Using Capsule Networks with Thermometer Encoding to Defend Against Adversarial Attacks

Sahaj Garg [*1,2] **Jordan Alexander** [*1,3] **Tanay Kothari** [*1,4]

Abstract

Adversarial attacks have been shown to construct examples that drastically reduce the performance of classification models. One recently proposed defense against adversarial attacks is to discretize the input in a method called thermometer encoding. We apply thermometer encoding to capsule networks, a recently proposed computer vision architecture that has also demonstrated state-of-the-art resistance to adversarial attacks. The capsule network with thermometer encoding outperforms our baseline CNN and vanilla capsule network when trained adversarially, and performs comparably to a CNN with thermometer encoding. The gains by applying thermometer encoding to capsule networks may improve adversarial resistance on more complex tasks where capsule networks have already demonstrated state-of-the-art resistance.

1. Introduction & Related Work

Classification models have been shown to be vulnerable to adversarial examples–inputs from the dataset with slight modifications that have been specifically selected in order to increase the probability of a classification error (Goodfellow et al., 2014). These adversarial attacks introduce security vulnerabilities into machine learning models, which has important implications for sensitive applications such as autonomous vehicles, signature verification, and others.

Considerable research has gone into understanding adversarial attacks and developing mechanisms to resist them. It has been hypothesized by Goodfellow et. al. that one of the primary reasons for this vulnerability is that the decision boundaries of CNNs are predominantly linear. Adding to the problem, adversarial examples are highly transferrable from one model to another, even when models are trained on different datasets or have different architectures (Szegedy et al., 2013).

Different approaches have been taken to dramatically reduce the effectiveness of adversarial attacks. One example, introduced by Goodfellow et. al, incorporates the fast gradient sign method adversarial attack directly into the objective function as a regularizer. Alternative approaches, including the one used in this paper, involve supplementing every minibatch with corresponding adversarial examples (Huang et al., 2015). Recently, adversarial training using adversarial examples generated via the basic iterative method (BIM) attack as well as randomly added noise was demonstrated to be robust to many different types of attacks on MNIST (Madry et al., 2017).

Building on recent work, thermometer encoding has been proposed as a way of defending against adversarial attacks (Anonymous, 2018b). This paper demonstrated state of the art resistance against the strongest known adversarial attacks on MNIST while sacrificing little accuracy on a clean (no adversarial examples) test-set. Intuitively, by discretizing inputs into vectors (similar to one-hot encodings by binning data), switching from one bin to another is nonlinear and even nondifferentiable, which precludes attack effectiveness. The details are discussed in the methods section of this paper.

The above adversarial testing methods have been predominantly applied to standard computer vision architectures, including shallow softmax classifiers and CNNs. A recent computer vision architecture, capsule networks, was shown to have state-of-the-art performance on MNIST (Sabour et al., 2017) and also perform well against adversarial attacks (Anonymous, 2018a). The reason for this resistance is likely due to a combination of the novel dynamic routing procedure and reconstruction loss, which are described in more detail in this paper's methods.

In this paper, we seek to test the effectiveness of applying the novel discretization method, thermometer encoding, to capsule networks. This paper tests the thermometer encoded capsule network against its vanilla version, a baseline CNN, and a thermometer encoded CNN. We find that capsule networks with thermometer encoding outper-

[*]Equal contribution [1]Stanford University, Stanford, CA [2]sahajg@stanford.edu [3]jfalex@stanford.edu [4]tkothari@stanford.edu.

Proceedings of the CS229 Final Project Session, Stanford, CA, 2017.

Figure 1. MNIST test set images (top row) and corresponding adversarial examples (bottom row). From left to right, the attacks are FGSM against CNN, DGA against CNN with thermometer encoding, FGSM against capsule network, DGA against capsule network with thermometer encoding.

form vanilla capsule networks, and perform comparably to CNNs with thermometer encoding. Because the model demonstrates improved accuracy, we recommend further study of the benefits of thermometer encoding applied to more complex datasets.

2. Dataset and Features

We use the MNIST (LeCun et al., 1998) dataset, a compilation of 70,000 (55,000 train, 5,000 val, 10,000 test) 28x28 images of handwritten greyscale digits from 0 to 9, to evaluate our models. Our models are trained using clean training examples and corresponding adversarial examples that are generated using the fast gradient sign method (FGSM) and discrete gradient ascent (DGA), depending on the model. FGSM and DGA are described in the methods. Figure 1 showcases examples of original input images and their corresponding FGSM/DGA adversarial attacks. Depending on the model, we either train solely on the clean train set, or train on a combination of this set and dynamically generated adversarial examples.

