many heuristic routing algorithms have been proposed [10, 25, 16] since [23] was published. However, recent work [19] shows that all routing algorithms proposed so far perform even worse than a uniform/random routing procedure.

From both numerical analysis and empirical experiments, our investigation reveals that the dynamic routing procedure contributes neither to the generalization ability nor to the affine robustness of CapsNets. Therefore, it is infeasible to improve the affine robustness by modifying the routing procedure. Instead, we investigate the limitations of the CapsNet architectures and propose a simple solution. Namely, we propose to apply an identical transformation function for all primary capsules and replace the routing by a simple averaging procedure (noted as No Routing).

[1] Each example is an MNIST digit with a small affine transformation.

Our contributions of this work can be summarized as follows: 1) We revisit the dynamic routing procedure of CapsNets; 2) We investigate the limitations of the current CapsNet architecture and propose a more robust affine Capsule Networks (Aff-CapsNet); 3) Based on extensive experiments, we investigate the properties of CapsNets trained without routing. Besides, we demonstrate the superiority of Aff-CapsNet.

The rest of this paper is organized as follows: Section 2 first reviews CapsNets and related work. Section 3 investigates the effectiveness of the routing procedure by (un)rolling the forward and backward passes of the iterative routing iterations. Section 4 shows the limitations of current CapsNets on the affine transformations and proposes a robust affine CapsNet (Aff-CapsNet). Section 4 conducts extensive experiments to verify our findings and proposed modifications. The last two sections discuss and conclude our work.

2. Background and Related Work

In this section, we first describe the CapsNets with dynamic routing and then review related work.

2.1. Fundamentals of Capsule Networks

CapsNets [23] encode entities with capsules. Each capsule is represented by an activity vector (e.g., the activation of a group of neurons), and elements of each vector encode the properties of the corresponding entity. The length of the activation vector indicates the confidence of the entity's existence. The output classes are represented as high-level capsules.

A CapsNet first maps the raw input features to low-level capsules and then routes the low-level capsules to high-level ones. For instance, in image classification tasks, a CapsNet starts with one (or more) convolutional layer(s) that convert the pixel intensities into low-level visual entities. A following capsule layer of the CapsNet routs low-level visual entities to high-level visual entities. A CapsNet can have one or more capsule layers with routing procedures.

Given a low-level capsule $\boldsymbol{u}_i$ of the L-th layer with N capsules, a high-level capsule $\boldsymbol{s}_j$ of the $(L+1)$-th layer with M capsules, and a transformation matrix $\boldsymbol{W}_{ij}$, the routing process is

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij}\boldsymbol{u}_i, \qquad \boldsymbol{s}_j = \sum_i^N c_{ij}\hat{\boldsymbol{u}}_{j|i} \tag{1}$$

where c_{ij} is a coupling coefficient that models the degree with which $\hat{\boldsymbol{u}}_{j|i}$ is able to predict $\boldsymbol{s}_j$. The capsule $\boldsymbol{s}_j$ is shrunk to a length in (0, 1) by a non-linear squashing function $g(\cdot)$, which is defined as

$$\boldsymbol{v}_j = g(\boldsymbol{s}_j) = \frac{\|\boldsymbol{s}_j\|^2}{1+\|\boldsymbol{s}_j\|^2}\frac{\boldsymbol{s}_j}{\|\boldsymbol{s}_j\|} \tag{2}$$

The coupling coefficients $\{c_{ij}\}$ are computed by an iterative routing procedure. They are updated so that high agreement ($a_{ij} = \boldsymbol{v}_j^T\hat{\boldsymbol{u}}_{j|i}$) corresponds to a high value of c_{ij}.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{3}$$

where initial logits b_{ik} are the log prior probabilities and updated with $b_{ik} = b_{ik} + a_{ij}$ in each routing iteration. The coupling coefficients between a i-th capsule of the L-th layer and all capsules of the $(L+1)$-th layer sum to 1, i.e., $\sum_{j=1}^M c_{ij} = 1$. The steps in Equations 1, 2, and 3 are repeated K times in the routing process, where $\boldsymbol{s}_j$ and c_{ij} depend on each other.

2.2. Related Work

Routing Algorithms: Many papers have improved the routing-by-agreement algorithm. [27] generalizes existing routing methods within the framework of weighted kernel density estimation and proposes two fast routing methods with different optimization strategies. [6] proposes an attention-based routing procedure with an attention module, which only requires a fast forward-pass. The agreement a_{ij} can also be calculated based on a Gaussian distribution assumption [10, 2] or distance measures [16] instead of the simple inner product.

Since the routing procedure is computationally expensive, several works propose solutions reducing the complexity of the iterative routing process. [25] formulates the routing strategy as an optimization problem that minimizes a combination of clustering-like loss and a KL distance between the current coupling distribution and its last states. [17] approximates the expensive routing process with two branches: a master branch that collects primary information from its direct contact in the lower layer and an aide branch that replenishes the master branch based on pattern variants encoded in other lower capsules.

Understanding the Routing Procedure: [4] incorporates the routing procedure into the training process by making coupling coefficients trainable, which are supposed to be determined by an iterative routing process. The coupling coefficients are independent of examples, which stay unchanged in the testing phase. What they proposed is simply to reduce the iterative updates to a single forward pass with prior coupling coefficients. [5] removes the routing procedure completely and modifies the CapsNet architectures. Their pure CapsNets achieve competitive performance. However, it has not been investigated how the properties of their CapsNets, e.g., the robustness to affine transformation, will be affected by the removal of the routing procedure. Furthermore, [19] shows that many routing procedures [23, 10, 25, 16] are heuristic, and perform even worse than a random routing assignment.

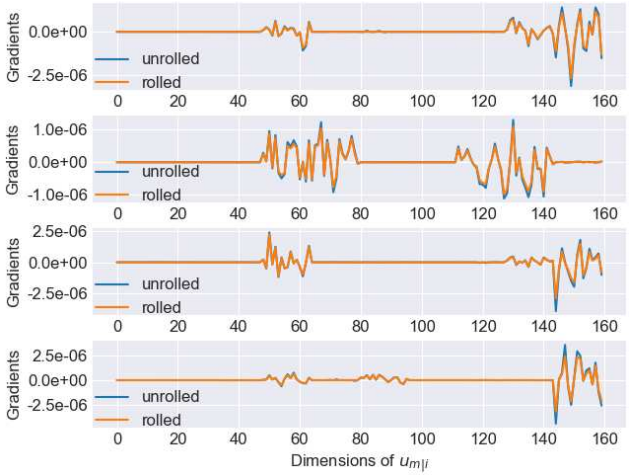


Figure 1: The gradients of the loss w.r.t. randomly choosen $\hat{\boldsymbol{u}}_{m|i}$ are visualized. The blue lines correspond to the unrolled routing iterations in Gradient Backpropagation, while the yellow lines to rolled routing iterations.

3. Revisiting the Dynamic Routing of CapsNets

In this section, we analyze dynamic routing, both theoretically and empirically. By unrolling the backpropagation of the routing procedure and rolling the forward propagation of the routing procedure, we show which role the routing procedure plays in CapsNets.

3.1. Backpropagation through Routing Iterations

The forward pass of an iterative routing process can be written as the following iterative steps

$$\begin{aligned} \boldsymbol{s}_j^{(t)} &= \sum_i^N c_{ij}^{(t)} \hat{\boldsymbol{u}}_{j|i} \\ \boldsymbol{v}_j^{(t)} &= g(\boldsymbol{s}_j^{(t)}) \\ c_{ij}^{(t+1)} &= \frac{\exp(b_{ij} + \sum_{r=1}^{t} \boldsymbol{v}_j^{(r)} \hat{\boldsymbol{u}}_{j|i})}{\sum_k \exp(b_{ik} + \sum_{r=1}^{t} \boldsymbol{v}_k^{(r)} \hat{\boldsymbol{u}}_{k|i})} \end{aligned} \tag{4}$$

where the superscript $t \in \{1, 2, ...\}$ is the index of an iteration. The $c_{ij}^{(1)}$ and b_{ij} are initialized as in Equation 3.

Assuming that there are K iterations and the classification loss is $\mathcal{L}(\boldsymbol{y}, \boldsymbol{t})$, where $\boldsymbol{y} = (\|\boldsymbol{v}_1^{(K)}\|, \cdots, \|\boldsymbol{v}_M^{(K)}\|)$ is the prediction and $\boldsymbol{t}$ the target, the gradients through the routing procedure are

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{u}}_{m|i}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} c_{im}^{(K)} + \sum_{j=1}^{M} \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_j^{(K)}} \frac{\partial \boldsymbol{v}_j^{(K)}}{\partial \boldsymbol{s}_j^{(K)}} \hat{\boldsymbol{u}}_{j|i} \frac{\partial c_{ij}^{(K)}}{\partial \hat{\boldsymbol{u}}_{m|i}} \tag{5}$$

The gradients are propagated through the unrolled routing iteration via the second item of Equation 5, which is also the main computational burden of the expensive routing procedure in CapsNets. By unrolling this term, we prove that

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{u}}_{m|i}} \approx C \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} c_{im}^{(K)} \tag{6}$$

where C is a constant, which can be integrated into the learning rate in the optimization process (see the proof in Appendix A). The approximation means that the gradients flowing through $c_{ij}^{(K)}$ in Equation 5 can be ignored. The $c_{ij}^{(K)}$ can be treated as a constant in Gradient Backpropagation, and the routing procedure can be detached from the computational graph of CapsNets.

To confirm Equation 6 empirically, we visualize $\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{u}}_{m|i}}$. Following [23], we train a CapsNet on the MNIST dataset. The architecture and the hyper-parameter values can be found in Appendix B. We first select capsule predictions $\hat{\boldsymbol{u}}_{j|i}$ randomly prior to the routing process and then visualize their received gradients in two cases: 1) unrolling the routing iterations as in [23]; 2) rolling the routing iterations by taking all c_{ij} as constants in Gradient Backpropagation (i.e., ignoring the second item in Equation 5). As shown in each plot of Figure 1, the gradients of the two cases (blue lines and yellow lines) are similar to each other.

In this section, we aim to show that the intrinsic contribution of the routing procedure is to identify specified constants as coupling coefficients $c_{ij}^{(K)}$. Without a doubt, both computational cost and memory footprint can be saved by rolling the routing iterations in Gradient Backpropagation. The computational graphs of the two cases can be found in Appendix C.

3.2. Forward Pass through Routing Iterations

The forward iterative routing procedure can be formulated as a function, mapping capsule predictions $\hat{\boldsymbol{u}}$ to coupling coefficients, i.e., $\hat{\boldsymbol{u}} \rightarrow \boldsymbol{C}^{(K)} = \{c_{ij}^{(K)}\}$ where the indexes of low-level capsules i vary from 1 to N and the indexes of high-level capsules j vary from 1 to M. Given an instance, without loss of generality, we assume the ground-truth class is the M-th (i.e., $\boldsymbol{v}_M$). With the idea behind the CapsNet, the optimal coupling coefficients $\boldsymbol{C}^* = \{c_{ij}^*\}$ of the instance can be described as

$$\begin{aligned} \boldsymbol{C}^* = \max_{\{c_{ij}\}} f(\hat{\boldsymbol{u}}) = \max_{\{c_{ij}\}} (& \sum_i^N c_{iM} \hat{\boldsymbol{u}}_{M|i} g(\sum_i c_{iM} \hat{\boldsymbol{u}}_{M|i}) \\ & - \sum_j^{M-1} \sum_i^N c_{ij} \hat{\boldsymbol{u}}_{j|i} g(\sum_i c_{ij} \hat{\boldsymbol{u}}_{j|i})) \end{aligned} \tag{7}$$

where the first term describes the agreement on the target class, and the second term corresponds to the agreement on non-ground-truth classes. The optimal coupling coefficient $\boldsymbol{C}^*$ corresponds to the case where the agreement on the target class is maximized, and the agreement on the non-ground-truth classes is minimized.

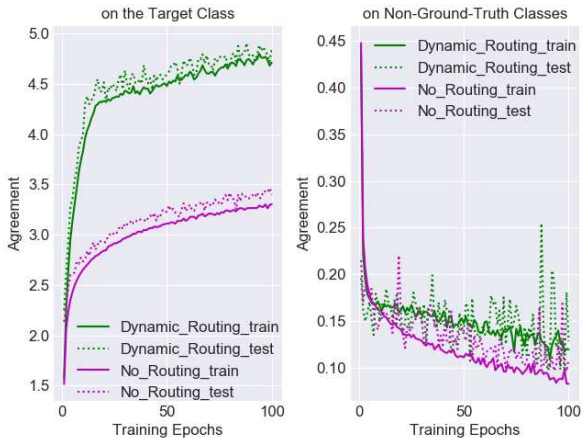


Figure 2: The green lines correspond to the model with dynamic routing, while the magenta ones to the model without routing procedure. For both models, the agreement on the target class increases with training time, and the agreement on the non-ground-truth classes decreases. The values are averaged over the whole training or test dataset.

Many routing algorithms differ only in how they approximate $\boldsymbol{C}^*$. For instance, the original work [23] approximates $\boldsymbol{C}^*$ with an iterative routing procedure. Without requiring iterative routing steps, [4] makes $\{b_{ij}\}$ trainable to approximate $\{c^*_{ij}\}$. Their proposal can be understood as only one-step routing with learned prior coupling coefficients. By further reformulation, we show that the optimal $\boldsymbol{s}^*_j$ can be learned, without a need for coupling coefficients, as

$$\boldsymbol{s}^*_j = \sum_i^N c^*_{ij}\hat{\boldsymbol{u}}_{j|i} = \sum_i^N c^*_{ij}\boldsymbol{W}^*_{ij}\boldsymbol{u}_i = \sum_i^N \boldsymbol{W}'_{ij}\boldsymbol{u}_i. \tag{8}$$

In the training process, the transformation matrix $\boldsymbol{W}_{ij}$ is updated via Gradient Decent Method. The coupling coefficients c_{ij} are determined by the agreement between low-level capsules and the corresponding high-level capsules. The training process ends up with parameter values $\boldsymbol{s}^*_j, \boldsymbol{W}^*_{ij}, c^*_{ij}$. As shown in Equation 8, the CapsNet can achieve the same results by simply learning a transformation matrix $\boldsymbol{W}'_{ij}$ without c^*_{ij}. In other words, the connection strengths c^*_{ij} between low-level capsules and high-level capsules can be learned implicitly in the transformation matrix $\boldsymbol{W}'_{ij}$. Therefore, we can conclude that different ways to approximate $\boldsymbol{C}^*$ do not make a significant difference since the coupling coefficients will be learned implicitly.

We visualize the implicit learning process of the coupling coefficients. In our experiments, we introduce the no-routing approach, where we remove the iterative routing procedure by setting all coupling coefficient c_{ij} as a constant $\frac{1}{M}$. In each training epoch, the agreement on the target class and on the non-ground-truth classes is visualized in Figure 2. As a comparison, we also visualize the corresponding agreement values of CapNets with the dynamic routing process. We can observe that, during the training process, the agreement on the target class increases (in the left plot) for both cases, and the agreement on the non-ground-truth classes decreases (in the right plot). In other words, $f(\hat{\boldsymbol{u}})$ increases in both CapNets with/without routing procedure, meaning that the coupling coefficients can be learned implicitly.

In summary, the affine robustness of CapsNet can not be contributed to the routing procedure. We conclude that it is not infeasible to improve the robustness of CapsNet by modifying the current routing-by-agreement algorithm.

4. Affine Robustness of Capsule Networks

Besides the dynamic routing process, the other difference between CapsNets and traditional CNNs is the CapsNet architecture. CapsNets represent each entity with a capsule and transform it to high-level entities employing transformation matrices. In this section, we investigate the limitation of the transformation process in terms of affine robustness and propose robust affine capsule networks.

4.1. The Limitation of CapsNets

The CapsNet starts with two convolutional layers, which converts the pixel intensities to form primary (low-level) capsules (e.g., the red cuboid in Figure 3 is a capsule $\boldsymbol{u}_i$). Each primary capsule has a certain receptive field (e.g., the image patch $\boldsymbol{x}_i$ marked with the yellow rectangle). For all inputs, the coordinates of the receptive field of $\boldsymbol{u}_i$ are the same. In other words, a primary capsule can only see a specific area in input images. We denote the corresponding converting process by $\boldsymbol{u}_i = p_i(\boldsymbol{x}_i)$.

Each primary capsule is transformed to high-level capsules with the corresponding transformation matrix. Each transformation matrix $\boldsymbol{W}_{ij}$ learns how to transform the i-th low-level capsule to the j-th high-level one, i.e., $\hat{\boldsymbol{u}}_{j|i} = t_{j|i}(\boldsymbol{u}_i)$. The transformation process corresponding to the input patch $\boldsymbol{x}_i$ can be described as

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij}\boldsymbol{u}_i = t_{j|i}(\boldsymbol{u}_i) = t_{j|i}(p_i(\boldsymbol{x}_i)). \tag{9}$$

The transformation matrix $\boldsymbol{W}_{ij}$ can only make meaningful transformations for the entities that have, at some point, appeared in the position of $\boldsymbol{x}_i$. The input domain of the transformation function $t_{j|i}(\cdot)$ is $\mathbb{U}_i$.

In the testing phase, if novel affine transformations are conducted on the input, the corresponding transformation process $t_{j|i}(p_i(\boldsymbol{x}'_i))$ are not meaningful since $p_i(\boldsymbol{x}'_i)$ is not in the input domain $\mathbb{U}_i$. In other words, the transformation matrix $\boldsymbol{W}_{ij}$ does not describe a meaningful transformation since the entities of $\boldsymbol{x}'_i$ have never appeared in the position of the patch $\boldsymbol{x}_i$ during training. Hence, the CapsNet is limited in its generalization ability to novel affine transformations of inputs.

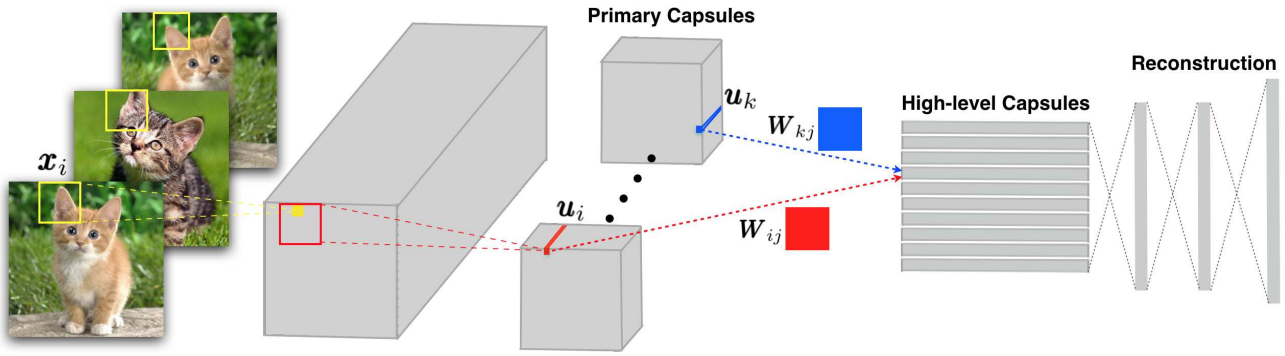


Figure 3: Illustration of the limitations of CapsNets: The transformation matrix $\boldsymbol{W}_{ij}$ can only transform $\boldsymbol{u}_i$ to high-level capsules, while $\boldsymbol{W}_{kj}$ can only make meaningful transformations on $\boldsymbol{u}_k$. When an input is transformed (e.g., rotated), the receptive field corresponding to $\boldsymbol{u}_i$ is not $\boldsymbol{x}_i$ any more. For the novel $\boldsymbol{u}_i$, the transformation process using $\boldsymbol{W}_{ij}$ can fail.

4.2. Robust Affine Capsule Networks

To overcome the limitation above, we propose a very simple but efficient solution. Concretely, we propose to use the same transformation function for all primary capsules (i.e., ensuring $t_{j|i}(\cdot) \equiv t_{j|k}(\cdot)$). We implement a robust affine capsule network (Aff-CapsNet) by sharing a transformation matrix. Formally, for Aff-CapsNets, we have

$$\boldsymbol{W}_{ij} = \boldsymbol{W}_{kj}, \ \forall i, k \in \{1, 2, \cdots, N\} \tag{10}$$

where N is the number of primary capsules. In Aff-CapsNets, the transformation matrix can make a meaningful transformation for all primary capsules since it learns how to transform all low-level capsules to high-level capsules during training. The transformation matrix sharing has also been explored in a previous publication [21]. The difference is that they aim to save parameters, while our goal is to make CapsNets more robust to affine transformations.

From another perspective, primary capsules and high-level capsules correspond to local coordinate systems and global ones, respectively. A transformation matrix is supposed to map a local coordinate system to the global one. One might be wondering that the transformation from each local coordinate system to a global one requires a specific transformation matrix. In existing architectures, the coordinate system is high-dimensional. Hence, a single shared transformation matrix is able to make successful transformations for all local coordinate systems.

5. Experiments and Analysis

The experiments include two parts: 1) We train CapsNets with different routing mechanisms (including no routing) on popular standard datasets and compare their properties from many perspectives; 2) We show that Aff-CapsNets outperform CapsNets on the benchmark dataset and achieves state-of-the-art performance. For all the experiments of this section, we train models with 5 random seeds and report their averages and variances.

5.1. Effectiveness of the Dynamic Routing

In Section 3, we show that the routing mechanism can be learned implicitly in CapsNets without routing procedure. Our experiments in this section aim to investigate if the advantages of CapsNets disappear when trained with no routing. We consider the following routing procedures in our training routines:

1. **Dynamic-R**: with standard dynamic routing in [23];
2. **Rolled-R**: with a rolled routing procedure by treating coupling coefficients as constants during Gradient Backpropagation, as analyzed in Section 3.1;
3. **Trainable-R**: one-step routing with trainable coupling coefficients, as in [4];
4. **No-R**: without routing procedure, which is equivalent to the uniform routing in [19, 5].

We train CapsNets with different routing procedures described above on four standard datasets, namely, MNIST [15], FMNIST [26], SVHN [18] and CIFAR10 [13]. The performance is reported in Table 1.

Given the performance variance for each model, the performance between different models is relatively small. The reason behind this is that coupling coefficients can be learned in transformation matrices implicitly, and all the models possess a similar transformation process. The models trained with **No-R** do not prevent the learning of coupling coefficients. We can also observe that the models with **Trainable-R** or **No-R** show a slightly better performance than the other two. To our understanding, the reason is that they do not suffer the polarization problem of coupling coefficients [17].

Datasets	MNIST	FMNIST	SVHN	CIFAR10
Dynamic-R	99.41(± 0.08)	92.12(± 0.29)	91.32(± 0.19)	74.64(± 1.02)
Rolled-R	99.29(± 0.09)	91.53(± 0.22)	90.75(± 0.52)	74.26(± 0.94)
Trainable-R	99.55(± 0.04)	92.58(± 0.10)	92.37(± 0.29)	76.43(± 1.11)
No-R	99.54(± 0.04)	92.53(± 0.26)	92.15(± 0.29)	76.28(± 0.39)

Table 1: The performance of CapsNets with different routing procedures on different standard datasets is shown, where the standard (untransformed) test datasets are used. We can observe that the routing procedures do not improve performance.

From this experiment, we can only conclude that the routing procedure does not contribute to the generalization ability of CapsNets. In work [23], CapsNets show many superior properties over CNNs, besides the classification performance. In the following, we analyze the properties of CapsNets with **No-R** and compare them with CapsNets with **Dynamic-R**.

5.1.1 On learned Representations of Capsules

When training CapsNets, the original input is reconstructed from the activity vector (i.e., instantiation parameters) of a high-level capsule. The reconstruction is treated as a regularization technique. In CapsNets with **Dynamic-R** [23], the dimensions of the activity vector learn how to span the space containing large variations. To check such property of CapsNets with **No-R**, following [23], we feed a perturbed activity vector of the ground-truth class to decoder network.

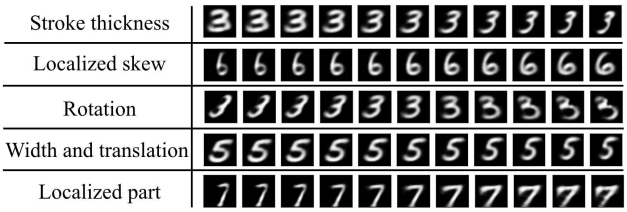


Figure 4: Disentangled Individual Dimensions of Capsules: By perturbing one dimension of an activity vector, the variations of an input image are reconstructed.

The perturbation of the dimensions can also cause variations of the reconstructed input. We show some examples in Figure 4. The variations include stroke thickness, width, translation, rotation, and various combinations. In Figure 5, we also visualize the reconstruction loss of the models with **Dynamic-R** and the ones with **No-R**. The CapsNets with **No-R** show even less reconstruction error and can reconstruct inputs better.

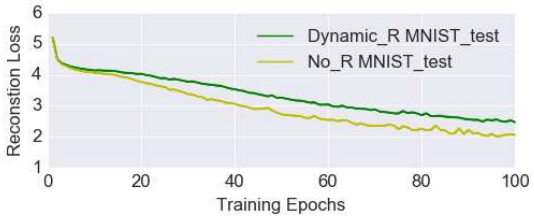


Figure 5: The average reconstruction loss of CapsNets with **Dynamic-R** and **No-R** on the test dataset is shown in each epoch of the training process.

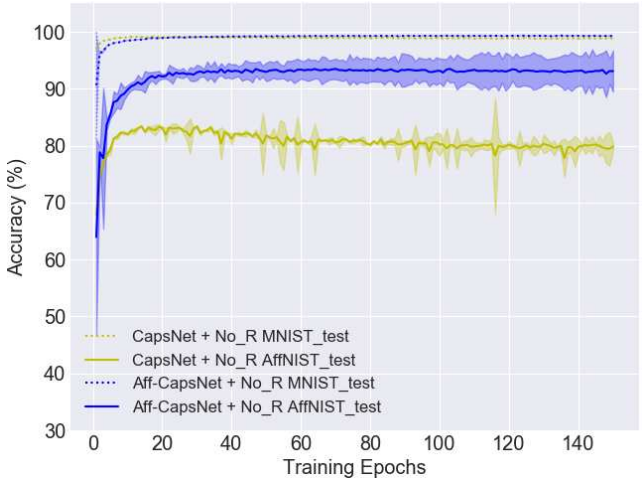
5.1.2 Parallel Attention Mechanism between Capsules

Dynamic routing can be viewed as a parallel attention mechanism, in which each high-level capsule attends to some active low-level capsules and ignores others. The parallel attention mechanism allows the model to recognize multiple objects in the image even if objects overlap [23]. The superiority of the parallel attention mechanism can be shown on the classification task on MultiMNIST dataset [9, 23]. Each image in this dataset contains two highly overlapping digits. CapsNet with dynamic routing procedure shows high performance on this task.

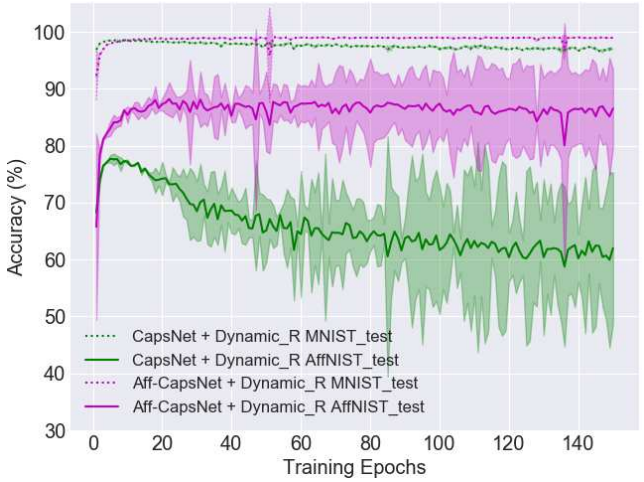
In this experiment, we show that the parallel attention mechanism between capsules can be learned implicitly, even without the routing mechanism. Following the experimental setting in [23], we train a CapsNet with **No-R** on the same classification task of classifying highly overlapping digits. The model **No-R** achieves 95,49% accuracy on the test set, while the one with **Dynamic-R** achieves 95% accuracy. The removal of the routing procedure does not make the parallel attention mechanism of CapsNets disappear.

5.1.3 Robustness to Affine Transformation

CapsNets are also known for their robustness to affine transformation. It is important to check whether the removal of the routing procedure affects the affine robustness. We conduct experiments on a standard benchmark task. Following [23], we train CapsNets with or without routing procedure on the MNIST training dataset and test them on



(a) Without a routing procedure: the test accuracy of CapsNets and Aff-Capsnets on on the expanded MNIST test set and the AffNIST test set.



(b) With the dynamic routing: the test accuracy of CapsNets and Aff-Capsnets on the expanded MNIST test set and the AffNIST test set.

Figure 6: For both cases (with or without routing procedure), Aff-CapsNets clearly outperform CapsNets on the AffNIST test dataset.

the affNIST dataset. The images in the MNIST training dataset are placed randomly on a black ground of 40×40 pixels to match the size of images in affNIST dataset. The CNN baseline is set the same as in [23].

It is hard to decide if one model is better at generalizing to novel affine transformations than another one when they achieved different accuracy on untransformed examples. To eliminate this confounding factor, we stopped training the models when they achieve similar performance, following [23]. The performance is shown in Table 2. Without routing procedure, the CapsNets show even better affine robustness.

In summary, our experiments show that the dynamic routing procedure contributes neither to the generalization ability nor to the affine robustness. Due to the high affine robustness of CapsNet cannot be attributed to the routing procedure: Instead, it is the inductive bias (architecture) of CapsNets that contributes to the affine robustness.

5.2. Affine Robustness of Aff-CapsNets

In Section 4, we proposed Aff-CapsNets that are more robust to the novel affine transformations of inputs. In this experiment, we train Aff-CapsNets with **Dynamic-R** and **No-R** respectively. As a comparison, we also train CapsNets with or without dynamic routing correspondingly.

We visualize the test accuracy on the expanded MNIST test set and the AffNIST test set. The performance is shown in Figure 6. The lines show the averaged values, while the colored areas around the lines describe the variances caused by different seeds. Figure 6a shows the accuracy of models trained without a routing procedure. We can observe that

Models	Test on MNIST	Test on AffNIST
CNN [23]	99.22%	66%
Dynamic-R [23]	99.23%	79%
No-R	99.22%	81.81%

Table 2: The performance on the expanded MNIST test set and the AffNIST test set.

the Aff-CapsNets constantly shows better accuracy than CapsNets on AffNIST. To a great extent, our Aff-CapsNets covers the performance gap between the test accuracy on untransformed examples and that on transformed ones.

In addition, the Aff-CapsNet architecture is still effective, even when the dynamic routing is applied in training (see Figure 6b). We can also observe that the CapsNets with dynamic routing overfit to the current viewpoints. With the training process going on, the coupling coefficients are polarized (become close to 0 or 1) [17]. The polarization of the coupling coefficient causes the overfitting. Furthermore, the training with dynamic routing is more unstable than without routing. The variance of model test performance in Figure 6b is much bigger than the ones in Figure 6a.

We now compare our model with previous work. In Table 3, we list the performance of CNN variants and CapsNet variants on this task. Without training on AffNIST dataset, our Aff-CapsNets achieve state-of-the-art performance on AffNIST test dataset. This experiment shows that the proposed model is robust to input affine transformation.

