Training Deep Capsule Networks

David Peer
University of Innsbruck
David.Peer@student.uibk.ac.at

Sebastian Stabinger
University of Innsbruck
Sebastian@Stabinger.name

Antonio Rodriguez-Sanchez
University of Innsbruck
Antonio.Rodriguez-Sanchez@uibk.ac.at

December 27, 2018

Abstract

The capsules of Capsule Networks are collections of neurons that represent an object or part of an object in a parse tree. The output vector of a capsule encodes the so called instantiation parameters of this object (e.g. position, size, or orientation). The *routing-by-agreement* algorithm routes output vectors from lower level capsules to upper level capsules. This iterative algorithm selects the most appropriate parent capsule so that the active capsules in the network represent nodes in a parse tree. This parse tree represents the hierarchical composition of objects out of smaller and smaller components. In this paper, we will show experimentally that the routing-by-agreement algorithm does not ensure the emergence of a parse tree in the network. To ensure that all active capsules form a parse tree, we introduce a new routing algorithm called *dynamic deep routing*. We show that this routing algorithm allows the training of deeper capsule networks and is also more robust to white box adversarial attacks than the original routing algorithm.

1 Introduction

Convolutional neural networks (CNNs) usually contain pooling layers to reduce computational complexity and provides translational invariance, as a side effect they lose information about feature locations. To detect objects in upper level layers it is important for neurons to make good use of the precise positions of different lower level parts of this object. Preserving the location information of detected features would therefore be desirable, but is hard to achieve in the context of CNNs.

Position is only one of a whole set of parameters describing the appearance of a feature; Size, rotation, and scale being other examples of the so called *instantiation parameters*. An elegant way to deal with these instantiation parameters was introduced by Hinton et al. (2011). The scalar-output neurons are replaced with vector-output *capsules*, which are collections of neurons. A capsule represents an object, or part of an object, and the *activity-vector* of the capsule encodes the instantiation parameters of this part. Hinton et al. (2011) hypothesise that, even if the viewing condition of an object changes, the same capsules stay active, representing the same parts, but the instantiation parameters change. This property is called *equivariance*.

This viewpoint-equivariance can be used to build relationships between capsules of different layers to solve the problem of assigning parts to wholes. Sabour et al. (2017) introduced an algorithm called *routing-by-agreement* that creates this relationship iteratively. The output of a capsule should be routed only to the (one) appropriate parent and therefore an active capsule can be interpreted as a node in a parse tree.

We will define two metrics which measure how closely the routing between capsules matches the structure of a parse tree and how dynamic this parse tree is, i.e. to what extend different inputs lead to a different structure of the parse tree.

We will also introduce a new routing algorithm that prefers the selection of a single parent over multiple parent capsules. Therefore, the capsule network represents a parse tree structure over a fully connected capsule network. This modification also enables the successful training of deeper capsule networks and is more robust to white box adversarial attack.

In the next chapter we will summarize existing work. Different metrics to evaluate the parse tree structure are described in section 3. A new routing algorithm called *dynamic deep routing* is introduced in section 4. In section 5 we describe and discuss the results of our experiments.

