# Generalized Capsule Networks with Trainable Routing Procedure

**Zhenhua Chen** [1] **Chuhua Wang** [1] **Tiancong Zhao** [1] **David Crandall** [1]

## Abstract

Capsule Networks have been found to be effective in modeling spatial features with fewer parameters, and thus to generalize better than CNNs. However, the routing procedure is designed to maximize the similarities between adjacent capsules rather than to maximize a global objective function, and the routing number has to be set manually. We introduce Generalized CapsNets (G-CapsNets) to overcome these disadvantages by incorporating the routing procedure into the optimization process. G-CapsNets not only address the optimization of the coupling coefficients but also avoid the computational overhead of the routing procedure. We implement two versions of G-CapsNets, fully-connected and convolutional, using Caffe. Experiments show that G-CapsNets achieve better performance on MNIST and CIFAR10 than classic CapsNets, and generalize better on GAN-generated synthetic images. We also test G-CapsNets on robustness to white-box & black-box adversarial attack.

## 1. Introduction

Convolutional neural networks (CNNs) (Lecun et al., 1998) have been found work very well on many computer vision problems, in part because the initial layers can learn simple features whereas later layers combine simple features into complex ones. However, a fundamental issue with CNNs is that the spatial hierarchies between simple and complex objects are not well captured, which can prevent CNNs from generalizing well. For example, to detect a face, a CNN reports a high confidence if it finds face parts (eyes, nose, ears, etc.) even if their spatial configuration is not sensible (since the spatial relationships do not contribute to the loss). It is still possible for CNNs to capture spatial relationships, but they may need deeper networks and many negative samples.

For the same reason, CNNs do not generalize well across different angles of the same object (Hinton et al., 2018). Capsule Networks, on the other hand, are composed of capsules: groups of neurons whose activity vectors represent the instantiation parameters of a specific type of entity such as an object or an object part (Sabour et al., 2017). Capsules help naturally preserve the spatial hierarchies between object parts, and thus can achieve similar performance with fewer parameters. Sabour et al. (2017) demonstrate this on MNIST (LeCun & Cortes (1998)), CIFAR10 (Krizhevsky et al.) and smallNORB (LeCun et al. (2004)).

Given these promising properties, many researchers have explored variants and applications of CapsNets. For example, Rawlinson et al. (2018) adapt CapsNets to unsupervised learning by sparsifying the last capsule layer, while Jaiswal et al. (2018) treat CapsNets as the backbone for GANs and achieve lower error rate on both MNIST and CIFAR10. O' Neill (2018) replace normal neural networks with CapsNets for face verification, while Duarte et al. (2018) and Liu et al. (2018) use CapsNets as backbones for video classification and object localization, respectively.

In spite of many successful applications, the routing procedure of CapsNets is computationally expensive, which limits their scalability. The issue becomes more serious when a CapsNet becomes deeper. For example, in Sabour et al. (2017), the routing number is set as 3 because a smaller or larger value would cause degradation of performance. For a 10-layer CapsNet, assuming we have to try 3 routing numbers for each layer, then $3^{10}$ combinations must be tested to find the best routing number assignment. This problem may significantly limit the scalability and efficiency of CapsNets.

To overcome this issue, in this paper we propose Generalized CapsNets (G-CapsNets). By "generalized," we mean we can train a CapsNet just like training a standard neural network, and the (local) optimality of the coupling coefficients is guaranteed. The key idea of G-CapsNet is to incorporate the routing procedure into the overall optimization procedure, which makes the coupling coefficients trainable instead of being calculated by dynamic (Sabour et al. (2017)) or EM routing (Hinton et al. (2018)). The differences between CapsNets and G-CapsNet are summarized in

[*]Equal contribution [1]School of Informatics, Computing, and Engineering, Indiana University Bloomington, USA. Correspondence to: Zhenhua Chen <chen478@iu.edu>, David Crandall <djcran@indiana.edu>.

*ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, Long Beach, California, 2019. 

| | CapsNets | G-CapsNets |
|---|---|---|
| Optimization | SGD + heuristic | SGD |
| Routing procedure | agreement | training |
| Times of Routing | meta parameter | 1 |
| Coefficients sum | 1 | any number |

*Table 1.* CapsNets versus G-CapsNets. CapsNets use a routing procedure called "routing by agreement (dynamic routing, EM routing);" thus the optimization of CapsNets is a hybrid of Stochastic Gradient Descent (SGD) and this heuristic algorithm, while G-CapsNets use pure SGD. CapsNets require several iterations of the routing procedure to find appropriate coupling coefficients whose sum has to be 1 for each layer, while the coupling coefficients in G-CapsNets are treated as ordinary parameters and thus can be any number and only need to be calculated once for each pass.

Table 1.

We also test the scalability, generalization, and robustness of G-CapsNets. We find that, compared to vanilla CNNs, G-CapsNets show better scalability and generalization by using fewer parameters which is consistent with what CapsNets have claimed. However, we find that G-CapsNets are as vulnerable as CNNs to white-box adversarial attacks, which is inconsistent with what the original CapsNets (Hinton et al., 2018) have found. Considering that G-CapsNets also fall into the slot of CapsNets, so if G-CapsNet is not robust to adversarial white-box attack, then the claim that CapsNet is robust to adversarial white-box attack is questionable. We find similar evidence in (Marchisio et al., 2019; Yoon, 2017). (Marchisio et al., 2019) develop a method (the attack belongs to black-box but similar to white-box) to attack CapsNets successfully while (Yoon, 2017) find that CapsNets are vulnerable to various attack types (including white-box attack). We conjecture that the robustness of (Hinton et al., 2018) comes from randomness introduced during the routing procedure rather than the network itself.

## 2. Related work

Other researchers have found the routing procedure to be computationally expensive and have tried to improve it. For example, Wang & Liu (2018) introduce a regularizer based on KL divergence to minimize the clustering loss between capsules of adjacent layers. Li et al. (2018) try to reduce computational cost by adopting two extra branches to approximate the routing process. Both of these papers propose an alternative procedure to replace the expensive "routing by agreement." However, the approximation of the routing procedure is restricted to calculating the coupling coefficients accurately which is not necessarily consistent with the final objective function. G-CapsNets, on the other hand, simplify the routing procedure and make it serve the whole objective function directly.

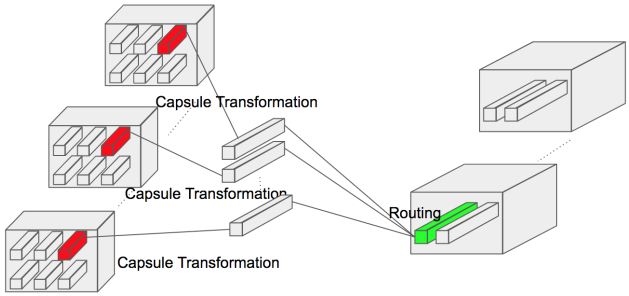


*Figure 1.* The structure of G-CapsNets. The capsules in the lower layers are transformed into intermediate capsules with some target dimension, then the routing procedure combines these transformed capsules to form new capsules in the upper layer.

## 3. Our approach: Generalized CapsNets

Similar to a CapsNet, each capsule layer of a G-CapsNet has two operations: capsule transformation and capsule routing. As Figure 1 shows, capsule transformation performs dimension transformation between adjacent capsule layers, while capsule routing combines the transformed capsules.

### 3.1. Capsule transformation & routing

Capsule transformation happens between adjacent capsule layers, converting one type of capsule $\mathbf{u_i}$ into another type of capsule $\mathbf{u_{j|i}}$ through a matrix operation $\mathbf{W_{ij}}$,

$$\mathbf{u_{j|i}} = \mathbf{W_{ij}}\mathbf{u_i}.$$

In theory, we can transform any type of capsule into any other type of capsule. The capsules can be tensors of any shape as long as we have the appropriate transformation matrix. For example, in Sabour et al. (2017), 8-d capsules are transformed into 16-d capsules, while in Hinton et al. (2018), $4 \times 4$ capsules are transformed into $4 \times 4$.

