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.

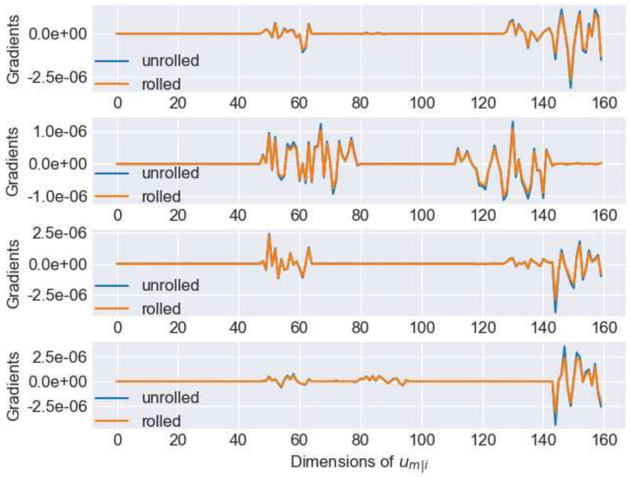


Figure 1: The gradients of the loss w.r.t. randomly choosen $\hat{\boldsymbol{u}}_{m|i}$ are visualized. The blue lines correspond to the unrolled routing iterations in Gradient Backpropagation, while the yellow lines to rolled routing iterations.

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., $\hat{\boldsymbol{u}} \to \boldsymbol{C}^{(K)} = \{c_{ij}^{(K)}\}$ 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.

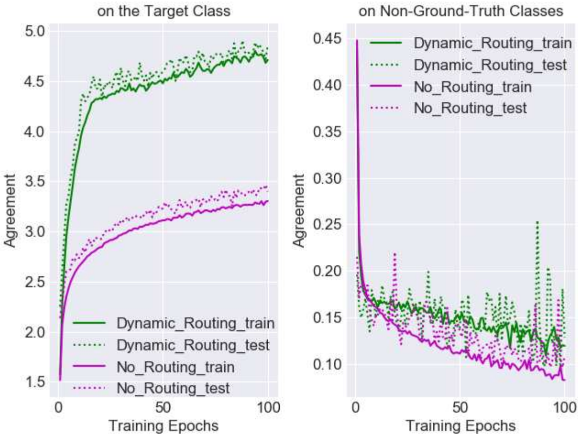


Figure 2: The green lines correspond to the model with dynamic routing, while the magenta ones to the model without routing procedure. For both models, the agreement on the target class increases with training time, and the agreement on the non-ground-truth classes decreases. The values are averaged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

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

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

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

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.

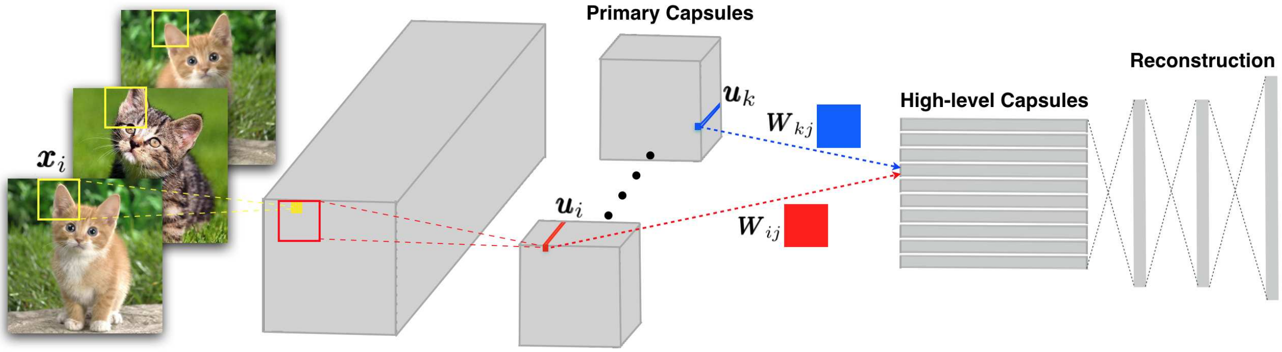


Figure 3: Illustration of the limitations of CapsNets: The transformation matrix $\boldsymbol{W}_{ij}$ can only transform $\boldsymbol{u}_i$ to high-level capsules, while $\boldsymbol{W}_{kj}$ can only make meaningful transformations on $\boldsymbol{u}_k$. When an input is transformed (e.g., rotated), 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.