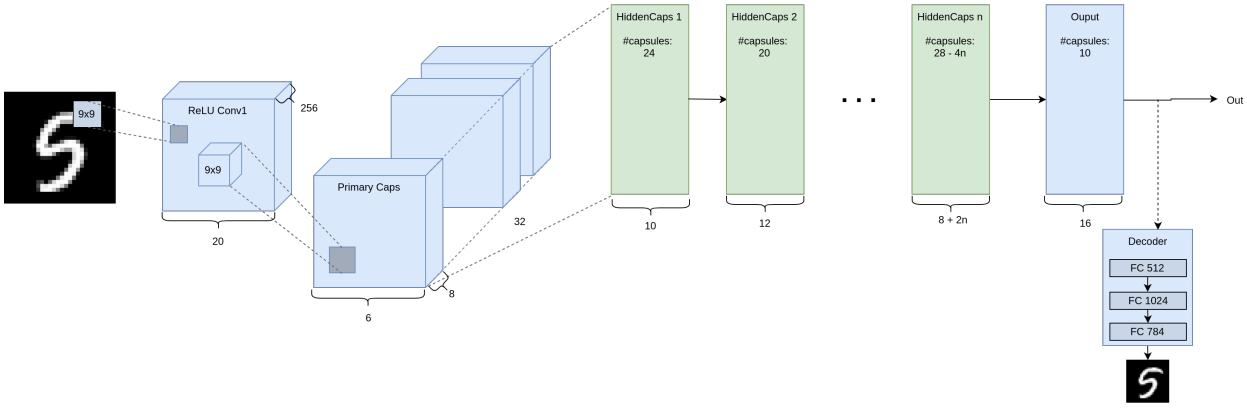


Figure 1: Capsule network(s) that are used for the experiments in this paper. The blue layers represent the original CapsNet architecture, the green layers are those that we added for the experiments. An architecture with 0 hidden layers is exactly the same as the architecture proposed by Sabour et al. (2017). The number of capsules in the hidden layers decreases by 4 with each additional layer. The dimension of the activity vectors increases as suggested by Sabour et al. (2017). A decoder network is added at the output layer to reconstruct the input image, acting as a regularizer.

2 Related Work

Capsules and the idea of equivariant instantiation parameters were introduced by Hinton et al. (2011), where the authors also showed how such a capsule can be trained by backpropagating the difference between the actual and the target outputs.

Sabour et al. (2017) introduced an architecture named CapsNet containing two layers of such capsules as shown in Figure 1 (blue). They also presented an iterative routing algorithm called *routing-by-agreement* to route instantiation parameters between different capsule layers as shown in algorithm 1. The coupling coefficient c_{ij} represents a probability to couple from lower level capsules i to upper level capsules j and is calculated using the dot product between the predicted vector $\hat{u}_{j|i}$ and the current activation v_j of the capsule. The authors hypothesize that this ensures that a lower level capsule couples with the most appropriate parent capsule of the next layer such that a parse-tree structure is established as the routing proceeds.

A different routing algorithm was developed by Hinton et al. (2018), which is based on the Expectation-Maximization algorithm. One reason the original routing-by-agreement algorithm is still popular might be its ease of implementation in common deep learning frameworks. In addition, the algorithm is easily adapted to new tasks.

Some examples include the works of Mobiny and Nguyen (2018) for lung cancer screening, Duarte et al. (2018) for detecting action in movies or Rawlinson et al. (2018) for unsupervised learning. This latter work

Algorithm 1 Routing-by-agreement algorithm as presented by Sabour et al. (2017).
$\forall$ capsules i in layer l and j in layer $l+1$ with r routing iterations and predictions $\hat{u}_{j|i}$

procedure ROUTINGBYAGREEMENT($\hat{u}_{j|i}, r, l$)
  $\forall b_{ij}, b_{ij} \leftarrow 0$
  for r iterations **do**
    $c_{ij} \leftarrow \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}$
    $s_j \leftarrow \sum_i c_{ij} \hat{u}_{j|i}$
    $v_j \leftarrow \frac{||s_j||^2}{1+||s_j||^2} \frac{s_j}{||s_j||}$
    $b_{ij} \leftarrow b_{ij} + v_j \cdot \hat{u}_{j|i}$
  end for
end procedure

provided an interesting insight about the routing-by-agreement algorithm, namely that a missing mask *"[...] causes a loss of equivariances and other desirable capsule qualities."*. The authors just assumed that routing-by-agreement produces a parse-tree and therefore only used the largest coupling coefficient from a lower level capsule to an upper level capsule by masking out all capsules other than the parent. The validity of this assumption will be evaluated in this paper.

Ancheng et al. (2018) evaluated the accuracy of a capsule network using a different number of routing iterations and found that the entropy of the coupling coefficient decreases as the training progresses. The authors also found out that routing is mostly random at the beginning of training, and suggest that a capsules should not be too confident about its parent capsule in the early training

stage, or it might lead to bad performance of the network. They called this issue *early over routing*. Ancheng et al. (2018) also show that the entropy of the coupling coefficient is still large after 7 routing iterations, indicating that a capsule sends its activity vector to all capsules in the following layer and not only to one parent capsule.