Capsule routing ensures capsules in lower layers are scaled and sent to their parent capsules in higher layers. From another point of view, capsule routing can be considered as a clustering procedure in which the capsules in the upper layer are centers while the capsules in the lower layer are points that need to be chosen. Capsule routing combines information to forge new capsules,

$$\mathbf{v_j} = \sum_i c_{ij}\mathbf{u_{j|i}}.$$

Note that the coupling coefficients $c_{ij}$ are not acquired by the "routing-by-agreement" but depend on the optimization target. Thus the coupling coefficients could be any value, depending on the loss term and the regularizer term of the

whole network. In other words, G-CapsNets choose the routing coefficients that best fit the loss function, and thus the sum of the coupling coefficients ($\sum_i c_{ij}$) does not have to be 1.

Although the two operations of G-CapsNets are similar to CapsNets, there is a fundamental difference: the routing procedure of CapsNets serves as the optimization of similarities between capsules while the routing procedure in G-CapsNets serves as a step for the global optimization target.

### 3.2. Activation & loss function

The idea of a squash function in a Capsule Network is to map the length of a capsule to a number between 0 and 1. We adopt $\mathbf{v_j^{'}} = \left(1 - \frac{1}{\mathbf{e}^{\|\mathbf{v_j}\|}}\right) \frac{\mathbf{v_j}}{\|\mathbf{v_j}\|}$ as suggested by Edgar et al. (2017), rather than the one in Sabour et al. (2017), to achieve faster convergence. The margin loss function (see Equation 1) is the same as in (Sabour et al., 2017),

$$L_k = T_k * \max(0,\ m^+ - \|\mathbf{v_k}\|)^2 + \lambda * (1 - T_k) * \max(0,\ \|\mathbf{v_k}\| - m^-)^2. \quad (1)$$

## 4. Experiments

### 4.1. G-CapsNets on MNIST & CIFAR10

#### 4.1.1. Fully connected G-CapsNets on MNIST

For G-CapsNets, we adopt the same architecture as Sabour et al. (2017). The first convolutional layer outputs 256 feature maps. The second convolutional layer outputs 256 feature maps or 32×6×6 8D capsules. For the final layer, we replace the dynamic routing procedure with our trainable routing procedure. Please check our released code or Sabour et al. (2017) for more details. We lso adopt the same baseline described in Sabour et al. (2017), with three convolutional layers of 256, 256, and 128 channels. The kernels and strides are 5x5 and 1. The last convolutional layer is followed by two fully-connected layers of size 328 and 192 and a 10-class softmax classifier.

We call the capsule structure here (and the one in Sabour et al. (2017)) "fully-connected CapsNet" since each capsule in the higher layer connects to every capsule in the lower layer. As Table 2 shows, no matter the reconstruction involved, G-CapsNets achieves better performance than CNNs or CapsNets (Sabour et al., 2017). Note that the accuracy of the baseline and CapsNets reported here is lower than in Sabour et al. (2017) due to differences in the data augmentation (pixel-shifting) adopted by Sabour et al. (2017) as well as different frameworks (Caffe versus TensorFlow).

| Algorithm | error rate(%) | param # |
|---|---|---|
| vanilla CNN | $0.91_{\pm 0.09}$/$36.54_{\pm 0.78}$ | 35.4M/9.6M |
| CapsNets** | $1.24_{\pm 0.10}$/$40.88_{\pm 0.67}$ | 8.2M/2.67M |
| FC G-CapsNets | $0.67_{\pm 0.05}$/$32.70_{\pm 0.47}$ | 8.2M/2.67M |
| FC G-CapsNets* | $0.66_{\pm 0.03}$/- | 6.8M/- |
| Conv G-CapsNets | $0.84_{\pm 0.09}$/$33.53_{\pm 0.52}$ | 6.9M/2.67M |
| Conv G-CapsNets* | $0.86_{\pm 0.1317}$/- | 5.5M/- |
| Multi G-CapsNets | -/$34.29_{\pm 0.33}$ | -/716K |

*Table 2.* Error rate versus number of parameters on MNIST & CIFAR10. Note that "*" means no reconstruction, and "CapsNets**" is the algorithm that comes from (Sabour et al., 2017). Each number before and after slash is the corresponding error rate of MNIST and CIFAR10 respectively.

#### 4.1.2. Convolutional G-CapsNets on MNIST

Similar to the structure in Hinton et al. (2018), we build a convolutional version of G-CapsNets. The same type of capsules of different positions share the same transformation matrices. We use a 6x6 kernel for the last capsule layer and a 4x4 matrix As Table 2 shows, convolutional G-CapsNets achieve better performance compared to the baseline by using fewer parameters.

