MATRIX CAPSULES WITH EM ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

A capsule is a group of neurons whose outputs represent different properties of the same entity. We describe a version of capsules in which each capsule has a logistic unit to represent the presence of an entity and a 4x4 pose matrix which could learn to represent the relationship between that entity and the viewer. A capsule in one layer votes for the pose matrix of many different capsules in the layer above by multiplying its own pose matrix by viewpoint-invariant transformation matrices that could learn to represent part-whole relationships. Each of these votes is weighted by an assignment coefficient. These coefficients are iteratively updated using the EM algorithm such that the output of each capsule is routed to a capsule in the layer above that receives a cluster of similar votes. The whole system is trained discriminatively by unrolling 3 iterations of EM between each pair of adjacent layers. On the smallNORB benchmark, capsules reduce the number of test errors by 45% compared to the state-of-the-art. Capsules also show far more resistance to white box adversarial attack than our baseline convolutional neural network.

1 INTRODUCTION

Convolutional neural nets are based on a simple fact and a simple mechanism: a vision system needs to use the same knowledge at all locations in the image; this can be achieved by tying the weights of features detectors so that features learned at one location are available at other locations. Convolutional capsules extend the sharing of knowledge across locations to include knowledge about the part-whole relationships that characterize a familiar shape. Viewpoint changes have complicated effects on pixel intensities but simple, linear effects on the pose matrix that represents the relationship between an object or object-part and the viewer. The aim of capsules is to make good use of this underlying linearity, both for dealing with viewpoint variation and improving segmentation decisions.

Capsules use high-dimensional coincidence filtering: a familiar object can be detected by looking for agreement between votes for its pose matrix. These votes come from parts that have already been detected. A part produces a vote by multiplying its own pose matrix by a transformation matrix that represents the viewpoint invariant relationship between the part and the whole. As the viewpoint changes, the pose matrices of the parts and the whole will change in a coordinated way so that any agreement between votes from different parts will persist. Finding tight clusters of high-dimensional votes that agree in a mist of irrelevant votes is one way of solving the problem of assigning parts to wholes. It is non-trivial because we cannot grid the high-dimensional space. To solve this challenge, we use a fast iterative process called ”routing-by-agreement” that updates the probability with which a part is assigned to a whole based on the proximity of the vote coming from that part to the votes coming from other parts that are assigned to the whole. This is a powerful segmentation principle that allows knowledge of familiar shapes to drive segmentation, rather than just using low-level cues such as proximity or agreement in color or velocity.

1.1 PREVIOUS WORK ON CAPSULES

Hinton et al. (2011) used a transformation matrix in a transforming autoencoder that learned to transform a stereo pair of images into a stereo pair from a slightly different viewpoint. However, that system required the transformation matrix to be supplied externally. More recently, routing-by-agreement was shown to be very effective for segmenting highly overlapping digits (Sabour et al. (2017)), but that system has several deficiencies that we have defeated in our work:

1. It uses the length of the pose vector to represent the probability that the entity represented by a capsule is present. To keep the length less than 1 requires an unprincipled non-linearity that prevents there from being any sensible objective function that is minimized by the iterative routing procedure.
2. It uses the cosine of the angle between two pose vectors to measure their agreement. Unlike the log variance of a Gaussian cluster, the cosine is not good at distinguishing between quite good agreement and very good agreement.
3. It uses a vector of length n rather than a matrix with n elements to represent a pose, so its transformation matrices have n^2 parameters rather than just n.

1.2 Related work

Among the multiple recent attempts on bringing underlying affine manifold of images to CNNs, the two main streams are attempts on gaining viewpoint invarient vs equivarient. Spatial transformer networks (Jaderberg et al. (2015) seeks viewpoint invarient by changing the sampling of CNNs according to a selection of affine transformations. De Brabandere et al. (2016) extends spatial transformer networks where filters are adapted during inference depending on the input. They generate different filters for each locality in the feature map rather than applying the same transformation to all filters. Their approach is a step toward input covariance detection from traditional pattern matching frameworks like standard convolutional neural networks (LeCun et al. (1990)). Dai et al. (2017) also improves upon STNs by generalizing the sampling method of filters. Our work differs substantially in that a unit is not activated based on the matching score with a filter (either fixed or dynamically changing during inference). In our case, a capsule is activated only if the transformed poses coming from the layer below match each other. This is a more effective way to capture covariance and leads to models with many fewer parameters that generalize better.