3 Metrics for dynamic parse-tree evaluation

In this chapter we describe the properties that a routing algorithm should have if the goal is to produce a parse tree structure and define multiple metrics to measure those properties. A dynamic routing algorithm for capsule networks should have the following properties:

1. Capsules in the network should represent a node in a parse tree, i.e. a capsule should only be coupled to one parent, but one parent can be coupled to multiple children.

2. The coupling of lower level capsules with parent capsules should be dynamic, i.e. the parse tree should change with the input.

We can measure these properties with the following metrics:

1. To evaluate the existence of a parse tree structure, the solution is to analyze the coupling coefficient c_{ij}. For all lower level capsules there should exist only one preferred capsule, the parent node of the parse tree. The coupling to all other capsules in the layer above should be small. Therefore, a parse tree is created whenever the max coupling c_{ij} from a lower level capsule i to a upper level capsule j is large. We call this metric *average maximum* c_{ij} and it can be calculated with Equation 1.

   For I capsules in layer L, J capsules in layer $L+1$ and M training examples:

$$\frac{1}{MI}\sum_{m=1}^{M}\sum_{i=1}^{I}\max_{j=1}^{J} c_{ij}^{m} \tag{1}$$

   where i.e. c_{ij}^{m} is the coupling in example m from the lower level capsule i to the upper level capsule j and $\max_{j=1}^{J}$ is the maximum of all values with j between 1 and J.

2. The second property can be measured with the maximum standard deviation of each lower level capsule for one mini-batch. A large value indicates that a capsule changes the parent capsule whenever the input changes, whereas a low value indicates that a capsule sends its activity vector always to the same parent. We call this metric *maximum batch standard deviation* and it can be calculated with Equation 2.

   For I capsules in layer L, J capsules in layer $L+1$ and M training examples:

$$\frac{1}{I}\sum_{i=1}^{I}\max_{j=1}^{J}\left(\sqrt{\frac{1}{M}\sum_{m}^{M}\left(c_{ij}^{m}-\underset{m=1}{\overset{M}{\text{mean}}}(c_{ij}^{m})\right)^{2}}\right) \tag{2}$$

   where $\max_{j=1}^{J}$ is the maximum of all values with j between 1 and J and $\text{mean}_{m=1}^{M}$ is the mean of all values with m between 1 and M.

Table 1: Example that shows how to calculate the metrics for dynamic tree evaluation

Inputs	c_{00}	c_{01}	c_{02}	c_{03}	Max. c_{ij}
0	0.1	0.1	0.1	0.7	0.7
1	0.1	0.7	0.1	0.1	0.7
2	0.1	0.1	0.1	0.7	0.7
3	0.1	0.7	0.1	0.1	0.7
4	0.1	0.1	0.1	0.7	0.7
Batch std.	0.0	0.3	0.0	0.3	

Table 1 shows an example of the coupling from one lower level capsule to 4 parent capsules for 5 different input samples. The maximum c_{ij} is calculated for each row separately. The average maximum c_{ij} of all rows measures if the capsule couples with only one parent capsule (large value) or with multiple capsules (small value) for all input samples. The batch standard deviation measures how the parse tree changes between different input samples and is calculated for each column. The maximum batch standard deviation is then used to evaluate if the parse tree is dynamic (large value) or static (small value).

4 Dynamic deep routing

We will now present a novel routing algorithm that satisfies the aforementioned properties of a good routing algorithm. We call it the *dynamic deep routing* algorithm.

The routing-by-agreement algorithm calculates the log prior probability with $b_{ij} = b_{ij} + cos(\alpha) * ||v_j|| * ||\hat{u}_{j|i}||$ where α is the angle between both vectors. The activity vector $||v_j||$ is the nonlinear sum of all lower level predictions weighted by the coupling coefficient c_{ij}.

