A Comparative Analysis of CapsNet and ResNet on Rotation Invariance

Weimin Huang*, Hanzi Jiang*, Yuting Shao*
University of Toronto
`{cheryl.huang, hanzi.jiang, yuting.shao}@mail.utoronto.ca`

Abstract

Residual Neural Network (ResNet) has achieved excellent performance in many computer vision tasks, yet similar to other Convolutional Neural Networks (CNNs), it fails to capture the spatial relationship between features. In an attempt to address this issue, Capsule Network (CapsNet) implements the idea of encoding both the probabilities and orientations of features in units of capsules. In this paper, we apply rotational transformations to the test images and compare the classification performance of ResNet and CapsNet. We choose the less commonly-used QuickDraw dataset, and demonstrate that CapsNet may not yield the same rotation invariant results as it has been applied to the simpler MNIST data.

1 Introduction

CNNs achieve translational invariance with their pooling layers, but they are not robust to rotation. To address this limitation, the Capsule Network (CapsNet) implements an approach based on activity vectors and dynamic routing between capsules. It has been known for its equivariance and invariance properties, yet no literature has specifically analyzed its rotation invariance. Moreover, most experiments on the performance of CapsNet are conducted on simple datasets like MNIST. In this paper, we explore further and perform image classification on a subset of the QuickDraw dataset [4]. We investigate the performance of CapsNet and Residual Neural Network (ResNet) - one of the most common architectures for CNN - in the presence of rotations. The performance of models with different training data sizes and routing iterations is compared, and sensitivity analysis is conducted. Finally, strengths and weaknesses of each model are discussed.

2 Related Works

Drawback of Traditional CNN. Despite ResNet's excellent performance on many computer vision tasks, it has a major drawback - the relative position of the features is not taken into account [1]. In the processing of max pooling, only the most active neurons will proceed to the next layer, resulting in the loss of the spatial relationships between features.

Capsules. In contrast, CapsNet replaces the scalar-output feature detectors from CNNs with vector-outputs. The CapsNet consists of a group of capsules and a decoder. A capsule [2] is a group of neurons, and each capsule learns to recognize an implicitly defined feature. A capsule encodes the state (position, orientation) of the feature as the direction of its output vector and the probability of the feature's existence as the length. The decoder uses the output vectors of all capsules to reconstruct the original image, and the model aims to minimize the mean squared distance between the reconstructed image and the original one.

Dynamic Routing. CapsNet also replaces the max-pooling subsampling technique with routing-by-agreement (pseducode is below) [8]. The squash function is a non-linear function to make the vector

Code implementation: https://github.com/HanziJiang/CapsNet-ResNet-Performance-Analysis length not over 1, and it is defined the paper [8]. The basic idea is that a capsule in a lower-level layer sends its output vector to higher-level capsules if the latter "agrees" with the former. The coupling coefficient of two adjacent-level capsules is calculated by the dynamic routing process for r iterations. This routing-by-agreement mechanism has been shown to achieve better performance compared to the pooling technique [9].

Algorithm 1 Routing algorithm.

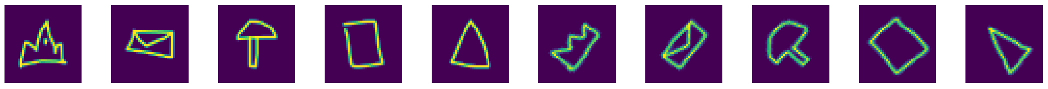
```
1: procedure ROUTING($\hat{u}_{j|i}, r, l$)
2:     for all capsule i in layer l and capsule j in layer (l+1): $b_{ij} \leftarrow 0$
3:     for r iterations:
4:         for all capsules i in layer l: $c_i \leftarrow softmax(b_i)$
5:         for all capsule j in layer (l+1): $s_j \leftarrow \sum_i c_{ij} \hat{u}_{j|i}$
6:         for all capsule j in layer (l+1): $v_j \leftarrow squash(s_j)$
7:         for all capsule i in layer l and capsule j in layer (l+1): $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$
8:     return $v_j$
```

Controversy on CapsNet. Although CapsNet is designed to achieve equivariance or invariance, some studies have reported it may not work as expected and even produces worse results than simple baseline algorithms [6, 7]. In addition, despite previous papers [8] claiming that CapsNet can be trained with a smaller number of samples in comparison with classical CNNs, Mukhometzianov et al. (2018) suggest that more complex datasets still require as many images as classical CNNs [6]. The overall performance of CapsNets in different tasks and in different types of dataset is unknown.