Models	Trained on AffNIST?	MNIST	AffNIST
Marginal. CNN [28]	Yes	97.82%	86.79%
TransRA CNN[1]	Yes	99.25 %	87.57%
BCN [3]	Mix*	97.5%	91.60%
CNN [23]	No	99.22%	66%
Dynamic-R [23]	No	99.23%	79%
GE-CAPS [16]	No	-	89.10%
SPARSECAPS [22]	No	99%	90.12%
Aff-CapsNet + **No-R**	No	99.23%	**93.21**$_{(\pm 0.65)}$**%**

Table 3: Comparison to state-of-the-art performance on the benchmark task.

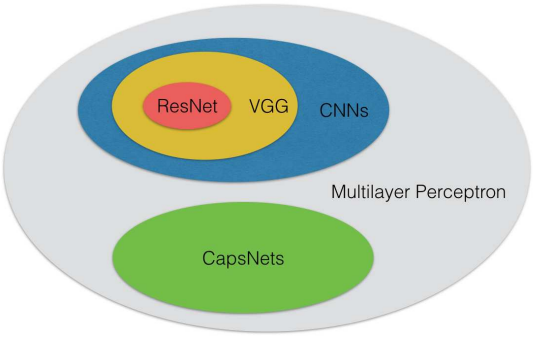


Figure 7: The relationship between different CNN architectures and Capsule Network architectures.

6. Discussion

The difference between the regular CNNs, Aff-Capsnet and CapsNets: Each neuron in the convolutional layer is connected only to a local spatial region in the input. However, each element in a capsule layer (with or without dynamic routing) is connected to all elements of all input capsules. By considering global information, the features extracted by the capsule layer might be more useful for some tasks, e.g., affine-transformed image classification or semantic image segmentation.

What is the difference between a fully connected (FC) layer and the capsule layer without dynamic routing? In an FC layer, each neuron is also connected to all neurons of the preceding layer. Compared with FC layers, convolutional layers show inductive biases, which are Local Connection and Parameter Sharing. Similarly, capsule layers might show a new inductive bias, namely, a new way to combine activations of the preceding layer.

The relationship between CapsNet architectures and CNN architectures is illustrated in Figure 7. CapsNets might be considered as new architectures parallel to CNNs. In the past years, our community has focused on exploring CNN architectures manually or automatically. The figure illustrates that there is "space" outside of the CNN paradigm: CapsNets, or even other unexplored options.

Going Deeper with CapsNets: One way to make CapsNets deep is to integrate advanced techniques of training CNNs into CapsNets. The integration of skip connections [8, 21] and dense connections [11, 20] have been proven to be successful. Instead of blindly integrating more advanced techniques from CNN into CapsNets, it might be more promising to investigate more into the effective components in CapsNets. Our investigation reveals that the dynamic routing procedure contributes neither to the generalization ability nor to the affine robustness of CapsNets. Such conclusion is helpful for training CapsNets on large scale datasets, e.g., the ImageNet 1K dataset [7].

Application of CapsNets to Computer Vision Tasks Besides the object recognition task, CapsNets are also applied to many other computer vision tasks, for examples, object segmentation [14], image generation models [12, 24], and adversarial defense [10]. It is not clear whether routing procedures are necessary for these tasks. If routing is not required here as well, the architectures of CapsuleNets can be integrated into these vision tasks with much less effort.

The Necessity of the Routing Procedure in CapsNets [23] demonstrated many advantages of CapsNets with dynamic routing over CNNs. However, our investigation shows that all the advantages do not disappear when the routing procedure is removed. Our paper does not claim that routing does not have any benefits but rather poses the question to the community: *What is the routing procedure really good for?* If the routing procedure is not necessary for a given task, CapsNets have the chance of becoming an easier-to-use building block.

7. Conclusion

We revisit the dynamic routing procedure of CapsNets. Our numerical analysis and extensive experiments show that neither the generalization ability nor the affine robustness of CapsNets is reduced by removing the dynamic routing procedure. This insight guided us to focus on the CapsNet architecture, instead of various routing procedures, to improve the affine robustness. After exploring the limitation of the CapsNet architecture, we propose Aff-CapsNets, which improves affine robustness significantly using fewer parameters.

Since this work mainly focused on the robustness to affine transformation, we investigate the standard CapsNets with dynamic routings. Other beneficial properties have also been shown in improved CapsNets, like adversarial robustness and viewpoint invariance. Further analysis of these properties will be addressed in future work.

References

[1] Shuhei Asano. *Proposal of transformation robust attentive convolutional neural network*. PhD thesis, Waseda University, 2018.

[2] Mohammad Taha Bahadori. Spectral capsule networks. In *ICML Workshop*, 2018.

[3] Simyung Chang, John Yang, SeongUk Park, and Nojun Kwak. Broadcasting convolutional network for visual relational reasoning. In *ECCV*, pages 754–769, 2018.

[4] Zhenhua Chen and David Crandall. Generalized capsule networks with trainable routing procedure. In *ICML Workshop*, 2019.

[5] Zhenhua Chen, Xiwen Li, Chuhua Wang, and David Crandall. Capsule networks without routing procedures. In *ICLR open review submissions*, 2020.

[6] Jaewoong Choi, Hyun Seo, Suii Im, and Myungjoo Kang. Attention routing between capsules. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[9] Geoffrey E Hinton, Zoubin Ghahramani, and Yee Whye Teh. Learning to parse images. In *Advances in neural information processing systems*, pages 463–469, 2000.

[10] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *ICLR*, 2018.

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[12] Ayush Jaiswal, Wael AbdAlmageed, Yue Wu, and Premkumar Natarajan. Capsulegan: Generative adversarial capsule network. In *ECCV*, pages 0–0, 2018.

[13] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[14] Rodney LaLonde and Ulas Bagci. Capsules for object segmentation. In *International Conference on Medical Imaging with Deep Learning*, 2018.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Jan Eric Lenssen, Matthias Fey, and Pascal Libuschewski. Group equivariant capsule networks. In *Advances in Neural Information Processing Systems*, pages 8844–8853, 2018.

[17] Hongyang Li, Xiaoyang Guo, Bo DaiWanli Ouyang, and Xiaogang Wang. Neural network encapsulation. In *ECCV*, pages 252–267, 2018.

[18] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[19] Inyoung Paik, Taeyeong Kwak, and Injung Kim. Capsule networks need an improved routing algorithm. *ArXiv*, abs/1907.13327, 2019.

[20] Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti R Bathula. Multi-level dense capsule networks. In *Asian Conference on Computer Vision*, pages 577–592. Springer, 2018.

[21] Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, and Ranga Rodrigo. Deepcaps: Going deeper with capsule networks. In *CVPR*, pages 10725–10733, 2019.

[22] David Rawlinson, Abdelrahman Ahmed, and Gideon Kowadlo. Sparse unsupervised capsules generalize better. *arXiv preprint arXiv:1804.06094*, 2018.

[23] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866, 2017.

[24] Raeid Saqur and Sal Vivona. Capsgan: Using dynamic routing for generative adversarial networks. In *Science and Information Conference*, pages 511–525. Springer, 2019.

[25] Dilin Wang and Qiang Liu. An optimization view on dynamic routing between capsules. In *ICLR Worksop*, 2018.

[26] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[27] Suofei Zhang, Quan Zhou, and Xiaofu Wu. Fast dynamic routing based on weighted kernel density estimation. In *International Symposium on Artificial Intelligence and Robotics*, pages 301–309. Springer, 2018.

[28] Jian Zhao, Jianshu Li, Fang Zhao, Xuecheng Nie, Yunpeng Chen, Shuicheng Yan, and Jiashi Feng. Marginalized cnn: Learning deep invariant representations. In *BMVC*, 2017.

Supplementary Material

PaperID 9208

Appx. A: Proof of Equation 6

Given that there are K iterations and the classification loss of is $\mathcal{L}(\boldsymbol{y}, \boldsymbol{t})$, where $\boldsymbol{y} = (\|\boldsymbol{v}_1^{(K)}\|, \cdots, \|\boldsymbol{v}_M^{(K)}\|)$ is the prediction and $\boldsymbol{t}$ the target, the gradients through the routing procedure are

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{u}}_{m|i}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} c_{im}^{(K)} + \sum_{j=1}^{M} \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_j^{(K)}} \frac{\partial \boldsymbol{v}_j^{(K)}}{\partial \boldsymbol{s}_j^{(K)}} \hat{\boldsymbol{u}}_{j|i} \frac{\partial c_{ij}^{(K)}}{\partial \hat{\boldsymbol{u}}_{m|i}} \tag{1}$$

As described in the paper, the coupling coefficients of the Digit Layer are computed as

$$c_{ij}^{(t+1)} = \frac{exp(B_{ij} + \sum_{r=1}^{t} \boldsymbol{v}_j^{(r)} \hat{\boldsymbol{u}}_{j|i})}{\sum_k exp(b_{ik} + \sum_{r=1}^{t} \boldsymbol{v}_k^{(r)} \hat{\boldsymbol{u}}_{k|i})} = \frac{exp(B_{ij})}{\sum_k exp(B_{ik})} \tag{2}$$

where the superscript t is the index of an iteration, and $B_{ik} = b_{ik} + \sum_{r=1}^{t} \boldsymbol{v}_k^{(r)} \hat{\boldsymbol{u}}_{k|i}$.

When unrolling the routing procedure (a factor of the second term in Equation 1), we have

$$\begin{aligned} \frac{\partial c_{ij}^{(K)}}{\partial \hat{\boldsymbol{u}}_{m|i}} = c_{ij}^{(K)}(1 - c_{ij}^{(K)}) \frac{\partial B_{ij}^{(K-1)}}{\partial \hat{\boldsymbol{u}}_{m|i}} \\ + \sum_{k=1 \& k \neq j}^{M} c_{ij}^{(K)} c_{ik}^{(K)} \frac{\partial B_{ij}^{(K-1)}}{\partial \hat{\boldsymbol{u}}_{m|i}} \end{aligned} \tag{3}$$

Since $c_{ik} \in (0, 1)$, by unrolling the above formulation further, we have $\frac{\partial c_{ij}^{(K)}}{\partial \hat{\boldsymbol{u}}_{m|i}} \approx 0$.

At end of the training process, the coupling coefficients are polarized. There are close either to 1 or to 0. When c_{im} is close to 1, the second term in Equation 1 can be ignored. We have

$$\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{u}}_{m|i}} \approx \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} c_{im}^{(K)} \tag{4}$$

When c_{im} is close to 0, We have

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{u}}_{m|i}} \approx & \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} c_{im}^{(K)} \\ & + \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} \hat{\boldsymbol{u}}_{m|i} c_{im}^{(K)}(1 - c_{im}^{(K)}) \frac{\partial B_{im}^{(K-1)}}{\partial \hat{\boldsymbol{u}}_{m|i}} \\ = & \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} c_{im}^{(K)}(1 + \hat{\boldsymbol{u}}_{m|i}(1 - c_{im}^{(K)}) \frac{\partial B_{im}^{(K-1)}}{\partial \hat{\boldsymbol{u}}_{m|i}}) \\ = & C \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}_m^{(K)}} \frac{\partial \boldsymbol{v}_m^{(K)}}{\partial \boldsymbol{s}_m^{(K)}} c_{im}^{(K)} \end{aligned} \tag{5}$$

where C is a constant for the given $\hat{\boldsymbol{u}}_{m|i}$. The constant can be absorbed into the learning rate when propagated back to scale the gradients of network parameters.

Appx. B: Experimental Setting of CapsNet

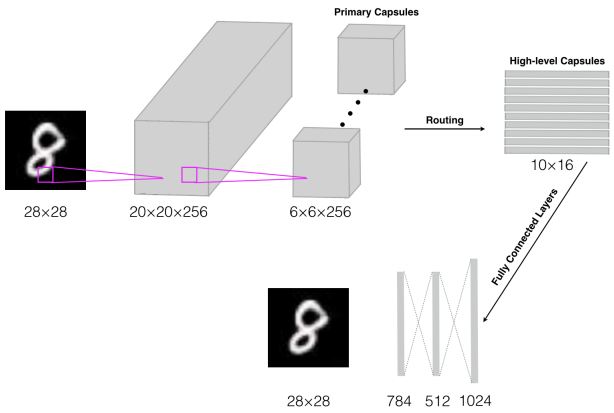


Figure 1. The Architecture of CapsNet used in the Experiments.

Training batch size	128
Training epochs	100
Learning rate	0.001
Routing iterations	3
Reconstruction weight	0.0005
Optimizer	Adam

Table 1. The Hyper-parameters of the Training Process.

Appx. C: Visualizing Computational Graphs

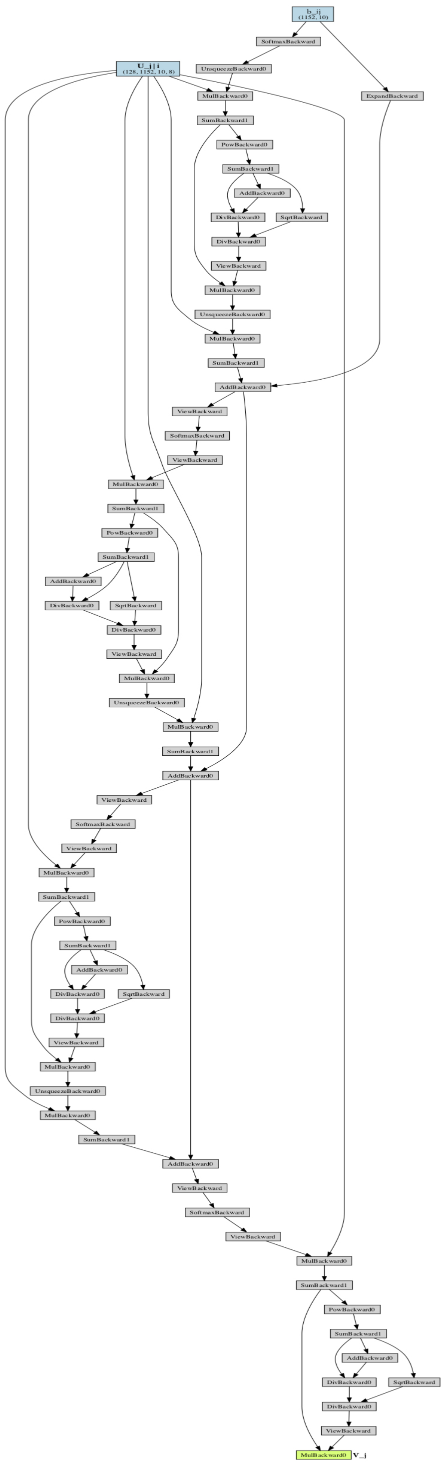


Figure 2. The computational graph of computing $\frac{\partial \hat{v}_j}{\partial \hat{\boldsymbol{u}}_{j|i}}$ where the coupling coefficients are treated as a function value of $\hat{\boldsymbol{u}}_{j|i}$. The gradients are propagated through the iterative routing iterations.

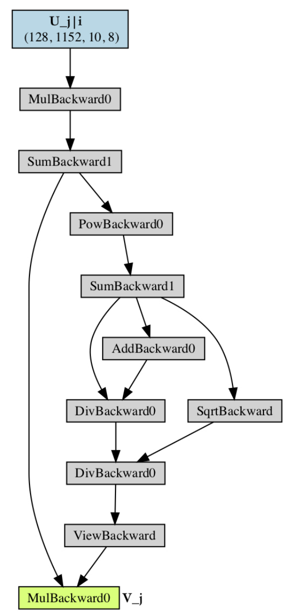


Figure 3. The computational graph of computing $\frac{\partial \hat{v}_j}{\partial \hat{\boldsymbol{u}}_{j|i}}$ where the coupling coefficients are treated as constants in gradent backpropagation.

Chapter 5

Capsule Networks VS. CNNs on the Robustness

Capsule Network is Not More Robust than Convolutional Network

Jindong Gu[1], Volker Tresp[1], Han Hu[2]
University of Munich[1]
Microsoft Research Asia[2]
jindong.gu@outlook.com, volker.tresp@siemens.com, hanhu@microsoft.com

Abstract

The Capsule Network is widely believed to be more robust than Convolutional Networks. However, there are no comprehensive comparisons between these two networks, and it is also unknown which components in the CapsNet affect its robustness. In this paper, we first carefully examine the special designs in CapsNet that differ from that of a ConvNet commonly used for image classification. The examination reveals five major new/different components in CapsNet: a transformation process, a dynamic routing layer, a squashing function, a marginal loss other than cross-entropy loss, and an additional class-conditional reconstruction loss for regularization. Along with these major differences, we conduct comprehensive ablation studies on three kinds of robustness, including affine transformation, overlapping digits, and semantic representation. The study reveals that some designs, which are thought critical to CapsNet, actually can harm its robustness, i.e., the dynamic routing layer and the transformation process, while others are beneficial for the robustness. Based on these findings, we propose enhanced ConvNets simply by introducing the essential components behind the CapsNet's success. The proposed simple ConvNets can achieve better robustness than the CapsNet.

1. Introduction

The Capsule network (CapsNet) [24] was proposed to address the intrinsic limitations of convolutional networks (ConvNet) [14], such as the exponential inefficiency and the lack of robustness to affine transformations. In recent years, It has been sugested that CapsNets have the potential to surpass the dominant convolutional networks in these aspects [24, 8, 21, 3, 2, 16]. However, there lack comprehensive comparisons to support this assumption, and even for some reported improvements, there are no solid ablation studies to figure out which ones of the components in CapsNets are, in fact, effective.

In this paper, we first carefully examine the major differences in design between the capsule networks and the common convolutional networks adopted for image classification. A common convolutional network follows a simple algorithm flow, using a backbone convolutional network to extract image features, a global average pooling layer plus a linear layer to produce the classification logits (or optionally several fully connected layers [13]), and an N-way SoftMax loss to drive the learning. To be better aligned with the capsule (vector) representations, the capsule networks introduce several special components. These components involve (see Fig. 1 for detailed architectures):

- a non-shared transformation module, in which the primary capsules are transformed to execute votes by non-shared transformation matrices;
- a dynamic routing layer to automatically group input capsules to produce output capsules with high agreements in each output capsule;
- a squashing function, which is applied to *squash* the capsule vectors such that their lengths distribute in the range of $[0, 1)$;
- a marginal classification loss to work together with the *squashed* capsule representations;
- a class-conditional reconstruction sub-network with a reconstruction loss, targeting at recovering the original image from the capsule representations. This sub-network acts as a regularization force, in complementary to the classification loss.

Unlike previous studies [24, 8] which usually takes CapsNet as a whole to test its robustness, we instead try to study the effects of each of the above components in their effectiveness on robustness. We consider the three different aspects shown in [24]:

- the robustness to affine transformations,
- the ability to recognizing overlapping digits,
- the semantic representation compactness.

Our investigations reveal that some widely believed benefits of Capsule networks could be wrong:

1. The ConvNets baseline adopted in comparison with CapsNets is weak [24]. Concretely, there is no global average pooling layer before the classification head in this baseline, which sacrifices the ability of spatial invariance to some extent and is harmful for generalization to novel views. In fact, a ConvNet with an additional global average pooling layer can outperform CapsNet by a large margin in the robustness to affine transformation;

2. The dynamic routing actually may harm the robustness to input affine transformation, in contrast to the common belief;

3. The high performance of CapsNets to recognize overlapping digits can be mainly attributed to the extra modeling capacity brought by the transformation matrices.

4. Some components of CapsNets are indeed beneficial for learning semantic representations, e.g., the conditional reconstruction and the squashing function, but they are mainly auxiliary components and can be applied beyond CapsNets.

In addition to these findings, we also enhance common ConvNets by the useful components of CapsNet, and achieve greater robustness. The paper is organized as follows: Sec. 2 introduces the CapsNet and related work. In Sec. 3, we examine the behavior of CapsNets and ConvNets on three kinds of robustness, one by one, and component by component. The last section concludes our work and discusses future work.

2. Background and Related Works

Capsule Network with Dynamic Routing [24]: The CapsNet architecture is shown in Fig. 1. CapsNet first extracts feature maps of shape (C, H, W) from pixel intensities with two standard convolutional layers where C, H, W are the number of channels, the height, and the width of the feature maps, respectively. The extracted feature maps are reformulated as primary capsules $(C/D_{in}, H, W, D_{in})$ where D_{in} is the dimensions of the primary capsules. There are $M = C/D_{in} * H * W$ primary capsules in total. Each capsule $\boldsymbol{u}_i$, a D_{in}-dimensional vector, consists of D_{in} units across D_{in} feature maps at the same location. Each primary capsule is transformed to make a vote with a transformation matrix $\boldsymbol{W}_{ij} \in \mathbb{R}^{(D_{in} \times N * D_{out})}$, where N is the number of output classes and D_{out} is the dimensions of output capsules. The vote is

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{u}_i \boldsymbol{W}_{ij}. \tag{1}$$

The routing mechanism takes all votes into consideration and identify a weight c_{ij} for each vote $\hat{\boldsymbol{u}}_{j|i}$. Concretely, the routing process iterates over the following three steps

$$\begin{aligned} \boldsymbol{s}_j^{(t)} &= \sum_i^N c_{ij}^{(t)} \hat{\boldsymbol{u}}_{j|i}, \\ \boldsymbol{v}_j^{(t)} &= g(\boldsymbol{s}_j^{(t)}), \\ c_{ij}^{(t+1)} &= \frac{\exp(b_{ij} + \sum_{r=1}^{t} \boldsymbol{v}_j^{(r)} \hat{\boldsymbol{u}}_{j|i})}{\sum_k \exp(b_{ik} + \sum_{r=1}^{t} \boldsymbol{v}_k^{(r)} \hat{\boldsymbol{u}}_{k|i})}, \end{aligned} \tag{2}$$

where the superscript t is the index of an iteration starting from 1 and $g(\cdot)$ is a squashing function that maps the length of the vector $\boldsymbol{s}_j$ into the range of $[0, 1)$. The b_{ik} is the log prior probability. The squashing function is

$$\boldsymbol{v}_j = g(\boldsymbol{s}_j) = \frac{\|\boldsymbol{s}_j\|^2}{1 + \|\boldsymbol{s}_j\|^2} \frac{\boldsymbol{s}_j}{\|\boldsymbol{s}_j\|}. \tag{3}$$

The length of the final output capsule $\boldsymbol{v}_j$ corresponds to the output probability of the j-th class. The margin loss function is applied to compute the classification loss

$$\begin{aligned} L_k =& T_k \max(0, m^+ - \|\mathbf{v}_k\|)^2 \\ &+ \lambda(1 - T_k) \max(0, \|\mathbf{v}_k\| - m^-)^2 \end{aligned} \tag{4}$$

where $T_k = 1$ if the object of the k-th class is present in the input. As in [24], the hyper-parameters are often empirically set as $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$.

A reconstruction sub-network reconstructs the input image from all N output capsules with a masking mechanism. The ones corresponding to the non-ground-truth classes are masked with zeros before being transfered to the reconstruction sub-network. Due to the masking mechanism, only the capsule of the ground-truth class is visible for the reconstruction. Hence, the reconstruction process is called class-conditional reconstruction. The reconstruction loss is computed as a regularization term in the loss function.

Capsule Network Follow-Ups: Many routing mechanisms have been proposed to improve the performance of CapsNet, such as Expectation-Maximization Routing [8], Self-Routing [6], Variational Bayes Routing [23], Straight-Through Attentive Routing [1], and Inverted Dot-Product Attention routing [25]. To reduce the parameters of CapsNet, a matrix or a tensor has been used to represent an entity instead of a vector [8, 21]. The size of the learnable transformation matrix can be reduced by the matrix/tensor representations. Another way to improve CapsNets is to integrate advanced modules of ConvNets into CapsNets, e.g., by skip connections [7, 21] and dense connections [11, 18].

Besides, the robustness of CapsNet has also been intensively investigated. Both new routing mechanisms [8] and new architectures [12] can improve the affine transformation robustness. The work [3] achieves the best performance

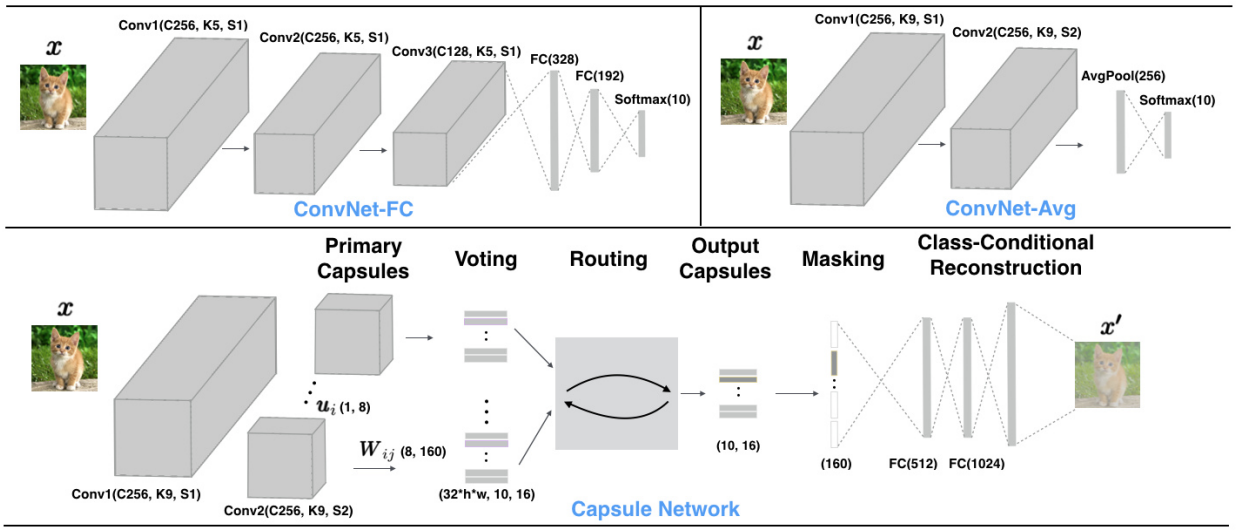


Figure 1: The overview of ConvNet and CapsNet architectures: The ConvNet-FC is a naive ConvNet architecture, while ConvNet-Avg is the one commonly used in image classifications. The CapsNet consists of primary capsule extraction, a transformation process, a routing process, and a class-conditional reconstruction, which is far more complex than ConvNets.

on the transformation robustness benchmark by simply removing the dynamic routing and by sharing the transformation matrix. The work also revealed that the high transformation robustness of CapsNets could not be attributed to the dynamic routing mechanism. The work [4] replaces the dynamic routing with a multi-head attention-based graph pooling approach to achieve better interpretability. The replacement of the routing does not harm the robustness of CapsNet, even though it is the fundamental part of CapsNets. These claims further motivate us to investigate the individual components of CapsNet.

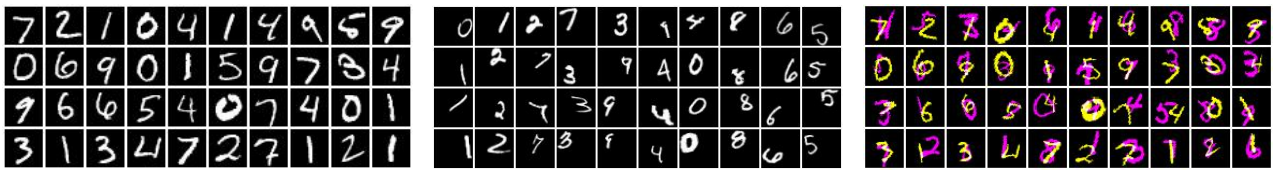
Additionally, CapsNet with new routing mechanisms can achieve high adversarial robustness [6]. However, the work [17] shows CapsNet can be fooled as easily as ConvNet. Recent work shows that the class-conditional reconstruction sub-network of CapsNet is useful to detect adversarial examples [20, 19]. The work [5] designs the first attack method specific for CapsNet, which reduces the robust accuracy and increases the rate to pass the adversarial detection. Due to the attack-defense arms race, it is difficult to draw a solid conclusion on the adversarial robustness of CapsNet. Hence, in this work, we mainly focus on the advantage of CapsNet demonstrated in [24].

3. Empirical Studies on Capsule Network

In this section, we conduct empirical studies on the robustness of CapsNets. Before we dive into the studies, we first introduce the architectures of CapsNets and ConvNets. The CapsNet we focus on in this work is Capsule Networks with dynamic routing [24]. Since the research on CapsNets is still at a primary stage, the work [24] compares their CapsNet with a LeNet-type ConvNet [14], called ConvNet-FC. The ConvNet-FC and CapsNet are illustrated in Fig. 1 on 28×28 MNIST images. The notation Conv(C, K, S) stands for a convolutional layer where C, K, S are the number of channels, the kernel size, and the stride size, respectively. FC(N) is a fully connected layer where N is the number of output units. All Conv and FC are followed by a ReLU activation function.

ConvNet-FC: The simple ConvNet baseline used in [24] is Conv(256, 5, 1) + Conv(256, 5, 1) + Conv(128, 5, 1) + FC(328) + FC(192) + Softmax(10). The three standard convolutional layers and two fully connected layers are applied to extract features from input images. An N-way Softmax is applied to obtain the output distribution. During training, cross-entropy loss is typically applied.

CapsNet: The CapsNet with Dynamic Routing in [24] is Conv(256, 9, 1) + Conv(256, 9, 2) + Dynamic Routing, followed by a reconstruction sub-network, FC(512) + FC(1024) + FC(28×28). The feature maps are computed with the two standard convolutional layers. The extracted feature maps (256, H, W) is reshaped into primary capsules (32*H*W, 8) where H and W are the height and width of feature maps. The primary capsules are squashed by the squashing function in Equation (3) and then transformed to make votes with the learned transformation matrices (32*H*W, 8, 160). The vote of each primary capsule is 160-dimensional. The dynamic routing in Equation (2) is applied to the votes to identify their weights. The output of the dynamic routing is 160-dimensional, i.e. representing 10 16-dimensional output capsules. The squashing function in Equation (3) is applied to output capsules to map their



(a) MNIST dataset (b) AffNIST dataset (c) MultiMNIST dataset

Figure 2: Visualization of datasets: While MNIST dataset corresponds to standard hand-written digits, AffNIST dataset consists of affine-transformed MNIST images. MultiMNIST dataset consists of images with two overlapping digits. In the figure, the two overlapping digits are marked with two different colors, i.e., yellow and magenta.

lengths into [0, 1). The length of an output capsule is interpreted as a class output probability. In the training process, the margin loss in Equation (4) is applied as the classification loss. In the class-conditional reconstruction process, a masking mechanism is applied to output capsules, where the capsules, corresponding to non-ground-truth classes, are masked with zeros. The input image is reconstructed from the masked output capsules. The reconstruction loss is used to regularize the training process.

By comparing the two networks, we can summarize 5 major differences between ConvNets and CapsNets, namely, a transformation process, a dynamic routing layer, a squashing function, the use of a marginal loss instead of a cross-entropy loss, and a class-conditional reconstruction regularization. With these differences, CapsNet outperforms ConvNet-FC in terms of robustness to affine transformation and overlapping digits recognition as well as in learning compact semantic representations. In this section, we will investigate these advantages one by one. In each of our studies, we attempt to answer the following questions:

1. Do ConvNet-FC and CapsNets perform differently?
2. Which components of CapsNets make the difference?
3. How bridge the gap between the two networks?

3.1. Robustness to Input Affine Transformation

Settings: To examine the transformation robustness of both models, we use the popular benchmark [24, 3] where models are trained on MNIST and tested on AffNIST. In AffNIST [24], the original 28×28 MNIST images are first padded with 6 pixels to 40×40 image and then affine transformed, namely, rotation within 20 degrees, shearing within 45 degrees, scaling from 0.8 to 1.2 in both vertical and horizontal directions, and translation within 8 pixels in each direction. In the training dataset, the 28×28 MNIST images are placed randomly on a black background of 40×40 pixels without further transformation. The image examples are visualized in Fig. 2. The performance on both MNIST and AffNIST test datasets is reported. All scores are averaged over 5 runs across this paper.