The success of CNNs has motivated many researchers to extend the translational equivariance built in to CNNs to include rotational equivariance (Cohen & Welling (2016), Dieleman et al. (2016), Oyallon & Mallat (2015)). The recent approach in Harmonic Networks (Worrall et al. (2017)) achieves rotation equivariant feature maps by using circular harmonic filters and returning both the maximal response and orientation using complex numbers. This shares the basic representational idea of capsules: By assuming that there is only one instance of the entity at a location, we can use several different numbers to represent its properties. They used a fixed number of streams of rotation orders. By enforcing the equality of the sum of rotation orders along any path they achieve patch-wise rotation equivariance. This approach is more parameter-efficient than data augmentation approaches, duplicating feature maps, or duplicating filters (Fasel & Gatica-Perez (2006), Laptev et al. (2016)). Our approach encodes general viewpoint equivarience rather than only Affine 2D rotations. Symmetry networks (Gens & Domingos (2014)) uses iterative Lucas-Kanade optimization to find poses that are supported by the most low-level features. Their key weakness is that the iterative algorithm always starts at the same poses, rather than the mean of the bottom-up votes.

Lenc & Vedaldi (2016) proposes a feature detection mechanism (DetNet) that is equivariant to affine transformations. DetNet is designed to detect the same points in the image under different viewpoint variations. This effort is orthogonal to our work but DetNet might be a good way to implement the de-rendering first-stage that activates the layer of primary capsules.

Our routing algorithm can be seen as an attention mechanism. In this view it is related to work of Gregor et al. (2015), where they improved the decoder performance in a generative model by using Gaussian kernels to attend to different parts of the feature map generated by the encoder. Vaswani et al. (2017) uses a softmax attention mechanism to match parts of the query sequence to parts of the input sequence for the translation task and when generating an encoding for the query. They show improvement upon previous translation efforts using recurrent architectures. Our algorithm has attention in the opposite direction. The competition is not between the lower-level capsules that a higher-level capsule might attend to. It is between the higher-level capsules that a lower-level capsule might send its vote to.

2 Using EM for routing-by-agreement

Let us suppose that we have already decided on the poses and activation probabilities of all the capsules in a layer and we now want to decide which capsules to activate in the layer above and how to assign each active lower-level capsule to one active higher-level capsule. For explanatory purposes, we start by assuming that all of the transformation matrices are the identity matrix because this makes it easy to map the assignment task onto the task of fitting a mixture of Gaussians. Each capsule in the higher layer corresponds to a Gaussian and each active capsule in the lower layer corresponds to a data-point (or a fraction of a data-point if the capsule is partially active). The only difference from a standard mixture of Gaussians is that we have far too many Gaussians so we need to introduce a penalty to stop us from assigning every data-point to a different Gaussian.

Using the minimum description length principle we have a choice when deciding whether or not to activate a higher-level capsule. **Choice 0**: if we do not activate it, we must pay a fixed cost per data-point for describing the means of all the lower-level capsules that are assigned to a higher-level capsule (assuming an improper uniform prior). For fractional assignments we pay that fraction of the fixed cost. **Choice 1**: if we do activate the higher-level capsule we must pay a fixed cost for coding its mean and the fact that it is active and then pay additional costs, pro-rated by the assignment probabilities, for describing the discrepancies between the lower-level means and the values predicted for them when the mean of the higher-level capsule is used to predict them via the inverse of the transformation matrix. A much simpler way to compute the cost of describing the discrepancies is to use the squared distance between the mean of the higher-level capsule and the vote from the lower-level capsule, scaled by the variance of the higher-level capsule. This is incorrect because the transformation matrix may compress or expand probability density but we use it anyway because it saves us from having to perform any additional transformations during the iterative routing process. The difference in cost between these two choices, is put through the logistic function on each iteration to determine the higher-level capsule's activation probability.

If the transformation matrices are not the identity matrix, the iterative routing process differs from fitting a standard mixture of Gaussians and the difference makes it converge much faster. The higher-level capsules all get different views of the same set of data-points because they see the data-points transformed by different part-whole relationships. This means that data-points that form a tight cluster from the perspective of one capsule may be widely scattered from the perspective of another capsule. These different perspectives break the symmetry of the Gaussians much more strongly than simply starting with different means and variances.