3 Methods and Algorithm

3.1 Experimental Settings

Dataset. The QuickDraw dataset [4] is a collection of sketch of objects drawn by hand. Due to hardware constraints, the models are trained and evaluated on a subset of the QuickDraw data containing five classes: triangle, square, mushroom, crown, and envelope. To investigate the rotation invariance property, after training on the original images, the models are evaluated on a test dataset consisting of rotated images.



(a) *Crown* (b) *Envelope* (c) *Mushroom* (d) *Square* (e) *Triangle* (f) *Crown* (g) *Envelope* (h) *Mushroom* (i) *Square* (j) *Triangle*

Figure 1: Five categories are chosen from QuickDraw. The dataset is split into train, validation, and test set with a ratio of 6:2:2. The train and validation set contain original images as shown in subfigures (a)-(e). The test set contains rotated images. Subfigures (f)-(j) show the images rotated by 45 degrees. The size of all images are changed from 28 by 28 to 40 by 40 in our experiments due to rotation.

CapsNet architecture. We choose the traditional CapsNet with dynamic routing [8]. The network is composed of a convolutional layer, a stack of 32 capsules, and a decoder structure. During training, the vector outputs of the capsules are masked out, except for the target class. We feed them into the decoder, which consists of three fully connected layers. The network aims to minimize the margin loss [8], plus the mean squared reconstruction loss scaled down by 0.00001. We determine this scaling factor by grid search, and the factor is different from the original paper [8], which uses 0.0005. This change makes sense because the reconstruction loss is likely to be larger when the dataset is more complex; the reconstruction loss should be weighted less. Otherwise, it will dominate the margin loss. Our new scaling factor allows a 4 percent higher validation accuracy and faster convergence. We borrow from the existing codebases [5, 3] with modifications to the model size, which are described in Figure 2.

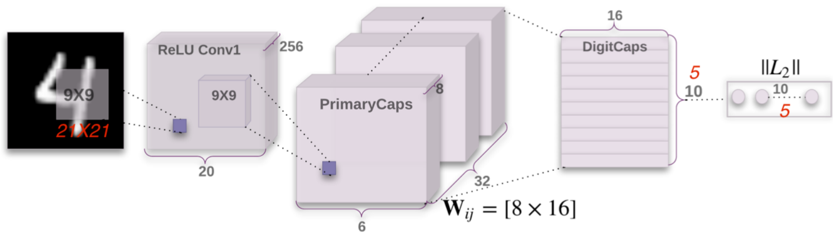


Figure 2: The traditional CapsNet structure [8]; our modifications are in red.

CNN baseline. We choose ResNet-18 as the baseline CNN algorithm. It has 11.7 million parameters, similar to our implementation of CapsNet, which has 9.2 million parameters. We replace the last convolutional layer and fine-tune this model to perform the downstream classification task.

3.2 Methods

To test the robustness of CapsNet to rotations, we train our two models on the padded training set. There is no rotation other than natural skewness in the original QuickDraw dataset. We then test the models on images rotated by angles, ranging from -180 to 180 degrees, with a step of step 10 degrees. There are two sets of experiments; we compare how the validation and test accuracy changes for each one.

Exp1 - Different training samples. We sample 5,000 images for each of the five categories, and randomly split them into 0.6:0.2:0.2 train:validation:test datasets. We train our CapsNet and ResNet until at least one of them converges. We also make sure that both of them have similar validation accuracy so that their test performance is comparable. Then the same experiment is performed again, except that the training set is reduced to 0.16 of its original size. This is done by randomly sampling from the original training dataset. The test and validation datasets remain the same. In this way, we reduce the amount of data the model sees during training to investigate how training sample size affects the models' performance.

