On the Vulnerability of Capsule Networks to Adversarial Attacks

Felix Michels [*1] Tobias Uelwer [*1] Eric Upschulte [*1] Stefan Harmeling [1]

Abstract

This paper extensively evaluates the vulnerability of capsule networks to different adversarial attacks. Recent work suggests that these architectures are more robust towards adversarial attacks than other neural networks. However, our experiments show that capsule networks can be fooled as easily as convolutional neural networks.

1. Introduction

Adversarial attacks change the input of machine learning models in a way that the model outputs a wrong result. For neural networks these attacks were first introduced by Goodfellow et al. (2014) with alarming results. Recently capsule networks (CapsNets) (Sabour et al., 2017) have been shown to be a reasonable alternative to convolutional neural networks (ConvNets). Frosst et al. (2018) state that CapsNets are more robust against white-box adversarial attacks than other architectures. Adversarial robustness of CapsNets has been previously studied by Marchisio et al. (2019), but with focus on the evaluation of their proposed attack. Detecting adversarial examples using the reconstruction quality of the CapsNets has been investigated by Frosst et al. (2018). Also Peer et al. (2018) have briefly discussed the application of the fast gradient sign method (FGSM) (Goodfellow et al., 2014) on CapsNets. Hinton et al. (2018) report results of the FGSM on CapsNets using EM routing. Another established approach for CapsNets is the dynamic routing algorithm (Sabour et al., 2017).

In this paper, we will focus on these variants of CapsNets and investigate their robustness against common adversarial attacks. In particular, we compare the results of four different attacks on CapsNets trained on different datasets and examine the transferability of adversarial perturbations. We will show that CapsNets are in general not more robust to adversarial attacks than ConvNets.

Our paper is structured as followed: in Sec. 2 we recapitulate the idea of CapsNets and the dynamic routing algorithm. In Sec. 3 we describe the attacks we apply to CapsNets. We summarize our experiments and results in Sec. 4.

[*]Equal contribution [1]Department of Computer Science, Heinrich-Heine-Universität Düsseldorf, Germany. Correspondence to: Felix Michels <felix.michels@hhu.de>, Tobias Uelwer <tobias.uelwer@hhu.de>, Eric Upschulte <eric.upschulte@hhu.de>, Stefan Harmeling <harmeling@hhu.de>.

ICML 2019 Workshop on Security and Privacy of Machine Learning, Long Beach, California. Copyright 2019 by the author(s).

2. Capsule Networks and Dynamic Routing

The concept of vector capsules and the dynamic routing algorithm was proposed by Sabour et al. (2017). In an essence, neurons are grouped into vectors, so-called capsules. Each capsule vector is dedicated to a distinct abstract entity, i.e. a single object class in a classification setting. The norm of a capsule vector encodes the probability of the represented object being present in the input, while the vector orientation encodes the object's characteristics. Thus, CapsNets aim to develop dedicated representations that are distributed into multiple vectors in contrast to convolutional networks that utilize an entangled representation in a single vector at a given location. This allows the application of linear transformations directly to the representations of respective entities. Spatial relations, which can be implemented as a matrix product, can thus be modeled more efficiently.

CapsNets are organized in layers. Initially, the original CapsNet applies a convolutional layer. The resulting feature maps are then processed by the primary capsule layer. Internally, it applies a series of convolutional layers on its own, each yielding a spatial grid of capsules. Within all capsule layers the *squashing* function serves as a vector-to-vector non-linearity that squashes each capsule vector length between 0 and 1 while leaving the orientation unaltered. Subsequently, convolutional or densely connected capsule layers can be applied. While the latter does not utilize weight sharing, convolutional capsule layers share the kernels over the spatial grid, as well as capsules from the previous layer. These layers effectively estimate output capsules based on respective input capsules. The dynamic routing algorithm determines each agreement between estimate and iteratively calculated output capsule. This is done by introducing a scalar factor, the so-called routing coefficient, for each connection between an estimate and respective output. Such an output is defined as the sum over all respective estimates, weighted by their routing coefficients. Theoretically, that means information flows where it is needed, both during forward and backpropagation. This non-parametric procedure supports the goal of capsules with clean dedicated representations. To improve results, an additional capsule may be used within the routing algorithm to serve as a dead end for information that may not be linked to known abstract capsule categories. This is also referred to as the *none-of-the-above* category.

3. Adversarial Attacks

Adversarial attacks can be performed in different settings: *white-box* attacks compute the gradient of the networks output with respect to the input, whereas in the *black-box* setting such calculations are not possible. Furthermore, adversarial attacks can be classified into *targeted* attacks, where the goal of the attack is that the network assigns a chosen label to the manipulated image, and *untargeted* attacks, where the attacker's goal is to fool the network in the sense that it missclassifies a given image.