3. Methods

Our methods include eight models. Our models are both clean-trained and adversarially-trained versions of a baseline CNN, CNN with thermometer encoding, capsule network, and capsule network with thermometer encoding.

3.1. Baseline CNN

Our baseline is a CNN implemented in Tensorflow (Abadi et al., 2015) with two convolutional layers, two max pooling layers, and one fully connected layer. Hyperparameters are specified in the example CNN in the Tensorflow repository (Kleinfeld, 2017).

3.2. Capsule Network Architecture

For our primary model, we use a capsule network. A capsule network is a convolutional architecture with two major modifications. First, capsule networks group neurons into capsules, whose output is a vector. The elements of this vector output represent different properties of the same feature. The norm of the vector, instead of the activation of a neuron, represents the presence of a feature.

The second major innovation in capsule networks is the routing mechanism between layers. In convolutional networks, information is routed from layer to layer using max pooling, which takes the maximum over a region of neurons. This preserves existence of features, but loses spatial information. Max pooling is replaced by dynamic routing, an unsupervised learning algorithm that is applied for three iterations between layers in a capsule network.

The unsupervised learning problem for dynamic routing between layers is as follows: The x's are capsules in layer l, and the latent variable z's are capsules in the next layer. In essence, each capsule in a lower layer is explained by being 'part of' a capsule in a higher order layer. For example, a capsule corresponding to a vertical line in a MNIST classifier could be active because it is part of either a 1, 4, or 7 in the final classification capsule layer. The routing algorithm learns the probability that each capsule in layer l is activated because it is part of a capsule in layer $l+1$, and multiplies the connection from one capsule to the next (a weight matrix) by this probability (the routing coefficient). The likelihood of a connection is based on the cosine similarity between the output of capsule x_i in layer l and the tentative input to capsule z_c in layer $l+1$. Informally, this routing algorithm routes information to capsules where the most agreement is observed. Note that this routing procedure is dynamically computed every forward pass for every image for three iterations. The full dynamic routing algorithm is presented in Figure 2.

Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\mathbf{u}}_{j|i}, r, l$)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j$

Figure 2. The routing procedure in (Sabour et al., 2017).

In addition, capsule networks decode the 16 dimensional vector corresponding to each digit prediction to an image. The network is trained with a low-weight L2 norm reconstruction loss in addition to the standard classification loss of the network. The implementation used is a modified version of the CapsLayer GitHub repository (Liao, 2017). The hyperparameters used for the capsule network are directly borrowed from Sabour et. al. because these hyperparameters maximize accuracy on MNIST in their paper.

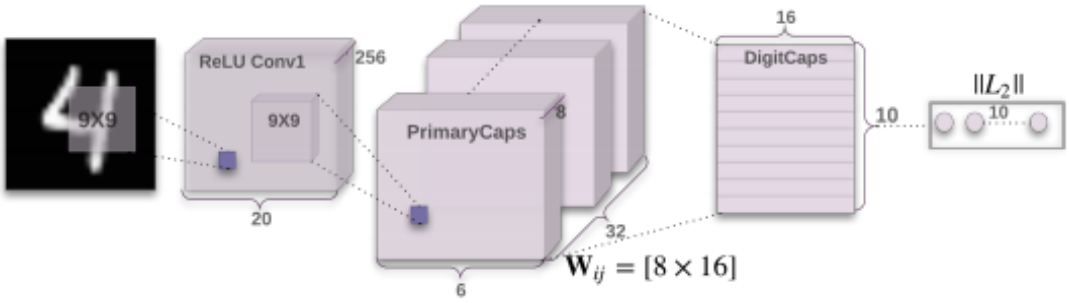

Figure 3. A graphical depiction of the Capsule Network architecture, from (Sabour et al., 2017).