#### 4.1.3. G-CapsNets on CIFAR10

For CIFAR10, we build a similar G-CapsNet structure as the one used for MNIST, except that: (1) we use 64 feature maps (rather than 256 feature maps) in the first convolutional layer, and (2) we use an 8x8 kernel for the capsule layer and transform each capsule with an 8x8 matrix. The baseline shares the same structure as G-CapsNets for the first two layers, and has two fully-connected layers with outputs of 512 and 10. As Table 2 shows, both fully-connected CapsNets and convolutional CapsNets achieve better performance than either the baseline or CapsNets. We also develop a multi-layer version of G-CapsNets which achieves better performance than either CNNs or CapsNets, as Table 2 shows. Please refer to the supplementary appendix for more details.

### 4.2. Generalization and robustness of G-CapsNets

CapsNets are better at capturing the relationship of different spatial features, so it makes sense to believe that CapsNets should be more generalizable and robust. To test if this, we use AC-GANs (Auxiliary Classifier Generative Adversarial Networks) proposed by (Odena et al., 2017) to generate synthetic images. Specifically, we generate 4,000 artificial MNIST-like images (400 images for each digit), as Figure 2 shows. AC-GAN encodes both class labels as well as noise to generate synthetic images. The discriminator of AC-GAN also outputs a distribution over the source and the

| Epochs | Error rate (%) | | |
|---|---|---|---|
| | Baseline | FC G-CapsNets | Conv G-CapsNets |
| 1 | 19.800 | 23.500 | 19.725 |
| 2 | 5.600 | 3.250 | 4.050 |
| 3 | 0.725 | 0.000 | 0.700 |
| 4 | 0.001 | 0.000 | 0.000 |
| 5 | 0.003 | 0.000 | 0.001 |

*Table 3.* Error rate on MNIST-like images generated by AC-GAN (Odena et al., 2017), for the baseline (35.4M parameters), fully-connected G-CapsNets (6.8M parameters), and convolutaional G-CapsNets (5.5M parameters).

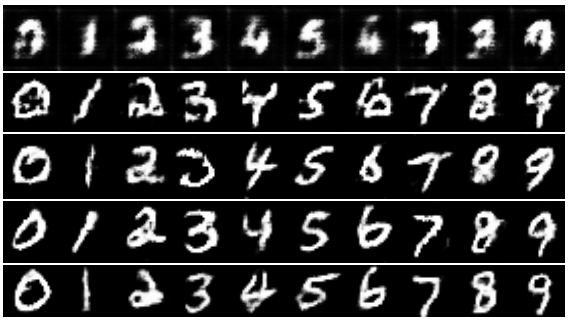

*Figure 2.* The generated MNIST-like images. From top to bottom, the number of training epochs is 1, 2, 3, 4 and 5.

class labels (please refer to the original paper for details). The discriminator contains four convolutional layers with 32, 64, 128, and 256 feature maps. Each convolutional layer is followed by a leaky ReLU layer and a dropout layer. The generator first uses a fully-connected layer to map the latent vector (100, 1) to a (3, 3, 384) tensor, then up-samples the tensor to a (7, 7, 192) tensor, a (14, 14, 96) tensor and a (28, 28, 1) tensor. The learning rate (2e-4) and beta1 (0.5) are those recommended by Radford et al. (2015).

These images are generally recognizable for humans but not the same as the original images in MNIST (in terms of distribution) since they are generated in the early stage of AC-GAN. These two properties make these images useful for testing generalization.

The underlying assumption of CapsNets is that they are better at capturing the spatial features in an image and thus need far fewer parameters than standard neural networks. As Table 3 shows, both the fully connected version and convolution version of G-CapsNets achieve better performance than the baseline, which uses more parameters. However, both the baseline and G-CapsNets show no significant difference when the number of epochs is larger than 3. We argue this is because the generated images have already converged to the same distribution as in MNIST. Note that it makes more sense to compare the performance over three network structures rather than across epochs since the generated images vary each time.