Throughout this paper we denote the input image as $x \in [0,1]^{n\times n}$, the neural network's output logits as $Z(x)$ and the perturbation as δ. If $F(x)$ is the output of the network interpretable as probability, then $F(x) = \mathrm{softmax}(Z(x))$ in the case of the ConvNet and $Z(x) = \mathrm{arctanh}(2F(x)-1)$ in the case of the CapsNet. We refer to the label assigned to x by the networks as $C(x)$ and to the correct label of x by $C^*(x)$. Furthermore, we denote the i-th entry of $Z(x)$ as $Z(x)_i$.

3.1. Carlini-Wagner Attack

The Carlini-Wagner (CW) attack (2017) is a targeted white-box attack and performed by solving the following constrained optimization problem

$$\begin{aligned} \underset{\delta}{\text{minimize}} \quad & ||\delta||_2 + c\cdot \max(G(x,\delta,t) - Z(x)_t, -\kappa) \\ \text{subject to} \quad & x+\delta \in [0,1]^{n\times n}, \end{aligned} \tag{1}$$

where $G(x,\delta,t) := \max_{i\neq t}(Z(x+\delta)_i)$ and $c > 0$. The parameter $\kappa > 0$ controls the confidence. The optimal value for c, i.e. the smallest value, that results in an adversarial example, is found using a binary search. To ensure the box-constraint on $x+\delta$ the authors suggested the following transform of variables

$$\delta = \frac{1}{2}(\tanh(w)+1) - x, \tag{2}$$

where the $\tanh$-function is applied componentwise. After this transformation the optimization problem is treated as unconstrained and can be solved in terms of w using the Adam optimizer (Kingma & Ba, 2014). Carlini and Wagner (2017) also proposed two different approaches to handle the box-constraint: projected gradient descent and clipped gradient descent. For details we refer the reader to the original work (Carlini & Wagner, 2017).

3.2. Boundary Attack

The idea of the boundary attack as introduced by Brendel et al. (2017) is to sample a perturbation which leads to a missclassification of the original image $x^{(0)} := x$. Additionally, the desired perturbation should have the smallest possible norm. The initial perturbation $\delta^{(0)}$ is sampled componentwise from a uniform distribution $\delta^{(0)}_{ij} \sim \mathcal{U}(0,1)$. Initial perturbations, which are not missclassified, are rejected. During the attack adversarial images are constructed iteratively $x^{(k+1)} := x^{(k)} + \delta^{(k)}$ by a random walk close to the decision boundary. During this random walk the following three conditions are enforced by appropriate scaling and clipping of the image and the perturbation:

1. The new image $x^{(k+1)}$ is in the range of a valid image, i.e. in $x^{(k+1)} \in [0,1]^{n\times n}$.
2. The proportion of the size of the perturbation $\delta^{(k)}$ and the distance to the given image is equal to a given parameter γ.
3. The reduction of the distance from the adversarial image to the original image $d(x,x^{(k)}) - d(x,x^{(k+1)})$ is proportional to $d(x,x^{(k)})$ with $\nu > 0$.

The parameters γ and ν are adjusted dynamically, similarly to Trust Region methods.

3.3. DeepFool Attack

DeepFool is an untargeted white-box attack developed by Moosavi-Dezfooli et al. (2016). The authors found that minimal adversarial perturbations for affine multiclass classifiers can be computed exactly and quickly, by calculating the distance to the (linear) decision boundaries and making an orthogonal projection to the nearest boundary. DeepFool initializes $\delta^{(0)} \leftarrow 0$ and then iteratively approximates F with its first degree Taylor polynomial at $x+\delta^{(k)}$, computes a perturbation $\Delta\delta^{(k)}$ for this approximation as described above and updates $\delta^{(k+1)} \leftarrow \delta^{(k)} + \Delta\delta^{(k)}$. For better results, we restrict the norm of $\Delta\delta^{(k)}$ each step k.

3.4. Universal Adversarial Perturbations

A universal perturbation is a single vector $\delta \in \mathbb{R}^{n\times n}$, such that $C(x+\delta) \neq C^*(x)$ for multiple x sampled from the input image distribution. This concept was proposed by Moosavi-Dezfooli et al. (2017) and we use a variation of their algorithm, which we briefly describe in the following. As long as the accuracy on the test set is above a previously chosen threshold, repeat these steps:

Table 1. Average perturbation norms for each attack and architecture.

Attack	Network	MNIST	Fashion	SVHN	CIFAR10
CW	ConvNet	1.40	0.51	0.67	0.37
	CapsNet	1.82	0.50	0.60	0.23
Boundary	ConvNet	3.07	1.24	2.42	1.38
	CapsNet	3.26	0.93	1.88	0.72
DeepFool	ConvNet	1.07	0.31	0.41	0.23
	CapsNet	2.02	0.55	0.80	0.16
Universal	ConvNet	6.71	2.61	2.46	2.45
	CapsNet	11.45	5.31	8.59	2.70

Table 2. Fooling rates of adversarial examples calculated for a CapsNet and evaluated on a ConvNet and vice versa. For the universal attack we report the accuracy on the whole test set.

