Generalized Capsule Networks with Trainable Routing Procedure

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

Abstract

CapsNets proposed by Sabour et al. (2017) has been found effective in modeling spatial features with much fewer parameters. However, the routing procedure is designed to maximize the similarities between adjacent capsules other than to maximize the global objective function. The routing number, which has to be set manually, is also difficult to optimize. We introduce the Generalized GapsNets (G-CapsNets) to overcome above disadvantages by incorporating the routing procedure into the optimization process. G-CapsNets not only guarantee the optimization of the coupling coefficients but also avoid the high computational overhead of the routing procedure. We implement two versions of G-CapsNets (fully-connected and convolutional) on CAFFE (Jia et al. (2014)) and compared them with CapsNets. The experiment shows that G-CapsNets achieve better performance than CapsNets on MNIST & CIFAR10. We also test G-CapsNets on the robustness to white-box & black-box attack, the generalization ability on GAN-generated synthetic images, and the scalability for deep neural networks.

1. Introduction

Convolutional neural networks (CNN) have been found effective in many computer vision problems since proposed (Lecun et al., 1998). CNNs are composed of covolutional layers stacking on each other. Each layer is a set of neurons. Each neuron is a scalar, whose responsibility is detecting some particular feature. Many neurons fire together to detect some high-level spatial features. However, the same objects could have varied spatial relationships, and the CNNs have to stack more layers to model a particular spatial relationship. If each neuron of CNNs is a tensor other than scalar, then the necessary number of parameters to model the same spatial relationship would be much fewer. This is the basic idea of CapsNet. According to Sabour et al. (2017), a capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part. Intuitively, CapsNet can better model spatial relationship by using much fewer parameters. The experiments in Sabour et al. (2017) show some intriguing results in MNIST (LeCun & Cortes (2010)), CIFAR10 (Krizhevsky et al.) and smallNORB (LeCun et al. (2004)).

Given the good property of CapsNets, many researchers explore the applications of CapsNets, for exmple, (Rawlinson et al., 2018) adapt CapsNets to unsupervised learning by sparsify the last capsule layer. (Jaiswal et al., 2018) treat CapsNets as the backbone for GAN and achieve lower error rate on both MNIST and CIFAR10. (O' Neill, 2018) replace normal neural networks with CapsNets for face verification. Similarly, (Duarte et al., 2018) and (Liu et al., 2018) also use CapsNets as backbones for video classification and object localization separately.

In spite of many successful applications, the routing procedure of CapsNets is computationally expensive which limits the scalability of CapsNets. The issue becomes more serious when a CapsNet becomes deeper. For example, in the paper (Sabour et al., 2017), the routing number has to be set as 3. A routing number that is smaller than 3 or larger than 3 would cause degradation of performance. For a 10-layer CapsNet, assume we have to try 3 routing numbers for each layer, then totally we have to try 3^{10} times to find the best routing number assignment in theory. That is why the scalability and efficiency of CapsNets are questioned.

To overcome this issue, we propose Generalized CapsNet. 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 CapsNet is incorporating the routing procedure to the overall optimization procedure, which makes the coupling coefficients trainable instead of being calculated by dynamic routing (Sabour et al. (2017)) or EM routing (Hinton et al. (2018)). Please refer to section 3 for more details. We also test the scalability, generalization, and robustness of G-CapsNets. We find that, compared to vanilla CNNs,

[*]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>.

arXiv preprint version

CapsNets show better scalability and generalization by using fewer number of parameters. However, we also find that G-CapsNets is as vulnerable as vanilla CNNs facing a white/black box attack according to our experiment. This is different from the discovery of (Hinton et al., 2018) who claim that CapsNets is more robust facing white-box attacks. We conjecture that the robustness of (Hinton et al., 2018) comes from introduced randomness during the routing procedure other than the network itself.

2. Rlated work

The idea of CapsNets attracts much attention recently. (Xi et al., 2017) explore whether stacking more capsule layers or adding more convolutional layers will result in better performance. (Wang & Liu, 2018) introduce a regularizer of KL divergence to minimize the clustering loss between capsules of adjacent layers. (Li et al., 2018) tries to soften the issue of computational cost by adopting two extra branches to approximate the routing process.

Despite CapsNets have many variations, all of them above adopt the same type of routing procedure called "routing by agreement". The primary issue of these routing procedures in ((Sabour et al., 2017; Hinton et al., 2018; Li et al., 2018; Wang & Liu, 2018)) is that they are not part of the global optimization procedure and thus the optimization cannot be guaranteed. Besides, the procedures of "routing by agreement" bring extra computationally overhead as well as extra work to adjust the meta-parameter – routing number for each capsule layer. For example, many papers, including (Xi et al., 2017; Rawlinson et al., 2018; Liu et al., 2018; O' Neill, 2018) etc., found that the routing procedure is computationally expensive and the routing numbers have an affect on the performance. We can image that this issue of computational overhead will become more serious as a capsule network becomes deeper. G-CapsNets introduced by this paper simplify the routing procedure and achieve better or comparable performance than CapsNets.

