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

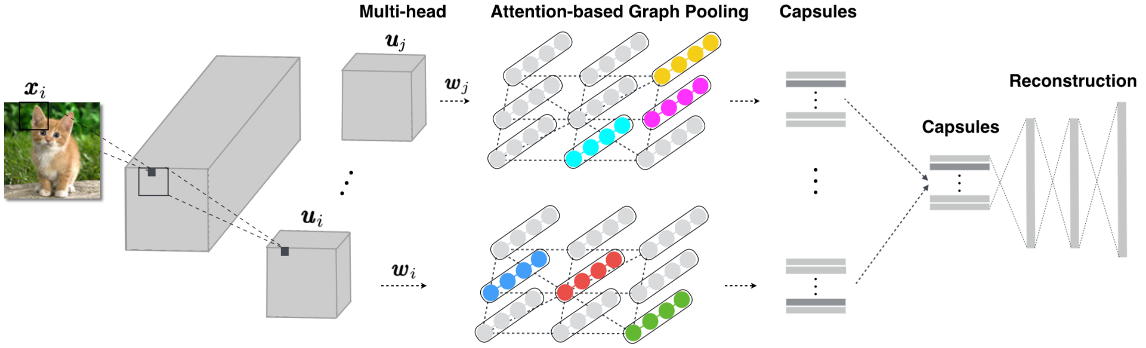


Figure 1: The illustration of GraCapsNets: The extracted primary capsules are transformed and modeled as multiple graphs. The pooling result on each graph (head) corresponds to one vote. The votes on multiple graphs (heads) are averaged to generate the final prediction.

```
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

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

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

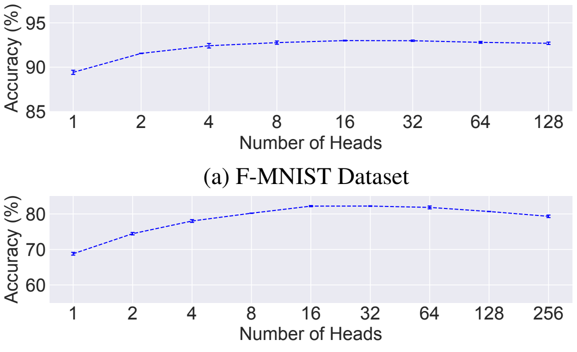
The datasets, MNIST (LeCun et al. 1998), F-MNIST (Xiao, Rasul, and Vollgraf 2017) and CIFAR10 (Krizhevsky et al. 2009), are used in this experiment. The data preprocessing, the arhictectures and the training procedure are set identically to (Sabour, Frosst, and Hinton 2017) (See Supplement A). Correspondingly, 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

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)

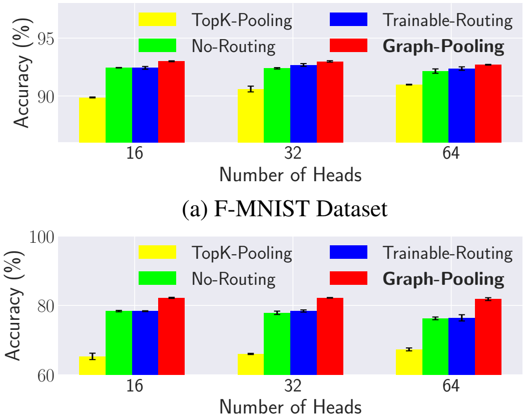
Table 2: Comparison to state-of-the-art CapsNets performance on the benchmark datasets.



(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.

with more heads achieves better performance in general. 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.