Besides ConvNet-FC and CapsNet, we include the state-of-the-art model on the benchmark in this experiment, namely, Aff-CapsNet. It simplifies CapsNet by removing dynamic routing and sharing the transformation matrix in the transformation process.

Following [24, 3], the Adam optimizer is used to train the models with an initial learning rate of 0.001 and a batch size of 128. In CapsNet, the reconstruction loss is scaled down by 0.0005 so that it does not dominate the margin loss during training. It is hard to decide which model is more robust to affine transformations when they achieved different accuracy on untransformed examples. To eliminate this confounding factor, we stopped training the models when they achieve similar performance (i.e. about 99.22%), following [24].

Models	#Para.	MNIST	AffNIST
GE-CapsNet [15]	-	98.42	89.10
SPARSECAPS [22]	-	99	90.12
SCAE [12]	-	98.5	92.21
EM-CapsNet [8]	-	99.2	93.1
ConvNet-FC [24]	35.4M	99.22	66
CapsNet [24]	13.5M	99.23	79
CapsNet-NoR [3]	13.5M	99.22	81.81
Aff-CapsNet-DR [3]	7.5M	99.22	89.03
Aff-CapsNet [3]	7.5M	99.23	93.21
ConvNet-Avg	5.3M	99.22	**94.11**

Table 1: Comparison on the transformation robustness benchmark: The generalization performance to AffNIST is reported when models achieve similar performance on MNIST test dataset. Our simple ConvNet-Avg is more robust than CapsNet to input affine transformations.

Results and Analysis: The performance is reported in Tab. 1. We can observe that there is a gap between ConvNet-FC and CapsNet. As reported in [24, 3], the CapsNet outperforms ConvNet-FC, and Aff-CapsNet outperforms CapsNet. We take Aff-CapsNet (a simplified CapsNet) as a baseline and conduct further ablation studies on the components of CapsNet in Tab. 2. We report the model test performance on both un-transformed MNIST test images and novel affine-transformed ones. No early stopping is applied in the ablation studies.

Factors	Routing	Shared TransM	Squash-fn	Reconstion	Loss	Train-MNIST	Test-MNIST	Test-AffNIST
Routing	NoR	✓	✓	✓	MarginLoss	100	99.29(± 0.13)	93.55(± 1.47)
	DR	-	-	-	-	100	99.21(± 0.31)	**90.07**(± 0.98)
Shared TransM	NoR	✓	✓	✓	MarginLoss	100	99.29(± 0.13)	93.55(± 1.47)
	-	×	-	-	-	100	98.98(± 0.04)	**80.49**(± 0.34)
Squash-fn	NoR	✓	✓	✓	MarginLoss	100	99.29(± 0.13)	93.55(± 1.47)
	-	-	×	-	-	99.75	97.93(± 0.13)	**80.42**(± 0.39)
Reconstruction	NoR	✓	✓	conditional	MarginLoss	100	99.29(± 0.13)	93.55(± 1.47)
	-	-	-	normal	-	100	99.43(± 0.28)	95.09(± 0.56)
	-	-	-	×	-	100	99.39(± 0.26)	93.49(± 0.46)
Loss	NoR	✓	✓	✓	MarginLoss	100	99.29(± 0.13)	93.55(± 1.47)
	-	-	-	-	CE Loss	100	99.27(± 0.05)	94.67(± 0.43)

Table 2: The performance on MNIST training dataset, MNIST test dataset, and AffMNIST test dataset are reported, respectively (in percentage %). Dynamic Routing (DR) and Margin loss are even harmful to the transformation robustness, while the squashing function (Squash-fn) and the shared transformation matrix (Shared TransM) are beneficial.

The transformation process can be seen as a fully connected (FC) layer since the transformation matrices therein are equivalent to the parameters of an FC layer. *Why is Aff-CapsNet more robust than CapsNet?* The transformation robustness of CapsNet can be improved by sharing the transformation matrix. When the transformation matrix is shared and no routing is applied in Aff-CapsNet, the transformation process is essential to conduct group 1×1 convolutional operations, global average pooling operations, and an average operation on the pooling results of different groups. A further study shows that the number of groups has no effect on the robustness (see Supplement A). Hence, we attribute the superior performance of the sharing transformation matrix to the global average pooling operation. *Why is CapsNet more robust than ConvNet-FC?* The ConvNet-FC has two fully connected layers, while CapsNet has a functionally similar one. Another difference between them is the kernel size. Our study shows that large kernels are also beneficial to achieve transformation robustness (see Tab. 3). This argument also echoes our claim above. Namely, both global average pooling and large kernels improve the robustness by increasing receptive fields.

In Tab. 2, the dynamic routing is even harmful to the transformation robustness, which is also supported by the Tab. 1. In addition, when no squashing function is applied, CapsNet has to regress the capsule length to extreme values (e.g., 0 or 1), which is a hard task and leads to unsatisfying performance (even on the training dataset). The margin loss can slightly weaken the transformation robustness of CapsNet, while reconstruction makes no difference to it. The non-conditional reconstruction slightly improves the performance since it updates all capsules in each training iteration.

Based on our findings, we propose a new simple ConvNet baseline, called **ConvNet-Avg**. It starts with the two convolutional layers and terminates with a global average pooling and an output layer, which is also a common architecture used in image classification. The cross-entropy loss is applied to train the model. To make a fair comparison, we use the same convolutional layers as in CapsNet and Aff-CapsNet, namely, Conv(256, 9, 1) + Conv(256, 9, 2) + Global AvgPool + FC(10) (see Fig. 1). It is hard to decide which model is better at generalizing to affine transformations when they achieved different accuracy on untransformed examples. We follow previous work and stop training the models when they achieve similar test performance (99.22%). As shown in Tab. 1, our simple ConvNet-Avg achieves slightly better performance with fewer parameters.

Conclusions: 1) Compared to ConvNet-FC, CapsNet achieves better test performance with fewer parameters on AffNIST. We attribute the gap to the kernel size. 2) Dynamic routing can harm the transformation robustness of CapsNet. When the routing is removed, the uniform average of votes (i.e., NoR) aggregates the global information better. 3) Our baseline ConvNet-Avg outperforms CapsNets significantly. It consists of only convolutional layers and a global average pooling layer, and no advanced component from SOTA ConvNets. The simplicity of ConvNet-Avg indicates that CapsNets are even less robust to affine transformation than ConvNets in a fair comparison.

3.2. Recognizing overlappping digits

Settings: The work [24] shows that the CapsNet is able to recognize overlapping digits by segmenting them. To check this property, we use the MultiMNIST dataset, which is generated by overlaying a digit on top of another digit but from a different class. Specifically, a 28×28 MNIST image with a digit is first shifted up to 4 pixels in each direction resulting in a 36×36 image. The resulting image is overlaid to another image from different classes but the same set (training dataset or test dataset). For each image in MNIST, we can create N (from 1 to 1K) images. See Fig. 2c for some examples from data.

Kernels	K(3, 3)			K(5, 5)			K(7, 7)			K(9, 9)			K(11, 11)		
Models	#Para.	A_{std}	A_{aff}	#Para.	A_{std}	A_{aff}	#Para.	A_{std}	A_{aff}	#Para.	A_{std}	A_{aff}	#Para.	A_{std}	A_{aff}
CapsNet	16.1M	96.31	61.36	14.4M	98.18	70.34	13.5M	98.74	75.82	13.5M	99.26	79.12	14.3M	99.1	86.79
ConvNet-FC	49.5M	96.54	64.57	35.4M	99.23	66.08	25.2M	99.03	66.76	18.8M	-	-	16.19M	-	-
ConvNet-Avg	0.59M	97.14	**86.58**	1.70M	98.58	**90.95**	3.23M	99.1	**92.31**	5.30M	99.22	**94.11**	7.96M	99.34	**90.58**

Table 3: The effect of the kernel sizes on the transformation robustness of different models: Both standard accuracy (A_{std}) and the generalization accuracy (A_{aff}) on transformed data are reported. The large kernels make positive contributions to the transformation robustness. When the same kernel size is applied, ConvNet-Avg outperforms both ConvNet-FC and CapsNet.

The classification of an image with overlapping digits is correct if both digits are correctly classified (the top 2 output classes match the ground truth). The margin loss can be applied to compute the classification loss. In the ConvNet baselines, the sigmoid function is applied to logits instead of softmax to obtain output probabilities since this is a multi-target classification task, and the binary cross-entropy loss is applied to compute the classification loss.

In the training process, the CapsNet is first applied to the overlapping digits to obtain output capsules. During reconstruction, a ground-truth class is picked at a time, and the capsule corresponding to the class is kept for the reconstruction while others are masked with zeros. In other words, we run the reconstruction sub-network twice, each for one digit. The reconstruction loss can be computed similarly since the images of individual digits are available.

Results and Analysis: The overlapping digit recognition performance is reported in Tab. 4 where the individual components of CapsNets are ablated. The reconstruction sub-network helps to improve the recognition performance. However, it does not have to be class-conditional. The reconstruction loss regularizes the training process so that the information about both digits is encoded in features and high-level capsules. The margin loss can be directly applied to a multi-target classification task, which outperforms the standard binary cross-entropy loss. Both the reconstruction and the margin loss can be applied to enhance a ConvNet.

When a vector representation is applied, the squashing function plays an important role. When applying the squashing function to the primary capsules, the feature maps are group-wise normalized. The information is communicated across different channels, which can help to better disentangle overlapping digits. Additionally, CapsNet has to regress the non-squashed capsule length to certain values. Since the regression task is hard, CapsNets achieve unsatisfying performance on both the training and test dataset. The analysis echoes the one in Sec. 3.1.

The dynamic routing process identifies the weights for votes, which results in a higher modeling capacity than the uniform averaging operation on votes. Other components that support the CapsNet's modeling capacity are the transformation matrices. When a shared transformation matrix is applied, the model performance drops dramatically. We check the ConvNet-Avg on this task and observe that CapsNet outperforms ConvNet-Avg significantly. The reason behind this is that the global pooling operation can be harmful for recognizing overlapping digits since it aggregates a feature map into a single unit. The convolutional layer itself is not able to disentangle the overlapping digits into different feature maps. In CapsNet, the transformation process acts as a fully connected layer, which avoids the global average pooling. Hence, we argue that the high modeling capacity is the essential reason why CapsNet performs well on the overlapping digits recognition task.

Figure 3: ConvNet-FC(LK) outperforms CapsNet on MultiMNIST dataset with different data sizes.

FC layers in ConvNet-FC can maintain richer information (features at all locations) for distinguishing overlapping digits. Note that the baseline ConvNet-FC in [24] has a smaller kernel size than in CapsNet. Hence, we propose to apply ConvNet-FC with large kernels (ConvNet-FC(LK)) to this overlapping digits recognition task. In ConvNet-FC(LK), we also reduce the units of fully connected layers to save parameters so that it can be compared to CapsNets. When the same large kernel is applied, ConvNet-FC(LK) outperforms the CapsNet and sets a new SOTA on this benchmark (97.11% vs. 95.18%). When different training data sizes and different kernel sizes are applied in the experiments, the simple ConvNet-FC(LK) outperforms the CapsNet consistently (See Fig. 3 and Supplement B).

Conclusions: 1) All the components contribute to the ability of CapsNet to recognize overlapping digits. 2) The transformation process with a non-shared transformation matrix and a dynamic routing to weight votes bring high modeling capacity, which essentially supports the high performance of CapsNet in this task. 3) The simple ConvNet-FC(LK) with similar parameters performs better than CapsNet on this benchmark, which indicates that CapsNet is not more robust than ConvNet to recognize overlapping digits.

Factors	Routing	Shared TransM	Squash-fn	Reconstion	Loss	Train-MultiMNIST	Test-MultiMNIST
Routing	DR	×	✓	✓	MarginLoss	94.03	93.26(± 0.24)
	NoR	-	-	-	-	90.28	90.07(± 0.29)
Shared TransM	DR	×	✓	✓	MarginLoss	94.03	93.26(± 0.24)
	-	✓	-	-	-	86.92	**86.44**(± 0.37)
Squash-fn	DR	×	✓	✓	MarginLoss	94.03	93.26(± 0.24)
	-	-	×	-	-	87.71	**87.24**(± 0.53)
Reconstruction	DR	×	✓	conditional	MarginLoss	94.03	93.26(± 0.24)
	-	-	-	normal	-	93.83	93.19(± 0.30)
	-	-	-	×	-	90.28	90.17(± 0.26)
Loss	DR	×	✓	✓	MarginLoss	94.03	93.26(± 0.24)
	-	-	-	-	BCE Loss	91.64	91.19(± 0.35)

Table 4: The ablation study on components of CapsNet: The performance of models trained on 6M overlapping digits. All individual components make positive contributions to the ability to recognize overlapping digits. The transformation matrices contribute the most; the performance drops dramatically if a shared transformation matrix is applied.

3.3. Semantic Capsule Representations

Settings: In CapsNets, when a single element in a capsule is perturbated, the reconstructed images are visually changed correspondingly [24], see Fig. 4d. The visual changes often correspond to human-understandable semantic object variations. In this experiment, we investigate which components support the semantic representations. Since this property is mainly demonstrated by a reconstruction sub-network, we introduce three models below:

ConvNet-CR: This ConvNet baseline has the same number of parameters as in CapsNet and the same reconstruction sub-network. Its architecture is Conv(256, 9, 1) + Conv(256, 9, 2) + FC(160), where 160 corresponds to the dimensions of output capsules and the parameters in FC(160) corresponds to the non-shared transformation matrices of CapsNet. The 160 activations are grouped into 10 groups where each group corresponds to an output capsule. The sum of 16 activations in each vector corresponds to a logit. The sigmoid function is applied to each logit to obtain the output probability. The reconstruction sub-network reconstructs the input from (the 160 activations) with a masking mechanism, similar to that in CapsNet.

ConvNet-R: In this baseline, an output layer FC(10) is built on the 160 activations of ConvNet-CR instead of grouping them. The reconstruction sub-network of ConvNet-CR reconstructs the input from the 160 activations directly, without the masking mechanism.

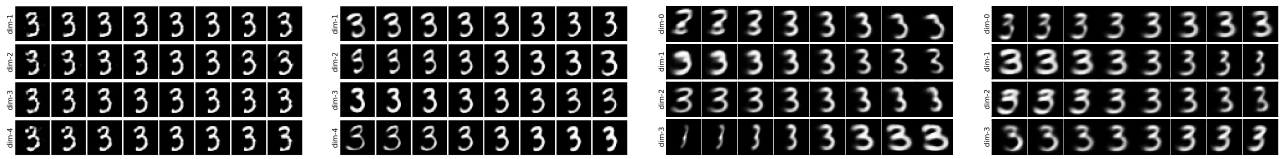
ConvNet-CR-SF: This baseline equips ConvNet-CR with the squashing function in Equation (3). The feature maps from Conv(256, 9, 2) are mapped into vectors with the same shape of primary capsules, and the vectors are squashed. Each element of the vectors is fully connected to 160 units of the next layer. The 160 activations are grouped to obtain the 10 output vectors. The vectors are similarly squashed so that their lengths stand for the output probability of the corresponding class. This baseline is equivalent to CapsNet without a routing mechanism (CapsNet-NoR).

In CapsNet, several units can correspond to a similar semantic concept. An interesting question to investigate is, what percentage of neurons strongly react to changes of a given latent factor. We propose a metric to evaluate such compactness. Given a latent factor z (e.g. rotation) and an image X, we compute the semantic compactness score with the following steps:

1. Creating a list of images with different rotation degrees;
2. Obtaining their representation vectors via forward inferences (the vectors of ground-truth classes are kept);
3. Computing the variance of the vectors in each dimension $\boldsymbol{Var}$ and normalize them by their sum $\boldsymbol{Var}_n$;
4. Computing the KL divergence between the normalized variance values $\boldsymbol{Var}_n$ and a uniform prior.

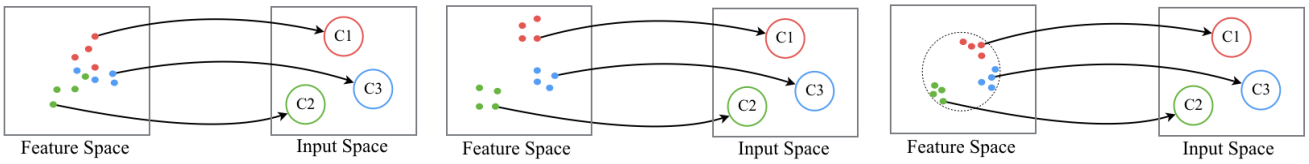
The compactness score is averaged over the whole dataset. The higher the score is, the more compact the semantic representation becomes. The intuition behind the score is that, if only one unit changes when images are rotated, the normalized variance will be one-hot, and the relative entropy to uniform prior is the maximum.

Results and Analysis: After training, we perform the capsule perturbation experiments on the 160 activations, as in [24]. In CapsNet, we tweak one dimension of capsule representations by intervals of 0.05 in the range [-0.2, 0.2]. The reconstructed images are visualized in Fig. 4d. The semantic changes of images can be observed, e.g., the rotation and the stroke thickness. We find that the reconstructed images in ConvNets stay almost unchanged visually when perturbing the corresponding activation with the same range. The observation can be caused by the too-small perturbation range for the unit activations. Hence, we increase the range gradually until the reconstructed image cannot be recognized where we reach the range of [-8, 8]. The reconstructed images are shown in Fig. 4. In ConvNet-R, the



(a) on ConvNet-R (b) on ConvNet-CR (c) on ConvNet-CR-SF (d) on CapsNet

Figure 4: The reconstructed images are shown when a single unit is perturbed. The reconstruction only helps when the class-conditional masking mechanism is applied. The squashing function improves the visual response further.



(a) Reconstruction in ConvNet-R (b) Reconstruction in ConvNet-CR (c) Reconstruction in ConvNet-CR-SF

Figure 5: The reconstruction from feature space to the input space: In ConvNet-R, the capsule representations of different classes are entangled in feature space; the ones in ConvNet-CR are clearly separated due to the class-conditional masking mechanism. When a squashing function is applied to squash the vector, the representations live within a manifold. The representation constraints improve the network's ability to extrapolate object variations.

Datasets	MNIST					
Factors	Rotation	Trans-X	Trans-Y	Scale	Shear-X	Shear-Y
ConvNet-R	0.0003	0.0016	0.0009	0.0004	0.0003	0.0007
ConvNet-CR	0.0028	0.0038	0.0032	0.0052	0.0058	0.0022
ConvNet-CR-SF	**0.0325**	**0.2010**	**0.3192**	**0.0146**	**0.0476**	**0.0506**
CapsNet	0.0031	0.0107	0.0464	0.0026	0.0098	0.0021

Table 5: The representation compactness: The class-conditional reconstruction and the squashing function improve the compactness, while dynamic routing reduces it.

semantics of reconstructed images is not sensitive to all individual dimensions in Fig. 4a. In ConvNet-CR, where the class-conditional reconstruction is applied, the changes of representation unit also cause the semantic changes of reconstructed images in Fig. 4b. When the squashing function is applied, the representations in ConvNet-CR-CF strongly react to the perturbations in Fig. 4c.

Both the class-conditional reconstruction mechanism and the squashing function can help ConvNets to learn meaningful semantic representations. The two components characterize the function learned by the reconstruction sub-network, which maps representations from feature space back to input space. We illustrate the characteristics of these functions in Fig. 5, using an example with a 2D input space and 3 output classes. The ConvNet-R reconstructs inputs from the features that are entangled to some degree. In ConvNet-CR, the features of different classes are perfectly separated since the features are class-conditional. The ConvNet-CR-CF constrains the feature space further by squashing the vectors so that they live inside a manifold. We also report the compactness score of each model in Tab. 5. We speculate that it is these constraints that improve the representation's compactness. More experiments on the FMNIST dataset can be found in Supplement C.

Conclusions: Both the class-conditional reconstruction and the squashing function help CapsNet learn meaningful semantic representations, while dynamic routing is even harmful. The two components can be integrated into ConvNets, where ConvNet-CR-SF learns better semantic compact representations than CapsNets.

4. Conclusion

We reveal 5 major differences between CapsNets and ConvNets and study 3 properties of CapsNets. We show that dynamic routing is harmful to CapsNets in terms of transformation robustness and semantic representations. In each presented task, a simple ConvNet can be built to outperform the CapsNet significantly. We find that there is no single ConvNet that can outperform CapsNet in all cases. Hence, we conclude that *CapsNets with dynamic routing are not more robust than ConvNets*. We leave further explorations for future work, e.g., concerning different datasets, and other properties of CapsNets, and other CapsNets.

The dynamic routing aggregates information from low-level entities into high-level ones. The aggregation can be also be done by a graph pooling operation [4]. In ConvNets, the relationship between low-level entities is also explored in aggregation [9, 10]. More aggregation approaches wil be explored in future work.

References

[1] Karim Ahmed and Lorenzo Torresani. Star-caps: Capsule networks with straight-through attentive routing. In *Advances in Neural Information Processing Systems*, pages 9101–9110, 2019.

[2] Fabio De Sousa Ribeiro, Georgios Leontidis, and Stefanos Kollias. Introducing routing uncertainty in capsule networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[3] Jindong Gu and Volker Tresp. Improving the robustness of capsule networks to image affine transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7285–7293, 2020.

[4] Jindong Gu and Volker Tresp. Interpretable graph capsule networks for object recognition. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[5] Jindong Gu, Baoyuan Wu, and Volker Tresp. Effective and efficient vote attack on capsule networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[6] Taeyoung Hahn, Myeongjang Pyeon, and Gunhee Kim. Self-routing capsule networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7658–7667, 2019.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 770–778, 2016.

[8] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations (ICLR)*, 2018.

[9] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[10] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019.

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4700–4708, 2017.

[12] Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. In *Advances in Neural Information Processing Systems*, pages 15512–15522, 2019.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Jan Eric Lenssen, Matthias Fey, and Pascal Libuschewski. Group equivariant capsule networks. In *Advances in Neural Information Processing Systems*, pages 8844–8853, 2018.

[16] Vittorio Mazzia, Francesco Salvetti, and Marcello Chiaberge. Efficient-capsnet: Capsule network with self-attention routing. *arXiv preprint arXiv:2101.12491*, 2021.

[17] Felix Michels, Tobias Uelwer, Eric Upschulte, and Stefan Harmeling. On the vulnerability of capsule networks to adversarial attacks. 2019.

[18] Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti R Bathula. Multi-level dense capsule networks. In *Asian Conference on Computer Vision*, pages 577–592. Springer, 2018.

[19] Yao Qin, Nicholas Frosst, Colin Raffel, Garrison Cottrell, and Geoffrey Hinton. Deflecting adversarial attacks. *arXiv preprint arXiv:2002.07405*, 2020.

[20] Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell, and Geoffrey Hinton. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. In *International Conference on Learning Representations (ICLR)*, 2020.

[21] Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, and Ranga Rodrigo. Deepcaps: Going deeper with capsule networks. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10725–10733, 2019.

[22] David Rawlinson, Abdelrahman Ahmed, and Gideon Kowadlo. Sparse unsupervised capsules generalize better. *arXiv preprint arXiv:1804.06094*, 2018.

[23] Fabio De Sousa Ribeiro, Georgios Leontidis, and Stefanos D Kollias. Capsule routing via variational bayes. In *AAAI*, pages 3749–3756, 2019.

[24] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems (NeurIPS)*, pages 3856–3866, 2017.

[25] Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov. Capsules with inverted dot-product attention routing. In *International Conference on Learning Representations (ICLR)*, 2020.

Capsule Network is Not More Robust than Convolutional Network

(Supplementary Materials)

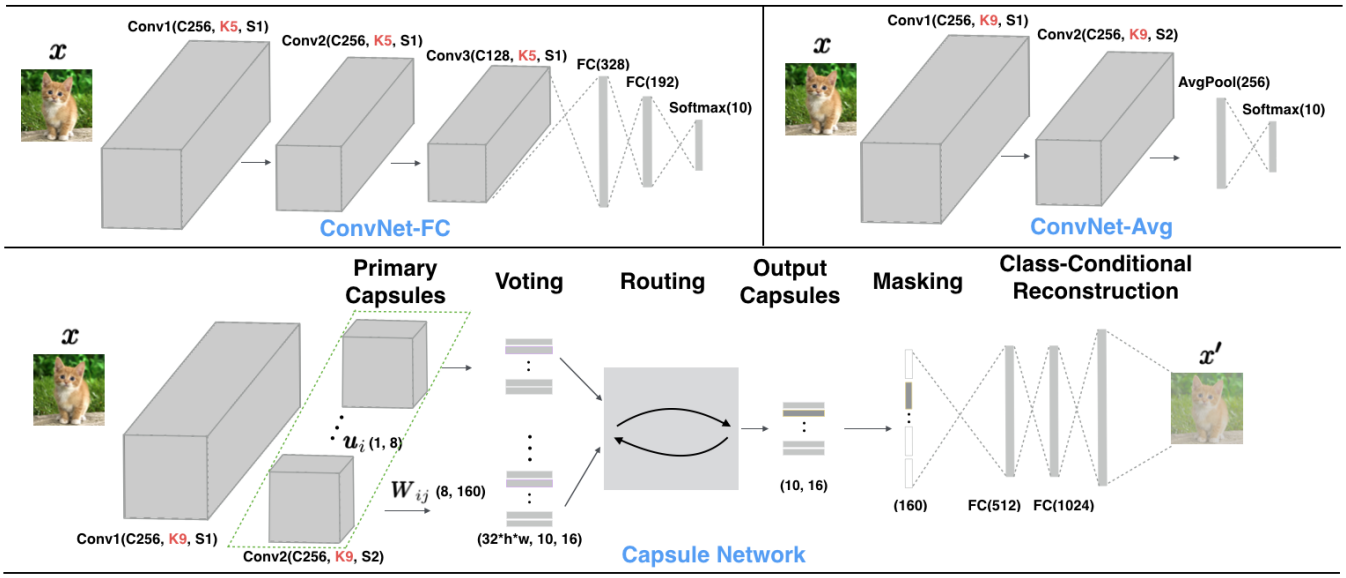


Figure 1: The overview of ConvNet and CapsNet architectures: The kernel sizes are marked with red color, and the capsule types (the groups) are marked with a dashed green box.

1. Supplement A: the effect of the number of capsule groups and the capsule sizes

In Section 3.1 of the paper, we study the effect of components of CapsNet on its affine transformation robustness. This supplement section shows the effect of architecture configurations on the transformation robustness, namely, the number of capsule dimensions (capsule size) and the number of capsule groups (marked with a green box in Fig. 1).

Factors	Num-groups	Caps-size	Routing	SharedM	Squash-fn	Reconstion	Loss	Test-MNIST	Test-AffNIST
Num-groups	1	8	NoR	✓	squash	✓	MarginLoss	99.26(± 0.27)	93.73(± 0.51)
	2	-	-	-	-	-	-	99.21(± 0.44)	92.92(± 0.62)
	8	-	-	-	-	-	-	99.45(± 0.38)	94.03(± 0.31)
	32	-	-	-	-	-	-	99.32(± 0.23)	93.55(± 0.39)
Caps-size	32	8	NoR	✓	squash	✓	MarginLoss	99.28(± 0.33)	93.55(± 0.61)
	16	16	-	-	-	-	-	99.24(± 0.31)	93.99(± 0.24)
	8	32	-	-	-	-	-	99.19(± 0.49)	94.01(± 0.37)

Table 1: The CapsNets with various architectures are trained only on MNIST dataset. The performance on MNIST training dataset, MNIST test dataset, and AffMNIST test dataset are reported, respectively (in percentage %). When CapsNet architecture is configured differently, the corresponding performance is reported. Given the variance, the number of capsule groups and the capsule size have no effect on the transformation robustness of CapsNet.

The performance on MNIST training data, MNIST test data, and AffNIST test data are reported in Tab. 1. The different architecture configurations with the same parameters make no difference in the generalization performance, given the variance. This study shows that the number of capsule dimensions and the number of capsule groups have no effect on the generalization ability of CapsNet to input affine transformations.

2. Supplement B: the effect of the kernel sizes on the ability to recognize overlapping digits

This study investigates the effect of the kernel sizes on the overlapping digits recognization ability of different models, such as ConvNet-FC and CapsNet. The performance of the models with different kernel sizes is reported in Tab. 2. To be noted that the model size of ConvNet-FC becomes smaller when large kernels are applied. The reason behind this is that the large kernels lead to smaller feature maps in case of no padding, which further leads to smaller units in the fully connected layer in ConvNet-FC. In CapsNet, the smaller feature maps lead to smaller transformation matrices.

Kernels	K(3, 3)		K(5, 5)		K(7, 7)		**K(9, 9)**		K(11, 11)	
Models	#Para.	A_{std}	#Para.	A_{std}	#Para.	A_{std}	#Para.	A_{std}	#Para.	A_{std}
CapsNet	13.0M	76.01	11.6M	78.92	11.1M	79.65	11.4M	80.22	12.5M	81.97
ConvNet-FC	38.7M	**85.23**	26.7M	**85.78**	18.5M	**85.77**	14.1M	**85.19**	13.5M	**85.34**

Table 2: The effect of the kernel sizes on the overlapping digits recognization ability of different models: The application of large kernels can improve the model's ability to recognize overlapping digits. ConvNet-FC outperforms CapsNet, even when the same model size is kept.

In the table, given a kernel size and the data size ($\times 10$), we report both the model size and the accuracy to classify overlapping digits of each model. ConvNet-FC outperforms CapsNet on this overlapping digits recognization task when the same kernel size is applied. Especially, when the kernel size 9×9 is applied, the model sizes of ConvNet-FC and CapsNet are similar, and ConvNet-FC outperforms CapsNet by 3.37%.

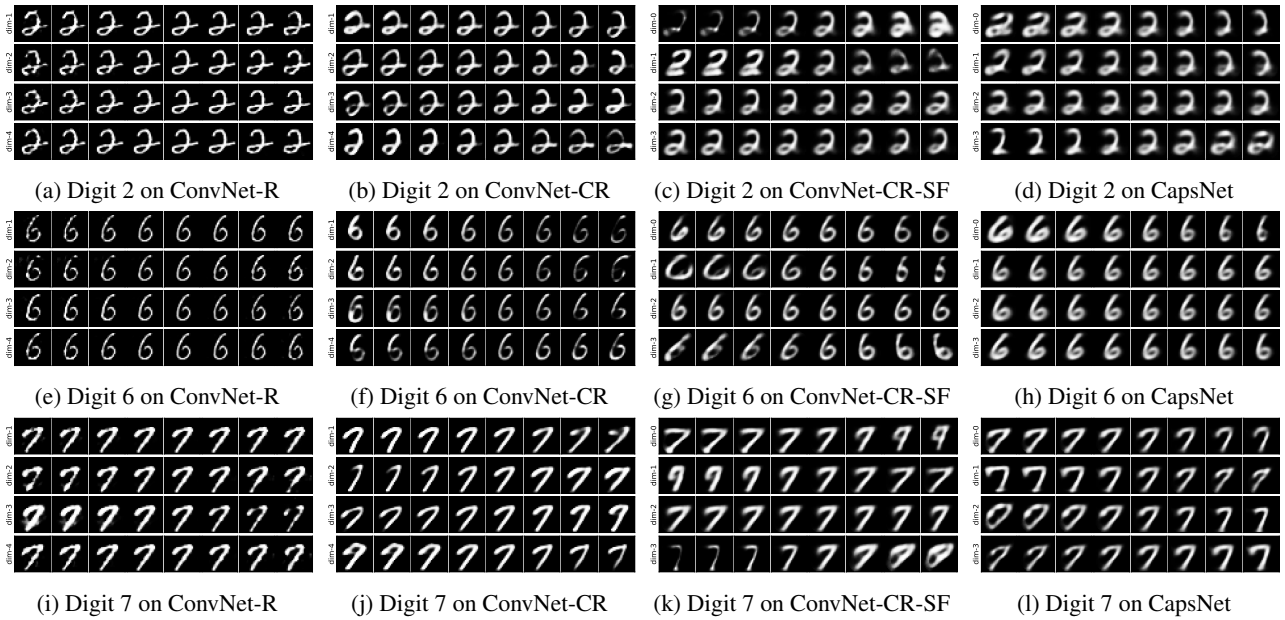
3. Supplement C: Semantic Representations

In CapsNets, when a single element of the vector representation is perturbated, the reconstructed images are also visually changed correspondingly. We conduct the same experiment on different models we build, namely, ConvNet-R, ConvNet-CR, and ConvNet-CR-SF as well as CapsNet. The more figures on MNIST dataset are shown in Fig. 2.