3. Our approach: Generalized CapsNet

3.1. Structure of Generalized CapsNet

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

3.1.1. Capsule transformation

Capsule transformation happens between adjacient capsule layers. It converse one type of capsules into another type of capsules. As Equation 2 shows, $\mathbf{u_i}$ is transformed to $\mathbf{u_{j|i}}$ by multiply it self with $\mathbf{W_{ij}}$. In theory, we can transform any type of capsules into any other type of capsules. 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-dimension capsules are transformed into 16-dimension capsules while in Hinton et al. (2018) 4×4 capsules are transformed into 4×4 capsules.

$$\mathbf{u_{j|i}} = \mathbf{W_{ij}}\mathbf{u_i} \tag{1}$$

3.1.2. Capsule routing

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 $\mathbf{v_j}$, as Equation 2 shows.

$$\mathbf{v_j} = \sum_i c_{ij}\mathbf{u_{j|i}} \tag{2}$$

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

Although the two operations of G-CapsNets seem the same as CapsNets, there exists a fundamental difference between them. The routing procedure of CapsNets serves as the optimization of similarities between capsules while the routing procedure in G-CapsNets servers as a step for the global optimization target.

3.2. Activation & loss function

The idea of squash function is to map the length of a capsule to a number between 0 and 1. G-CapsNets also adopts squash function as the activation function. As Equation 3 and Equation 4 show, $\mathbf{v_j}$ are squashed capsules and $\mathbf{v_j'}$ are capsules before the squash operation. We adopt a squash function suggested by Edgar et al. (2017) (as Equation 4 shows) in our experiment which results in faster convergence than Sabour et al. (2017) (as Equation 3 shows).

$$\mathbf{v_j'} = \frac{\|\mathbf{v_j}\|^2}{1 + \|\mathbf{v_j}\|^2} \frac{\mathbf{v_j}}{\|\mathbf{v_j}\|} \tag{3}$$

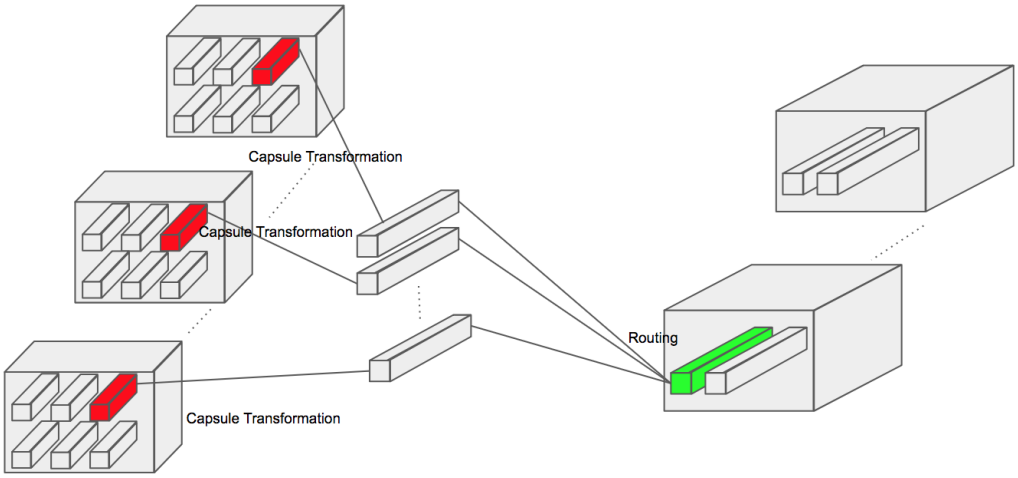

Figure 1. The structure of G-CapsNets.

$$\mathbf{v_j}^{'} = \left(1 - \frac{1}{\mathbf{e}^{\|\mathbf{v_j}\|}}\right) \frac{\mathbf{v_j}}{\|\mathbf{v_j}\|} \tag{4}$$

For G-CapsNets, we adopt the same margin loss function in (Sabour et al., 2017), as Equation 5 shows. m^+ is the upper threshold of the length of a capsule. In other words, capsules with a length bigger than m^+ are considered convergence. Similarly, m^- is the lower threshold. λ is

$$L_k = T_k * \max(0,\ m^+ - \|\mathbf{v_k}\|)^2 + \lambda * (1 - T_k) * \max(0,\ \|\mathbf{v_k}\| - m^-)^2 \tag{5}$$

Algorithm	error rate(%)	param #
vanilla CNN	0.83	35.4M
CapsNets (Sabour et al., 2017)	1.1	8.2M
full connected G-CapsNets	0.66	8.2M
full connected G-CapsNets*	0.66	6.8M
Convolutional G-CapsNets	0.75	6.9M
Convolutional G-CapsNets*	0.70	5.5M

Table 1. Error rate VS number of parameters on MNIST. Note that "*" means no reconstruction,

4. Experiments

4.1. G-CapsNets on MNIST & CIFAR10

4.1.1. Full connected G-CapsNets on MNIST

We adopt the same baseline as described in paper Sabour et al. (2017) which has 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. The final layer is a 10-class softmax classifier.