3.3. Thermometer Encoding

Thermometer encoding is a discretization of input that is analogous to one-hot encoding binned data. In our case, we use 16 bins, an optimal demonstrated by (Anonymous, 2018b). Consider the example of the seventh bin. A one-hot encoder sets just the seventh entry of the vector to 1. However, a thermometer encoder sets the first seven entries to one, preserving some relationship between encodings of different buckets.

$$\text{One-hot: } 100 \rightarrow [0000001000000000]$$
$$\text{Thermometer: } 100 \rightarrow [1111111000000000]$$

More generally, consider a quantization function b. Set buckets $0 < b_1 < b_2 < ... < b_k = 1$. In our architecture, we set $k = 16$ and $b_i = i/16$. Then, for a real number $\theta \in [0, 1]$ let $b(\theta)$ be the largest index $\alpha \in \{1, ..., k\}$ such that $\theta \leq b_\alpha$. Now, consider the index function $\tau(j)_l = 1\{l \geq j\}$. We define our thermometer encoding function for a given input x as: $f_{therm}(x)_i = \tau_{b_i}(b(x_i))$.

3.4. Adversarial Attacks and Training

3.4.1. Fast Gradient Sign Method (FGSM)

For our baseline models with continuous input that has not been thermometer-encoded, we use the Fast Gradient Sign Method (FGSM) to generate adversarial examples (Goodfellow et al., 2014). FGSM constructs an attack such that each new pixel is within at most a small constant ϵ of its original value. It does this by taking the gradient of the loss with respect to the input, and modifying the input in the direction that maximizes the loss. Essentially:

$$x_{adv} := x + \epsilon \text{sign}(\nabla_x J(\theta, x, y))$$

3.4.2. Discrete Gradient Ascent (DGA)

For our models with discrete input that has been thermometer-encoded, we use Discrete Gradient Ascent (DGA) to generate adversarial examples (Anonymous, 2018b). DGA is an iterative attack that moves each pixel x_i, or $(z_i)^t$ after t iterations of the attack, to the bucket within ϵ of its true value that is likely to do the most harm. We apply DGA for one iteration, analogous to FGSM. More precisely:

$$z_i^0 = f_{therm}(x_i + U(-\epsilon, \epsilon)) \quad (1)$$

$$\text{harm}(z_i^t)_l = (z_i^t - \tau(l))^T \cdot \frac{\partial L(z^t)}{\partial z_i^t} \cdot 1\{\exists \gamma(-\epsilon \leq \gamma \leq \epsilon) \mid b(x_i + \gamma) = l\} \quad (2)$$

$$z_i^{t+1} = \tau(\arg\max(\text{harm}(z_i^t))) \quad (3)$$

3.4.3. Adversarial Training

Adversarial training is done dynamically at train-time. At the end of every mini-batch of 100 train examples, 100 corresponding adversarial examples are computed. The network is then trained on these adversarial examples.

4. Results

4.1. Experiments

As explained in the methods, eight models were constructed for this experiment: both clean-trained and adversarially trained versions of a baseline CNN, a CNN with thermometer encoding, a capsule network, and a capsule network with thermometer encoding. For adversarially trained models, the model was trained on newly generated adversarial examples every mini-batch during train time. Each architecture was trained for either a maximum of thirty epochs or until validation error began increasing, a sign of overfitting. Models were trained on a Tesla K80 GPU using the Adam optimizer.

At test-time, for each individual model, we evaluated accuracy on the 10,000 test examples as well as on the 10,000 corresponding adversarial white-box attacks to the model. In addition, we evaluated every model against the adversarially generated test-set (source) from each of the other models. This was done to test how robust each model was to black box attacks.

Figure 4. Reconstructions for capsule networks (top) and capsule networks with thermometer encoding (bottom). From left to right: original, adversarial attack, reconstructed clean example, reconstructed adversarial example. Note the similarity between reconstructions for the clean and adversarial examples.

16 bins for thermometer encoding were chosen after hyperparameter tuning as a reasonable maximization of clean test-set accuracy while still allowing for discretization. Results are omitted for space.

4.2. Results

Capsule network reconstructions for FGSM and DGA generated attacks are depicted in Figure 4. Note that reconstructions learn to ignore the adversarial perturbations because during adversarial train-time, they are trained to reconstruct the correct images. This likely contributes to the capsule network's ability to effectively resist adversarial examples. An especially impressive characteristic of the capsule network is that its reconstructions appear of similar quality to the original images.