Attack	Network	MNIST	Fashion	SVHN	CIFAR10
CW	ConvNet	0.8%	1.2%	2.8%	2.4%
	CapsNet	2.0%	2.0%	3.8%	2.0%
Boundary	ConvNet	8.8%	9.5%	10.5%	13.4%
	CapsNet	14.2%	14.6%	12.9%	26.1%
DeepFool	ConvNet	4.3%	8.5%	13.5%	11.8%
	CapsNet	0.9%	10.9%	10.8%	14.1%
Universal	ConvNet	4.9%	20.4%	35.0%	25.9%
	CapsNet	38.2%	25.7%	53.4%	47.2%

1. Initialize $\delta^{(0)} \leftarrow 0$.
2. Sample a batch $X^{(k)} = \{x_1^{(k)}, ..., x_N^{(k)}\}$ of images with $\forall x \in X^{(k)} : C(x + \delta^{(k)}) = C^*(x)$.
3. For each $x_i^{(k)}$ compute a perturbation $\delta_i^{(k+1)}$ using FGSM (Goodfellow et al., 2014).
4. Update the perturbation:

$$\delta^{(k+1)} \leftarrow \delta^{(k)} + \frac{1}{N}\sum_{i=0}^{N} \delta_i^{(k+1)}$$

Since this method depends on the FGSM it is a white-box attack.

4. Experiments

4.1. Datasets and Network Architectures

We train models on each of the following benchmark datasets: MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011) and CIFAR-10 (Krizhevsky & Hinton, 2009). Each dataset consists of ten different classes. As a baseline architecture we use a ConvNet which we trained on each of the datasets using batch-normalization (Ioffe & Szegedy, 2015) and dropout (Srivastava et al., 2014). Since training CapsNets can be rather difficult in practice, we had to carefully select appropriate architectures:

Like Sabour et al. (2017) we use a three layer CapsNet for the MNIST dataset, where we only used 64 convolutional kernels in the first layer. For the Fashion-MNIST and the SVHN dataset we use two convolutional layers at the beginning (32 and 32 channels with 3×3 filter for Fashion-MNIST, 64 and 256 channels with 5×5 filter for SVHN), followed by a convolutional capsule layer with 8D capsules, 32 filter with size 9×9 and a stride of 2, and finally a capsule layer with one 16D capsule per class. Since CapsNets have problems with more complex data like CIFAR10 (Xi et al., 2017), we use a modified DCNet (Phaye et al., 2018) with three convolutional capsule layers and so-called none-of-the-above category for the dynamic routing for this dataset. We train all CapsNet architectures using the margin loss and the reconstruction loss for regularization (Sabour et al., 2017).

For each dataset we calculate 1000 adversarial examples on images randomly chosen from the test set using the DeepFool attack and the boundary attack. For the Carlini-Wagner attack we calculate 500 adversarial examples again on random samples from the test set (with hyperparameter $\kappa = 1$). The target labels too are chosen at random, but different from the true labels. To evaluate the performance of universal perturbation we split the test set in ten parts and compute ten adversarial perturbations according to the procedure described in Sec. 3.4 on each part.

None of the attacks restrict the norm of the perturbation. This means, the Carlini-Wagner, the boundary and the DeepFool attack generate only valid adversarial examples. In the case of the universal perturbation, we stop once accuracy falls below 50%.

4.2. Results

We are aware of the fact that the test accuracies shown in Tab. 3 of our models are not state-of-the-art. However, we found our models to be suitable for the given task, since the similar performances of ConvNets and CapsNets ensure comparability.

We also compare the average Euclidean norm of the perturbation for each attack, dataset and network. The results are displayed in Tab. 1. Our main result is that applying the Carlini-Wagner attack on the CapsNets yields smaller adversarial perturbations than on the ConvNet. Nevertheless,

Table 3. Test accuracies achieved by our networks.

Network	MNIST	Fashion-MNIST	SVHN	CIFAR10
ConvNet	99.39%	92.90%	92.57%	88.22%
CapsNet	99.40%	92.65%	92.35%	88.21%

for most of the dataset we found that the DeepFool attack performs worse on the CapsNets.

To compare the transferability of adversarial examples we calculate perturbations on the ConvNet and apply those to the CapsNet and vice versa (see Tab. 2). In case of the (targeted) Carlini-Wagner (CW) attack we define a network *fooled* if the perturbed image is classified with the target label. For the Carlini-Wagner attack, the boundary attack and the DeepFool attack our results fit to those displayed in Tab. 1. Especially the perturbations calculated using the universal attack seem to generalize well on the other architecture. For this attack we also found out that the smaller perturbations calculated on the ConvNet can be successfully transferred to CapsNets, while the other way around this approach was less effective, although the norms of the perturbations for the CapsNets are very large.