### 4.3. The robustness of G-CapsNets on black box attack

Hinton et al. (2018) found that CapsNets are comparable to CNNs regarding robustness to black-box attacks despite using fewer parameters. Our experiments also support this claim. Specifically, we adopt LeNet as the substitute model to generate perturbations for each test image (10K testing images in MNIST) based on FSGM (Goodfellow et al. (2015)). We restrict the maximum perturbation of each pixel to be 8 ($L_\infty \leq 8$). We found that the accuracy of the baseline, the fully-connected G-CapsNets, and the convolutional G-CapsNets drop sharply to 11.35%, 8.92%, and 11.35%, respectively, after the attack. This suggests that G-CapsNets are as vulnerable as standard neural networks.

### 4.4. The robustness of G-CapsNets on white-box attack

Hinton et al. (2018) found that CapsNets are robust to white-box adversarial attack due to numerical instability ( Nayebi & Ganguli (2017)) as well as the smaller percentage of zero values in the gradient. However, their testing was based on FGSM (Goodfellow et al., 2015) which is not a strong attack technique. Inspired by Universal Adversarial Perturbations (Moosavi Dezfooli et al., 2017), we trained a generative model to generate universal perturbations during training (offline), and then applied the generated perturbations to all testing images. We found that G-CapsNets do not show stronger robustness to the attack. Please check the supplementary appendix for more details.

## 5. Conclusion

G-CapsNets incorporate the capsule routing procedure into the overall optimization process, avoiding the need to set routing times for each capsule layer. The two versions of G-CapsNets, fully-connected G-CapsNets and convolutional G-CapsNets, achieve better performance on MNIST and CIFAR10 compared to both CNNs and CapsNets. G-CapsNets also show good scalability as well as generalization ability. Finally, we evaluated the robustness of G-CapsNets against both the white-box attack and black-box attack, and found that G-CapsNets are vulnerable to both types of attacks.

## References

Duarte, K., Rawat, Y. S., and Shah, M. Videocapsulenet: A simplified network for action detection. *CoRR*, abs/1805.08162, 2018. URL http://arxiv.org/abs/1805.08162.

Edgar, X., Selina, B., and Yang, J. Capsule network performance on complex data. *CoRR*, 2017. URL

https://arxiv.org/abs/1712.03480.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Hinton, G. E., Sabour, S., and Frosst, N. Matrix capsules with EM routing. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJWLfGWRb.

Jaiswal, A., AbdAlmageed, W., Wu, Y., and Natarajan, P. CapsuleGAN: Generative Adversarial Capsule Network. *CoRR*, February 2018.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10. URL http://www.cs.toronto.edu/~kriz/cifar.html.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 1998. URL http://yann.lecun.com/exdb/mnist/.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

LeCun, Y., Huang, F., and Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

Li, H., Guo, X., Dai, B., Ouyang, W., and Wang, X. Neural network encapsulation. *ECCV*, 2018.

Liu, W., Barsoum, E., and Owens, J. D. Object localization and motion transfer learning with capsules. *CoRR*, abs/1805.07706, 2018. URL http://arxiv.org/abs/1805.07706.

Marchisio, A., Nanfa, G., Khalid, F., Abdullah Hanif, M., Martina, M., and Shafique, M. CapsAttacks: Robust and Imperceptible Adversarial Attacks on Capsule Networks. *CoRR*, art. arXiv:1901.09878, Jan 2019.

Moosavi Dezfooli, S. M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9, 2017. doi: 10.1109/Cvpr.2017.17. URL http://infoscience.epfl.ch/record/226156.

Nayebi, A. and Ganguli, S. Biologically inspired protection of deep networks from adversarial attacks. *CoRR*, March 2017.

O' Neill, J. Siamese Capsule Networks. *CoRR*, May 2018.

Odena, A., Olah, C., and Shlens, J. Conditional image synthesis with auxiliary classifier GANs. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2642–2651, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/odena17a.html.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. URL http://arxiv.org/abs/1511.06434.

Rawlinson, D., Ahmed, A., and Kowadlo, G. Sparse unsupervised capsules generalize better. *CoRR*, abs/1804.06094, 2018. URL http://arxiv.org/abs/1804.06094.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *CoRR*, abs/1710.09829, 2017. URL http://arxiv.org/abs/1710.09829.

Wang, D. and Liu, Q. An optimization view on dynamic routing between capsules, 2018. URL https://openreview.net/forum?id=HJjtFYJDf.