As for G-CapsNets, we also adopt the same architecture as in paper 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 the original paper written by Sabour et al. (2017) for more details.

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 1 shows, no matter whether the reconstruction involved, G-CapsNets can always achieve better performance than the vanilla CNN as well as CapsNets (Sabour et al., 2017). Note that the accuracy of the baseline and the CapsNets reported here is lower than in Sabour et al. (2017), we consider the difference is caused by the data augmentation (pixel-shifting) adopted by Sabour et al. (2017) as well the different frameworks (Caffe VS TensorFlow).

4.1.2. Convolutional G-CapsNets on MNIST

Similar to the structure in Hinton et al. (2018), we also 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" to transform capsules (from 4x4 matrix capsules to 4x4 matrix capsules). As Table 1 shows, convolutional G-CapsNets achieves 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. There are two differences. One is that we adopt 64 feature maps (other than 256 feature maps) in the first convolutional layer. The other one is we adopt 8x8 kernel for the capsule layer and transform each capsule by using an 8x8 matrix. The baseline shares the same structure as the G-CapsNets used here for the first two layers, then it has two fully-connected layers with an output of 512 and 10 separately. As Table 2 shows, both fully-connected CapsNet, and convolutional CapsNet achieve better performance than the baseline as well as CapsNets.

4.1.4. 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 3 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). The performance of the Multi-layer CapsNet is shown at the last line of Table 2.

Algorithm	error rate(%)	param #
vanilla CNN	36.62	9.6M
CapsNet (Sabour et al., 2017)	40.3	2.67M
FC G-CapsNets	32.11	2.67M
FC G-CapsNets*	33.00	2.67M
Conv G-CapsNets	32.92	2.67M
Multi-layer G-CapsNets	34.21	716K

Table 2. Error rate VS number of parameters on CIFAR10. Note that "*" means the alternate way of stacking capsules.

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 3. The structure of Multi-layer G-CapsNets on CIFAR10.

4.2. The generalization of G-CapsNets

CapsNet is better at capturing the relationship of different spatial features, so it makes sense to believe that CapsNet has a better generalization ability. To verify if this is true, we use AC-GAN (Auxiliary Classifier Generative Adversarial Network) proposed by (Odena et al., 2017) to generate synthetic images. Specifically, we generate 4k artificial MNIST-like images (400 images for each digit), as Figure 2 shows. AC-GAN encodes both class labels as well as the noise to generate synthetic images. The discriminator of AC-GAN also output distribution over the sources and the class labels. Please refer to the original paper for details. The discriminator contains four convolutional layers which have 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 perceptible for human beings 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. Being perceptible means these images have enough clues to tell their classes and being generated in the early stage means they do not share the same distribution as the authentic MNIST. Both properties make these images perfect for testing generalization.

The underlying assumption of G-CapsNets is that they are better at capturing the spatial features in an image and thus they need far less number of parameters than standard neural networks. As Table 4 shows, both full connected version, and convolutions version of G-CapsNets achieve better performance than the Baseline which uses much more parameters. However, both the baseline and the G-CapsNets show no significant difference when the number of epochs is larger than 3. We argue that the reason is that the generated

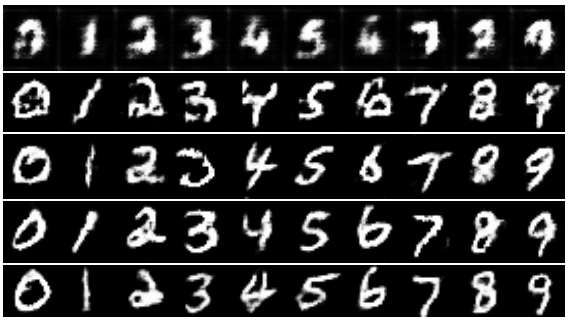

Figure 2. The generated MNIST-like images. From top to bottom, the number of training epochs is 1, 2, 3, 4 and 5.

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 other than across epochs since the generated images vary each time.