In addition, we also verify our claims on FMNIST dataset. Similarly, we visualize the reconstructed images under different perturbations in Fig. 3. We also report the semantic compactness score of the learned representations in Tab. 3. Given the perturbation range, we also reduce the perturbation interval to show more intermediate images in Fig. 4 and Fig. 5. All the visualizations, as well as the table, show consistent results with the ones on MNIST. Namely, both the class-conditional reconstruction mechanism and the squashing function can help ConvNet learn meaningful semantic representations.

Datasets	MNIST						FMNIST					
Factors	Rotation	Trans-X	Trans-Y	Scale	Shear-X	Shear-Y	Rotation	Trans-X	Trans-Y	Scale	Shear-X	Shear-Y
CNN-R	0.0003	0.0016	0.0009	0.0004	0.0003	0.0007	0.0005	0.0004	0.0006	0.0005	0.0009	0.0002
CNN-CR	0.0028	0.0038	0.0032	0.0052	0.0058	0.0022	0.0038	0.0019	0.0012	0.0016	0.0042	0.0031
CNN-CR-SF	**0.0325**	**0.2010**	**0.3192**	**0.0146**	**0.0476**	**0.0506**	**0.0062**	**0.0078**	**0.0233**	**0.0092**	**0.0074**	**0.0159**
CapsNet	0.0031	0.0107	0.0464	0.0026	0.0098	0.0021	0.0018	0.0017	0.0022	0.0013	0.0022	0.0018

Table 3: The representation compactness: The class-conditional reconstruction and the squashing function improve the compactness, while dynamic routing reduces it. This claim is true on both MNIST and FMNIST datasets.

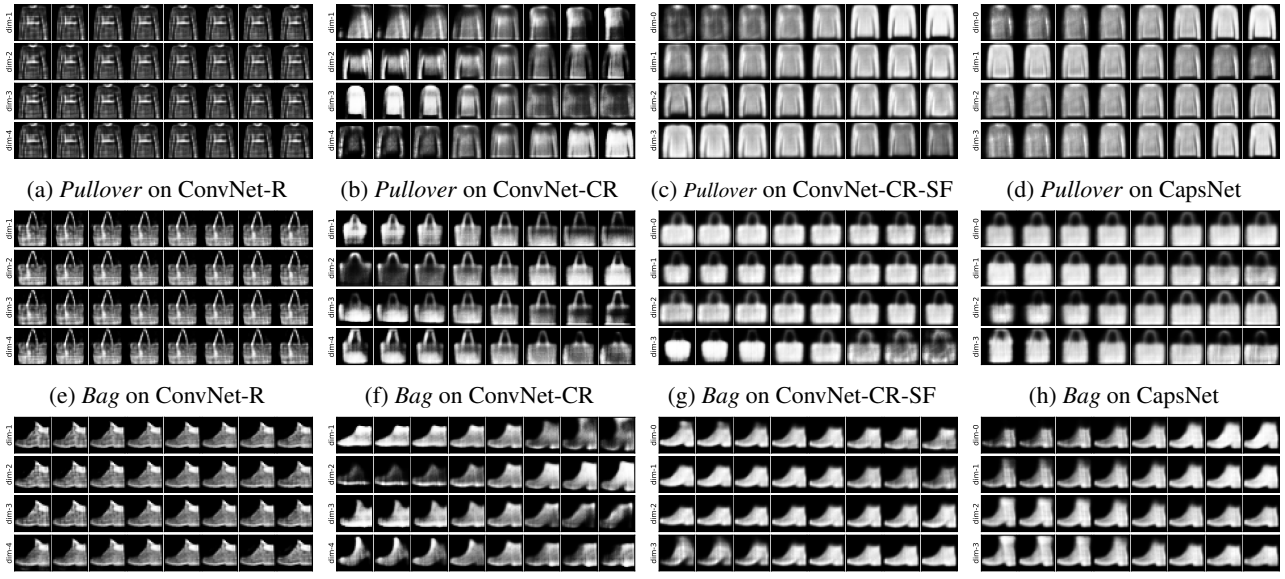


(a) Digit 2 on ConvNet-R (b) Digit 2 on ConvNet-CR (c) Digit 2 on ConvNet-CR-SF (d) Digit 2 on CapsNet

(e) Digit 6 on ConvNet-R (f) Digit 6 on ConvNet-CR (g) Digit 6 on ConvNet-CR-SF (h) Digit 6 on CapsNet

(i) Digit 7 on ConvNet-R (j) Digit 7 on ConvNet-CR (k) Digit 7 on ConvNet-CR-SF (l) Digit 7 on CapsNet

Figure 2: The reconstructed images on MNIST dataset are shown when a single unit of representation is perturbed. We show the images on some classes where images and classes are selected randomly. The reconstruction only helps when the class-conditional masking mechanism is applied. The squashing function improves the visual response further.

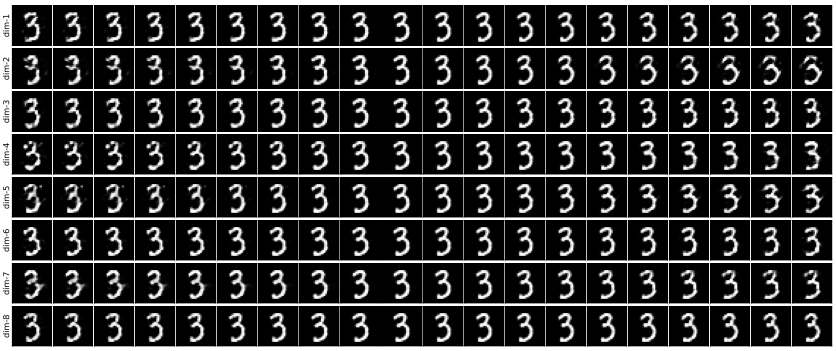


(a) *Pullover* on ConvNet-R (b) *Pullover* on ConvNet-CR (c) *Pullover* on ConvNet-CR-SF (d) *Pullover* on CapsNet

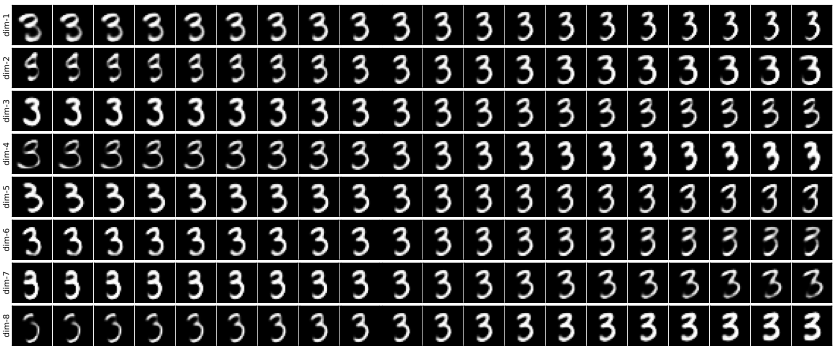
(e) *Bag* on ConvNet-R (f) *Bag* on ConvNet-CR (g) *Bag* on ConvNet-CR-SF (h) *Bag* on CapsNet

(i) *boot* on ConvNet-R (j) *boot* on ConvNet-CR (k) *boot* on ConvNet-CR-SF (l) on CapsNet

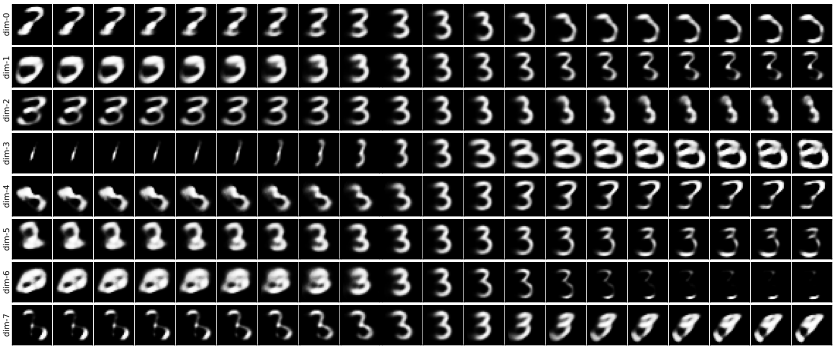
Figure 3: The reconstructed images on FMNIST dataset are shown when a single unit of representation is perturbed. We show the images on some classes where images and classes are selected randomly. They are *Pullover*, *Bag*, and *boot*. The observation is consitent with the one on MNIST.



(a) on ConvNet-R



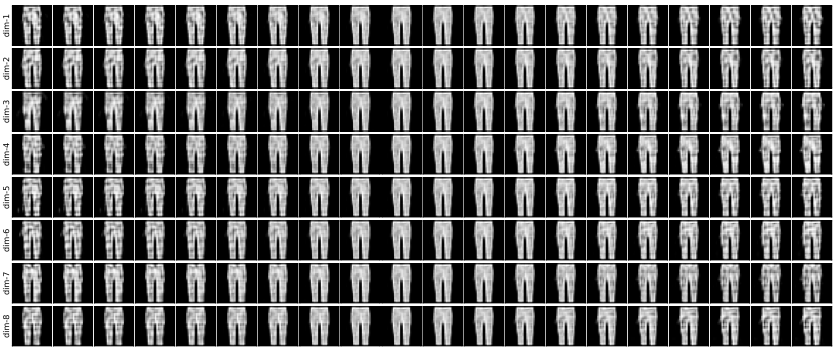
(b) on ConvNet-CR



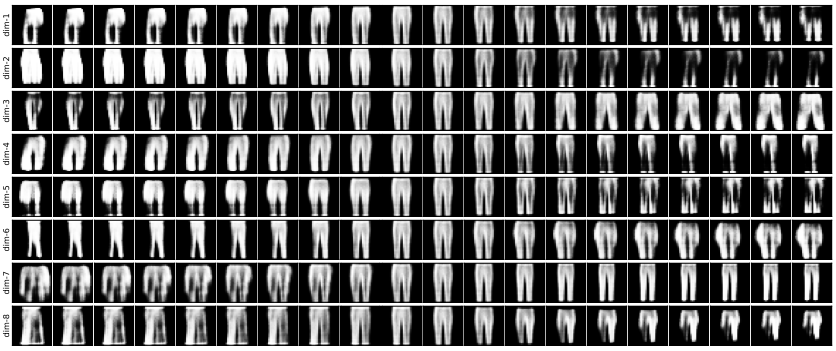
(c) on ConvNet-CR-SF

(d) on CapsNet

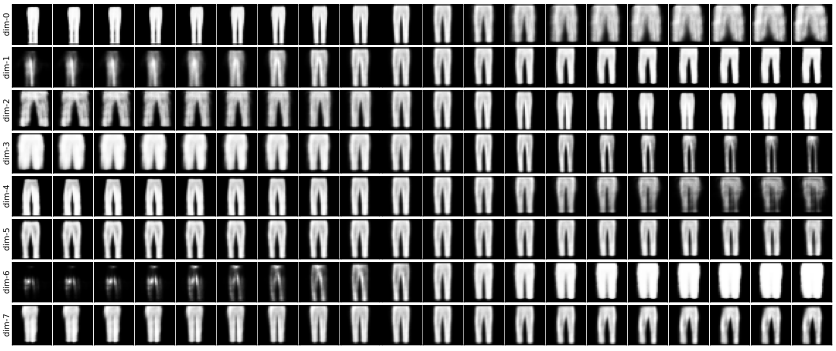
Figure 4: The reconstructed images on MNIST dataset are shown when a single unit of representation is perturbed. We reduce the perturbation interval to obtain more reconstructed images. The conclusion is the same. Namely, the reconstruction only helps when the class-conditional masking mechanism is applied, and the squashing function improves the visual response further.



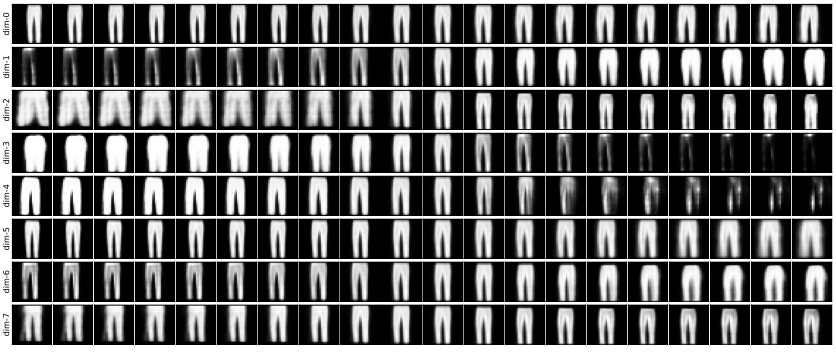
(a) on ConvNet-R



(b) on ConvNet-CR



(c) on ConvNet-CR-SF



(d) on CapsNet

Figure 5: The reconstructed images on FMNIST dataset are shown when a single unit of representation is perturbed. The reconstruction only helps when the class-conditional masking mechanism is applied. The squashing function improves the visual response further.

Chapter 6

Efficient and Effective Vote Attack on Capsule Networks

EFFECTIVE AND EFFICIENT VOTE ATTACK ON CAPSULE NETWORKS

Jindong Gu[1,4], **Baoyuan Wu**[2,3], **Volker Tresp**[1,4]
University of Munich, Germany[1]
The Chinese University of Hong Kong, Shenzhen, China[2]
Shenzhen Research Institute of Big Data, Shenzhen, China[3]
Corporate Technology, Siemens AG, Munich, Germany[4]
jindong.gu@outlook.com, wubaoyuan@cuhk.edu.cn, volker.tresp@siemens.com

ABSTRACT

Standard Convolutional Neural Networks (CNNs) can be easily fooled by images with small quasi-imperceptible artificial perturbations. As alternatives to CNNs, the recently proposed Capsule Networks (CapsNets) are shown to be more robust to white-box attacks than CNNs under popular attack protocols. Besides, the class-conditional reconstruction part of CapsNets is also used to detect adversarial examples. In this work, we investigate the adversarial robustness of CapsNets, especially how the inner workings of CapsNets change when the output capsules are attacked. The first observation is that adversarial examples misled CapsNets by manipulating the votes from primary capsules. Another observation is the high computational cost, when we directly apply multi-step attack methods designed for CNNs to attack CapsNets, due to the computationally expensive routing mechanism. Motivated by these two observations, we propose a novel vote attack where we attack votes of CapsNets directly. Our vote attack is not only effective but also efficient by circumventing the routing process. Furthermore, we integrate our vote attack into the detection-aware attack paradigm, which can successfully bypass the class-conditional reconstruction based detection method. Extensive experiments demonstrate the superior attack performance of our vote attack on CapsNets.

1 INTRODUCTION

A hardly perceptible small artificial perturbation can cause Convolutional Neural Networks (CNNs) to misclassify an image. Such vulnerability of CNNs can pose potential threats to security-sensitive applications, *e.g.*, face verification (Sharif et al., 2016) and autonomous driving (Eykholt et al., 2018). Besides, the existence of adversarial images demonstrates that the object recognition process in CNNs is dramatically different from that in human brains. Hence, the adversarial examples have received increasing attention since it was introduced (Szegedy et al., 2014; Goodfellow et al., 2015).

Many works show that network architectures play an important role in adversarial robustness (Madry et al., 2018; Su et al., 2018; Xie & Yuille, 2020; Guo et al., 2020). As alternatives to CNNs, Capsule Networks (CapsNets) have also been explored to resist adversarial images since they are more biologically inspired (Sabour et al., 2017). The CapsNet architectures are significantly different from those of CNNs. Under popular attack protocols, CapsNets are shown to be more robust to white-box attacks than counter-part CNNs (Hinton et al., 2018; Hahn et al., 2019). Furthermore, the reconstruction part of CapsNets is also applied to detect adversarial images (Qin et al., 2020).

In image classifications, CapsNets first extract primary capsules from the pixel intensities and transform them to make votes. The votes reach an agreement via an iterative routing process. It is not clear how these components change when CapsNets are attacked. By attacking output capsules directly, the robust accuracy of CapsNets is 17.3%, while it is reduced to 0 on the counter-part CNNs in the same setting. Additionally, it is computationally expensive to apply multi-step attacks (*e.g.*, PGD (Madry et al., 2018)) to CapsNets directly, due to the costly routing mechanism. The two observations motivate us to propose an effective and efficient vote attack on CapsNets.

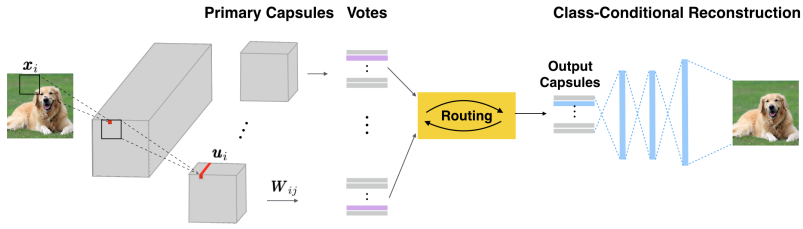


Figure 1: The overview of Capsule Networks: the CapsNet architecture consists of four components, *i.e.*, primary capsule extraction, voting, routing, and class-conditional reconstruction.

The contributions of our work can be summarised as follows: 1). We investigate the inner working changes of CapsNets when output capsules are attacked; 2). Motivated by the findings, we propose an effective and efficient vote attack; 3). We integrate the vote attack in the detection-aware attack to bypass class-conditional reconstruction based adversarial detection. The next section introduces background knowledge and related work. Sec. 3 and 4 investigate capsule attack and introduce our vote attack, respectively. The last two sections show experiments and our conclusions.

2 Background Knowledge and Related Work

Capsule Networks The overview of CapsNets is shown in Figure 1. CapsNets first extract primary capsules $\boldsymbol{u}_i$ from the input image $\boldsymbol{x}$ with pure convolutional layers (or CNN backbones). Each primary capsule $\boldsymbol{u}_i$ is then transformed to make votes for high-level capsules. The **voting process**, also called transformation process, is formulated as

$$\hat{\boldsymbol{u}}_{j|i} = \boldsymbol{W}_{ij}\boldsymbol{u}_i. \tag{1}$$

Next, a dynamic routing process is applied to identify weights c_{ij} for the votes $\hat{\boldsymbol{u}}_{j|i}$, with $i \in \{1, 2, \ldots, N\}$ corresponding to indices of primary capsules and $j \in \{1, 2, \ldots, M\}$ to indices of high-level capsules. Specifically, the **routing process** iterates over the following three steps

$$\boldsymbol{s}_j^{(t)} = \sum_i^N c_{ij}^{(t)} \hat{\boldsymbol{u}}_{j|i}, \qquad \boldsymbol{v}_j^{(t)} = g(\boldsymbol{s}_j^{(t)}), \qquad c_{ij}^{(t+1)} = \frac{\exp(b_{ij} + \sum_{r=1}^{t} \boldsymbol{v}_j^{(r)} \hat{\boldsymbol{u}}_{j|i})}{\sum_k \exp(b_{ik} + \sum_{r=1}^{t} \boldsymbol{v}_k^{(r)} \hat{\boldsymbol{u}}_{k|i})}, \tag{2}$$

where the superscript t indicates the index of iterations starting from 1, and $g(\cdot)$ is a squashing function (Sabour et al., 2017) that maps the length of the vector $\boldsymbol{s}_j$ into the range of $[0, 1)$. The b_{ik} is the log prior probability. Note that the routing process is the most expensive part of CapsNets.

The final output capsules are computed as $\boldsymbol{v}_j = g(\sum_{i=1}^{N} c_{ij} * \hat{\boldsymbol{u}}_{j|i})$ where c_{ij} is the output of the last routing iteration. The output capsules are represented by vectors, the length of which indicates the confidence of the entitys' existence. In the training phase, the class-conditional reconstruction net reconstructs the input image from the capsule corresponding to the ground-truth class t, *i.e.*, $\hat{\boldsymbol{x}} = r(\boldsymbol{v}_t)$. The reconstruction error $d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2$ works as a regularization term. All above notations will be used across this manuscript.

To improve CapsNets (Sabour et al., 2017), various routing mechanisms have been proposed, such as (Hinton et al., 2018; Zhang et al., 2018; Hahn et al., 2019; Tsai et al., 2020). The advanced techniques of building CNNs or GNNs have also been integrated into CapsNets successfully. For example, the multi-head attention-based graph pooling is applied to replace the routing mechanism (Gu & Tresp, 2020b). The CNN backbones are applied to extract more accurate primary capsules (Rajasegaran et al., 2019; Phaye et al., 2018). To understand CapsNets, (Gu & Tresp, 2020a) investigates the contribution of dynamic routing to the input affine transformation robustness. This work focuses on its contribution to the adversarial robustness.

(Hinton et al., 2018; Hahn et al., 2019) demonstrated the high adversarial robustness of CapsNets. However, it has been shown in (Michels et al., 2019) that the robustness does not hold for all attacks. In addition, many defense strategies proposed for CNNs are circumvented by later defense-aware white-box attacks (Athalye et al., 2018). Given the previous research line, we argue that it is necessary to explore CapsNet architecture-aware attacks, before we give any claim on the robustness

of CapsNets. To the best of our knowledge, there is no attack specifically designed for CapsNets in current literature.

Adversarial Attacks Given the outputs $f(\boldsymbol{x})$ of an input in a CNN, attacks fool the model by creating perturbations to increase the loss $\mathcal{L}(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y})$ where $\mathcal{L}(\cdot)$ is the standard cross-entropy loss and $\boldsymbol{\delta}$ indicates a ℓ_p-bounded perturbation. The one-step *Fast Gradient Sign Method* (FGSM (Goodfellow et al., 2015)) creates perturbations as

$$\boldsymbol{\delta} = \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{\delta}} \mathcal{L}(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y})). \tag{3}$$

The multi-step *Projected Gradient Descent* (PGD (Madry et al., 2018)), is defined as

$$\boldsymbol{\delta} \leftarrow \text{clip}_{\epsilon}(\boldsymbol{\delta} + \alpha \cdot \text{sign}(\nabla_{\boldsymbol{\delta}} \mathcal{L}(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y}))). \tag{4}$$

Other popular multi-step attacks also include *Basic Iteractive Method* (BIM (Kurakin et al., 2017)) *Momentum Iterative Method* (MIM (Dong et al., 2018)). Besides, C&W attack (Carlini & Wagner, 2017b) and Deepfool (Moosavi-Dezfooli et al., 2016) are popular strong attacks on the ℓ_2-norm constraint.

Adversarial Detection Besides adversarial attack and defense (Madry et al., 2018; Chen et al., 2020; Li et al., 2020), adversarial detection has also received much attention (Xu et al., 2017; Ma et al., 2020). Many CNN-based adversarial detection methods were easily bypassed by constructing new loss functions (Carlini & Wagner, 2017a). Adversarial images are not easily detected. The most recent work (Qin et al., 2020) leverages the class-conditional reconstruction net of CapsNets to detect adversarial images.

Given any input $\boldsymbol{x}$, the predictions and the corresponding capsule are $f(\boldsymbol{x})$ and $\boldsymbol{V}$, respectively. The input is flagged as an adversarial image, if the reconstruction error is bigger than a pre-defined threshold $\|r(\boldsymbol{v}_p) - \boldsymbol{x}\|_2 > \theta$ where $p = \arg\max f(\boldsymbol{x})$ is the predicted class. The reconstruction net $r(\cdot)$ reconstructed the input from the capsule $\boldsymbol{v}_p$ of the predicted class. The choice of θ involves a trade-off between false positive and false negative detection rates. Instead of tuning this parameter, the work (Qin et al., 2020) simply sets it as the 95th percentile of benign validation distances. A strong detection-aware reconstructive attack is also proposed to verify the effectiveness of the proposed detection method in (Qin et al., 2020). The reconstructive attack is a two-stage optimization method where it first creates a perturbation $\boldsymbol{\delta}$ to fool the prediction as in Equation (5), and updates the perturbation further to reduce the reconstruction error as in Equation (6),

$$\boldsymbol{\delta} \leftarrow \text{clip}_{\epsilon}(\boldsymbol{\delta} + \alpha \cdot \beta \cdot \text{sign}(\nabla_{\boldsymbol{\delta}} \mathcal{L}(f(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y}))), \tag{5}$$

$$\boldsymbol{\delta} \leftarrow \text{clip}_{\epsilon}(\boldsymbol{\delta} + \alpha \cdot (1-\beta) \cdot \text{sign}(\nabla_{\boldsymbol{\delta}} \left\|r(\boldsymbol{v}_{f(\boldsymbol{x})}), \boldsymbol{x}\right\|_2), \tag{6}$$

where α is the step size, and β is a hyper-parameter to balance the losses in the two stages.

3 Capsule Attack on Capsule Networks

Attack Formulation. In CNNs, under certain constraints, the adversary finds adversarial perturbation of an instance by maximizing the classification loss. In CapsNets, the length of output capsules corresponds to the output probability of the classes. Similarly, the adversarial perturbation can be obtained by first mapping the length of output capsules to logits $Z(\boldsymbol{x})_j = \log(\|\boldsymbol{v}_j\|_2)$ and solving the maximization problem in Equation (7). In this formulation, output capsules are attacked directly, which is called **Caps-Attack**.

$$\boldsymbol{\delta}^* = \underset{\boldsymbol{\delta} \in \mathcal{N}_\epsilon}{\arg\max}\, \mathcal{H}(Z(\boldsymbol{x}+\boldsymbol{\delta}), \boldsymbol{y}) = \mathcal{L}(\text{softmax}(Z(\boldsymbol{x}+\boldsymbol{\delta})), \boldsymbol{y}), \tag{7}$$

where $\mathcal{N}_\epsilon = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_p \leq \epsilon\}$ with $\epsilon > 0$ being the maximal perturbation. This optimization problem can be naturally solved using the algorithm designed for the attack against CNNs, such as FGSM (see Equation (3)) (Goodfellow et al., 2015) and PGD (see Equation (4)) (Madry et al., 2018).

Analysis. In CapsNets, the primary capsule $\boldsymbol{u}_i$ can make a positive or negative vote for the j-th class or abstain from voting. It depends on the relationship between $\boldsymbol{v}_j$ and $\hat{\boldsymbol{u}}_{j|i}$. The vote from $\boldsymbol{u}_i$ for the j-th class is positive if $\cos(\boldsymbol{v}_j, \hat{\boldsymbol{u}}_{j|i}) > 0$, otherwise negative if $\cos(\boldsymbol{v}_j, \hat{\boldsymbol{u}}_{j|i}) < 0$. The similarity value $\cos(\boldsymbol{v}_j, \hat{\boldsymbol{u}}_{j|i}) = 0$ corresponds to abstention of the primary capsules.

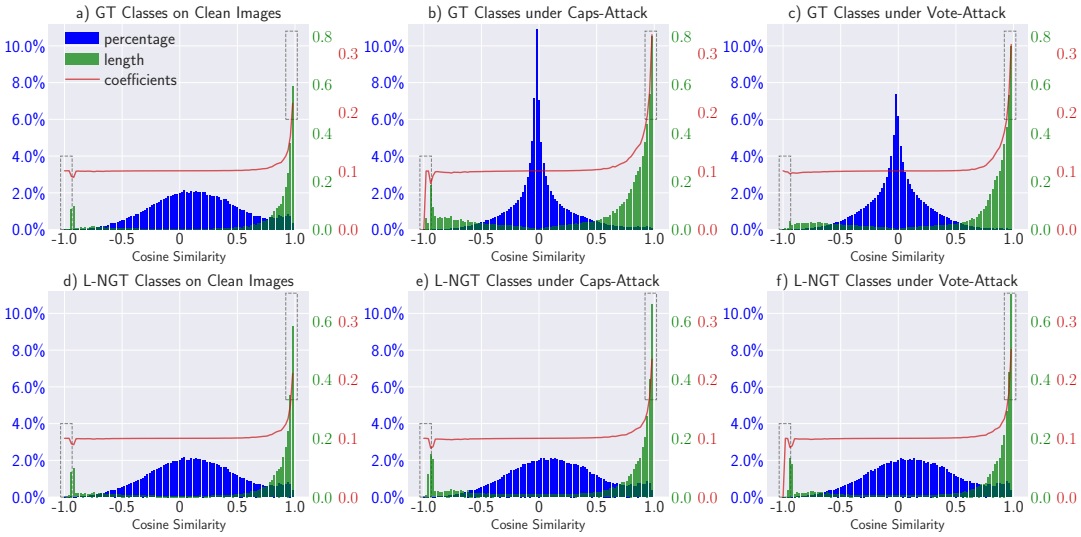


Figure 2: The left-to-right columns correspond to statistics of predictions on clean images, under Caps-Attack, and under Vote-Attack, respectively. The first row corresponds to the statistics on ground-truth classes, and the second row corresponds to the classes with the largest output probabilities that are not ground-truth (L-NGT) classes. In each subplot, the x-axis indicates the cosine similarity value between the vote $\hat{\boldsymbol{u}}_{j|i}$ and the output capsule $\boldsymbol{v}_j$. The blue histogram shows the percentage of votes falling in bins divided by the similarity values in x-axis. The green histogram corresponds to the strength of votes (the averaged length of the votes $\hat{\boldsymbol{u}}_{j|i}$). The red curve presents the averaged weight (*i.e.*, c_{ij}, see Equation (2)) of votes at each bin. Please refer to the main context for more in-depth analysis of this figure.

How do the votes change when CapsNets are attacked by adversarial images? We investigate this question with experiments and visualize the results. We firstly train a CapsNet with Dynamic Routing (DR-CapsNet) (Sabour et al., 2017) on the CIFAR10 dataset (Krizhevsky et al., 2009). With the standard well-trained DR-CapsNet (92.8% test accuracy), we classify all clean images in the test dataset and extract all votes $\hat{\boldsymbol{u}}_{j|i}$ and output capsules $\boldsymbol{v}_j$ of the ground-truth (GT) classes. We compute $\cos(\boldsymbol{v}_j, \hat{\boldsymbol{u}}_{j|i})$ in all classifications and split them into 100 equal-width bins in the range of $[-1, 1]$. In each bin, we compute the averaged length of all $\hat{\boldsymbol{u}}_{j|i}$ and average of all coupling coefficients c_{ij} therein. Note that c_{ij} identified by the routing process stands for the weights of the vote $\hat{\boldsymbol{u}}_{j|i}$. The results are visualized in Figure 2a. The majority of primary capsules make positive votes (more votes with positive similarity values in blue bins).