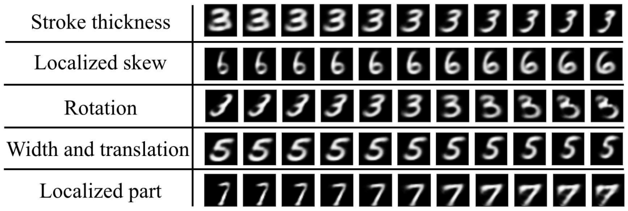


Figure 4: Disentangled Individual Dimensions of Capsules: By perturbing one dimension of an activity vector, the variations of an input image are reconstructed.

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.

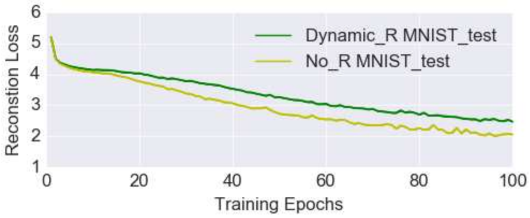


Figure 5: The average reconstruction loss of CapsNets with **Dynamic-R** and **No-R** on the test dataset is shown in each epoch of the training process.

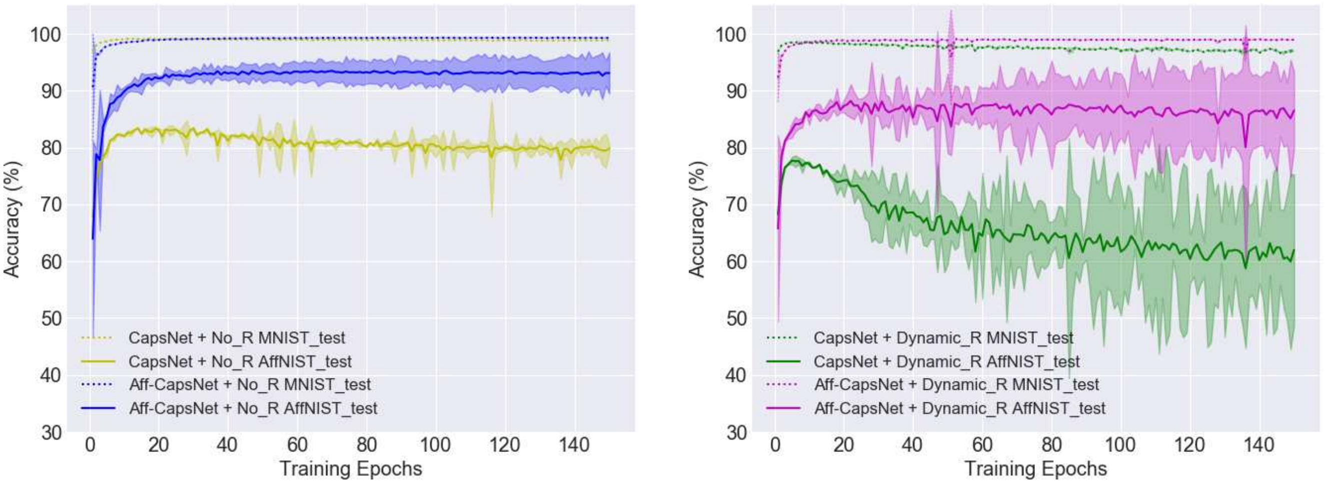
5.1.2 Parallel Attention Mechanism between Capsules

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.

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 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 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.

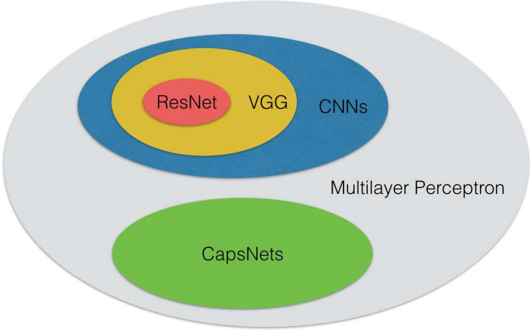


Figure 7: The relationship between different CNN architectures and Capsule Network architectures.

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.