Exp2 - Different routing iterations. The same 5,000-per-category training, validation and test datasets are used. There are also two experiments. In the first one, the routing iteration is 1; in the second one, the routing iteration is 3. By comparing the validation and test accuracy, we can determine if the routing algorithm is still effective when applied to a more complex dataset.

4 Experiments and Discussions

Some reconstructed images from the validation set are shown in Appendix Figures 4 and 5. Interestingly, when the number of training samples increases from 480 to 3,000 per class, the quality of the reconstructed images does not improve much against the darker background. The validation accuracy differs by less than 1.5%, so our CapsNet is robust to changes in training sample size.

4.1 Sensitivity Analysis

Sensitivity to the initial learning rate value: When implementing CapsNet, we use the Adam optimizer with the exponentially decaying learning rate. In the experiment, we keep gamma constant (0.92) and test different initial learning rate values. Appendix Table 1 shows how the classification accuracy on the validation dataset varies as the initial learning rate value increases. When the learning rate is too large (e.g. 0.01), the best accuracy of CapsNet on the validation dataset is only 20.5%. The reason is that, if too high, the learning rate can make the model jump over the minimum or even lead to divergence. When the initial learning rate is 0.0001, we get the best performance, achieving the highest validation accuracy of 98.16%.

Sensitivity to the batch size: In Appendix Table 2, we examine the effect of batch size on the classification accuracy of the validation dataset. As the batch size increases, the validation accuracy decreases. When the batch size is 8, the best accuracy (98.06%) can be achieved. However, in this test, we keep all other hyperparameters the same, so a model with a larger batch size may not converge at the end of training. In addition, we find that models with smaller batches tend to converge faster, though the training progress is slower. Smaller batch size also makes the model generalize better, which is preferred.

Sensitivity to the number of samples per class: To test the influence of the number of samples per class on the validation accuracy, we choose five different samples for each class, as shown in Appendix Table 3. We observe that the number of samples per class is highly correlated with the validation accuracy. As expected, CapsNet performs better when given more labelled samples per class. When the number of examples per class is 5,000, we have the best validation accuracy of 98.62%. Adding more examples decreases the generalization error.

4.2 Robustness under Rotational Transformation

To compare the performance of ResNet and CapsNet in the presence of rotational transformation, we compute the test accuracy of the two models on the test dataset with rotated images. A plot of classification test accuracy vs rotation angle is created for each of the models, shown in Figure 3.

For both models, the test accuracy is the highest when the angle is 0, which is evaluated on untransformed images. As the angle deviates from 0, the classification accuracy starts to decrease. There is a symmetric pattern for clockwise and counter-clockwise rotations. The accuracy has local peaks when images are rotated by 90 and 180 degrees. The reason is that the dataset contains the Square and Triangle classes. When they are rotated by certain degrees, the shape is identical to un-rotated ones, yielding a higher test accuracy.

The differences in the classification test accuracy for ResNet and CapsNet demonstrate that CapsNet is more robust to rotational transformation than ResNet does. For most values of angles, the test accuracy of CapsNet is higher than that of ResNet. As the angle deviates from 0, the drop in accuracy of CapsNet is less drastic than that of ResNet. However, as the image rotates, the test accuracy of CapsNet drops by over 60%. This decrease in performance is significant. We conclude that even though our CapsNet model yields a slightly better result compared with ResNet, it is not robust to rotations on the QuickDraw dataset.

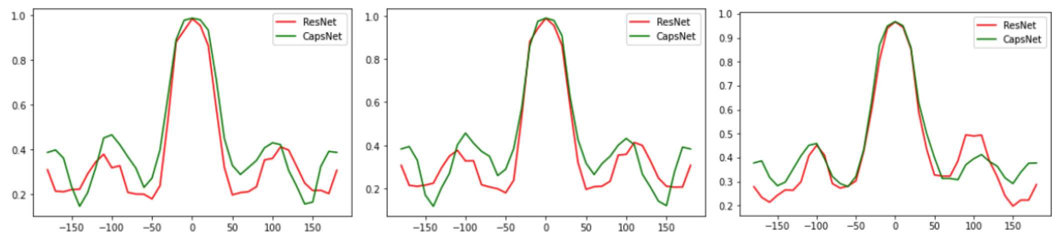


Figure 3: Test accuracy vs rotation angle for experiments (1) r=1, full training set; (2) r=3, full training set; (3) r=3, 0.16 training set

Limitations. Due to the computationally expensive routing mechanism, training CapsNets requires a lot of time and computational resources. One epoch takes around 15 minutes on a Google Colab GPU for our model. Some experiments with much better results have their models trained for over 14 hours [6]. Therefore, the model's performance might improve if we continue the training.