Yoon, J. Adversarial Attack to Capsule Networks, November 2017. URL https://github.com/jaesik817/adv_attack_capsnet.

| Layer | Layer parameters | param # |
|---|---|---|
| Conv#1 | (7, 7, 64), 1 | 9.4K |
| Conv#2 | (7, 7, 128), 2 | 401.4K |
| Conv Caps Trans#1 | (4, 4, 8), 1, (8, 8) | 131.1K |
| Conv Caps Routing#1 | (3, 3, 16), (7, 7, 8) | 56.4K |
| Conv Caps Trans#2 | (3, 3, 16), 2, (2, 2) | 4.6K |
| Conv Caps Routing#2 | (3, 3, 8), (3, 3, 16) | 10.4K |
| FC Caps Trans#1 | (16, 3, 3, 8), (8, 10) | 92.2K |
| FC Caps Routing#1 | (16, 3, 3, 8), 10 | 11.5K |
| Total | - | 716K |

*Table 1.* The structure of Multi-layer G-CapsNets.

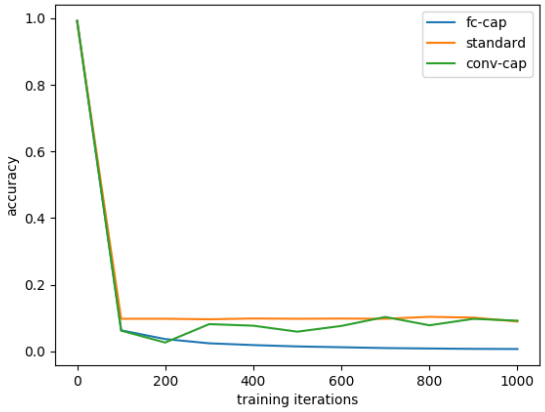


*Figure 1.* The decreasing curve of three network structures (standard: standard neural network, fc: fully-connected G-CapsNets, conv: convolutional G-CapsNets) with white-box attack

# 6. Supplementary Material

## 6.1. The structure of Multi-layer G-CapsNets on CIFAR10

To test the scalability of G-CapsNets, we build a network with two convolutional layers, two convolutional capsule layers, and one full connected layer, as Table 1 shows. We adopt one ReLU layer and one squash layer after each convolutional capsule layer and full connected capsule layer. Take the 'Conv Caps Transform#1' as an example, (4, 4, 8) means the kernel size is (4, 4) and the number of capsule feature maps is 8. The following number is the stride. The last (8, 8) refers to the transformation matrix. Namely, the G-CapsNets transform capsules (1x8) on the lower layer to capsules (1x8) on the upper layer. 'Conv Caps Routing#1' follows the 'Conv Caps Transform#1', whose responsibility is combining the information from each tensor with the shape of (3, 3, 16), and we have a total number of 7x7x8 combinations. For the full connected CapsNet, take the 'FC Caps Transform#1' layer as example, the multi-layer G-CapsNets transform each tensor with a shape of (16, 3, 3, 8) on the lower layer to a new tensor with a shape of (16, 3, 3, 8) on the upper layer. Then the layer 'FC Caps Routing#1' combines each tensor of shape (16, 3, 3, 8) to a new tensor of shape (1, 8). Since CIFAR has ten classes, the output of the final layer has a shape of (10, 8).

## 6.2. White adversarial attack on G-CapsNet

The structure of the generative model is similar to a GAN but with the discriminator unchanged. Specifically, the input of the generator is a 100-dimension latent vector whose values are between 0 and 1. The latent vector layer is followed by three deconvolutional layers (note that we apply Batch Normalization after each de-convolutional layer). The final layer of the generative branch outputs a tensor with the same dimension as the input image. The discriminator is the classification model we need to test. The loss function is to minimize the difference between the logits of a clean image and the logits of its manipulated version. The whole attack is untargeted, and we assign each under-attacked image a random incorrect label during training.

We apply this attack technique on the test set of MNIST which contains 1000 images. As Figure 1 shows, all three networks' accuracy drop sharply after 100 attacking iterations. This result is not consistent with what( Hinton et al. (2018)) found with the weaker FGSM attack. These results suggest that CapsNets and CNNs are both vulnerable to strong white-box attacks like UAP.