We primarily evaluate the adversarially trained models because these are most reflective in the context of adversarial resistance. The results for adversarially trained models are depicted in Figure 5. The first row represents performance on the clean dataset, and the diagonal of the 4x4 table represents performance against white-box attacks. Based on the results in the table, we can note that thermometer encoding improves the performance of both CNNs and capsule networks. This result is especially impressive because the thermometer encoded models only sacrifice marginal accuracy on the clean test set compared to their clean-trained counterparts (Figure 7). In addition, the vanilla capsule network outperforms the vanilla CNN. Overall, the CNN with thermometer encoding and the capsule network with thermometer encoding reflect similar, and mixed results.

Learning curves are presented in Figure 6. Full results for all clean-trained and adversarially trained models are presented in Figure 7.

4.3. Discussion

The comparisons described above have several important implications. First and foremost, because thermometer encoding improves capsule networks, we have confirmation that thermometer encoding as a resistance method generalizes well to other, more complex models. Although time constraints prevented testing capsule networks on more complex datasets where their adversarial gains are significant compared to CNNs, we may infer from these results that thermometer encoding could possibly have a significant positive effect on the accuracy of capsule networks on other datasets. The learning curves demonstrate that applying thermometer encoding to capsule networks has the same effect observed in (Anonymous, 2018b) where it dramatically reduces the amount of train time necessary for high adversarial accuracy while not sacrificing clean dev-set loss.

In addition to the primary result of demonstrated effectiveness of thermometer encoding to capsule networks, there are several other interesting insights from these experiments. First, not only do vanilla capsule networks outperform vanilla CNNs on SmallNORB (Anonymous, 2018a), they also outperform CNNs on datasets such as MNIST. We hypothesize that two of the reasons for this improved performance may be the fact that reconstructions can be

Source \ Target	CNN	Thermo CNN	Caps	Thermo Caps
Clean	0.9930	0.990	0.9950	0.9933
CNN	0.9718	0.9634	0.9278	0.9637
Thermo/CNN	0.8201	0.9466	0.9357	0.9389
Caps	0.8875	0.9763	0.9481	0.9727
Thermo/Caps	0.8936	0.9684	0.9556	0.9792

Figure 5. Test set results for adversarially trained models. The first row represents evaluation of each adv. trained model on the clean test set, and the next four rows represent evaluation on adversarial test sets generated by applying FGSM or DGA to each corresponding model.

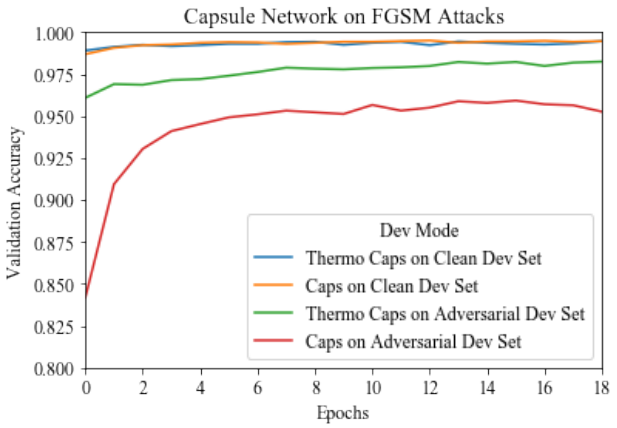

Figure 6. Learning curves for capsule networks trained with and without thermometer encoding, evaluated on the validation set and corresponding adversarial examples at the end of every epoch.

trained to ignore adversarial noise and that dynamic routing considers the relationships between part and whole and can consequently disregard individual pixel noise more effectively.

Finally, capsule networks allow for additional model interpretability in the case of incorrect classifications. Because reconstructions also demonstrate incorrect classifications, it may be clearer what part of adversarial examples are successful in fooling models.

5. Conclusion and Future Work

This research demonstrates the effectiveness of thermometer encoding applied to models other than convolutional networks. Specifically, thermometer encoding applied to capsule networks on MNIST improves performance when adversarially trained, and this performance is comparable to CNNs with thermometer encoding. Given the results of this paper, several immediate extensions are obvious and important. First, stronger attacks against both vanilla and thermometer encoded models should be implemented: namely Basic Iterative Method (BIM) (Kurakin et al., 2016) and Logit Space Projected Gradient Ascent (LSPGA) (Anonymous, 2018b), respectively. Evaluation under strong attacks will yield results that are more reflective of real-world tasks.

In addition, given the quality of results on MNIST, the same approach to adversarial resistance should be applied to datasets where capsule networks have demonstrated state of the art adversarial resistance. For example, the same set of experiments should be applied to the Small-NORB dataset (LeCun et al., 2004), which contains pictures of toys photographed from different viewpoints, on which capsule networks have been shown to perform well (Anonymous, 2018a).