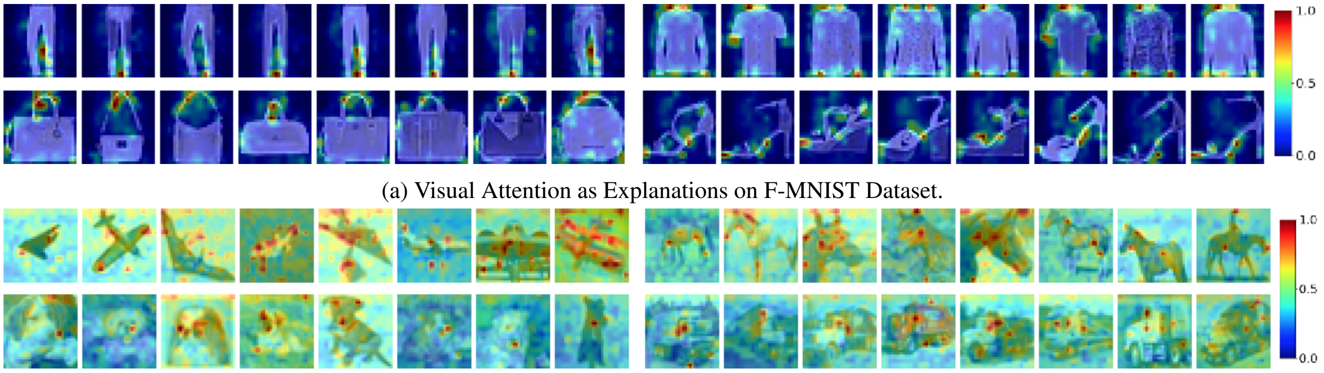
Ablation Study on Graph Pooling In GraCapsNets, we model the transformed capsules as multiple graphs. The spatial relationship between the transformed capsules is modeled in each graph. To investigate the effectiveness of the graph modeling, we compare GraCapsNets with closely related pooling operations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

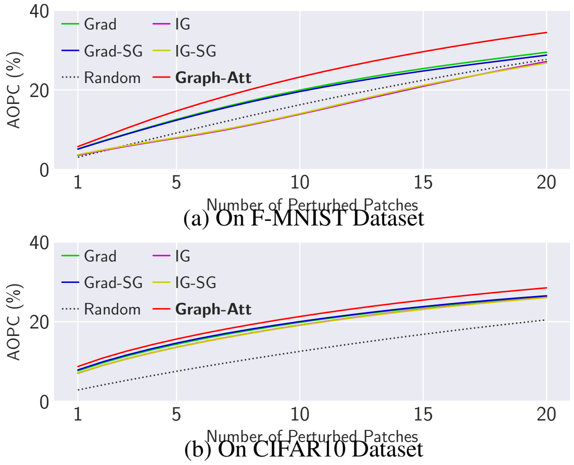
4.2 Classification Interpretability

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) (See Supplement B). 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 (see Supplement C).

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.



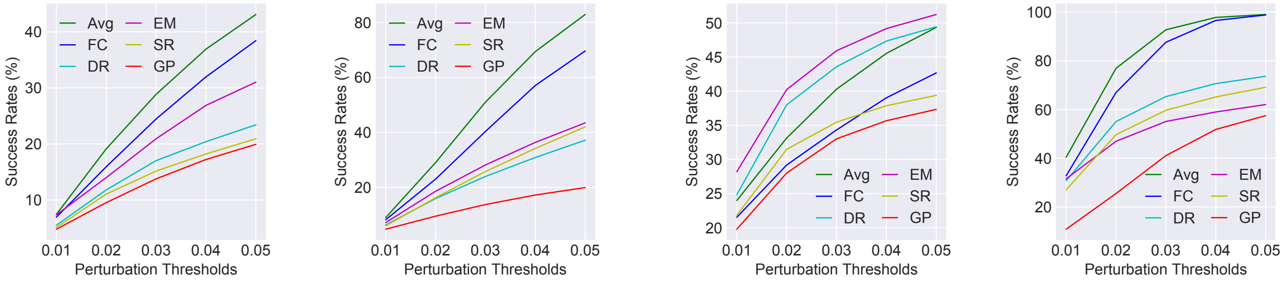
(a) On F-MNIST Dataset

(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 (more results in Supplement D). 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 Squared-Grad 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.

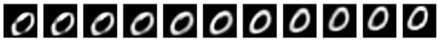
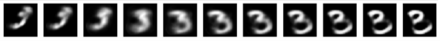
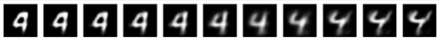
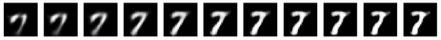
In this experiment, we use the trained models in Section 4.1. FGSM (Goodfellow, Shlens, and Szegedy 2015) (a primary attack method) and C&W (Carlini and Wagner 2017) are applied to create adversarial examples. Their hyperparameter settings are default in Adversarial Robustness 360 Toolbox (Nicolae et al. 2018) (See Supplement E). 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.

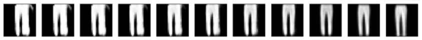
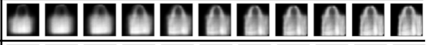
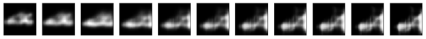
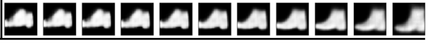
Figure 6 shows the success rates of CNNs (Avg, FC), CapsNets (DR, EM, SR) and our GraCapsNets (GP) under untargeted setting. Overall, CapsNets with various routing algorithms more robust than CNNs. Especially, when the strong attack C&W is used under a large threshold of 0.05, all the predictions of 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 (See Supplement F). 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)*.