The cost of explaining a whole data-point i by using capsule c that has an axis-aligned covariance matrix is simply the sum over all dimensions of the cost of explaining each dimension, h, of i. This is simply $-ln(P_{ih})$ where P_{ih} is the probability density of the h^{th} component of the vote from i under the capsule's Gaussian model for dimension h which has variance σ_h^2:

$$P_{ih} = \frac{1}{\sqrt{2\pi\boldsymbol{\sigma}_h^2}} e^{-\frac{(V_{ih}-\mu_h)^2}{2\sigma_h^2}}$$

$$ln(P_{ih}) = -\frac{(V_{ih}-\mu_h)^2}{2\sigma_h^2} - ln(\sigma_h) - ln(2\pi)/2 \tag{1}$$

$$\tag{2}$$

Summing over all lower-level capsules for a single dimension of one higher-level capsule we get:

$$cost_h = \sum_i -r_i ln(P_{ih}) = \frac{\sum_i r_i \sigma_h^2}{2\sigma_h^2} + (ln(\sigma_h) + ln(2\pi)/2)\sum_i r_i = (ln(\sigma_h) + k)\sum_i r_i \tag{3}$$

Where $\sum_i r_i$ is the amount of data assigned to c (*i.e.* the sum of its assignment probabilities, r_i), k is a constant and V_{ih} is the value on dimension h of the vote from capsule i to capsule c. V_{ih} is the product of the the transformation matrix W_{ic} that is learned discriminatively. Turning on c *increases* the description length for the means of the lower-level capsules assigned to c by the sum of the cost over all dimensions, so we define the activation function of capsules c to be:

$$a_c = logistic(\lambda(b - \sum_h cost_h)) \tag{4}$$

where $-b$ represents the cost of describing the mean of capsule c and λ is an inverse temperature parameter. We learn b discriminatively and set a fixed schedule for λ as a hyper-parameter.

For finalizing the pose parameters and activations of the capsules in layer $L + 1$ we run the EM algorithm for few iterations (normally 3) after the pose parameters and activations have already

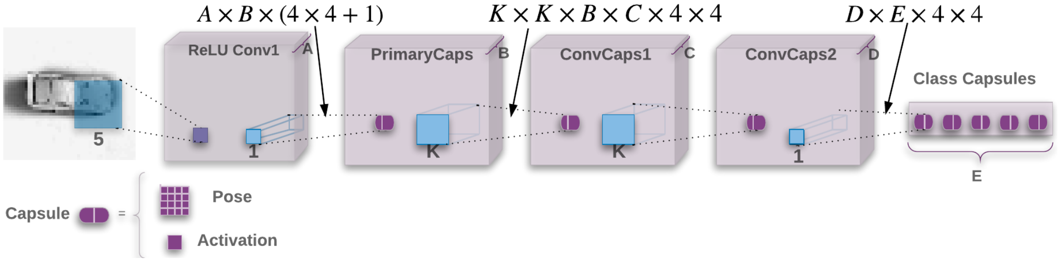

$A \times B \times (4 \times 4 + 1)$
$K \times K \times B \times C \times 4 \times 4$
$D \times E \times 4 \times 4$
ReLU Conv1
A
PrimaryCaps
B
ConvCaps1
C
ConvCaps2
D
Class Capsules
5
1
K
K
1
E
Capsule
=
Pose
Activation

Figure 1: A network with one ReLU convolutional layer followed by a primary convolutional capsule layer and two more convolutional capsule layers.

been finalized in layer L. The non-linearity implemented by a whole capsule layer is a form of cluster finding using the EM algorithm, so we call it **EM Routing**.

Procedure 1 Routing algorithm[1] returns **activation** and **pose** of the capsules in layer $L+1$ given the **activations** and **votes** of capsules in layer L. V_{ich} is an H dimensional vote from capsule i with activation a_i in layer L to capsule c in layer $L+1$. β_a, β_v are learned discriminatively and the inverse temperature λ increases at each iteration with a fixed schedule.

```
procedure EM ROUTING($\boldsymbol{a}$, $V$)
    $\forall i, c$: $R_{ic} \leftarrow 1/size(L+1)$
    for $t$ iterations do
        $\forall c$: $M_{c:}, S_{c:}, \boldsymbol{a}'_c \leftarrow$ M-STEP($R_{:c}$, $\boldsymbol{a}$, $V_{:c:}$)
        $\forall i$: $R_{i:} \leftarrow$ E-STEP($M$, $S$, $\boldsymbol{a}'$, $V_{i::}$)
    return $\boldsymbol{a}'$, $M$

procedure M-STEP($\boldsymbol{r}$, $\boldsymbol{a}$, $V'$)                    ▷ for one higher-level capsule
    $\forall i$: $\boldsymbol{r}'_i \leftarrow \boldsymbol{r}_i * \boldsymbol{a}_i$
    $\forall h$: $\boldsymbol{\mu}_h \leftarrow \frac{\sum_i \boldsymbol{r}'_i V'_{ih}}{\sum_i \boldsymbol{r}'_i}$
    $\forall h$: $\boldsymbol{\sigma}^2_h \leftarrow \frac{\sum_i \boldsymbol{r}'_i (V'_{ih} - \boldsymbol{\mu}_h)^2}{\sum_i \boldsymbol{r}'_i}$
    $cost_h \leftarrow (\beta_v + log(\boldsymbol{\sigma}_h)) \sum_i \boldsymbol{r}'_i$
    $a' \leftarrow sigmoid(\lambda(\beta_a - \sum_h cost_h))$
    return $\boldsymbol{\mu}, \boldsymbol{\sigma}, a'$

procedure E-STEP($\boldsymbol{a}'$, $S$, $M$, $V''$)                        ▷ for one lower-level capsule
    $\forall c$: $\boldsymbol{p}_c \leftarrow \frac{1}{\sqrt{\prod_h^H 2\pi \boldsymbol{S}^2_{ch}}} e^{-\sum_h^H \frac{(V''_{ch} - \boldsymbol{M}_{ch})^2}{2\boldsymbol{S}^2_{ch}}}$
    $\forall c$: $\boldsymbol{r}_c \leftarrow \frac{\boldsymbol{a}'_c p_c}{\sum_j \boldsymbol{a}'_j p_j}$
    return $\boldsymbol{r}$
```