In addition to these straightforward future steps, additional work should be done to examine the reasons behind improved adversarial resistance. As mentioned in the introduction, one hypothesized reason that adversarial attacks via FGSM tend to perform well is because of locally linear decision boundaries. Church-window plots may be used as a mechanism for visualizing the non-linearity of decision boundaries (Anonymous, 2018b). These plots ought to be generated for capsule networks, both with and without thermometer encoding, to determine whether dynamic routing and discretization improve nonlinearity in decision boundaries.

Overall, research in adversarial machine learning is nascent and growing. Similarly, the recently published capsule network architecture may pave the path for a new approach to computer vision. By combining these two approaches, we demonstrate high quality adversarial resistance on the MNIST dataset, and hope that this guides future research investigating the powers of these combined models.

6. Contributions and Acknowledgements

Sahaj Garg trained the capsule network models, implemented the FGSM and DGA adversarial attacks, and constructed the testing framework to evaluate all models against each other. Jordan Alexander built the CNN architecture, prototyped capsule network applications on different datasets, and modularized the codebase to allow for any additional models to be tested with thermometer encoding. Tanay Kothari implemented the thermometer encoding mechanism, sourced the papers and ideas about capsule networks and thermometer encoding, and hyperparameter tuned thermometer encoding. Vijay Krishnan acted as a mentor for this project, and provided guidance and direction in terms of which problems to approach.

Source \ Target		Clean				Adversarially Trained			
		CNN	Thermo CNN	Caps	Thermo Caps	CNN	Thermo CNN	Caps	Thermo Caps
Clean Test Set	Clean	0.993	0.99	0.995	0.9933	0.9867	0.9922	0.9928	0.9922
Adv. examples from clean trained	CNN	0.0939	0.4831	0.5565	0.5547	0.8552	0.9701	0.9432	0.9683
	Thermo CNN	0.6952	0.2781	0.5924	0.484	0.8872	0.9492	0.9566	0.9322
	Caps	0.8243	0.7283	0.7669	0.7301	0.8569	0.971	0.951	0.9665
	Thermo Caps	0.8787	0.6587	0.7251	0.7288	0.887	0.9225	0.9649	0.9225
Adv. examples from adv. Trained	CNN	0.6713	0.5881	0.6082	0.5836	0.9718	0.9634	0.9278	0.9637
	Thermo CNN	0.8163	0.5461	0.5486	0.4673	0.8201	0.9466	0.9357	0.9389
	Caps	0.8757	0.7923	0.7831	0.7581	0.8875	0.9763	0.9481	0.9727
	Thermo Caps	0.8893	0.702	0.7089	0.5909	0.8936	0.9684	0.9556	0.9792

Figure 7. Full results for all tested models. The columns represent target models, and the rows represent the test set used for evaluation, either the clean test set or adversarial examples generated from each of the other models. This demonstrates transferability of attacks.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Anonymous. Matrix capsules with em routing. *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=HJWLfGWRb.

Anonymous. Thermometer encoding: One hot way to resist adversarial examples. *International Conference on Learning Representations*, 2018b. URL https://openreview.net/forum?id=S18Su--CW.

Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. Explaining and Harnessing Adversarial Examples. 12 2014. URL http://arxiv.org/abs/1412.6572.

Huang, Ruitong, Xu, Bing, Schuurmans, Dale, and Szepesvári, Csaba. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015. URL http://arxiv.org/abs/1511.03034.

Kleinfeld, Sanders. Tensorflow examples. https://github.com/tensorflow/tensorflow/tree/r1.4/tensorflow/examples, 2017.

Kurakin, Alexey, Goodfellow, Ian J., and Bengio, Samy. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016. URL http://arxiv.org/abs/1607.02533.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Yann, Huang, Fu Jie, and Bottou, Leon. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pp. II–104. IEEE, 2004.

Liao, Huadong. Capsnet tensorflow. https://github.com/naturomics/CapsLayer, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv e-prints*, June 2017.

Sabour, Sara, Frosst, Nicholas, and Hinton, Geoffrey E. Dynamic Routing Between Capsules. 10 2017. URL http://arxiv.org/abs/1710.09829.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL http://arxiv.org/abs/1312.6199.