To obtain adversarial images, we apply PGD attack to the clean image classifications on the DR-CapsNet where 17.3% robust accuracy is obtained. Similarly, we extract corresponding information from the classifications of adversarial images on the ground-truth class and visualize the results in Figure 2b. The votes corresponding to $\cos(\boldsymbol{v}_j, \hat{\boldsymbol{u}}_{j|i}) \approx 0$ are invalid since they have only tiny impact on final prediction. The adversarial images make votes invalid by manipulating the votes and the weights of them. Concretely, the votes on adversarial images are $\hat{\boldsymbol{u}}'_{j|i}$. The voting weights identified by the routing process are c'_{ij}. Both are manipulated by adversarial images so that the output capsule $\boldsymbol{v}'_j = \sum_{i=1}^{N} c'_{ij} * \hat{\boldsymbol{u}}'_{j|i}$ is orthogonal to most votes $\hat{\boldsymbol{u}}'_{j|i}$. Namely, the adversarial images make the majority of votes invalid for the ground-truth class (the concentration of votes around the zero).

To understand how votes change on non-ground-truth classes, we also visualize the corresponding information on the classes with the **L**argest output probabilities that are **N**ot **G**round-**T**ruth classes (L-NGT classes) in Figure 2d and 2e. We mark differences between the two plots with dashed gray boxes. We can observe that the votes for L-NGT classes become stronger since both the coupling coefficients (the red line) and the strength of their positive votes (the green bins) become larger.

Drawbacks. The above analysis explains why the attack method originally designed for CNNs still works for CapsNets. The first drawback of Caps-Attack is its *limited effectiveness*. As will

be shown in later experiments, under the same attack method, CapsNets are much more robust than CNNs. Since the routing process is the main difference between CapsNets and CNNs, we attribute the higher robustness of CapsNets to the conjecture that the routing process obfuscates the gradients used to generate adversarial examples. One intuitive way to mitigate it is to approximate the routing process, *e.g.*, with Backward Pass Differentiable Approximation (BPDA) (Athalye et al., 2018). However, it is non-trivial to approximate the routing process with several routing iterations. The second drawback of Caps-Attack is the *low efficiency*. The widely used multi-step gradient-based attacks require many times forward and backward passes on the whole CapsNet to generate adversarial examples, *e.g.,* under PGD attack. Caps-Attack are computationally expensive due to the costly iterative routing mechanism of CapsNets.

4 VOTE ATTACK ON CAPSULE NETWORKS

The above two drawbacks of Caps-Attack inspire us that it is necessary to develop adversarial attack methods specifically for CapsNets, rather than directly applying the attack methods designed for CNNs to attack CapsNets. In this work, we propose to directly attack the votes (see Equation (8)) rather than the final output capsules of CapsNets, dubbed **Vote-Attack**. The behind rationale is that the vote $\hat{\boldsymbol{u}}_{j|i}$ exactly corresponds to the output class j, though it is an intermediate activation of CapsNets. Besides, when the votes from primary capsules are attacked, the corresponding weights (*i.e.*, c_{ij}, see Equation (2)) identified by the routing process will also be changed. Thus, the attacked votes could mislead the corresponding outputs of CapsNets.

Specifically, given an input-label pair $(\boldsymbol{x}, \boldsymbol{y})$, the N votes from primary capsules are $\hat{\boldsymbol{u}}_{-|i} = f_v^i(\boldsymbol{x})$ where $i \in \{1, 2, \ldots, N\}$. The average of the N votes is first computed and then squashed wtih the squashing fucntion $g(\cdot)$. The vector lengths of the squashed one correspond to output probabilities. Formally, the Vote-Attack on $\boldsymbol{x}$ is defined as

$$\boldsymbol{\delta}^* = \arg\max_{\boldsymbol{\delta} \in \mathcal{N}_\epsilon} \mathcal{H}(\log(g(\frac{1}{N}\sum_{i=1}^{N} f_v^i(\boldsymbol{x}+\boldsymbol{\delta}))), \boldsymbol{y}). \tag{8}$$

In the formulation above, we first average the votes and squash the averaged vote. There are two intuitive variants of the proposed Vote-attack. The one is to first squash their votes and then average the squashed votes. The other is to average the loss caused by all votes. Instead of opimizing on the loss computed on the squahed averaged vote, we can compute the loss of individual vote seperatedly and average them. More details about these two variants of our Vote-Attack can be found in Appendix A.

The maximization problem of Equation (8) can be approximately solved with popular attack method, *e.g.*, PGD attack. When PGD is taken as the underlying attack, the proposed Vote-Attack method can reduce the robust accuracy of DR-CapsNets from 17.3% (with Caps-Attack) to 4.83%.

Our Vote-Attack can also be extended to targeted attack by simply modifying the attack loss function of Equation (8) into $\boldsymbol{\delta}^* = \arg\max_{\boldsymbol{\delta} \in \mathcal{N}_\epsilon} l(\log(g(\frac{1}{N}\sum_{i=1}^{N} f_v^i(\boldsymbol{x}+\boldsymbol{\delta}))), \boldsymbol{t})$ where $\boldsymbol{t}$ is the target class.

Analysis. We also visualize the votes on the adversarial images created by our Vote-Attack. On the GT classes (see Figure 2c), our Vote-Attack increase the negative votes and decrease the positive votes, when compared to Caps-Attack in Figure 2b. On the L-NGT classes, the positive votes are strengthened further by our Vote-Attack, which leads to more misclassifications. See the difference in dashed gray boxes, where both the length of positive votes and the weights become larger (where the similarity values are about 1.0).

Advantages. It is interesting to find that the proposed Vote-Attack could alleviate the drawbacks of CapsNets. Firstly, since the routing process is excluded, Vote-Attack could mitigate the gradient obfuscation when computing the gradient to generate adversarial samples. Hence, the attack performance of Vote-Attack is expected to be higher than Caps-Attack. Secondly, since the costly routing process is removed from the attack method, Vote-Attack will be more efficient than Caps-Attack.

5 EXPERIMENTS

In this section, we verify our proposal via empirical experiments. We first show the effectiveness of Vote-Attack on CapsNets in the regular training scheme and the adversarial training one. We also show the efficiency of Vote-Attack. Besides, we apply Vote-Attack to bypass the recently proposed CapsNet-based adversarial detection method. All the reported scores are averaged over 5 runs.

5.1 EFFECTIVENESS OF VOTE ATTACK ON CAPSNETS

Models: We take ResNet18 as a CNN baseline. In couter-part CapsNets, we apply resnet18 backbone to extract primary capsules $\boldsymbol{u} \in (64 \times 4 \times 4, 8)$ where the outputs of the backbone are feature maps of the shape $(512, 4, 4)$ and 64 is the number of capsule groups, 8 is the primary capsule size. The primary capsules are transformed to make $64 \times 4 \times 4$ votes $\hat{\boldsymbol{u}} \in (64 \times 4 \times 4, 10, 16)$ with the learned transformation matrices $\boldsymbol{W} \in (64 \times 4 \times 4, 8, 160)$. The size of output capsule is 16, and 10 are the number of output classes. The votes $\hat{\boldsymbol{u}}$ reach an agreement $\boldsymbol{v} \in (10, 16)$ via the dynamic routing meachnism. The length of 10 output capsules are the probabilites of 10 output classes.

Datasets: The popular datasets CIFAR10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011) are used in this experiment. The standard preprocess is applied on CIFAR10 for training: 4 pixels are padded on an input of 32×32, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. For ℓ_∞-based attacks, the perturbation range is 0.031 (CIFAR10) and 0.047 (SVHN) for pixels ranging in [0, 1]. For ℓ_2-based attacks, the ℓ_2 norm of the allowed maximal perturbation is 1.0 for both datasets.

White-Box Attacks We train CNNs and CapsNets with the same standard training scheme where the models are trained with a batch size of 256 for 80 epochs using SGD with an initial learning rate of 0.1 and moment 0.9. The learning rate is set to 0.01 from the 50-th epoch. We apply popular ℓ_∞-based attacks (FGSM (Goodfellow et al., 2015), BIM (Kurakin et al., 2017), MIM (Dong et al., 2018),PGD (Madry et al., 2018)) and ℓ_2-based attacks (C&W attack (Carlini & Wagner, 2017b), Deepfool (Moosavi-Dezfooli et al., 2016)) to attack the well-trained models. The hyper-parameters mainly follow the Foolbox tool (Rauber et al., 2017). In CapsNets, Capsules and Votes are taken as targets to attack, respectively.

Table 1: The robust accuracy of ResNets and CapsNets are shown under popular attacks on CIFAR10 and SVHN datasets. Vote-Attack is much more effective than Caps-Attack and compatible with different underlying attacks.

Model	Target	FGSM	BIM	MIM	PGD	Deepfool-ℓ_2	C&W-ℓ_2
On **CIFAR10 Dataset**, the model accuracy are ResNet 92.18(±0.57) and CapsNet 92.80(±0.14).							
ResNet	Logits	16.6(±0.76)	0.15(±0.05)	0(±0)	0(±0)	0.08(±0.05)	0.24(±0.14)
CapsNet	Caps	44.55(±1.6)	24.43(±1.95)	21.69(±2.52)	17.3(±1.35)	26.55(±0.43)	18.91(±1.5)
	Votes	**26.21**(±1.66)	**8.12**(±0.13)	**9.20**(±3.44)	**4.83**(±0.05)	**20.83**(±0.78)	**6.66**(±0.32)
On **SVHN Dataset**, the model accuracy are ResNet 94.46(±0.14) and CapsNet 94.16(±0.02).							
ResNet	Logits	14.57(±2.73)	2.9(±0.47)	0.06(±0.02)	0.06(±0.02)	3.05(±0.45)	2.16(±0.1)
CapsNet	Caps	58.32(±1.34)	50.25(±0.88)	40.09(±1.65)	34.82(±2.11)	45.76(±1.17)	44.29(±1.07)
	Votes	**49.16**(±1.0)	**31.46**(±0.22)	**14.22**(±0.23)	**8.11**(±0.3)	**39.31**(±0.56)	**27.94**(±0.14)

The standard test accuracy and the robust accuracy under different attacks are reported in Table 1. The CapsNets and the counter-part CNNs achieve similar performance on normal test data. The strong attack PGD can mislead all the classifications of ResNet. However, it is less effective to attack output capsules. Our Vote-Attack can reduce the robust accuracy of CapsNets significantly across different attack methods. We also check the ℓ_0, ℓ_1, ℓ_2 norms of the perturbations created by different attack methods in Appendix B. In most cases, the different norms of perturbations corresponding to Vote-attack is similar to the ones to Caps-attack.

We also verify the effectiveness of Vote-Attack from other perspectives, such as, the targeted attacks, the transferability of adversarial examples and the adversarial robustness on affine-transformed inputs. The experimental details of the targeted Vote Attack are in the Appendix C. The transferability

of the created adversarial examples is investigated in Appendix D. The adversarial examples created by Vote-attack are more transferable than the ones by Caps-attack.

CapsNets are shown to be robust to input affine transformation (Sabour et al., 2017; Gu & Tresp, 2020a). When inputs are affine transformed, the votes in CapsNets also change correspondingly. We also verify the effectiveness of Vote-Attack in case of affine transformed inputs. We consider two cases: 1) The CapsNet built on standard convolutional layers (Sabour et al., 2017) is trained on MNIST dataset and tested on AffNIST dataset. 2) The CapsNet built on a backbone (i.e. ResNet18) is trained on the standard CIFAR10 training dataset and tested on affine-transformed CIFAR10 test images. In both cases, our Vote-Attack achieves higher attack success rates than Caps-Attack. More details about this experiment can be found in Appendix E. This experiment shows that our Vote-Attack is more effective than Caps-Attack when the inputs are affine-transformed.

Under Vote-Attack, the robust accuracy of CapsNets is still higher than that of counter-part CNNs. However, we did claim CapsNets are more robust for two reasons. 1) CapsNets possess more network parameters due to transformation matrices. 2) The potential attacks can reduce the robust accuracy further. This study demonstrates that the high adversarial robustness of CapsNets can be a fake sense, and we should be careful to draw any conclusion about the robustness of CapsNets.

Adversarial Training In this experiment, we verify the effectiveness of Vote-Attack in the context of Adversarial Training. We train models with adversarial examples created by Caps-Attack where PGD with 8 iterations is used. For training a more robust model, we also combine Vote-Attack and Caps-Attack to create adversarial examples where a new loss from the two attacks is used.

The underlying attack method used in this experiment is PGD with 40 iterations. The model performance is reported in Table 2 under different training schemes. We can observe that the Vote-Attack (corresponding to the last column) is more effective than Caps-Attack (corresponding to the second last column) under adversarial training. When we include Vote-Attack to improve the adversarial training (AT v.s. AT +Votes), the robust accuracy of CapsNets is increased under both Caps-Attack and Vote-Attack.

The Vote-Attack only attacks part of the model. During adversarial training, the model can adapt the routing process to circumvent the adversarial perturbations. Therefore, it is not effective to do adversarial training only using Vote-Attack.

Table 2: The robustness of CapsNets with different training schemes on CIFAR10 and SVHN datasets: Vote-Attack is also effective to attack models with adversarial training; It can also be applied to improve adversarial training.

Dataset	Traning	ResNet		CapsNet		
		A_{std}	Logits	A_{std}	Caps	Votes
CIFAR10	Natural	92.18(±0.57)	0	92.8(±0.14)	17.30(±1.35)	4.83(±0.05)
	AT	79.45(±1.27)	43.91(±0.62)	75.0(±0.04)	45.49(±0.78)	43.65(±0.85)
	AT + Votes	-	-	76.42(±0.37)	49.62(±0.56)	44.12(±0.32)
SVHN	Natural	94.46(±0.14)	0.06(±0.02)	94.16(±0.02)	34.82(±2.11)	8.11(±0.30)
	AT	87.9(±0.08)	36.05(±0.33)	86.0(±0.80)	33.40(±1.36)	30.44(±1.08)
	AT + Votes	-	-	83.89(±0.73)	39.13(±0.96)	34.92(±0.98)

5.2 Efficiency of Vote Attack On CapsNets

In the last subsection, we demonstrate the effectiveness of Vote-Attack from different perspectives. We now show the efficiency of Vote-Attack. In our Vote-Attack, no routing process is involved in both forward inferences and gradient backpropagations. To show the efficiency of Vote-Attack empirically, we record the time required by each attack to create a single adversarial example and average them across the CIFAR10 test dataset. A single Nvidia V100 GPU is used.

The required time is reported in Table 3. The time on SVHN dataset is almost the same as in CIFAR10 since both input space dimensions are the same (*i.e.,* 32, 32, 3). The column corresponding to A_{std} shows the time required to classify a single input image. Compared to the logit attack in CNNs, Caps-Attack in CapsNets requires more time to create adversarial examples since the dy-

namic routing is computationally expensive. Our Vote-Attack can create adversarial images without using the routing part, and reduce the required time significantly. However, the required time is still more than that on CNNs. The reason behind this is that the current deep learning framework is highly optimized on the convolutional operations, less on the voting process.

Table 3: The averaged time required by each attack to create an adversarial example is reported on CIFAR10 test dataset. Vote-Attack requiring less time is more efficient than Caps-Attacks.

Model	Target	A_{std}	FGSM	BIM	MIM	PGD	Deepfool	C&W
ResNet	Logits	4.14*ms*	12.13*ms*	83.34*ms*	165.76*ms*	324.53*ms*	186.77*ms*	409.38*ms*
CapsNet	Caps	5.65*ms*	17.45*ms*	120.75*ms*	242.97*ms*	471.81*ms*	607.84*ms*	612.79*ms*
	Votes		**14.89***ms*	**105.09***ms*	**196.11***ms*	**414.58***ms*	**295.28***ms*	**448.31***ms*

5.3 Bypassing Class-conditional Capsule Reconstruction based Detection

In this experiment, we demonstrate that class-conditional capsule reconstruction based detection can be bypassed by integrating our Vote-Attack in the detection-aware attack method. Following the work (Qin et al., 2020), we use the original CpasNet architecture (Sabour et al., 2017) for this experiment. The architecture details are shown as follows.

CapsNets, two standard convolutional layers, Conv1(C256, K9, S1), Conv2(C256, K9, S2), are used to extract primary capsules of shape (32×6×6, 8). The output capsule of shape (10, 16) can be obtained after the dynamic routing process. The output capsules will be taken as input for a reconstruction net with (FC160-FC512-FC1024-FC28×28). In the reconstruction process, only one of the output capsules is activated, others are masked with zeros. Since the input contains the class information, the reconstruction is class-conditional. The capsules corresponding to the ground-truth class will be activated during training, while the winning capsule (the one with maximal length) will be activated in the test phase.

Two CNN baseline models are considered. **CNN+CR** uses the same architecture without routing and group 160 activations into 10 groups where the sum of 16 activations is taken as a logit. The same class-conditional reconstruction mechanism is used. **CNN+R** does not group 160 activations and reconstructs the input from activations without a masking mechanism. More details of the baseline models can be found in (Qin et al., 2020).

Given an input, it will be flagged as adversarial examples if its reconstruction error is bigger than a given threshold $d(\boldsymbol{x}, \hat{\boldsymbol{x}}) > \theta$. Following (Qin et al., 2020), we set θ as 95th percentile of reconstruction errors of benign validation images, namely, 5% False postive rate. We report **Success Rate** $S = \frac{1}{K}\sum_i^N (f(\boldsymbol{x}+\boldsymbol{\delta}) \neq y)$ and **Undetected Rate** $R = \frac{1}{K}\sum_i^N (f(\boldsymbol{x}+\boldsymbol{\delta}) \neq y) \cap (d(\boldsymbol{x}, \hat{\boldsymbol{x}}) \leq \theta)$. Both detection-agnostic and detection-aware attacks introduced in Sec. 2 are considered.

Table 4: Different attacks are applied to circumvent the class-conditional reconstruction adversarial detection method on FMNIST dataset. The attack success rate and undetected rate (S/R) are reported for each attack. The integration of Vote-Attack in the detection-aware attack increases both the attack success rate and the undetected rate significantly.

Attacks	Model	Target	A_{std}	FGSM	BIM	PGD	C&W
Detection-agnostic Attack	CNN+R	Logits	90.95	85.8/63.3	100/80.0	100/75,7	86,4/68.8
	CNN+CR	Logits	91.79	89.4/66.4	97.4/70.4	97.9/67.9	77.3/77.1
	CapsNet	Caps	91.85	40.2/29.3	88.8/53.1	90.6/51.4	70.7/54.1
		Votes		**74.8**/46.1	**94.6**/59.2	**94.7**/55.3	**90.5**/50.1
Detection-aware Attack	CNN+R	Logits	90.95	85.3/77.3	99.7/95.0	100/92.1	-
	CNN+CR	Logits	91.79	89.3/75.9	96.3/82.3	96.2/81.2	-
	CapsNet	Caps	91.85	41.8/37.2	87.9/78.7	89.7/78.2	-
		Votes		**76.8/66.5**	**95.1/85.2**	**95.6/86.1**	-

The results on FMNIST dataset are reported in Table 4. In detection-agnostic attacks, we apply our Vote-Attack to attack CapsNets directly without considering the detection mechanism. The CapsNet

Figure 3: This figure shows the clean images and the corresponding adversarial images created by Caps-Attack and Vote-Attack in a targeted setting. The attack target class is set to the digit 0. The adversarial images created by the two attack methods are visually similar. The observation also echoes the previous findings in Appendix B, where we show that the perturbations created by Caps-Attack and Vote-Attack have similar norms.

used in this experiment is built on standard convolutional layers instead of backbones in previous experiments. Our Vote-Attack still achieve a higher success rate than Caps-Attack. It indicates that the Vote-Attack is effective across different architectures. Furthermore, the undetected rate is also increased correspondingly. In detection-aware attacks, the integration of our Vote-Attack increases the attack success rate and undetected rate significantly. More results on MNIST and SVHN datasets are shown in Appendix F.

Under the class-conditional capsule reconstruction based detection, some of the undetected examples are not imperceptible anymore, as shown in (Qin et al., 2020). Some images are flipped into the attack target classes when attacked, although a small perturbation threshold is applied. Some images are hard to flip, e.g., the ones with a big digit or thin strokes. We also visualize the adversarial examples created by Caps-Attack and our Vote-Attack in Figure 3. More figures and details are shown in Appendix G. We find that there is no obvious visual difference between the adversarial examples created by the two attacks. This finding echoes a previous experiment, where we compute the different norms (i.e., the ℓ_0, ℓ_1, ℓ_2 norms) of the created perturbations. The perturbations have similar norms (see Appendix B). Hence, the adversarial examples created by the two attacks are visually similar.

6 Conclusions and Future Work

We dive into the inner working of CapsNets and show how it is affected by adversarial examples. Our investigation reveals that adversarial examples can mislead CapsNets by manipulating the votes. Based on the investigation analysis, we propose an effective and efficient Vote-Attack to attack CapsNets. The Vote-Attack is more effective and efficient than Caps-Attack in both standard training and adversarial training settings. Furthermore, Vote-Attack also demonstrates the superiority in terms of the transferability of adversarial examples as well as the adversarial robustness on affine-transformed data. Last but not least, we apply our Vote-Attack to increase the undetected rate significantly of the class-conditional capsule reconstruction based adversarial detection.

The idea of attacking votes of CapsNet can also be applied to different versions of CapsNets. However, some adaptions are required since different CapsNet versions can have significantly different architectures. For instance, in EM-CapsNet (Hinton et al., 2018), a capsule corresponding to an entity are represented by a matrix, and the confidence of the entity's existence is represented by the activation of a single neuron. The possible adaption could be attacking votes by flipping the neuron activations that represents the existence of entities. Recently, many capsule networks have been proposed, to name a few (Hinton et al., 2018; Zhang et al., 2018; Rawlinson et al., 2018; Hahn et al., 2019; Ahmed & Torresani, 2019; Gu & Tresp, 2020a; Tsai et al., 2020; Ribeiro et al., 2020). We leave the further exploration on different versions of CapsNet in future work.

Even though CapsNets still seem to be more robust than counter-part CNNs under our stronger Vote-Attack, it is too early to draw such a conclusion. We conjecture that the robust accuracy of CapsNets can be reduced further. In future work, we will explore more strong attacks as well as the certifications to compare the robustness of CNNs and CapsNets.

REFERENCES

Karim Ahmed and Lorenzo Torresani. Star-caps: Capsule networks with straight-through attentive routing. In *Advances in Neural Information Processing Systems*, pp. 9101–9110, 2019.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14, 2017a.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017b.

Weilun Chen, Zhaoxiang Zhang, Xiaolin Hu, and Baoyuan Wu. Boosting decision-based black-box adversarial attacks with random sign flip. In *European Conference on Computer Vision*, pp. 276–293. Springer, 2020.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 9185–9193, 2018.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International conference on learning representations (ICLR)*, 2015.

Jindong Gu and Volker Tresp. Improving the robustness of capsule networks to image affine transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7285–7293, 2020a.

Jindong Gu and Volker Tresp. Interpretable graph capsule networks for object recognition. 2020b. To appear in AAAI 2021.

Minghao Guo, Yuzhe Yang, Rui Xu, and Ziwei Liu. When nas meets robustness: In search of robust architectures against adversarial attacks. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Taeyoung Hahn, Myeongjang Pyeon, and Gunhee Kim. Self-routing capsule networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7658–7667, 2019.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations (ICLR)*, 2018.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations (ICLR)*, 2017.

Yiming Li, Baoyuan Wu, Yan Feng, Yanbo Fan, Yong Jiang, Zhifeng Li, and Shutao Xia. Toward adversarial robustness via semi-supervised robust training. *arXiv preprint arXiv:2003.06974*, 2020.

Chengcheng Ma, Weiliang Meng, Baoyuan Wu, Shibiao Xu, and Xiaopeng Zhang. Efficient joint gradient based attack against sor defense for 3d point cloud classification. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 1819–1827, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International conference on learning representations (ICLR)*, 2018.

Felix Michels, Tobias Uelwer, Eric Upschulte, and Stefan Harmeling. On the vulnerability of capsule networks to adversarial attacks. 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti R Bathula. Multi-level dense capsule networks. In *Asian Conference on Computer Vision*, pp. 577–592. Springer, 2018.

Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell, and Geoffrey Hinton. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. In *International Conference on Learning Representations (ICLR)*, 2020.

Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, and Ranga Rodrigo. Deepcaps: Going deeper with capsule networks. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10725–10733, 2019.

Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. URL `http://arxiv.org/abs/1707.04131`.

David Rawlinson, Abdelrahman Ahmed, and Gideon Kowadlo. Sparse unsupervised capsules generalize better. *arXiv preprint arXiv:1804.06094*, 2018.

Fabio De Sousa Ribeiro, Georgios Leontidis, and Stefanos D Kollias. Capsule routing via variational bayes. In *34-th Association for the Advancement of Artificial Intelligence*, 2020.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems (NeurIPS)*, pp. 3856–3866, 2017.

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pp. 1528–1540, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy? - a comprehensive study on the robustness of 18 deep image classification models. In *European Conference on Computer Vision (ECCV)*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International conference on learning representations (ICLR)*, 2014.

Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov. Capsules with inverted dot-product attention routing. In *International Conference on Learning Representations (ICLR)*, 2020.

Cihang Xie and Alan Loddon Yuille. Intriguing properties of adversarial training at scale. In *International conference on learning representations (ICLR)*, 2020.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Liheng Zhang, Marzieh Edraki, and Guo-Jun Qi. Cappronet: Deep feature learning via orthogonal projections onto capsule subspaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5814–5823, 2018.

A TWO VARIANTS OF VOTE ATTACK

We have another two choices when attacking votes in CapsNet directly. **Choices 1**: In Equation (8), we first average the votes and squash the averaged vote. Another choice is to first squash their votes and then average the squashed votes. Our experiments show that this option is similarly effective.

$$\boldsymbol{\delta}^* = \arg\max_{\boldsymbol{\delta}\in\nabla} \mathcal{H}(\log(\frac{1}{N}\sum_{i=1}^{N} g(f_v^i(\boldsymbol{x}+\boldsymbol{\delta}))), \boldsymbol{y}). \tag{9}$$

Choices 2: Another choice is to average the loss caused by all votes. Instead of opimizing on the loss computed on the squahed averaged vote, we can compute the loss of individual vote seperatedly and average them, namely,

$$\boldsymbol{\delta}^* = \arg\max_{\boldsymbol{\delta}\in\nabla} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(g(f_v^i(\boldsymbol{x}+\boldsymbol{\delta})), \boldsymbol{y}). \tag{10}$$

The loss of each vote can differ from each other significantly. The large part of loss can be caused by a small part of votes. In other words, the gradients of received by the input can be caused mainly by a few too strong votes. This choice is less effective, compared to the one in Equation (8).

We use the same emperimental setting as in Sec. 5. Under the same PGD attack on CIFAR10 dataset, the robust accuracy corresponding to the choice 1 is $4.06_{(\pm 1.12)}$, and it is effective, similar to Equation (8). The choice 2 with the robust accuracy $43.31_{(\pm 2.46)}$ does not work well since the gradients received by inputs are dominated only by a small part of votes.

B NORMS OF PERTURBATIONS CREATED BY DIFFERENT ATTACKS

On CIFAR10 and SVHN datasets, we compute the different norms of perturbations created by different attacks. On each dataset, we first select the examples that are successfully attacked by both Vote-Attack and Caps-Attack on CapsNets as well as the corresponding attack on ResNets from the test dataset. Then, we obtain the created perturbations created by the corresponding attacks. The ℓ_0, ℓ_1 and ℓ_2 norm of perturbations are shown in Table 5 on CIFAR10 dataset and Table 6 on SVHN dataset.

In most cases, Vote-Attack and Caps-Attack create perturbations with similar norms. Under BIM attack, we can observe that ℓ_1 and ℓ_2 norms corresponding to Vote-Attack are higher than the ones to Caps-Attack. Both are smaller than the ones corresponding to other multi-step attacks (**e.g.,** PGD). The reason behind this is that the BIM attack does not converge since only 10 iterations are used by default in FoolBox tool (50 iterations in PGD). Given the same iterations before the convergence, Vote-Attack accumulates the relatively consistent gradients. Vote-Attack converges faster than Caps-Attack, which explains our observation.

In addition, the ℓ_2 attack find the minimal perturbations to misled the classifier. The different norms of perturbations is small. Our Vote-Attack finds samller perturbations in SVHN dataset and similar ones in CIFAR10 dataset. This obervation indicate that the performance of attack method can also depend on the datasets.

C VOTE TARGETED ATTACK

We create adversarial examples in targeted attack settings on CIFAR10 and SVHN datasets. The used models are the same as in the untargeted setting. The target classes are selected uniformly at random from the non-ground-truth classes. The attack is successful if the created adversarial examples are classified as the corresponding target classes by the underlying classifier.

The attack success rate (%) is reported in Table 7. In the targeted attack setting, our Vote-Attack achieves a significantly higher attack success rate than Caps-Attack. This exeriment show that our Vote-Attack is still effective when extended to the targeted attack setting.

Table 5: The ℓ_0, ℓ_1, and ℓ_2 norms of perturbations created by different attacks are shown on CIFAR10 dataset. Overall, the perturbations created by our Vote-Attack have similar norms to the ones by Caps-Attack.

	Model	Target	FGSM	BIM	MIM	PGD	Deepfool-ℓ_2	C&W-ℓ_2
ℓ_0 **norm**	ResNet	Logits	3054.7	2797.1	3071.9	3057.4	1533.5	2942.2
	CapsNet	Caps	3054.5	2489.2	3071.6	3066.3	1431.7	2977.4
		Votes	3054.6	2741.1	2523.9	3065.7	1534.8	2978.1
ℓ_1 **norm**	ResNet	Logits	93.96	53.07	77.89	77.38	0.21	0.51
	CapsNet	Caps	93.93	28.79	78.86	53.36	0.32	0.42
		Votes	93.91	43.68	78.71	54.03	0.32	0.51
ℓ_2 **norm**	ResNet	Logits	1.7041	1.1089	1.4974	1.4849	0.0059	0.0105
	CapsNet	Caps	1.7037	0.6471	1.5066	1.1035	0.0092	0.0087
		Votes	1.7035	0.6753	1.5047	1.1155	0.0091	0.0104

Table 6: The ℓ_0, ℓ_1, and ℓ_2 norms of perturbations created by different attacks are shown on SVHN dataset. In ℓ_∞-attack methods, the perturbations created by our Vote-Attack have similar norms to the ones by Caps-Attack. In ℓ_2-attack methods, our Vote-attack can find smaller perturbations to fool the underlying classifier.

	Model	Target	FGSM	BIM	MIM	PGD	Deepfool-ℓ_2	C&W-ℓ_2
ℓ_0 **norm**	ResNet	Logits	3066.9	2854.1	3071.9	3066.0	1754.4	2972.7
	CapsNet	Caps	3067.4	2552.2	3071.8	3070.3	2103.4	2931.1
		Votes	3067.2	2587.6	3071.8	3069.7	875.0	2924.9
ℓ_1 **norm**	ResNet	Logits	94.83	59.18	80.75	111.37	0.48	0.65
	CapsNet	Caps	94.88	32.99	77.77	79.37	0.52	0.87
		Votes	94.86	35.06	78.00	80.33	0.24	0.15
ℓ_2 **norm**	ResNet	Logits	1.7136	1.2041	1.5357	2.1657	0.0141	0.0149
	CapsNet	Caps	1.7142	0.7283	1.4920	1.6464	0.0161	0.0172
		Votes	1.7140	0.7677	1.4954	1.6442	0.0074	0.0029

D TRANSFERABILITY OF ADVERSARIAL EXAMPLES

We also investigate the transferability of adversarial examples created by Caps-Attack and Vote-Attack on CIFAR10 dataset. We consider three models, VGG19 (Simonyan & Zisserman, 2015), ResNet18 and CapsNets. The PGD is used as the underlying attack. We measure the transferability using Transfer Sucess Rate (TSR).