2.1 THE OBJECTIVE FUNCTION FOR EM ROUTING

There is a free energy function that is reduced by each step of the routing. The recomputation of the means and the variances reduces an energy equal to the squared distances of the votes from the means, weighted by the assignment probabilities. The recomputation of the capsule activations reduces a free energy equal to the sum of the (scaled) costs used in the logistic function minus the entropy of the activation values. The recomputation of the assignment probabilities reduces the sum of the two energies above minus the entropies of the assignment probabilities.

3 THE CAPSULES MODEL

The general architecture of our model is shown in Fig. 1. The model starts with a 5x5 convolutional layer with 32 channels (A=32) and a stride of 2 with a ReLU non-linearity. All the other layers

[1] We use Matlab-like notation for indexing tensors and matrices. An index replaced with ":" indicates a slice over that index.

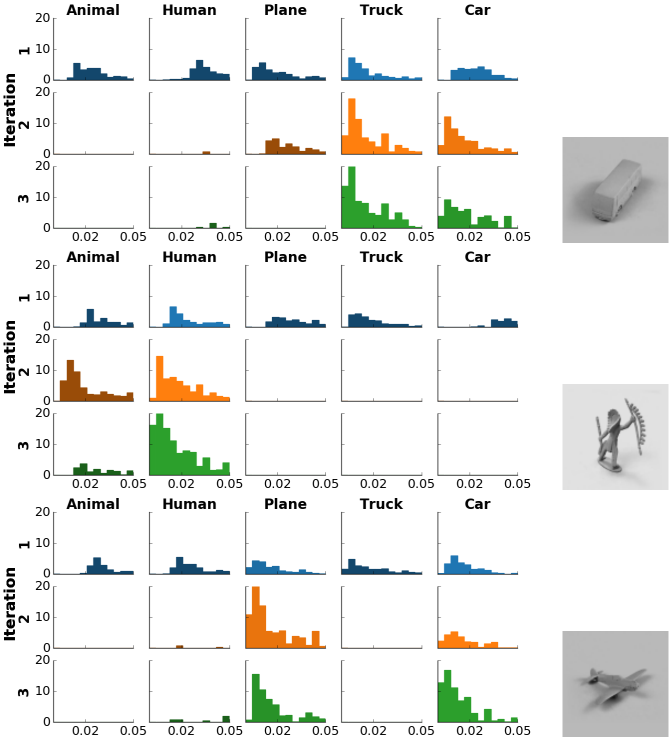


Figure 2: Histogram of distances of votes to the mean of each of the 5 final capsules after each routing iteration. Each distance point is weighted by its assignment probability. All three images are selected from the smallNORB test set. The routing procedure correctly routes the votes in the truck and the human example. The plane example shows a rare failure case of the model where the plane is confused with a car in the third routing iteration. The histograms are zoomed-in to visualize only votes with distances less than 0.05. Fig. A.2 shows the complete histograms for the "human" capsule without fixing the y axis.

are capsule layers starting with the primary capsule layer. Each capsule has a 4x4 pose matrix and one logistic activation unit. The 4x4 pose of each of the B=32 primary capsule types is a linear transformation of the output of all the lower layer ReLUs centered at that location. The activations of the primary capsules are produced by applying the sigmoid function to weighted sums of the same set of lower layer ReLUs.

The primary capsules are followed by two 3x3 convolutional capsule layers (K=3), each with 32 capsule types (C=D=32) with strides of 2 and one, respectively. The last layer of convolutional capsules is connected to the final capsule layer which has one capsule per output class.