5 Summary and Conclusions

In this paper, we present a comparative study of the Capsule Network in image classification, in the presence of rotation, and on the less commonly used QuickDraw dataset. No similar literature has been delivered to evaluate the rotation invariance of CapsNet on this dataset. We find that CapsNet performs slightly better than the ResNet18 when tested on rotated images, yet the latter has a simpler architecture and faster training process. The test accuracy drops by over 60% when the images are rotated, so we conclude our model is not robust to rotations when in similar situation. The routing iterations do not affect the test performance in our experiments. Since CapsNet preserves spacial information, its effect of rotation invariance may be more prominent when applied on 3D datasets.

Authurs' Contributions

We have worked collaboratively on this project. All the team members have helped in model training and hyperparameter tuning. Weimin Huang has built the file utilities and data loader for the QuickDraw dataset, and measured the performance under rotation with different angles for ResNet and CapsNet. Hanzi Jiang has built the training code for CapsNet and conducted experiments in changing the model's architecture. Yuting Shao has performed image transformation, modified the ResNet model to perform image classification on the QuickDraw dataset, and conducted hyperparameter sensitivity analysis experiments.

References

[1] Logan Engstrom et al. *A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations*. 2019. URL: `https://openreview.net/forum?id=BJfvknCqFQ`.

[2] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. "Transforming Auto-Encoders". In: *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*. Ed. by Timo Honkela et al. Vol. 6791. Lecture Notes in Computer Science. Springer, 2011, pp. 44–51. DOI: `10.1007/978-3-642-21735-7\_6`. URL: `https://doi.org/10.1007/978-3-642-21735-7\_6`.

[3] jindongwang. *jindongwang/Pytorch-CapsuleNet*. URL: `https://github.com/jindongwang/Pytorch-CapsuleNet`.

[4] Google Creative Lab. *googlecreativelab/quickdraw-dataset*. URL: `https://github.com/googlecreativelab/quickdraw-dataset`.

[5] Laubonghaudoi. *laubonghaudoi/CapsNet*$_{guide}$$_{Py}$$Torch$. URL: `https://github.com/laubonghaudoi/CapsNet_guide_PyTorch`.

[6] Rinat Mukhometzianov and Juan Carrillo. "CapsNet comparative performance evaluation for image classification". In: *CoRR* abs/1805.11195 (2018). arXiv: `1805.11195`. URL: `http://arxiv.org/abs/1805.11195`.

[7] Inyoung Paik, Taeyeong Kwak, and Injung Kim. "Capsule Networks Need an Improved Routing Algorithm". In: *Proceedings of The 11th Asian Conference on Machine Learning, ACML 2019, 17-19 November 2019, Nagoya, Japan*. Ed. by Wee Sun Lee and Taiji Suzuki. Vol. 101. Proceedings of Machine Learning Research. PMLR, 2019, pp. 489–502. URL: `http://proceedings.mlr.press/v101/paik19a.html`.

[8] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules". In: *CoRR* abs/1710.09829 (2017). arXiv: `1710.09829`. URL: `http://arxiv.org/abs/1710.09829`.

[9] Lei Zhao and Lei Huang. "Exploring Dynamic Routing As A Pooling Layer". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019, pp. 738–742. DOI: `10.1109/ICCVW.2019.00095`.