The TSR of different adversarial examples is reported in Table 8. The adversarial examples created on CNNs are more transferable. Especially, the ones created on ResNet18 can be transferred to CapsNets very well. The reason behind this is that CapsNets also the ResNet18 bone to extract primary capsules. By comparing the last two columns in Table 8, we can observe that the adversarial example created by Vote-Attack is more transferable than the ones created by Caps-Attack.

E ADVERSARIAL ROBUSTNESS ON AFFINE-TRANSFORMED DATA

CapsNets learn equivariant visual representations. When inputs are affine transformed, the votes also changes correspondingly. In this experiment, we aim to verify the effectiveness of Vote-Attack when inputs and their votes in Capsnets changed. The model is trained the same as before. We translate the test images with 2 pixels randomly and rotate the images within a given pre-defined degree.

Table 7: The targeted attack success rates (%) are shown on CIFAR10 and SVHN datasets. In the targeted attack setting, our Vote-Attack is significantly more effective than Caps-Attack when combined with popular attacks.

Model	Target	FGSM	BIM	MIM	PGD	Deepfool-ℓ_2	C&W-ℓ_2
On **CIFAR10 Dataset**, the model accuracy are ResNet 92.18(±0.57) and CapsNet 92.80(±0.14).							
ResNet	Logits	39.13(±3.11)	98.47(±0.68)	99.71(±0.33)	99.97(±0.04)	10.47(±0.11)	97.99(±1.39)
CapsNet	Caps	9.58(±0.16)	27.91(±2.11)	48.38(±0.21)	65.94(±0.92)	9.43(±0.48)	34.07(±1.38)
	Votes	**10.67**(±0.32)	**32.66**(±2.09)	**61.08**(±4.71)	**75.35**(±0.91)	**9.55**(±0.64)	**41.41**(±5.85)
On **SVHN Dataset**, the model accuracy are ResNet 94.46(±0.14) and CapsNet 94.16(±0.02).							
ResNet	Logits	43.06(±3.37)	91.72(±0.42)	98.15(±0.02)	99.78(±0.04)	11.84(±0.45)	93.97(±0.82)
CapsNet	Caps	5.82(±0.06)	38.58(±0.59)	49.04(±0.89)	68.94(±2.11)	6.82(±1.12)	44.64(±0.96)
	Votes	**7.28**(±1.73)	**48.25**(±1.02)	**65.35**(±0.28)	**91.68**(±1.06)	**7.57**(±1.06)	**62.93**(±0.55)

Table 8: The transferability of adversarial examples created on CNNs and CapsNets on CIFAR10 dataset: the ones created on CNNs are more transferable than on CapsNets; the ones created with Vote-Attack are more transferable than the ones with Caps-Attack.

		Attacks on Source Model			
		VGG19 (Logits)	ResNet18 (Logits)	CapsNet (Caps)	CapsNet (Votes)
Target Models	VGG19	83.79(±0.18)	93.94(±0.28)	35.64(±0.96)	41.49(±0.19)
	ResNet18	71.81(±1.04)	97.26(±1.84)	37.59(±6.25)	43.45(±8.13)
	CapsNet	80.38(±1.79)	97.53(±0.57)	46.43(±5.56)	55.34(±6.26)

The robust accuracy of affine-transformed images is shown in Table 9 on CIFAR10 dataset. Under different rotation degrees, our Vote-Attack is still effective. It consistently reduces the robust accuracy of CapsNets, when compared to Caps-Attack.

Table 9: When inputs are affine-transformed in CIFAR10 dataset, the Vote-Attack is still more effective to create adversarial examples than Caps-Attack.

Model	Target	(0, ±0°)	(±2, ±15°)	(±2, ±30°)	(±2, ±60°)	(±2, ±90°)
ResNet	A_{std}	92.18(±0.57)	85.64(±0.46)	68.11(±1.12)	48.47(±0.30)	42.07(±0.22)
	Logits	0	0	0	0	0
CapsNet	A_{std}	92.8(±0.14)	86.09(±0.39)	69.44(±1.96)	49.37(±2.43)	42.62(±1.64)
	Caps	17.3(±1.35)	5.82(±1.86)	2.89(±1.05)	1.63(±0.51)	1.11(±0.38)
	Votes	**4.83**(±0.05)	**1.15**(±0.38)	**0.54**(±0.22)	**0.32**(±0.16)	**0.23**(±0.08)

We also conduct experiments on AffNIST dataset. In this experiment, the original CapsNet architecture and the original CNN baseline in (Sabour et al., 2017) are used. The modes are trained on standard MNIST dataset and tested on AffNIST dataset. In AffNIST dataset, the MNIST images are transformed, namely, rotated, translated, scaled, or sheared. More details about this dataset are in this resource [1]. The perturbation threshold and the attack step size are set to 0.3 and 0.01, respectively. The other hyper-parameters are defaults in the Foolbox tool (Rauber et al., 2017).

The test accuracy on the untransformed test dataset (A_{std}), the accuracy on the transformed dataset (A_{aff}) and the robust accuracy under different attacks are reported in Table 10. Our Vote-Attack achieve higher attack success rates than Caps-Attack.

[1] https://www.cs.toronto.edu/ tijmen/affNIST/

Table 10: The test accuracy on the dataset with untransformed images and the one on the dataset with transformed images are reported (in %). CapsNet achieves better transformtation robustness than the original CNN baseline. The robust accuracy of different models are also reported under different attacks. We can observe that it is more effective to attack Votes instead of output capsules in CapsNet.

Model	Target	A_{std}	A_{aff}	FGSM	BIM	MIM	PGD
ResNet	Logits	99.22	66.08	10.18	0	0	0
CapsNet	Caps	99.22	79.12	15.61	4.27	1.01	0.48
	Votes			10.43	1.33	0	0

F BYPASSING CLASS-CONDITIONAL RECONSTRUCTION ON MNIST, FMNIST AND SVHN

The integration of our Vote-attack into detection-aware attack is effective to bypass the class-conditional reconstruction detection method. To verify this, we also conduct experiments on different datasets, such as MNIST and SVHN. The results are reported in Table 11. On the All three datasets, both detection-aware and detection-agnostic attacks achieve high attack success rate and undetected rate, when combined with our Vote-attack.

Table 11: Different attacks are applied to circumvent the class-conditional reconstruction adversarial detection method. The attack success rate and undetected rate (S/R) are reported for each attack. On all the three popular datasets, the integration of Vote-Attack in the detection-aware attack increases both the attack success rate and the undetected rate significantly.

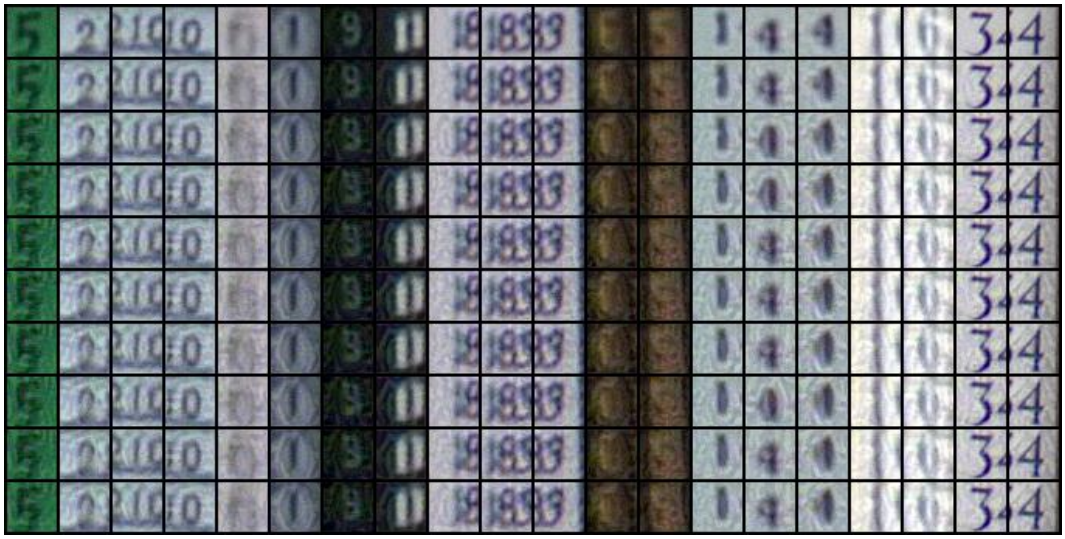
DataSet	Model	A_{std}	Attacks	Target	FGSM	BIM	PGD
MNIST	CapsNet	99.41	Detection-agnostic	Caps	13.3/6.3	73.3/31.7	77.9/33.1
				Votes	38.8/15.1	92.3/35.4	93.4/34.1
			Detection-aware	Caps	16.1/13.6	71.7/52.7	77.2/57.8
				Votes	44.8/34.9	92.1/66.8	93.4/67.1
FMNIST	CapsNet	91.85	Detection-agnostic	Caps	40.2/29.3	88.8/53.1	90.6/51.4
				Votes	74.8/46.1	94.6/59.2	94.7/55.3
			Detection-aware	Caps	41.8/37.2	87.9/78.7	89.7/78.2
				Votes	76.8/66.5	95.1/85.2	95.6/86.1
SVHN	CapsNet	91.32	Detection-agnostic	Caps	83.2/78.1	99.1/92.3	99.6/92.2
				Votes	95.5/88.8	99.9/93.2	99.9/93.3
			Detection-aware	Caps	84.2/80.1	97.8/95	97.8/94.7
				Votes	90.6/90.8	100/96.7	100/96.8

G VISUALIZING UNDETECTED ADVERSARIAL EXAMPLES

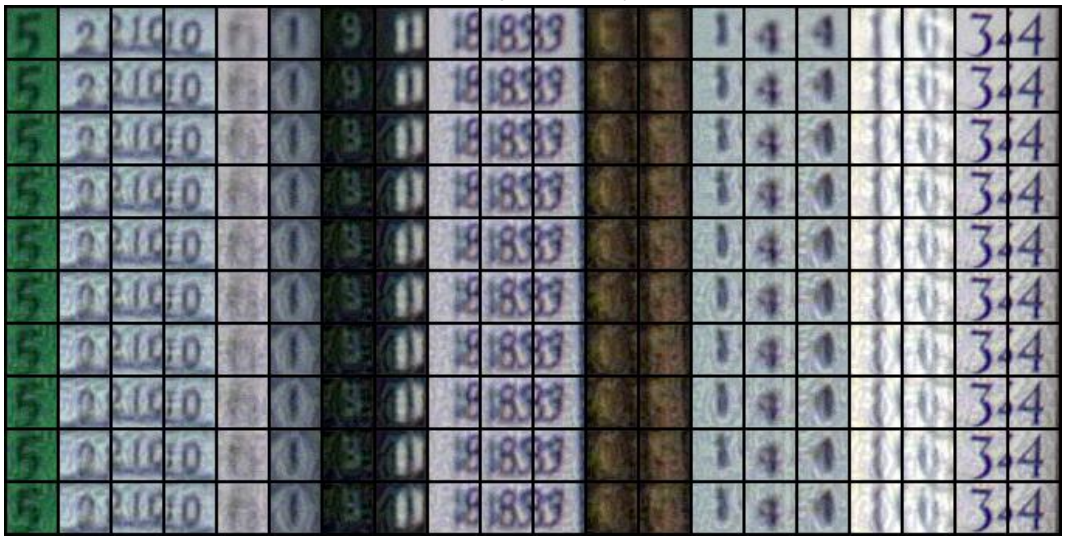
We also visualize the adversarial examples created by Caps-Attack and Vote-Attack in Figure 4. In this experiment, following (Qin et al., 2020), we use a detection-aware attack method and set the attack target class is 0. The standard setting 0.047 is used in the case of input range [0, 1], which corresponds to 12 of the pixel range of 255. In Figure 4, Some adversarial examples are flipped to target class to human perception, although the perturbation threshold is small. For some examples, it is hard to flip them, e.g., the ones with a big digit and thin strokes.

By comparing the adversarial examples created by Caps-Attack and Vote-Attack, we can find that there is no obvious visual difference between the adversarial examples. The observation also echos with our experiment in Appendix B. In that experiment, we compute the different norms of the perturbations created by different methods. The results in Table 5 and 6 show the perturbations created by Caps-Attack and Vote-Attack have similar norms. Hence, the adversarial examples created by Caps-Attack and Vote-Attack are also visually similar.

(a) Clean images in SVHN test dataset



(b) Adversarial images created by Vote-Attack



(c) Adversarial images created by Vote-Attack

Figure 4: The first subfigure shows clean test images of the SVHN dataset. The second subfigure shows the adversarial images created by Caps-Attack. Different rows correspond to different weights to reduce reconstruction error in Equation (6) (i.e., the second attack step in detection-aware attack method). Some images are flipped, and some hard ones are not. The images in the third subfigure are the adversarial images created by Vote-Attack. There is no obvious visual difference between the adversarial examples created by the two attacks. To be noted that the images are randomly selected (not cherry picked).

Chapter 7

Understanding Robustness of ViT to Patch Perturbation

Are Vision Transformers Robust to Patch Perturbations?

Jindong Gu[1] **Volker Tresp**[1] **Yao Qin**[2]

[1]*University of Munich*
[2]*Google Research*

Abstract

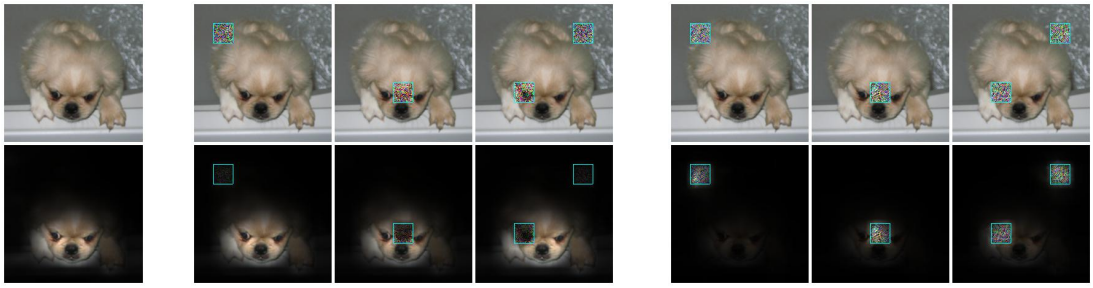
The recent advances in Vision Transformer (ViT) have demonstrated its impressive performance in image classification, which makes it a promising alternative to Convolutional Neural Network (CNN). Unlike CNNs, ViT represents an input image as a sequence of image patches. The patch-wise input image representation makes the following question interesting: How does ViT perform when individual input image patches are perturbed with natural corruptions or adversarial perturbations, compared to CNNs? In this work, we study the robustness of ViT to patch-wise perturbations. Surprisingly, we **find** that ViTs are more robust to naturally corrupted patches than CNNs, whereas they are more vulnerable to adversarial patches. Furthermore, we conduct experiments to **understand** the robustness to patch perturbations. We have revealed that the attention module can help improve the robustness of ViT by effectively ignoring natural corrupted patches. However, when ViTs are attacked by an adversary, the attention mechanism can be easily fooled to focus more on the adversarially perturbed patches and cause a mistake. Based on our analysis, we propose a simple method to **improve** the robustness of ViT to adversarial patches. Extensive qualitative and quantitative experiments verified our findings, understanding, and improvement of ViT robustness to patch-wise perturbations.

1 Introduction

Recently, Vision Transformer (ViT) has demonstrated impressive performance [7, 8, 10, 14, 15, 25, 46, 48, 49], which makes it become a potential alternative to convolutional neural networks (CNNs). Meanwhile, the robustness of ViT has also received great attention [5, 20, 38, 40, 41, 44]. On the one hand, it is important to improve its robustness for safe deployment in the real world. On the other hand, diagnosing the vulnerability of ViT can also give us a deeper understanding of its underlying working mechanisms. Existing works have intensively studied the robustness of ViT and CNNs when the whole input image is perturbed with natural corruptions or adversarial perturbations [2, 3, 5, 28, 40]. Unlike CNNs, ViT processes the input image as a sequence of image patches. Then, a self-attention mechanism is applied to aggregate information from all patches. In this work, instead, we study the robustness of ViT to patch-wise perturbations based on its special patch-based architecture.

In this work, two typical types of perturbations are considered to compare the robustness between ViTs and CNN (e.g., ResNets [16]). One is natural corruptions [17], which is to test models' robustness under distributional shift. The other is adversarial perturbations [13, 43], which are created by an adversary to specifically fool a model to make a wrong prediction. Surprisingly, we find ViT does not always perform more robustly than ResNet. When individual image patches are naturally corrupted, ViT performs more robustly than ResNet. However, when input image patch(s) are adversarially attacked, ViT shows a higher vulnerability than ResNet.

Digging down further, we conduct extensive qualitative and quantitative experiments to understand the robustness to patch perturbations. We have revealed that ViT's stronger robustness to natural corrupted patches and higher vulnerability against adversarial patches are both caused by the attention mechanism. Specifically,



(a) Clean Image (b) with Naturally Corrupted Patch (c) with Adversarial Patch

Figure 1: Images with patch-wise perturbations (top) and their corresponding attention maps (bottom). The attention mechanism in ViT can effectively ignore the naturally corrupted patches to maintain a correct prediction in figure b, whereas it is forced to focus on the adversarial patches to make a mistake in figure c. The images with corrupted patches are all correctly classified. The images with adversary patches in subfigure 1c are misclassified as *dragonfly*, *axolotl*, and *lampshade*, respectively.

the self-attention mechanism of ViT can effectively ignore the natural patch corruption, while it's also easy to manipulate the self-attention mechanism to focus on an adversarial patch. This is well supported by rollout attention visualization [1] on ViT. As shown in Fig. 1 (a), ViT successfully attends to the class-relevant features on the clean image, *i.e.*, the head of the dog. When one or more patches are perturbed with natural corruptions, shown in Fig. 1 (b), ViT can effectively ignore the corrupted patches and still focus on the main foreground to make a correct prediction. In Fig. 1 (b), the attention weights on the positions of naturally corrupted patches are much smaller even when the patches appear on the foreground. In contrast, when the patches are perturbed with adversarial perturbations by an adversary, ViT is successfully fooled to make a wrong prediction, as shown in Fig. 1 (c). This is because the attention of ViT is misled to focus on the adversarial patch instead.

When the attention is misled to focus on the adversarial patch, all patch embeddings are mainly based on the embedding of the adversarial patch, which leads to deviated image representation. An intuitive solution to mitigate this is to modify attention so that the attention is forced not to attend to a single patch. We propose a simple way to implement the solution. It is called Smoothed Attention where we scale the temperature of the *softmax* operation in the attention. In Smoothed Attention, the normal patches also contribute to patch embeddings of the next layer, which can boost the robustness of ViT.

Our main contributions can be summarized as follows:

- Based on a fair comparison, we **find** that ViT is more robust to natural patch corruption than ResNet, whereas it is more vulnerable to adversarial patch perturbation.
- We conduct extensive analysis to **understand** our observations. Specifically, we reveal that the self-attention mechanism can effectively ignore natural corrupted patches to maintain a correct prediction but be easily fooled to focus on adversarial patches to make a mistake.
- Inspired by our analysis, we show attention smoothing can **improve** the robustness of ViT against adversarial patches since smoothed attention does not focus on a single patch.

2 Related Work

Robustness of Vision Transformer. The robustness of ViT have achieved great attention due to its great success [2, 3, 4, 5, 18, 28, 29, 30, 33, 33, 34, 38, 40, 44, 50]. On the one hand, [5, 36] show that vision transformers are more robust to natural corruptions [17] compared to CNNs. On the other hand, [5, 36, 40]

Table 1: Comparison of popular ResNet and ViT models. The difference in model robustness can not be blindly attributed to the model architectures. It can be caused by different training settings. WS, GN and WD correspond to Weight Standardization, Group Normalization and Weight Decay, respectively.

Model	Pretraining	DataAug	Input Size	WS	GN	WD
ResNet [16]	N	N	224	N	N	Y
BiT [22]	Y	N	480	Y	Y	N
ViT [10]	Y	N	224/384	N	N	N
DeiT [46]	N	Y	224/384	N	N	N

demonstrate that ViT achieves higher adversarial robustness than CNNs under adversarial attacks. These existing works, however, mainly focus on investigating the robustness of ViT when a whole image is naturally corrupted or adversarially perturbed. Instead, our work focuses on patch perturbation, given the patch-based architecture trait of ViT. The patch-based attack [12, 20] and defense [32, 41] methods have also been proposed recently. Different from their work, we aim to understand ViT-based classifications under patch-based natural corruption and adversarial patch perturbation.

Adversarial Patch Attack. The seminal work [35] shows that adversarial examples can be created by perturbing only a small amount of input pixels. Further, [6, 24] successfully creates universal, robust, and targeted adversarial patches. These adversarial patches therein are often placed on the main object in the images. The works [11, 31] shows that effective adversarial patches can be created without access to the target model. However, both universal patch attacks and black-box attacks are weak to be used for our study. They can only achieve very low fooling rates when a single patch of ViT (only 0.5% of image) is attacked. In contrast, the white-box attack [21, 23, 26, 37, 47] can fool models by attacking only a very small patch. In this work, we apply the most popular adversarial patch attack in [21] to both ViT and CNNs for our study.

3 Experimental Settings to Compare ViT and ResNet

Fair Base Models. We list the state-of-the-art ResNet and ViT models and part of their training settings in Tab. 1. The techniques applied to boost different models are different, *e.g.*, pretraining. A recent work [3] points out the necessity of a fair setting. Our investigation finds weight standardization and group normalization have also a significant impact on model robustness (More in Apeendix A). This indicates that the difference in model robustness can not be blindly attributed to the model architectures if models are trained with different settings. Hence, we build fair models to compare ViT and ResNet as follows.

First, we follow [46] to choose two pairs of fair model architectures, DeiT-small vs. ResNet50 and DeiT-tiny vs. ResNet18. The two models of each pair (*i.e.* DeiT and its counter-part ResNet) are of similar model sizes. Further, we train ResNet50 and ResNet18 using the **exactly same setting** as DeiT-small and Deit-tiny in [46]. In this way, we make sure the two compared models, *e.g.*, DeiT-samll and ResNet50, have similar model sizes, use the same training techniques, and achieve similar test accuracy (See Apeendix A). The two fair base model pairs are used across this paper for a fair comparison.

Adversarial Patch Attack. We now introduce adversarial patch attack [21] used in our study. The first step is to specify a patch position and replace the original pixel values of the patch with random initialized noise δ. The second step is to update the noise to minimize the probability of ground-truth class, *i.e.* maximize the cross-entropy loss via multi-step gradient ascent [27]. The adversary patches are specified to align with input patches of DeiT.

Evaluation Metric. We use the standard metric **Fooling Rate (FR)** to evaluate the model robustness. First, we collect a set of images that are correctly classified by both models that we compare. The number of these collected images is denoted as P. When these images are perturbed with natural patch corruption or

Table 2: Fooling Rates (in %) are reported. DeiT is more robust to naturally corrupted patches than ResNet, while it is significantly more vulnerable than ResNet against adversarial patches. Bold font is used to mark the lower fooling rate, which indicates the higher robustness.

Model	# Naturally Corrupted Patches				# Adversarial Patches			
	32	96	160	196	1	2	3	4
ResNet50	3.7	18.2	43.4	49.8	**30.6**	**59.3**	**77.1**	**87.2**
DeiT-small	**1.8**	**7.4**	**22.1**	**38.9**	61.5	95.4	99.9	100
ResNet18	6.8	31.6	56.4	61.3	**39.4**	**73.8**	**90.0**	**96.1**
DeiT-tiny	**6.4**	**14.6**	**35.8**	**55.9**	63.3	95.8	99.9	100

adversarial patch attack, we use Q to denoted the number of images that are misclassified by the model. The Fooling Rate is then defined as FR = $\frac{Q}{P}$. The lower the FR is, the more robust the model is.

4 ViT Robustness to Patch-wise Perturbations

Following the setting in [46], we train the models DeiT-small, ResNet50, DeiT-tiny, and ResNet18 on ImageNet 1k training data respectively. Note that no distillation is applied. The input size for training is $H = W = 224$, and the patch size is set to 16. Namely, there are 196 image patches totally in each image. We report the clean accuracy in Apeendix A where DeiT and its counter-part ResNet show similar accuracy on clean images.

4.1 Patch-wise Natural Corruption

First, we investigate the robustness of DeiT and ResNet to patch-based natural corruptions. Specifically, we randomly select 10k test images from ImageNet-1k validation dataset [9] that are correctly classified by both DeiT and ResNet. Then for each image, we randomly sample n input image patches $\boldsymbol{x}_i$ from 196 patches and perturb them with natural corruptions. As in [17], 15 types of natural corruptions with the highest level are applied to the selected patches, respectively. The fooling rate of the patch-based natural corruption is computed over all the test images and all corruption types. We test DeiT and ResNet with the same naturally corrupted images for a fair comparison.

We find that both DeiT and ResNet hardly degrade their performance when a small number of patches are corrupted (*e.g.*, 4). When we increase the number of patches, the difference between two architectures emerges: DeiT achieves a lower FR compared to its counter-part ResNet (See Tab. 2). This indicates that DeiT is more robust against naturally corrupted patches than ResNet. The same conclusion holds under the extreme case when the number of patches $n = 196$. That is: the whole image is perturbed with natural corruptions. This is aligned with the observation in the existing work [5] that vision transformers are more robust to ResNet under distributional shifts.

In addition, we also increase the patch size of the perturbed patches, *e.g.*, if the patch size of the corrupted patch is 32×32, it means that it covers 4 continuous and independent input patches as the input patch size is 16×16. As shown in Fig. 2 (Left), even when the patch size of the perturbed patches becomes larger, DeiT (marked with red lines) is still more robust than its counter-part ResNet (marked with blue lines) to natural patch corruption.

4.2 Patch-wise Adversarial Attack

In this section, we follow [21] to generate adversarial patch attack and then compare the robustness of DeiT and ResNet against adversarial patch attack. We first randomly select the images that are correctly classified

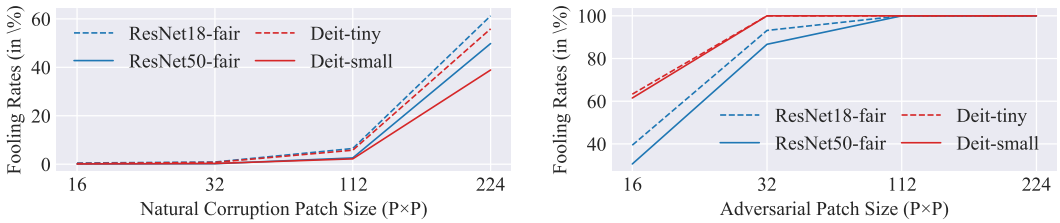


Figure 2: DeiT with red lines shows a smaller FR to natural patch corruption and a larger FR to adversarial patch of different sizes than counter-part ResNet.

by both models from imagenet-1k validation daset. Following [21], the ℓ_∞-norm bound, the step size, and the attack iterations are set to 255/255, 2/255, and 10K respectively. Each reported FR score is averaged over 19.6k images.

As shown in Tab. 2, DeiT achieves much higher fooling rate than ResNet when one of the input image patches is perturbed with adversarial perturbation. This consistently holds even when we increase the number of adversarial patches, sufficiently supports that DeiT is more vunerable than ResNet against patch-wise adversarial perturbation. When more than 4 patches ($\sim$2% area of the input image) are attacked, both DeiT and ResNet can be successfully fooled with almost 100% FR.

When we attack a large continuous area of the input image by increasing the patch size of adversarial patches, the FR on DeiT is still much larger than counter-part ResNet until both models are fully fooled with 100% fooling rate. As shown in Fig. 2 (Right), DeiT (marked with red lines) consistently has higher FR than ResNet under different adversarial patch sizes.

Taking above results together, we discover that DeiT is more robust to natural patch corruption than ResNet, whereas it is significantly more vulnerable to adversarial patch perturbation.

5 Understanding ViT Robustness to Patch Perturbation

In this section, we design and conduct experiments to analyse the robustness of ViT. Especially, we aim to obtain deep understanding of how ViT performs when its input patches are perturbed with natural corruption or adversary patches.

5.1 How ViT Attention Changes under Patch Perturbation?

We visualize and analyze models' attention to understand the different robustness performance of DeiT and ResNet against patch-wise perturbations. Although there are many existing methods, *e.g.*, [39, 42, 52], designed for CNNs to generate saliency maps, it is not clear yet how suitable to generalize them to vision transformers. Therefore, we follow [21] to choose the **model-agnostic** vanilla gradient visualization method to compare the gradient (saliency) map [51] of DeiT and ResNet. Specifically, we consider the case where DeiT and ResNet are attacked by adversarial patches. The gradient map is created as follow: we obtain the gradients of input examples towards the predicted classes, sum the absolute values of the gradients over three input channels, and visualize them by mapping the values into gray-scale saliency maps.

Qualitative Evaluation. As shown in Fig. 3 (a), when we use adversarial patch to attack a ResNet model, the gradient maps of the original images and the images with adversarial patch are similar. The observation is consistent with the one made in the previous work [21]. In contrast to the observation on ResNet, the adversarial patch can change the gradient map of DeiT by attracting more attention. As shown in Figure 3 (b), even though the main attention of DeiT is still on the object, part of the attention is misled to the adversarial patch. More visualizations are in Appendix B .

Table 3: Quantitative Evaluation. Each cell lists the percent of patches in which the maximum gradient value inside the patches is also the maximum of whole gradient map. SUM corresponds to the sum of element values inside patch divided by the sum of values in the whole gradient map. The average over all patches is reported.

	Towards ground-truth Class				Towards misclassified Class			
	SUM		MAX		SUM		MAX	
Patch Size	16	32	16	32	16	32	16	32
ResNet50	0.42	1.40	0.17	0.26	0.55	2.08	0.25	0.61
DeiT-small	**1.98**	**5.33**	**8.3**	**8.39**	**2.21**	**6.31**	**9.63**	**12.53**
ResNet18	0.24	0.74	0.01	0.02	0.38	1.31	0.05	0.13
DeiT-tiny	**1.04**	**3.97**	**3.67**	**5.90**	**1.33**	**4.97**	**6.49**	**10.16**

Quantitative Evaluation. We also measure our observation on the attention changes with the metrics in [21]. In each gradient map, we score each patch according to (1) the maximum absolute value within the patch (MAX); and (2) the sum of the absolute values within the patch (SUM). We first report the percentage of patches where the MAX is also the maximum of the whole gradient map. Then, we divide the SUM of the patch by the SUM of the all gradient values and report the percentage.

As reported in Tab. 3, the pixel with the maximum gradient value is more likely to fall inside the adversarial patch on DeiT, compared to that on ResNet. Similar behaviors can be observed in the metric of SUM. The quantitative experiment also supports our claims above that adversarial patches mislead DeiT by attracting more attention.

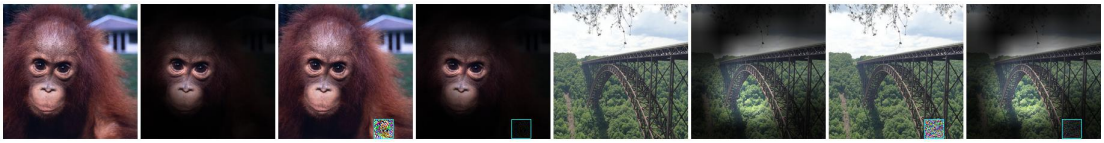
(a) on ResNet50 under Adversary Patch Attack

(b) on DeiT-small under Adversary Patch Attack

Figure 3: Gradient Visualization. the clean image, the images with adversarial patches, and their corresponding gradient maps are visualized. We use a blue box on the gradient map to mark the location of the adversarial patch. The adversary patch on DeiT attracts attention, while the one on ResNet hardly do.