When connecting the last convolutional capsule layer to the final layer we do not want to throw away information about the location of the convolutional capsules but we also want to make use of the fact that all capsules of the same type are extracting the same entity at different positions. We therefore share the transformation matrices between different positions of the same capsule type and add the scaled coordinate (row, column) of the center of the receptive field of each capsule to the first two elements of its vote. We refer to this technique as Coordinate Addition. This should encourage the shared final transformations to produce values for those two elements that represent the fine position of the entity relative to the center of the capsule's receptive field.

The routing procedure is used between each adjacent pair of capsule layers. For convolutional capsules, each capsule in layer $L + 1$ sends feedback only to capsules within its receptive field in layer L. Therefore each convolutional instance of a capsule in layer L receives at most *kernel_size* X *kernel_size* feedback from each capsule type in layer $L + 1$. The instances closer to the border of the image receive fewer feedbacks with corner ones receiving only one feedback per capsule type in layer $L + 1$.

Table 1: The effect of varying different components of our capsules model on smallNORB.

Routing iterations	Pose structure	Loss	Coordinate Addition	Test error rate
1	Matrix	Spread	Yes	9.7%
2	Matrix	Spread	Yes	2.2%
3	Matrix	Spread	Yes	**1.8**%
5	Matrix	Spread	Yes	3.9%
3	Vector	Spread	Yes	2.9%
3	Matrix	Spread	No	2.6%
3	Vector	Spread	No	3.2%
3	Matrix	Margin[2]	Yes	3.2%
3	Matrix	CrossEnt	Yes	5.8%
Baseline CNN with 4.2M parameters				5.2%
CNN of Cireşan et al. (2011) with extra input images & deformations				2.56%
Our Best model (third row), with multiple crops during testing				**1.4**%

3.1 Spread Loss

In order to make the training less sensitive to the initialization and hyper-parameters of the model, we use "spread loss" to directly maximize the gap between the activation of the target class (a_t) and the activation of the other classes. If the activation of a wrong class, a_i, is closer than the margin, m, to a_t then it is penalized by the squared distance to the margin:

$$L_i = (max(0, m - (a_t - a_i))^2, \quad L = \sum_{i \neq t} L_i \tag{5}$$

By starting with a small margin of 0.2 and linearly increasing it during training to 0.9, we avoid dead capsules.

4 Experiments

The smallNORB dataset (LeCun et al. (2004)) has gray-level stereo images of 5 classes of toy: airplanes, cars, trucks, humans and animals. There are 10 physical instances of each class which are painted matte green. 5 physical instances of a class are selected for the training data and the other 5 for the test data. Every individual toy is pictured at 18 different azimuths (0-340), 9 elevations and 6 lighting conditions, so the training and test sets each contain 24,300 stereo pairs of 96x96 images. We selected smallNORB as a benchmark for developing our capsules system because it is carefully designed to be a pure shape recognition task that is not confounded by context and color, but it is much closer to natural images than MNIST .

We downsample smallNORB to 48×48 pixels and normalize each image to have zero mean and unit variance. During training we randomly crop 32×32 patches and, after add random brightness and contrast to the image. During test we crop a 32×32 patch from the center of the image. We achieve $\mathbf{1.4}\%$ test error on smallNORB when averaging the class activations over multiple patches and 1.8% when only using the cetral crop. The best reported result on smallNORB is 2.56% (Cireşan et al. (2011)). To achieve this they added two additional stereo pairs of input images that are created by using an on-center off-surround filter and an off-center on-surround filter. They also applied affine distortions to the images. Our work also beats the Sabour et al. (2017) capsule work which achieves 2.7% on smallNORB.

We also tested our model on NORB which is a jittered version of smallNORB with added background and we achieved a 2.6% error rate which is on par with the state-of-the-art of 2.7% (Ciresan et al. (2012)).

As the baseline for our experiments on generalization to novel viewpoints we train a CNN which has two convolutional layers with 32 and 64 channels respectively. Both layers have kernel size 5 and

[2] The loss proposed by Sabour et al. (2017) for training Capsules.

Table 2: A comparison of the smallNORB test error rate of the baseline CNN and the capsules model on novel viewpoints when the capsules model is matched to the CNN for error rate on familiar viewpoints.

Test set	Azimuth		Elevation	
	CNN	Capsules	CNN	Capsules
Novel viewpoints	20%	13.5%	17.8%	12.3%
Familiar viewpoints	3.7%	3.7%	4.3%	4.3%

stride of 1 with a 2×2 max pooling. The third layer is a 1024 unit fully connected layer with dropout and connects to the 5-way softmax output layer. All hidden units use the ReLU non-linearity. We use the same image preparation for CNN baseline as described above for the capsule network. Our baseline CNN was the result of an extensive hyperparameter search over filter sizes, numbers of channels and learning rates.