The adversarial examples for the CapsNets calculated with the Carlini-Wagner, the boundary and the DeepFool attack are not visible for the human eye. Only the universal perturbations are observable (see Fig. 1).

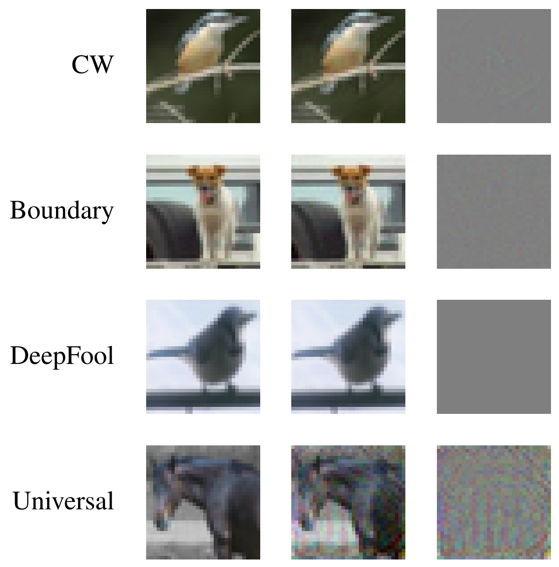

Figure 1. Original images from the CIFAR10 dataset (left), adversarial images (middle) and the corresponding perturbation (right) calculated for a CapsNet.

4.3. Visualizing Universal Perturbations

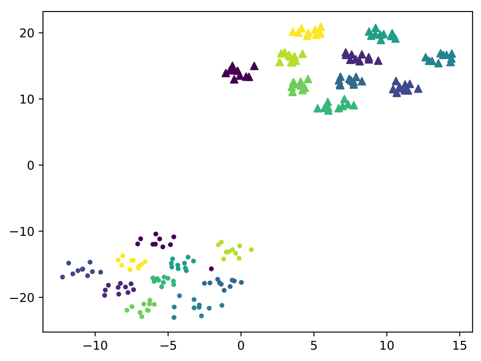

Figure 2. Two dimensional embedding of the universal perturbations calculated using t-SNE (Maaten & Hinton, 2008). The upper right cluster represents perturbations calculated on a ConvNet, whereas the lower left cluster represents those calculated on a CapsNet. Perturbations with the same color were created using the same subset of test data.

We also visualized the universal perturbations calculated for the CapsNet and for the ConvNet using t-SNE (Maaten & Hinton, 2008) and we observe that the perturbations for the CapsNet seem to be inherently different than the perturbations for the ConvNets (see Fig. 2).

5. Conclusion

Our experiments show that CapsNets are not in general more robust to white-box attacks. With sufficiently sophisticated attacks CapsNets can be fooled as easily as ConvNets. Our experiments also show that the vulnerability of CapsNets and ConvNets is similar and it is hard to decide which model is more prone to adversarial attacks than the other. Moreover, we showed that adversarial examples can be transferred between the two architectures.

To fully understand the possibly distinguishable roles of the convolutional and capsule layers with respect to adversarial attacks, we are currently examining the effects of attacks on the activation level of single neurons. However, this analysis is not finished yet and beyond the scope of this paper.

References

Brendel, W., Rauber, J., and Bethge, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Frosst, N., Sabour, S., and Hinton, G. DARCCC: Detecting adversaries by reconstruction from class conditional capsules. *arXiv preprint arXiv:1811.06969*, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hinton, G. E., Sabour, S., and Frosst, N. Matrix capsules with em routing. *International Conference on Learning Representations*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov): 2579–2605, 2008.

Marchisio, A., Nanfa, G., Khalid, F., Hanif, M. A., Martina, M., and Shafique, M. CapsAttacks: Robust and imperceptible adversarial attacks on capsule networks. *CoRR*, abs/1901.09878, 2019. URL http://arxiv.org/abs/1901.09878.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: A simple and accurate method to fool deep neural Networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature Learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Peer, D., Stabinger, S., and Rodríguez-Sánchez, A. J. Training deep capsule networks. *CoRR*, abs/1812.09707, 2018. URL http://arxiv.org/abs/1812.09707.

Phaye, S. S. R., Sikka, A., Dhall, A., and Bathula, D. R. Dense and diverse capsule networks: Making the capsules learn better. *CoRR*, abs/1805.04001, 2018. URL http://arxiv.org/abs/1805.04001.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 3856–3866. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Xi, E., Bing, S., and Jin, Y. Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*, 2017.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.