Besides the gradient analysis, another popular tool used to visualize ViT is Attention Rollout [1]. To further confirm our claims above, we also visualize DeiT with Attention Rollout in Fig. 4. The rollout attention also shows that the attention of DeiT is attracted by adversarial patches. The attention rollout is not applicable to ResNet. As an extra check, we visualize and compare the feature maps of classifications on ResNet. The average of feature maps along the channel dimension is visualized as a mask on the original image. The visualization also supports the claims above. More visualizations are in Appendix C. Both qualitative and quantitative analysis verifies our claims that the adversarial patch can mislead the attention of DeiT by attactting it.

However, the gradient analysis is not available to compare ViT and ResNet on images with natural corrupted

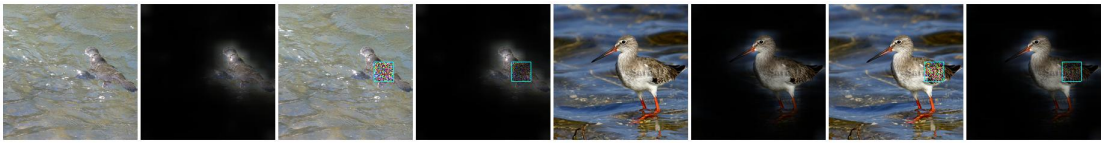


(a) Attention on ResNet18 under Adversary Patch Attack

(b) Attention on DeiT-tiny under Adversary Patch Attack

Figure 4: Attention Comparison between ResNet and DeiT under Patch Attack. The clean image, the adversarial images, and their corresponding attention are visualized. The adversary patch on DeiT attract attention, while the ones on ResNet hardly do.

(a) Attention on ResNet18 under Natural Patch Corruption



(b) Attention on DeiT-tiny under Natural Patch Corruption

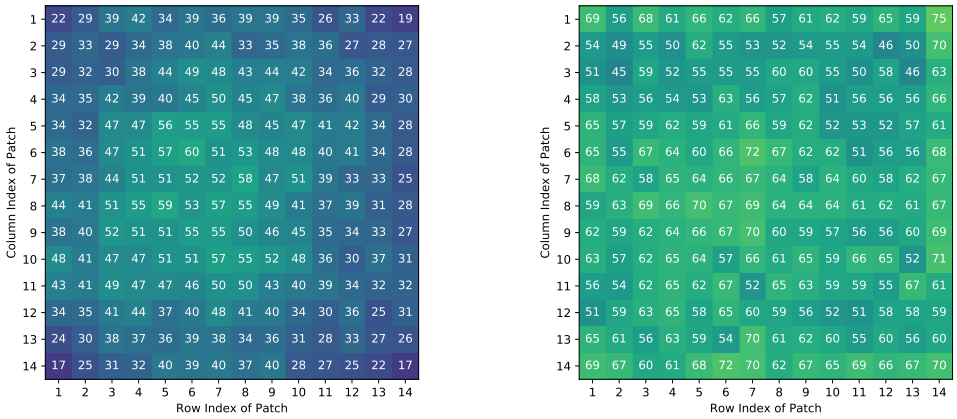
Figure 5: Attention Comparison between ResNet and DeiT under Natural Patch Corruption. The clean image, the naturally corrupted images, and their corresponding attention are visualized. The patch corruptions on DeiT are ignored by attending less to the corrupted patches, while the ones on ResNet are treated as normal patches.

patches. When a small number of patch of input images are corrupted, both Deit and ResNet are still able to classify them correctly. The slight changes are not reflected in vanilla gradients since they are noisy. When a large area of the input image is corrupted, the gradient is very noisy and semantically not meaningful. Due to the lack of a fair visualization tool to compare DeiT and ResNet on naturally corrupted images, we apply Attention Rollout to DeiT and Feature Map Attention visualization to ResNet for comparing the their attention.

The attention visualization of these images is shown in Fig. 5. We can observe that ResNet treats the naturally corrupted patches as normal ones. The attention of ResNet on natually patch-corrupted images is almost the same as that on the clean ones. Unlike CNNs, DeiT attends less to the corrupted patches when they cover the main object. When the corrupted patches are placed in the background, the main attention of DeiT is still kept on the main object. More figures are in Appendix D.

5.2 How Sensitive Is ViT Vulnerability to Attack Patch Positions?

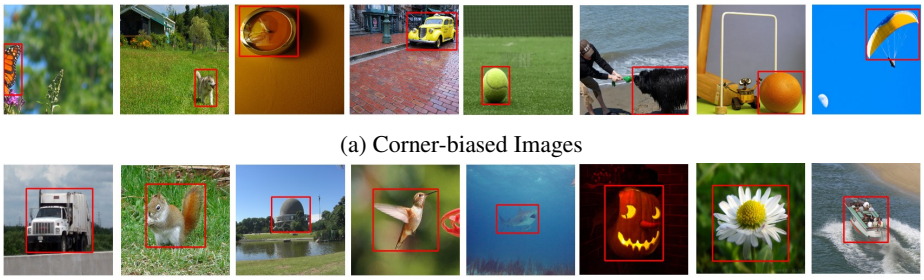
To investigate the sensitivity against the location of adversarial patch, we visualize the FR on each patch position in Fig. 6. We can clearly see that adversarial patch achieves higher FR when attacking DeiT-tiny than ResNet18 in different patch positions. Interestingly, we find that the FRs in different patch positions of



(a) Patch Attack FRs on ResNet18

(b) Patch Attack FRs on DeiT-tiny

Figure 6: Patch Attack FR (in %) in each patch position is visualized. FRs in different patch positions of DeiT-tiny are similar, while the ones in ResNet18 are center-clustered.



(a) Corner-biased Images

(b) Center-biased Images

Figure 7: Collection of two sets of biased data. The fist set contains only images with corner-biased object(s), and the other set contains center-biased images.

DeiT-tiny are similar, while the ones in ResNet18 are center-clustered. A similar pattern is also found on DeiT-small and ResNet50 in Appendix E.

Considering that ImageNet are center-biased where the main objects are often in the center of the images, we cannot attribute the different patterns to the model architecture difference without further investigation. Hence, we design the following experiments to disentangle the two factors, *i.e.*, model architecture and data bias. Specifically, we select two sets of correctly classified images from ImageNet 1K validation dataset. As shown in Fig. 7a, the first set contains images with corner bias where the main object(s) is in the image corners. In contrast, the second set is more center-biased where the main object(s) is exactly in the central areas, as shown in Fig. 7b.

We apply patch attack to corner-biased images (*i.e.*, the first set) on ResNet. The FRs of patches in the center area are still significantly higher than the ones in the corner (See Appendix F). Based on this, we can conclude that such a relation of FRs to patch position on ResNet is caused by ResNet architectures instead of data bias. The reason behind this might be that pixels in the center can affect more neurons of ResNet than the ones in corners.

Similarly, we also apply patch attack to center-biased images (the second set) on DeiT. We observe that the FRs of all patch positions are still similar even the input data are highly center-biased (See Appendix G).

Table 4: Transferability of adversarial patch across different patch positions of the the image. Translation X/Y stands for the number of pixels shifted in rows or columns. When they are shifted to cover other patches exactly, adversarial patches transfer well, otherwise not.

Trans-(X,Y)	(0, 1)	**(0, 16)**	**(0, 32)**	(1, 0)	**(16, 0)**	**(32, 0)**	(1, 1)	**(16, 16)**
ResNet50	0.06	0.31	0.48	0.06	0.18	0.40	0.08	0.35
DeiT-small	0.27	**8.43**	**4.26**	0.28	**8.13**	**3.88**	0.21	**4.97**
ResNet18	0.22	0.46	0.56	0.19	0.49	0.68	0.15	0.49
DeiT-tiny	2.54	**29.15**	**18.19**	2.30	**28.37**	**17.32**	2.11	**21.23**

Hence, we draw the conclusion that DeiT shows similar sensitivity to different input patches regardless of the content of the image. We conjecture it can be explained by the architecture trait of ViT, in which each patch equally interact with other patches regardless of its position.

5.3 Are Adversarial Patches on ViT Still Effective When Shifted?

The work [21] shows that the adversarial patch created on an image on ResNet is not effective anymore when shifted even a single pixel away. We also conduct similar experiments on DeiT. We find that the adversarial patch perturbation on DeiT does not transfer well either when only shifted a single-pixel away. However, when shifted to match another input patch exactly, the adversarial patch is still highly effective, as shown in Tab. 4.

Namely, the adversarial perturbation can be still effective when aligned with a different patch. The reason behind this is that, when the adversarial patch is switched to another patch, the network attention can still be misled as shown in Tab. 5. When shifted in a single pixel, the structure of perturbation is destroyed due to the patch split of DeiT. Additionally, We find that the adversarial patch perturbation barely transfers across images or models regardless of the alignment. Details can be found in Appendix H.

6 Improving ViT Robustness to Adversarial Patch

Given an input image $\boldsymbol{x} \in \mathbb{R}^{H\times W\times C}$, ViT [10] first reshapes the input $\boldsymbol{x}$ into a sequence of image patches $\{\boldsymbol{x}_i \in \mathbb{R}^{(\frac{H}{P}\cdot\frac{W}{P})\times(P^2\cdot C)}\}_{i=1}^{N}$ where P is the patch size and N is the number of patches. A class-token patch is concatenated to the patch sequence. A set of self-attention blocks is applied to obtain patch embeddings of the l-th block $\{\boldsymbol{x}_i^l\}_{i=1}^{N}$. The class-token patch embedding of the last block is mapped to the output.

The patch embedding of the i-th patch in the l-th layer is the weighted sum of all patch embedding $\{\boldsymbol{x}_j^{l-1}\}_{i=0}^{N}$ of the previous layer. The weights are the attention weights obtained from the attention module. Formally, the patch embedding $\boldsymbol{x}_i^l$ is computed with following equation

$$\boldsymbol{x}_i^l = \sum_{j=0}^{N} \alpha_{ij}\cdot \boldsymbol{x}_j^{l-1}, \qquad \alpha_{ij} = \frac{\exp(Z_{ij})}{\sum_{j=0}^{N}\exp(Z_{ij})} \tag{1}$$

where α_{ij} is the attention weight that stands for the attention of the i-th patch of the l-th layer to the j-th patch of the $(l\text{-}1)$-th layer. Z_{ij} is the scaled dot-product between the key of the j-th patch and the query of of the i-th patch in the $(l\text{-}1)$-th layer, i.e., the logits before *softmax* attention.

Given a classification, we denote the patch embedding of the clean image as $\boldsymbol{x}_i^{*l}$. When the $k-$th patch is attacked, the patch embedding of the i-th patch in the l-th layer deviates from $\boldsymbol{x}_i^{*l}$. The deviation distance is described as

$$d(\boldsymbol{x}_i^l, \boldsymbol{x}_i^{*l}) = \sum_{j=0}^{N}\alpha_{ij}\cdot\boldsymbol{x}_j^{l-1} - \sum_{j=0}^{N}\alpha_{ij}^{*}\cdot\boldsymbol{x}_j^{l-1}, \tag{2}$$

where α^*_{ij} is the attention weight corresponding to the clean image. Our analysis show that the attention is misled to focus on the attacked patch. In other words, α_{ik} is close to 1, and other attention weights are close to zero.

The original attention can be replaced by smoothed attention with temperature scaling in the *softmax* operation. Formally, the smoothed attention is

$$\alpha_{ij} = \frac{\exp(Z_{ij}/T)}{\sum_{j=0}^{N} \exp(Z_{ij}/T)}, \tag{3}$$

where $T(> 1)$ the hyper-parameter that determines the smoothness of the proposed attention. With the smoothed attention, the deviation of the patch embedding from the clean patch embedding is smaller (see proof in Appendix).

$$d(\boldsymbol{x}_i^{\Diamond l}, \boldsymbol{x}_i^{*l}) = \sum_{j=0}^{N} \alpha_{ij}^{\Diamond l} \cdot \boldsymbol{x}_j^{l-1} - \sum_{j=0}^{N} \alpha_{ij}^{*} \cdot \boldsymbol{x}_j^{l-1} < d(\boldsymbol{x}_i^{l}, \boldsymbol{x}_i^{*l}) \tag{4}$$

The smoothed attention forces self-attention not to focus on a single patch. By doing this, ViT becomes more robust to adversarial patches. We apply the method to ViT and report the results in Fig. 8. Under different temperatures, the smoothed attention can improve the adversarial robustness of ViT to adversarial patches and hardly reduce the clean accuracy. Note that we do not claim attention smoothing as a defense method against various patch attacks, which is not the focus of this paper. Our analysis with smoothed attention mainly aims to further verify our understanding of ViT.

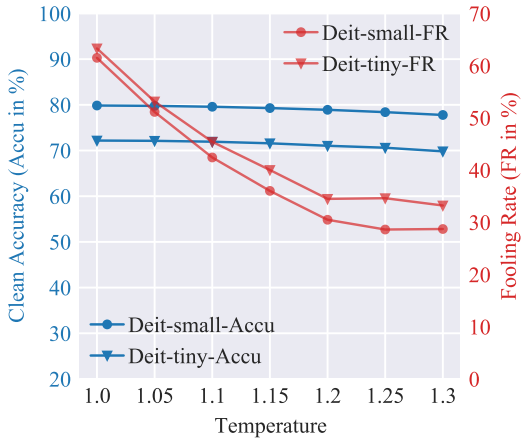


Figure 8: Improving robustness of ViT with Smoothed Attention.

7 Discussion

In previous sections, we leverage the state-of-the-art patch attack method to study the most primary ViT architecture and ResNet. In this section, we present our further investigation into different model variants and different patch attacks.

Investigation into More Models. Other than the architectures presented in the main paper, we also studied different versions of ViT [10, 25, 46], CNN [16, 19] as well as Hybrid architectures [14]. Following the experimental setting in section 3, we train all the models and report fooling rate on each model in Fig. 9. Four main conclusions can be drawn from the figure.

1). CNN variants are more robust than ViT models. 2). The robustness of LeViT model [14] with hybrid architecture (*i.e.*, Conv Layers + Self-Attention Blocks) lives somewhere between ViT and CNNs, as expected. 3). Swin Transformers [25] are as robust as CNNs since attention cannot be manipulated by a single patch due to hierarchical attention and the shifted windows therein. The self-attention in Swin Transformers is only conducted on patches within a local region. With shifted windows, a single patch will interact with patches from different groups in different layers. Both designs make effective adversarial patches challenging. That's the reason why Swin Transformer performs more robustly than popular ViTs. 4). Mixer-MLP [45] uses the same patch-based architecture as ViTs and has no attention module. Mixer-base with FR (31.36) is comparable to ResNet and more robust than ViTs. The results confirm that the vulnerability of ViT can be attributed to self-attention mechanism.

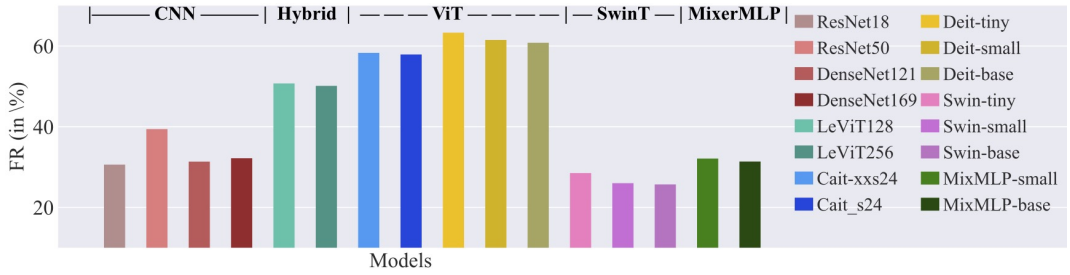


Figure 9: We report Fooling Rates on different versions of ViT, CNN as well as Hybrid architectures under Adversarial Patch Attack.

The attention smoothing by temperature scaling can improve the robustness of DeiT and Levit. Meanwhile, the improvement on Swin Transformers is only tiny since they will not focus on the single adversarial patch by natural design with the original hierarchical attention.

Investigation into More Patch Attacks. 1) Imperceptible Patch Attack In this work, we use unbounded local patch attacks where the pixel intensity can be set to any value in the image range $[0, 1]$. The adversarial patches are often visible, as shown in Fig. 1. In a more popular setting of adversarial attack and defense, the maximally allowed change of the input value is 8/225, in which the adversarial perturbation is imperceptible human vision. We also compare ResNet and DeiT under this setting.

In the case of a single patch attack, the attacker achieves FR of 2.9% on ResNet18 and 11.2% on DeiT-tiny. More scores and visualization of the images with imperceptible perturbation can be found in Appendix I. DeiT is still more vulnerable than ResNet when attacked with imperceptible patch perturbation. When the patch size to attack is set to be the whole image size, it is exactly the same as the standard attack. We show that both ResNet and DeiT can be easily fooled When the standard attack setting is applied.

2) Targeted Patch Attack. Targeted attack can be achieved by setting the attack objective to maximize the probability of the target class. We also compare DeiT and ResNet under the targeted attack above. In the experiment, we randomly select a target class except for the ground-truth class for each image. In the case of a single attack patch, the attacker achieves FR of 15.4% on ResNet18 and 32.3% on DeiT-tiny. Under targeted attack, DeiT is more vulnerable than ResNet. The claim also holds on the other model pair (ResNet50 7.4% vs. DeiT-small 24.9%). Visualization of adversarial patches is in Appendix J.

3) Patch Attack with Different Strength. In our experiment, as in [21], the attack iteration is set to 10k. We also check how many iterations are required to attack the classification successfully. The required iterations are averaged on all patch positions of the misclassified images. The required attack iterations on DeiT-tiny is less than that on ResNet18 (65 vs. 342). The observation also holds on DeiT-small and ResNet50 (294 vs. 455). This experiment shows DeiT is more vulnerable than ResNet from another perspective.

4) ViT (non)-specific Patch Attacks. When the adversarial patch is aligned perfectly with an ViT input patch, the patch attack can be seen as an instance of ViT-specific patch attack since there is no input patch in ResNet. As reported in Tab. 2, ViT is more vulnerable than ResNet. We also study ViT-agnostic patch attack where the adversarial patch of the same size as an input patch is placed to a random area of the image. The covered area can involve pixels from multiple input patches. We find that DeiT becomes less vulnerable to adversarial patch attack, *e.g.*, the FR on DeiT-small decreases from 61.5% to 47.9%. When the adversarial patch is not aligned with the input patch, *i.e.*, only part of patch pixels can be manipulated, the attention of DeiT is less likely to be misled. Under such ViT-agnostic patch attack, ViT is still more vulnerable than ResNet.

8 Conclusion

This work first shows an interesting observation on the robustness of ViT to patch perturbations. Namely, vision transformer (e.g., DeiT) is more robust to natural patch corruption than ResNet, whereas it is significantly more vulnerable against adversarial patches. A deep understanding of the observation is then provided. We reveal that the self-attention mechanism of ViT can effectively ignore natural corrupted patches but be easily misled to adversarial patches to make mistakes. Based on our analysis, we show attention smoothing can improve the robustness of ViT to adversarial patches. We hope this study can help the community better understand the robustness of ViT to patch perturbations.

References

[1] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[2] Ahmed Aldahdooh, Wassim Hamidouche, and Olivier Deforges. Reveal of vision transformers robustness against adversarial attacks. *arXiv:2106.03734*, 2021.

[3] Yutong Bai, Jieru Mei, Alan Yuille, and Cihang Xie. Are transformers more robust than cnns? *NeurIPS*, 2021.

[4] Philipp Benz, Soomin Ham, Chaoning Zhang, Adil Karjauv, and In So Kweon. Adversarial robustness comparison of vision transformer and mlp-mixer to cnns. *BMVC*, 2021.

[5] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. *ICCV*, 2021.

[6] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv:1712.09665v1*, 2017.

[7] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *ICCV*, 2021.

[8] Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. Visformer: The vision-friendly transformer. *ICCV*, 2021.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2020.

[11] Alhussein Fawzi and Pascal Frossard. Measuring the effect of nuisance variables on classifiers. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[12] Yonggan Fu, Shunyao Zhang, Shang Wu, Cheng Wan, and Yingyan Lin. Patch-fool: Are vision transformers always robust against adversarial perturbations? In *International Conference on Learning Representations*, 2021.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[14] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *ICCV*, 2021.

[15] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv:2103.00112*, 2021.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

[18] Haoqi Hu, Xiaofeng Lu, Xinpeng Zhang, Tianxing Zhang, and Guangling Sun. Inheritance attention matrix-based universal adversarial perturbations on vision transformers. *IEEE Signal Processing Letters*, 28:1923–1927, 2021.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[20] Ameya Joshi, Gauri Jagatap, and Chinmay Hegde. Adversarial token attacks on vision transformers. *arXiv:2110.04337*, 2021.

[21] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *International Conference on Machine Learning (ICML)*, 2018.

[22] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European Conference on Computer Vision (ECCV)*, 2020.

[23] Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

[24] Aishan Liu, Jiakai Wang, Xianglong Liu, Bowen Cao, Chongzhi Zhang, and Hang Yu. Bias-based universal adversarial patch attack for automatic check-out. In *European conference on computer vision*, pages 395–410. Springer, 2020.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021.

[26] Jinqi Luo, Tao Bai, and Jun Zhao. Generating adversarial yet inconspicuous patches with a single image (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15837–15838, 2021.

[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[28] Kaleel Mahmood, Rigel Mahmood, and Marten Van Dijk. On the robustness of vision transformers to adversarial examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7838–7847, 2021.

[29] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Ranjie Duan, Shaokai Ye, Yuan He, and Hui Xue. Towards robust vision transformer. *CVPR*, 2022.

[30] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Shaokai Ye, Yuan He, and Hui Xue. Rethinking the design principles of robust vision transformer. *arXiv:2105.07926*, 2021.

[31] Jan Hendrik Metzen, Nicole Finnie, and Robin Hutmacher. Meta adversarial training against universal patches. *arXiv preprint arXiv:2101.11453*, 2021.

[32] Norman Mu and David Wagner. Defending against adversarial patches with robust self-attention. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

[33] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *NeurIPS*, 2021.

[34] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Fahad Shahbaz Khan, and Fatih Porikli. On improving adversarial transferability of vision transformers. *ICLR*, 2022.

[35] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, 2016.

[36] Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners. *arXiv:2105.07581*, 2021.

[37] Yaguan Qian, Jiamin Wang, Bin Wang, Shaoning Zeng, Zhaoquan Gu, Shouling Ji, and Wassim Swaileh. Visually imperceptible adversarial patch attacks on digital images. *arXiv preprint arXiv:2012.00909*, 2020.

[38] Hadi Salman, Saachi Jain, Eric Wong, and Aleksander Mądry. Certified patch robustness via smoothed vision transformers. *arXiv:2110.07719*, 2021.

[39] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

[40] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. On the adversarial robustness of visual transformers. *arXiv:2103.15670*, 2021.

[41] Yucheng Shi and Yahong Han. Decision-based black-box attack against vision transformers via patch-

wise adversarial removal. *arXiv preprint arXiv:2112.03492*, 2021.

[42] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning (ICML)*, 2017.

[43] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *International Conference on Learning Representations (ICLR)*, 2014.

[44] Shiyu Tang, Ruihao Gong, Yan Wang, Aishan Liu, Jiakai Wang, Xinyun Chen, Fengwei Yu, Xianglong Liu, Dawn Song, Alan Yuille, et al. Robustart: Benchmarking robustness on architecture design and training techniques. *TPAMI*, 2021.

[45] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, 2021.

[46] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021.

[47] Jiakai Wang, Aishan Liu, Xiao Bai, and Xianglong Liu. Universal adversarial patch attack for automatic checkout using perceptual and attentional bias. *IEEE Transactions on Image Processing*, 31:598–611, 2021.

[48] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv:2006.03677*, 2020.

[49] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *arXiv:2106.14881*, 2021.

[50] Zhongzhi Yu, Yonggan Fu, Sicheng Li, Chaojian Li, and Yingyan Lin. Mia-former: Efficient and robust vision transformers via multi-grained input-adaptation. *arXiv preprint arXiv:2112.11542*, 2021.

[51] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 2014.

[52] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

A Training Setting Affect Model Robustness

We train ResNet18 on CIFAR10 in the standard setting [16]. To study the impact of training settings on model robustness, we train models with different input sizes (i.e., 32, 48, 64), with or without Weight Standardization and Group Normalization to regularize the training process. The foolong rate of single patch attack is reported. Especially, with our experiments, we find that Weight Standardization and Group Normalization can have a significant impact on model robustness (See Tab. 5). The two techniques are applied in BiT [22] to improve its performance. However, they are not applied to standard ViT and DeiT training settings. Hence, the robustness difference between ViT and BiT cannot be attributed to the difference between model architectures.

Note that a comprehensive study of the relationship between all factors of training and model adversarial robustness is out of the scope of this paper. We aim to point out that these factors can have an impact on model robustness to different extents. The robustness difference cannot be blindly attributed to the difference of model architectures. We need to build new fair base models to study the robustness of ResNet and ViT.

Table 5: Study of the training factors on the relation to model robustness: While the input size has minor impact on model robustness in the first tabular, Weight Standardization (WS) and Group Normalization (GN) can change model robustness significantly in the second tabular.

Model	Input Size		
ResNet18	32	48	64
Clean Accuracy	93.4	93.8	93.7
FR of Patch Attack	35.9	42.2	39.2

Model	Training Techniques			
ResNet18	No	WS	GN	WS + GN
Clean Accu	93.4	93.6	92.0	93.8
Patch Attack FR	35.9	51.3	52.6	71.1

Table 6: Fair base models. DeiT and counter-part ResNet are trained with the exact same setting. Two models of each pair achieve similar clean accuracy with comparable model sizes.

Model	Model Size	Clean Accuracy
ResNet50	25M	78.79
DeiT-small	22M	79.85
ResNet18	12M	69.39
DeiT-tiny	5M	72.18

B Gradient Visualization of Adversarial Images under Patch Attack

We first get the absolute value of gradient received by input and sum them across the channel dimension. The final values are mapped into gray image scale. We also mark the adversarial patch with a blue bounding box in the visualized gradient maps.

The adversarial patch noises with different patch size of 32 are shown on DeiT and ResNet in Fig. 14, 15. In each row of these figures, we fist show the clean image and visualize the gradients of inputs as a mask on the image. Then, we show the images with patch noises on different patch positions, and the gradient masks are also shown following the corresponding adversarial images.

C More Figures of Attention on Different Patch Sizes and Positions

In this appendix section, we show more Attention Rollout on DeiT and Feature Map Masks on ResNet. The adversarial patch noises are shown (i.e., P=32) in Fig. 16 and 17. In each row of these figures, we fist show the clean image and visualize the attention as a mask on the image. Then, we show the images with patch noises

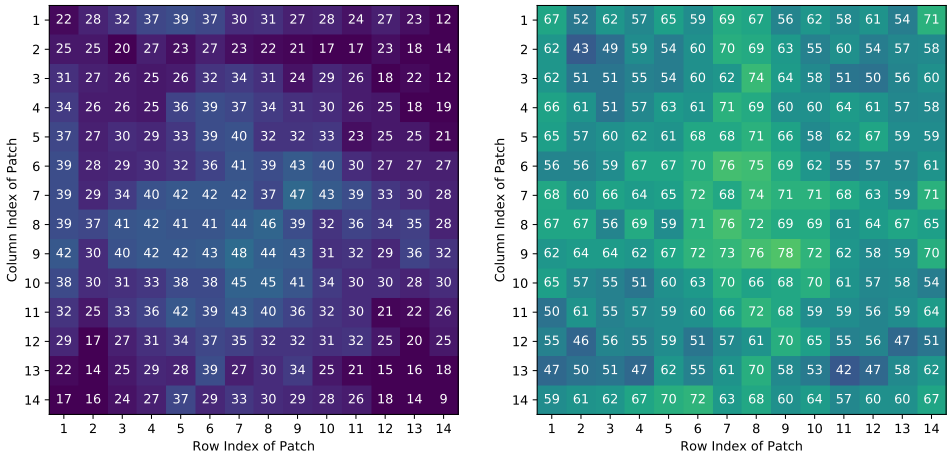
on different patch positions, and the attention masks are also shown following the correspond adversarial images.

D Attention under Natural Patch Corruption and Adversarial Patch Attack

The rollout attention on DeiT and Feature Map mask on ResNet on naturally corrupted images are shown in Fig. 18 and 19. We can observe that ResNet treats tha corrupted patches as normal ones. On DeiT, the attention is slightly distract by naturally corrupted patches when they are in the background. However, the main attention is still on the main object of input.

E Fooling Rates of Each Patch on ResNet50 and DeiT-small

The FRs in different patch positions of DeiT are similar, while the ones in ResNet are center-clustered. A similar pattern can also be found on DeiT-small and ResNet50 in Fig. 10.



(a) Adversarial Patch Attack FRs on ResNet50 (b) Adversarial Patch Attack FRs on DeiT-small

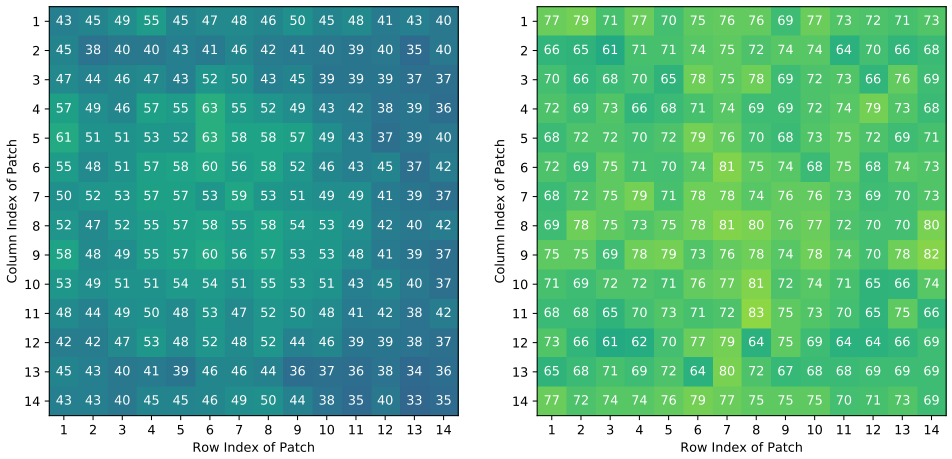
Figure 10: Patch Attack FR (in %) in each patch position is visualized on ResNet50 and DeiT-small.

F Fooling Rates of Each Patch on ResNet and DeiT on Corner-biased Data

In the coner-biased image set, the FR on ResNet is still center-clustered, as shown in Fig. 11a.

G Fooling Rates of Each Patch on ResNet and DeiT on Center-biased Data

In the center-biased image set, the FR on DeiT is still similar on different patch postions, as shown in Fig. 11b.



(a) FRs of ResNet18 on Corner-biased Data (b) FRs of DeiT-tiny on Center-biased Data

Figure 11: Patch Attack FR (in %) in each patch position is visualized on ResNet18 and DeiT-tiny on biased data.

H Transferability of Adversarial Patches across Images, Models, and Patch Positions

As shown in Tab. 7, the adversarial patch noise created on a given image hardly transfer to other images. When large patch size is applied, the patch noises on DeiT transfer slightly better than the ones on ResNet.

Table 7: Transferability of adversarial patch across images

Models	ResNet50	DeiT-small	ResNet18	DeiT-tiny
across images (Patch Size=16)	3.5	2.1	3.4	6.4
across images (Patch Size=112)	8.1	13.4	10.6	21.5

The transferbility of adversrial noise between Vision Transformer and ResNet has already explored in a few works. They show that the transferability between them is remarkablely low. As shown in Tab. 8, the adversarial patch noise created on a given image does not transfer to other models.