The CNN baseline achieves 5.2% test error rate on smallNORB and has 4.2M parameters. We deduce that the Cireşan et al. (2011) network has 2.7M parameters. By using small matrix multiplies we reduced the number of parameters by a factor of 15 to 310K compared with our baseline CNN (and 9 w.r.t Cireşan et al. (2011)). A smaller capsule network of $A = 64, B = 8, C = D = 16$ with only 68K trainable parameters achieves 2.2% test error rate which also beats the prior state-of-the-art.

Replacing the MDL-derived capsule activation term with posterior probabilities that are proportional to the sum of the assignment probabilities increases the test error rate on smallNORB to 4.5%. Tab. 1 summarizes the effects of the number of routing iterations, the type of loss and the use of matrices rather than vectors for the poses.

Exactly the same capsules model as Fig. 1 achieves 0.44% test error rate on MNIST. If the number of channels in the first hidden layer is increased to 256, it achieves 11.9% test error rate on Cifar10 (Krizhevsky & Hinton (2009)).

4.1 Generalization to novel viewpoints

A more severe test of generalization is to use a limited range of viewpoints for training and to test on a much wider range. We trained both our convolutional baseline and our capsule model on one third of the training data containing azimuths of (300, 320, 340, 0, 20, 40) and tested on the two thirds of the test data that contained azimuths from 60 to 280. In a separate experiment we trained on the 3 smaller elevations and tested on the 6 larger elevations.

It is hard to decide if the capsules model is better at generalizing to novel viewpoints because it is better at all viewpoints. To eliminate this confounding factor we stopped training the capsules model when its performance matched the baseline CNN on the third of the test set that used the training viewpoints. Then we compared these matched models on the two thirds of the test set with novel viewpoints. Results in Tab. 2 show that, compared with the baseline CNN, capsules with matched performance on familiar viewpoints reduce the test error rate on novel viewpoints by about 30% for both novel azimuths and novel elevations.

5 Adversarial Robustness

There is growing interest in the vulnerability of neural networks to adversarial examples; inputs that have been slightly changed by an attacker so as to trick a neural net classifier into making the wrong classification. These inputs can be created in a variety of ways, but straightforward strategies such as FGSM (Goodfellow et al. (2014)) have been shown to drastically decrease accuracy in convolutional neural networks on image classification tasks. We compared our capsule model and a traditional convolutional model on their ability to withstand such attack.

FGSM computes the gradient of the loss w.r.t. each pixel intensity and then changes the pixel intensity by a fixed amount ϵ in the direction that increases the loss. So the changes only depend on

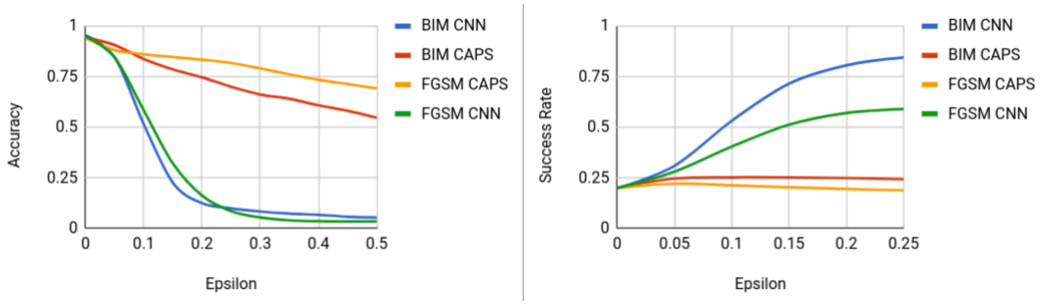


Figure 3: Accuracy against ϵ after an adversarial attack (left) and Success Rate after a targeted adversarial attack (right). The targeted attack results were evaluated by averaging the success rate after the attack for each of the 5 possible classes.

the sign of the gradient at each pixel. This can be extended to a targeted attack by updating the input to maximize the classification probability of a particular wrong class. We generated adversarial attacks using FGSM because it has only one hyper-parameter and it is easy to compare models that have very different gradient magnitudes. To test the robustness of our model we generated adversarial images from the test set using a fully trained model. We then reported the accuracy of the model on these images.

We found that our model is significantly less vulnerable to both general and targeted FGSM adversarial attacks; a small epsilon can be used to reduce a convolutional model's accuracy much more than an equivalent epsilon can on the capsule model (Fig. 3). It should also be noted that the capsule model's accuracy after the untargeted attack never drops below chance (20%) whereas the convolutional model's accuracy is reduced to significantly below chance with an epsilon of as small as 0.2.