Epochs	Baseline	FC G-CapsNets	Conv G-CapsNets
1	19.80	23.50	19.725
2	5.60	3.25	4.05
3	0.725	0.00	0.70
4	0.0005	0.00	0.00
5	0.00275	0.00	0.001

Table 4. Error rate on MNIST-like images that are generated by AC-GAN. FC CapsNet is full connected G-CapsNets and Conv G-CapsNets refers to convolutional G-CapsNets. The number of parameters of the baseline, FC G-CapsNets and Conv G-CapsNets are 35.4M, 6.8M and 5.5M separately.

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

Hinton et al. (2018) claimed that CapsNets are comparable to CNNs regarding robustness to black-box attacks despite using fewer parameters. Our experiment also supports this claim. Specifically, we adopt LeNet as the substitute model to generate perturbations for each testing images (10K testing images in MNIST) based on FSGM (Goodfellow et al. (2014)). We restrict the maximum perturbation for each pixel to be 8 ($L_\infty \leq 8$). As Table 5 shows, the accuracy of the baseline, the fully-connected G-CapsNets, and the convolutional G-CapsNets drops sharply. There are no significant differences among them, and we can thus conclude that G-CapsNets is as vulnerable as the standard neural networks.

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

Hinton et al. (2018) claimed 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, the whole testing is based on FGSM Goodfellow et al. (2014) which is not a strong attack technique. Inspired by Universal Adversarial Perturbation (UAP) Moosavi-Dezfooli et al. (2016), we trained a generative model to generate universal perturbations during training (offline), and then apply the generated perturbations to all testing images.

	Baseline	FC G-CapsNets	Conv G-CapsNets
accuracy	11.35%	8.92%	11.35%

Table 5. Black box attack on three types of networks. The number of parameters of the FC G-CapsNets and Conv G-CapsNets are 6.8M and 5.5M separately.

The structure of the generative model is similar to GAN but with the discriminator unchanged. Specifically, the input of the generator is a 100-dimension latent vector whose value is between 0 and 1. The latent vector layer is followed by three deconvolutional layers (note that we apply Batch Normalization after each de-convolutional layers). The final layer of the generative branch is supposed to output 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 testing set of MNIST which contains 10K images. As Figure 3 shows, all three networks' accuracy drop sharply after 100 attacking iterations. This result is not consistent with what Hinton et al. (2018) found. The CapsNets and CNNs are both vulnerable to strong white-box attacks like UAP.

5. Conclusion

G-CapsNets incorporate the capsule routing procedure into the whole optimization process which gets rid of the setting of routing times for each capsule layer and guarantees the optimization. The two versions of G-CapsNets (fully-connected G-CapsNets and convolutional G-CapsNets) achieve better performance on MNIST and CIFAR10 compared to both navilla CNNs and CaspNets. G-CapsNets also show the good scalability as well as generalization ability. Finally, we evaluate the robustness of G-CapsNets in the background of both the white-box attack and black-box attack. For all the measurements we adopt in this paper, G-CapsNets achieve either better (concerning the accuracy, number of parameters, generalization ability, scalability) or comparable (regarding the white-box attack and black-box attack) performance.

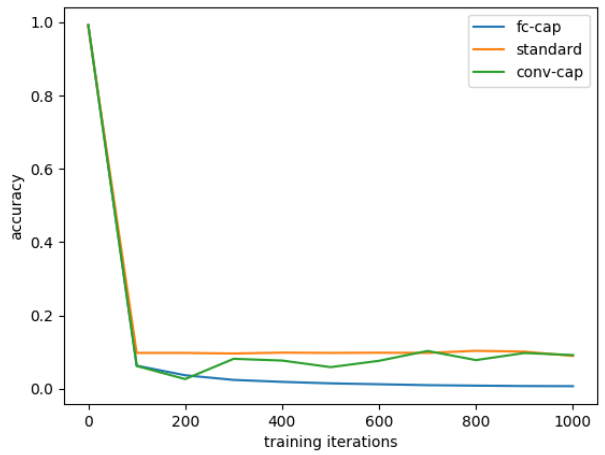

Figure 3. The decreasing curve of three network structures (standard: standard neural network, fc: fully-connected G-CapsNets, conv: convolutional G-CapsNets) with white-box attack

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. 2017. URL https://arxiv.org/abs/1712.03480.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, December 2014.

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. *ArXiv e-prints*, February 2018.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. 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 *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 2004.

Li, H., Guo, X., Dai, B., Ouyang, W., and Wang, X. Neural network encapsulation. *CoRR*, abs/1808.03749, 2018. URL http://arxiv.org/abs/1808.03749.

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.

Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016. URL http://arxiv.org/abs/1610.08401.

Nayebi, A. and Ganguli, S. Biologically inspired protection of deep networks from adversarial attacks. *ArXiv e-prints*, March 2017.

O' Neill, J. Siamese Capsule Networks. *ArXiv e-prints*, 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.

Xi, E., Bing, S., and Jin, Y. Capsule Network Performance on Complex Data. *ArXiv e-prints*, December 2017.