Table 8: Transferability of adversarial patch across models

	Patch Size=16							
Models	ResNet50	DeiT-small	ResNet18	DeiT-tiny	ResNet50	DeiT-small	ResNet18	DeiT-tiny
ResNet50	-	0.3	0.16	2.2	-	5.25	8	11.75
DeiT-small	0.04	-	0.09	1.79	5.5	-	9.25	12.25
ResNet18	0.09	0.22	-	1.9	5.75	5	-	12
DeiT-tiny	0.04	0.13	0.06	-	5.5	5	9.25	-

When they are transfered to another patch, the adversarial patch noises are still highly effective. However, the transferability of patch noise can be low, when the patch is not aligned with input patches. The claim on the patch noise with size of 112 is also true, as shown in Tab. 9.

Table 9: Transferability of adversarial patch across patch positions

Model	ResNet50	DeiT-small	ResNet18	DeiT-tiny
across positions (0, 4)	6.25	5.25	11.25	12.75
across positions (0, 16)	5.75	**34.5**	11.5	**54**
across positions (0, 64)	6	22	9.5	30.75
across positions (4, 0)	6.5	5.75	9.75	12.5
across positions (16, 0)	7.25	**35**	10.25	**54**
across positions (64, 0)	5.5	18.25	9.25	31
across positions (4, 4)	6	4.75	8.5	13.5
across positions (16, 16)	4.5	**18.5**	9	**33**
across positions (64, 64)	6	9.75	8.25	17.5

I More Settings and Visualization of Adversarial Examples with Imperceptible Noise

In the standard adversarial attack, the artificial noise can be placed anywhere in the image. In our adversarial patch attack, we conduct experiments with different patch sizes, which are multiple times the size of a single patch. The robust accuracy under different attack patch sizes is reported in Tab. 10. We can observe that DeiT is more vulnerable than ResNet under imperceptible attacks.

Table 10: Adversarial Patch Attack with Imperceptible Perturbation . FRs are reported in percentage.

Model	PatchSize=16	PatchSize=32	PatchSize=112	PatchSize=224
ResNet50	2.9	20.9	98.3	100
DeiT-small	4.1	38.7	100	100
ResNet18	3.1	26.0	99.1	100
DeiT-tiny	11.2	46.8	100	100

The clean images and the adversarial images created on different models are shown in Fig. 12. The adversarial perturbations created with imperceptible patch attack are imperceptible for human vision.

J Visualization of Adversarial Patch Noise

Besides reporting the FRs, we also visualize the adversarial patch perturbation created on ResNet and DeiT. The adversarial patch perturbation are shown in Fig. 13a and 13c. We are not able to recognize any object in the target class.

Following Karmon et al. 's LaVAN, we enhance the attack algorithm where we place the patch noise on different patch positions in different images in each attack iteration. From the visualization of the created noise in Fig. 13b and 13d, we can recognize the object/object parts of the target class on both ResNet and DeiT. In this section, we conclude that the recognizability of adversarial patch noise is dependent more on attack algorithms than the model architectures.

(a) Clean Images

(b) Adversarial Examples on ResNet18

(c) Adversarial Examples on DeiT-tiny

(d) Adversarial Examples on ResNet50

(e) Adversarial Examples on DeiT-small

Figure 12: Visualization of Adversarial Examples with Imperceptible Patch Noise: The adversarial images with patch noise of size 112 in the left-upper corner of the image are visualized. Please Zoom in to find the subtle difference.

(a) Patch Noise on ResNet50 under the 1st Setting

(b) Patch Noise on ResNet50 under the 2nd Setting

(c) Patch Noise on DeiT-small under the 1st Setting

(d) Patch Noise on DeiT-small under the 2nd Setting

Figure 13: Visualization of Adversarial Patch Perturbations under different Settings: In the 1st setting, the patch noise is created to fool a single classification in a given patch position. The goal in the 2nd setting to mislead the classifications of a set of images at all patch positions.

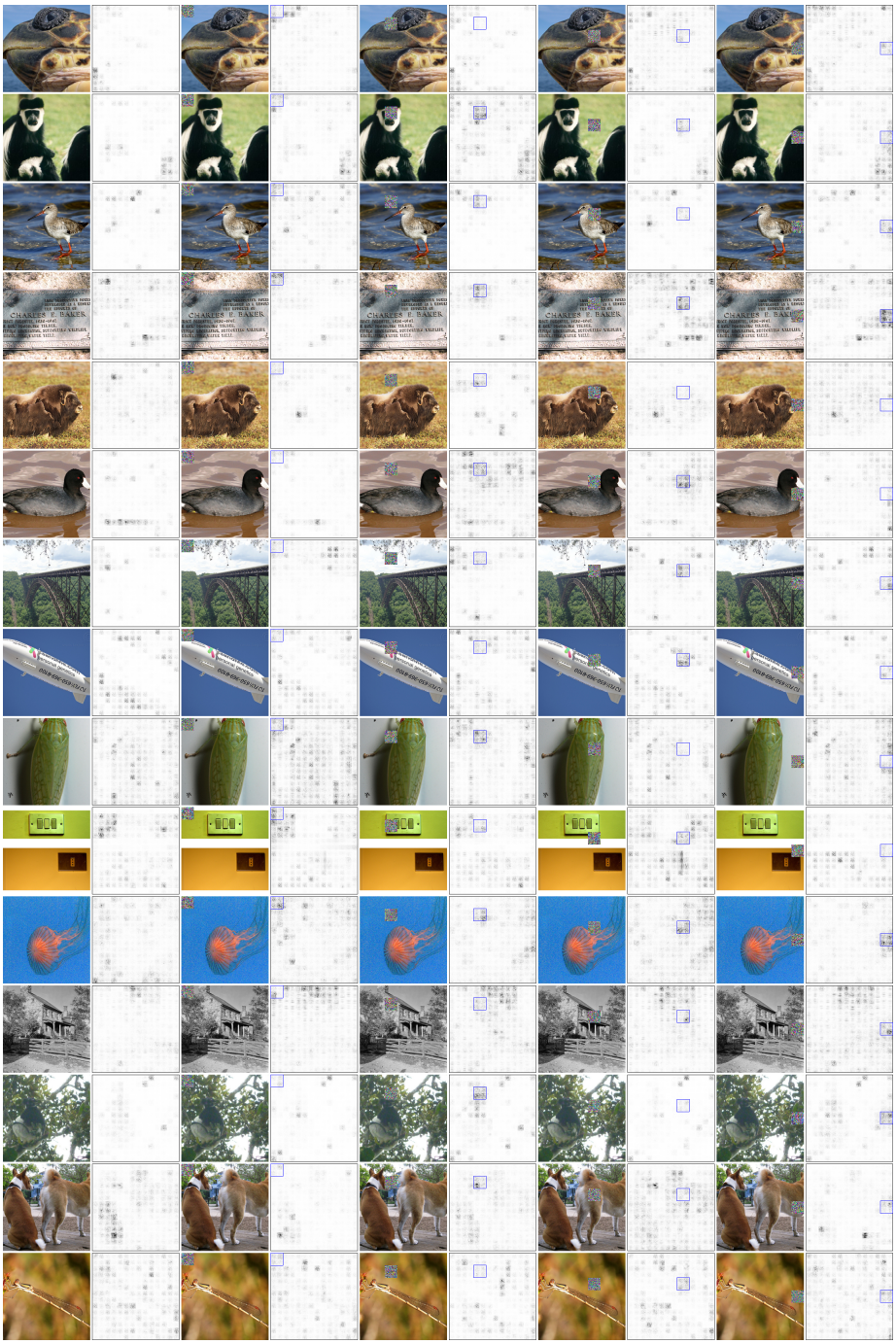


Figure 14: Gradient Visualization on DeiT-small with Attack Patch size of 32

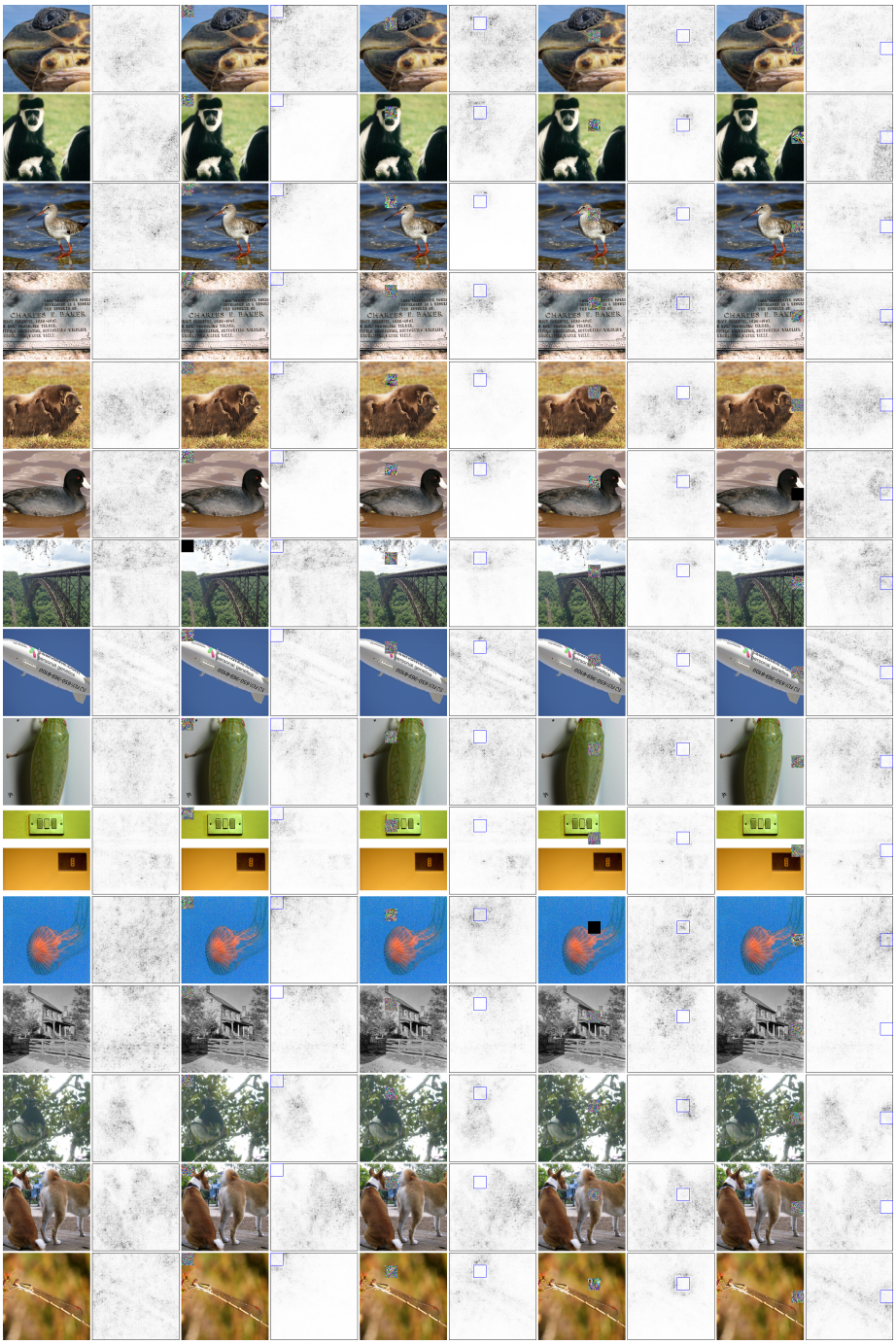


Figure 15: Gradient Visualization on ResNet50 with Attack Patch size of 32

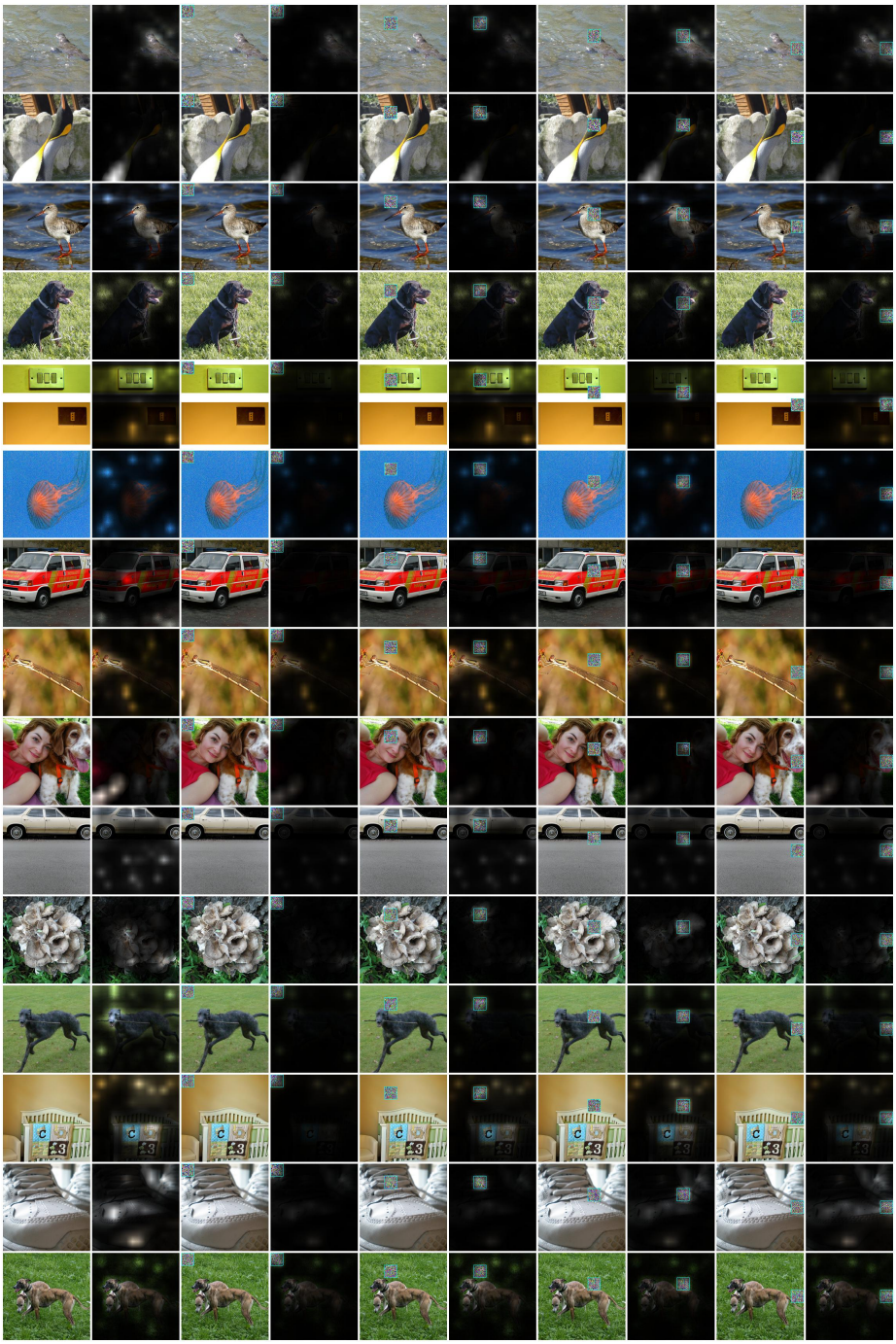


Figure 16: Rollout Attention on DeiT-small with Attack Patch size of 32 on Adversarial Images

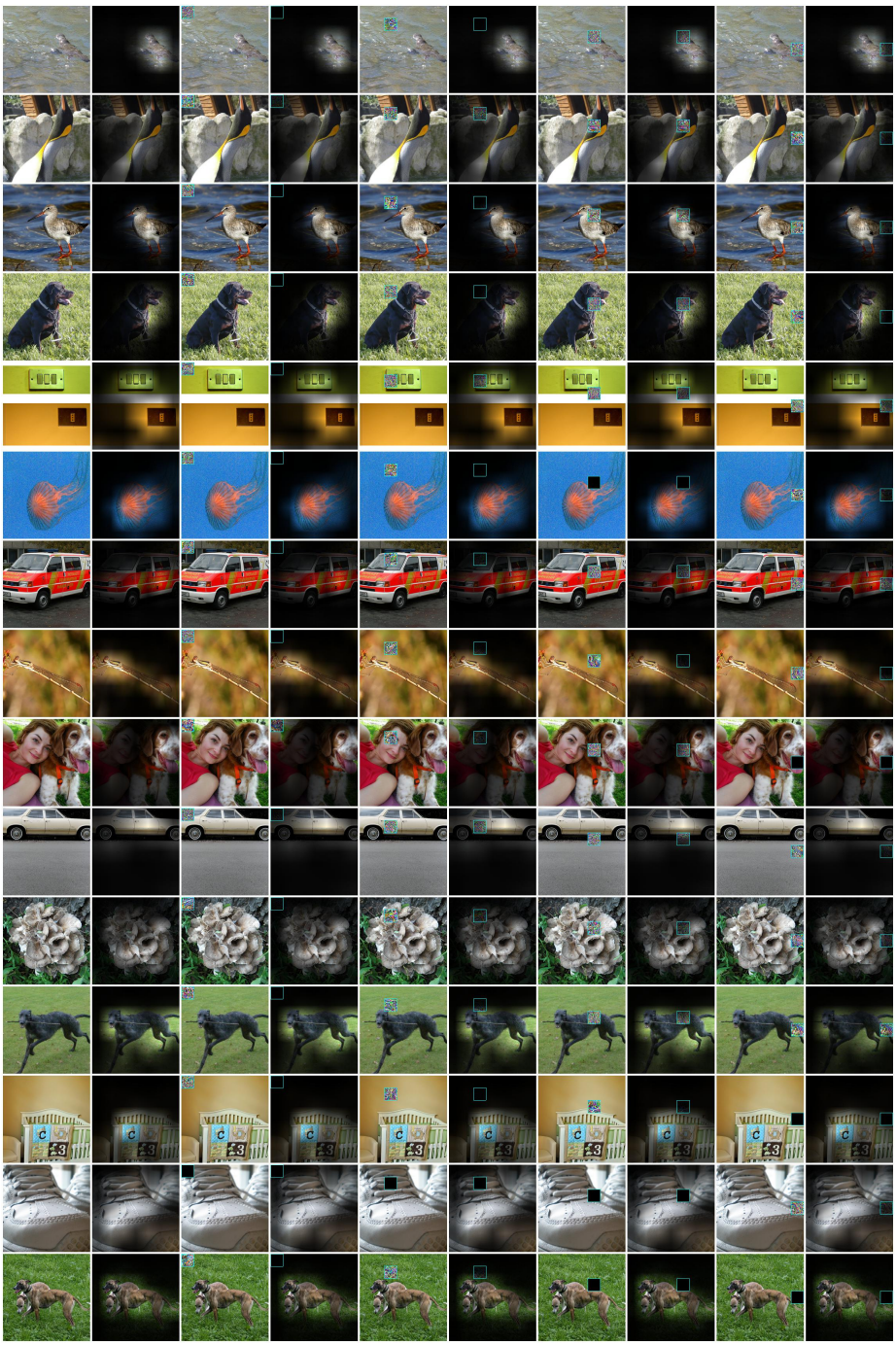


Figure 17: Averaged Feature Maps of ResNet50 as Attention with Attack Patch size of 32 on Adversarial Images

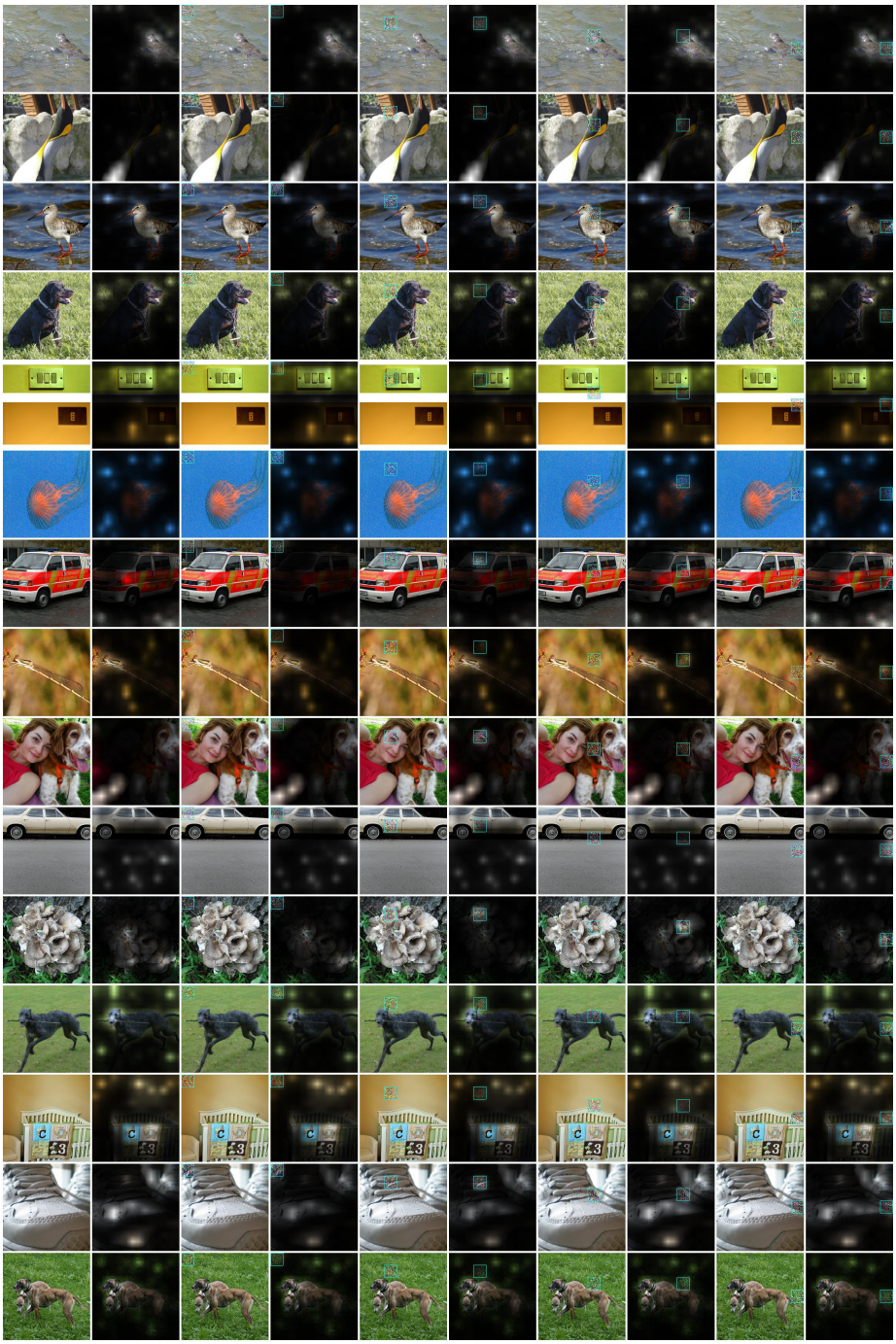


Figure 18: Rollout Attention on DeiT-small with Attack Patch size of 32 on Corrupted Images

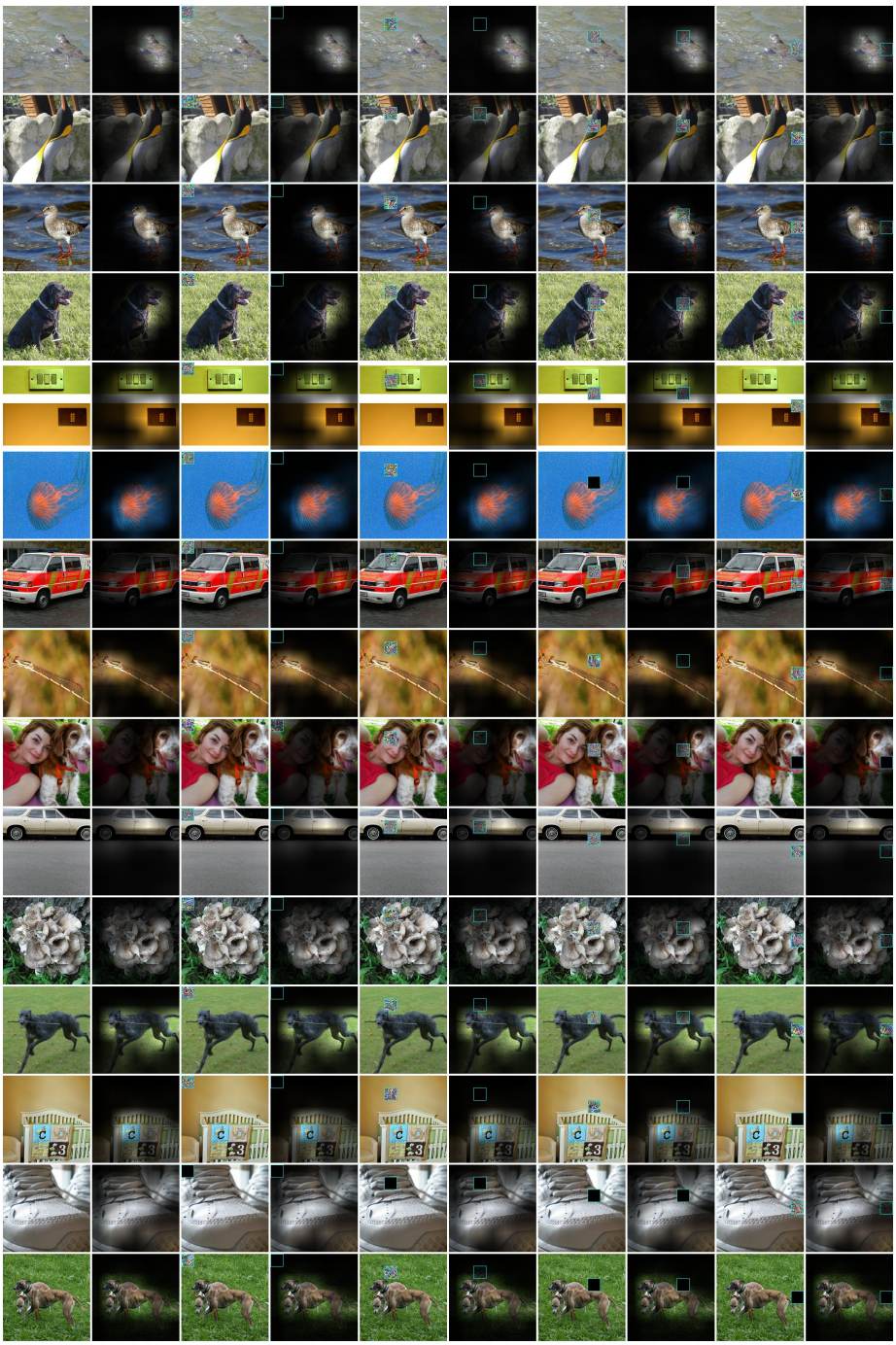


Figure 19: Averaged Feature Maps of ResNet50 as Attention with Attack Patch size of 32 on Corrupted Images

Chapter 8

Conclusion

In this dissertation, we have studied the explainability and robustness of deep visual classification models. Especially, we focus on the popular deep visual neural networks, namely, Convolutional Neural Networks (CNNs), Capsule Networks (CapsNets), and Vision Transformers (ViTs). From the perspective of explainability, we propose an explanation tool for CNNs and interpretable CapsNets and provide an understanding of ViT-based classifications. Besides, by studying the core building component of each component, e.g., dynamic routing in CapsNets and self-attention in ViTs, we reveal the weakness of their predictions when natural and adversarial perturbations are injected into input images. The contribution of this thesis will facilitate the application of existing popular deep visual classification models and inspires the development of more intelligent classifiers in the future.

In Chapter 2, we first evaluate the explanations generated by LRP and find that the generated explanations are not class-discriminative. To improve discriminativeness of the generated explanations, we propose the Contrastive Layer-wise Relevance Propagation (CLRP). Both qualitative and quantitative evaluations confirm that the CLRP is better than the LRP. As an explanation tool, our CLRP is able to generate class-discriminative, pixel-wise explanations for the individual CNN-based classification decisions.

In Chapter 3, we propose an interpretable Graph Capsule Networks (GraCapsNet). The built-in explanations for individual classifications of GraCapsNets can be created in an effective and efficient way. Surprisingly, our model also demonstrates some unexpected benefits, even though it replaces the fundamental part of CapsNets. Our GraCapsNets achieve better classification performance with fewer parameters and better adversarial robustness when compared to CapsNets. Besides, GraCapsNets still keep other advantages of CapsNets, namely, disentangled representations and affine transformation robustness.

When applied to classify images, our GraCapsNet is able to offer classification explanations in an effective and efficient way.

In Chapters 4 and 5, We first analyze the robustness of CapsNets to input affine transformation and attribute their robustness to CapsNet architecture instead of dynamic routing. Based on our exploration of the limitation of the CapsNet architecture, we propose AffCapsNets, which improves affine transformation robustness significantly using fewer parameters. Furthermore, we summarize 5 major differences between CapsNets and ConvNets and study 3 properties of CapsNets. We show that dynamic routing is harmful to CapsNets in terms of transformation robustness and semantic representations. In each presented task, a simple ConvNet can be built to outperform the CapsNet significantly. Overall, we conclude that *Dynamic Routing Capsule Network is Not More Robust than Primary Convolutional Neural Networks.*

In Chapter 6, we focus on the adversarial robustness of CapsNets. By revealing how it is affected by adversarial examples, our investigation reveals that adversarial examples can mislead CapsNets by manipulating the votes. We propose an effective and efficient Vote-Attack to attack CapsNets. The Vote-Attack is more effective and efficient than the standard Caps-Attack in both standard training and adversarial training settings. The adversarial robustness of CapsNets can be reduced to a similar level to the counterpart CNNs under our proposed Vote Attack.

In Chapter 7, based on the architectural traits of Vision Transformers, we study the following question interesting: How does ViT perform when individual input image patches are perturbed with natural corruptions or adversarial perturbations, compared to CNNs? Based on a fair comparison, we find that ViT is more robust to natural patch corruption than ResNet, whereas it is more vulnerable to adversarial patch perturbation. We conduct extensive analysis to understand our observations. Specifically, we reveal that the self-attention mechanism can effectively ignore natural corrupted patches to maintain a correct prediction but be easily fooled to focus on adversarial patches to make a mistake. Inspired by our analysis, we show attention smoothing can improve the robustness of ViT against adversarial patches since smoothed attention does not focus on a single patch. This chapter shows that self-attention boost the robustness to natural patch perturbations, but reduce the robustness to adversarial patch.

We argue that the adversarial vulnerability and the lack of explainability of deep visual classification models can be attributed to the difference between the current deep visual neural network-based classification and human visual recognition. The advanced mod-

ules of existing deep vision classification models, i.e., skip connections, dynamic routing, and self-attention, do not necessarily make image classification more aligned with human perception. In this dissertation, we develop tools to make the existing deep visual classifications more explainable and robust and reveal the large gap between the existing deep visual classifications and human-aligned object recognition. We believe that this dissertation will contribute to the application of existing popular deep visual classification models and inspire the development of more human-aligned classification models.

Different from the image classification task, it is often multi-goal oriented for our human perception to recognize objects. For example, during object recognition, we often easily locate objects, segment objects from the cluttered background, and recall the relationship of the object with other concepts.

In future work, we will study the robustness of visual structured models, e.g., object detection, semantic segmentation, and scene graph generation. Furthermore, we will explore integrating external knowledge to improve the robustness of vision systems, e.g., designing new architectures to leverage text knowledge for improving robustness. Besides the explainability and the robustness, we will also explore the efficiency of the visual systems to facilitate their applications. Especially, we are interested in improving efficiency from the perspective of cognitive perception.

Overall, my research goal is to build explainable, robust, and efficient visual systems. With my research, I am dedicated to making AI more intelligent and safer.