We also tested our model on the slightly more sophisticated adversarial attack of a Basic Iterative Method (Kurakin et al. (2016)) which is simply the aforementioned attack except we take multiple smaller steps when creating the adversarial image. Here too we find that our model is much more robust to the attack than the traditional convolutional model.

It has been shown that some robustness to adversarial attacks in models can be due to simply numerical instability in the calculation of the gradient Brendel & Bethge (2017). To insure that this was not the sole cause of our model's robustness we calculated the percentage of zeros values in the gradient with respect to the image in the capsule model and found it to be smaller than that of the CNN. Furthermore the capsule gradients, although smaller that those of the CNN, are only smaller by 2 orders of magnitude, as opposed to 16 orders of magnitude in Brendel et al.'s work.

Finally we tested our model's robustness to black box attacks by generating adversarial examples with a CNN and testing them on both our capsule model and a different CNN. We found that the capsule model did not perform noticeably better at this task than the CNN.

6 Conclusion

Building on the work of Sabour et al. (2017), we have proposed a new capsule network architecture in which each capsule has a logistic unit to represent the presence of an entity and a 4x4 pose matrix to represent the relationship between that entity and the viewer. We also introduced a new iterative routing procedure between capsule layers, making use of the EM algorithm, which allows the output of each lower-level capsule to be routed to a capsule in the layer above so that each higher-level capsule receives a cluster of similar pose votes, if such a cluster exists. This new architecture achieves significantly better accuracy on the smallNORB data set then the state-of-the-art CNN, reducing the number of errors by 45%. We have also shown it to be significantly more robust to white box adversarial attacks than a baseline CNN.

SmallNORB is an ideal data-set for developing new shape-recognition models precisely because it lacks many of the additional features of images in the wild. Now that our capsules model works well on NORB, we plan to implement an efficient version so that we can test much larger models on much larger data-sets such as ImageNet.

REFERENCES

Wieland Brendel and Matthias Bethge. Comment on" biologically inspired protection of deep networks from adversarial attacks". *arXiv preprint arXiv:1704.01547*, 2017.

Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3642–3649. IEEE, 2012.

Dan C Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*, 2011.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999, 2016.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *arXiv preprint arXiv:1703.06211*, 2017.

Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *Neural Information Processing Systems (NIPS)*, 2016.

Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1889–1898. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045590.

Beat Fasel and Daniel Gatica-Perez. Rotation-invariant neoperceptron. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pp. 336–339. IEEE, 2006.

Robert Gens and Pedro M Domingos. Deep symmetry networks. In *Advances in neural information processing systems*, pp. 2537–2545, 2014.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pp. 1462–1471, 2015.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 289–297, 2016.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pp. 396–404, 1990.

Yann LeCun, Fu Jie Huang, and Leon Bottou. 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.

Karel Lenc and Andrea Vedaldi. Learning covariant feature detectors. In *Computer Vision–ECCV 2016 Workshops*, pp. 100–117. Springer, 2016.

Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2865–2873, 2015.

Sara Sabour, Nicholas Fross, and Geoffrey E Hinton. Dynamic routing between capsules. In *Neural Information Processing Systems (NIPS)*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NIPS)*, 2017.

Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

A Supplementary figures

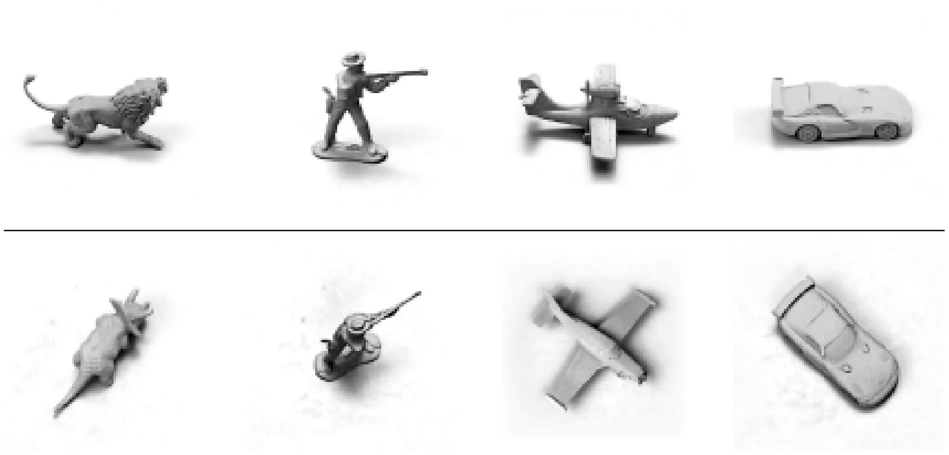


Figure A.1: Sample smallNORB images at different viewpoints. All images in first row are at azimuth 0 and elevation 0. The second row shows a set of images at a higher elevation and different azimuth.