Appendix

Table 1: Validation accuracy vs initial learning rate values

Accuracy	lr	Samples per class	Batch size	Gamma	Routing iteration	Epoch
0.9816	1e-5	5,000	8	0.92	1	10
0.9840	5e-5	5,000	8	0.92	1	10
0.9862	1e-4	5,000	8	0.92	1	10
0.9802	1e-3	5,000	8	0.92	1	10
0.2050	1e-2	5,000	8	0.92	1	10

Table 2: Validation accuracy vs batch size

Accuracy	Batch size	Samples per class	lr	Gamma	Routing iteration	Epoch
0.9806	8	5,000	1e-5	0.92	1	10
0.9800	16	5,000	1e-5	0.92	1	10
0.9772	32	5,000	1e-5	0.92	1	10
0.9730	64	5,000	1e-5	0.92	1	10
0.9694	128	5,000	1e-5	0.92	1	10

Table 3: Validation accuracy vs number of samples per class

Accuracy	Samples per class	lr	Batch size	Gamma	Routing iteration	Epoch
0.9630	1,000	1e-4	8	0.92	1	10
0.9770	2,000	1e-4	8	0.92	1	10
0.9827	3,000	1e-4	8	0.92	1	10
0.9835	4,000	1e-4	8	0.92	1	10
0.9862	5,000	1e-4	8	0.92	1	10

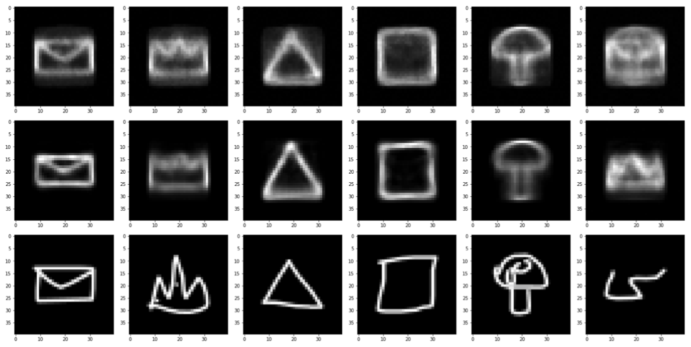


Figure 4: Top to bottom: reconstructed validation images of CapsNet trained on 3,000 training samples, 480 training samples, and target images; Left to right: mail, crown, triangle, square, mushroom, badly drawn mushroom.

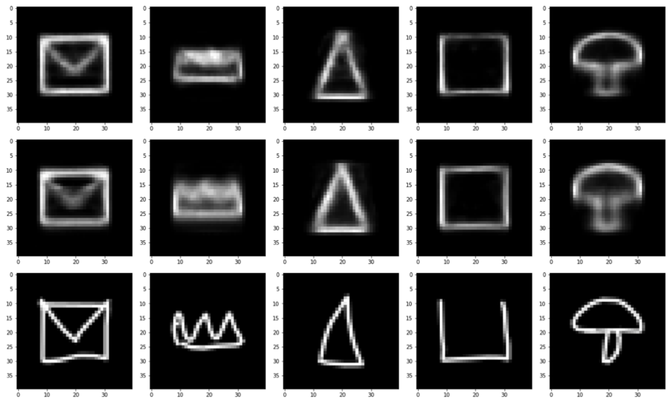


Figure 5: Top to bottom: reconstructed validation images of CapsNet with 1 routing iteration, 3 routing iterations, and target images; Left to right: mail, crown, triangle, square, mushroom.

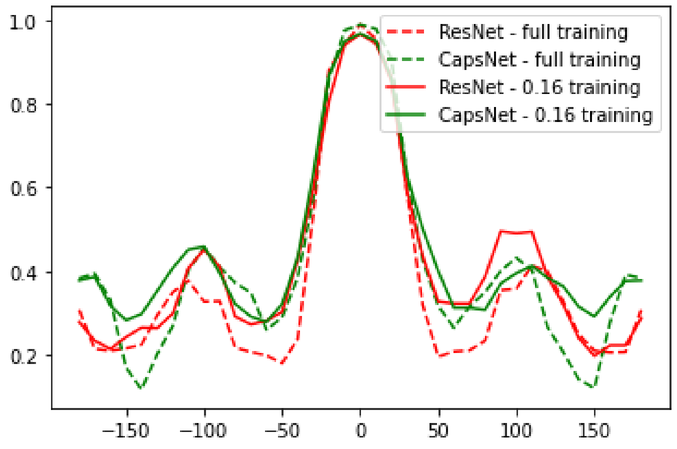


Figure 6: Test accuracy vs rotation angle on different training sample sizes. Interestingly, the models trained on fewer training samples yield better performance.