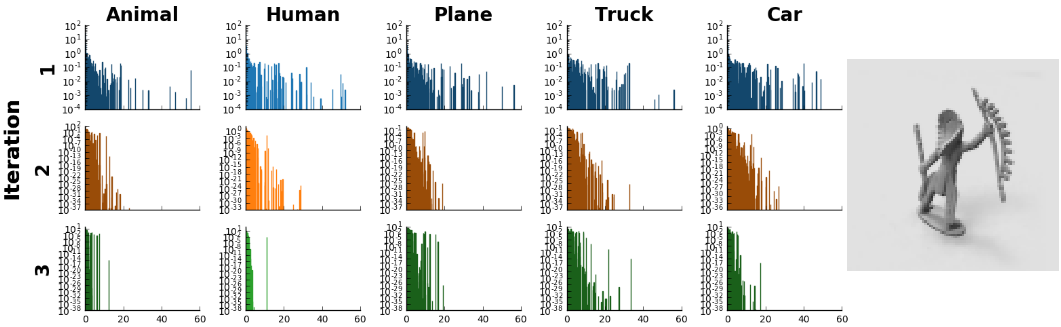
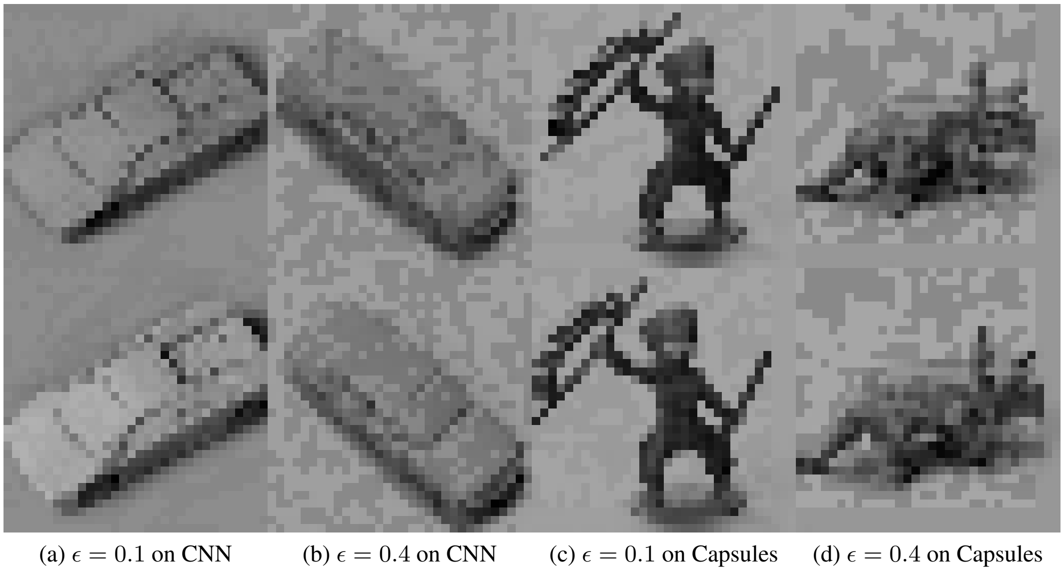


Figure A.2: Log scale histogram of distances between the receiving votes and the center of each of 5 final capsule. Each row visualizes the 5 histograms for iteration 1, 2 and 3. Unlike Fig. 2 the histograms are Log scale, not sharing their y axis and the considered distance range is 60 which results in non-empty histograms where most of the bins have height less than 1.0.

Figure A.3: Adverserial images generated with FGSM with $\epsilon = 0.1$ and $\epsilon = 0.4$ on the CNN model and the Capsule model.



(a) $\epsilon = 0.1$ on CNN (b) $\epsilon = 0.4$ on CNN (c) $\epsilon = 0.1$ on Capsules (d) $\epsilon = 0.4$ on Capsules

Figure A.4: Effect of tweaking the first two dimmensions of the capsules network trained on MNIST and reconstructing the image by passing the target capsule's pose through a 3 layer fully-connected decoding network. For this experiment the decoder was trained along the discriminative task and the squared distance of the reconstructed image with the original image was added to the discriminative loss. The decoder network is similar to the architecture Sabour et al. (2017) has used for reconstructions. In most of the digits adding same epsilon to the first two dimmensions of the pose matrix tends to shift the image toward the right boundary. In digit 6 adding the epsilon increases the distance to the right boundary of the image. In 3 and 5 it only shifts the bottom part toward the right boundary which results in a clock wise rotation. Since the decoder has only seen MNIST training data (without any transformation or augmentation), it can only reconstruct the part of the pose changes that is explainable by the MNIST digits.