The drawback of this method is that whenever the prediction vector $\hat{u}_{j|i}$ is large, the calculated b_{ij} also becomes large and the angle between both vectors has less influence on the coupling. This means that b_{ij} can be increased by simply increasing the values in the weight ma-

Algorithm 2 Dynamic deep routing algorithm.
$\forall$ capsules i in layer l with I capsules and j in layer $l+1$ with J capsules, r routing iterations and predictions $\hat{u}_{j|i}$

1: **procedure** DYNAMICDEEPROUTING($\hat{u}_{j|i}, r, l$)
2: $\quad\forall b_{ij}, b_{ij} \leftarrow 0$
3: $\quad$**for** r iterations **do**
4: $\quad\quad c_{ij} \leftarrow \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}$
5: $\quad\quad P_j \leftarrow \{c_{ij} | c_{ij} \geq \frac{1}{J}\}$
6: $\quad\quad s_j \leftarrow \sum_{p_{ij}}^{P_j} \frac{p_{ij}\hat{u}_{j|i}}{|P|}$
7: $\quad\quad v_j \leftarrow \frac{||s_j||^2}{1+||s_j||^2} \frac{s_j}{||s_j||}$
8: $\quad\quad t \leftarrow \frac{\log(0.9(J-1)) - \log(1-0.9)}{-0.5 * \text{mean}(||\hat{u}_{j|i} - v_j||)}$
9: $\quad\quad b_{ij} \leftarrow ||\hat{u}_{j|i} - v_j|| t$
10: $\quad$**end for**
11: **end procedure**

trix W_{ij} without actually decreasing the angle between the parent and child capsule. So the activity vector $||v_j||$ not necessarily represents a vector that is motivated by the largest agreement of lower level capsules. We assume that therefore the coupling from lower level capsules to parent capsules become only slightly stronger and the overall coupling stays close to uniform.

This can be resolved by replacing the dot product between the activity vector and the prediction by the negative euclidean distance such that a large weight matrix W_{ij} does not automatically produce a large agreement. This is shown in line 9 of algorithm 2.

For this solution to take place, the algorithm must be changed accordingly as follows. The length of the activity vector v_j should still represent the probability that an entity is present and therefore we use the same non-linear function shown in line 7 that limits the length of the vector v_j to 1. Therefore all activity vectors are contained within an n-dimensional hypersphere of radius 1, where the maximum possible distance between the prediction and the output vector is 2 whenever $\hat{u}_{j|i} = -v_j$ under the assumption that the predictions are somewhere inside the hypersphere of radius 1. So the maximum possible coupling coefficient for the parent capsule will be reached whenever $||\hat{u}_{j|i} - v_j|| = 0$ for the parent capsule and $||\hat{u}_{j|i} - v_j|| = 2$ for all other capsules. The maximum coupling coefficient that is possible for the parent capsule with i.e. 10 upper level capsules is therefore 0.45. This maximum coupling coefficient gets worse as the number of upper level layer capsules increases. For 128 capsules, the maximum possible value for c_{ij} for the parent capsule decreases to 0.05.

Therefore we multiply the euclidean distance by a scale factor t to allow larger coupling coefficients for the parent capsule. This factor is calculated such that the parent capsule couples with probability c_{ip} whenever the euclidean distance of the parent prediction is d_p and the distance to all other capsules is d_o where $d_p < d_o$. The coupling coefficient is calculated using the *softmax* function and therefore c_{ip} satisfies

$$c_{ip} = \frac{\exp(d_p t)}{\sum_0^{J-1} \exp(d_o t) + \exp(d_p t)} \tag{3}$$

where J is the number of parent capsules. By rewriting Equation 3 we can calculate the scale factor t with

$$t = \frac{\log(c_{ip}(J-1)) - \log(1-c_{ip})}{d_p - d_o} \tag{4}$$

Empirically we found out that setting $c_{ip} = 0.9$ whenever $d_p = \frac{d_o}{2}$ where $d_o \approx \text{mean}(||\hat{u}_{j|i} - v_j||)$ produces a strong coupling to the parent capsule but still allows a uniform coupling at the beginning of the training, which is important to avoid early over routing.

Due to the fact that we now use point predictions for agreements rather than angles between vectors we also replace the calculation of the activity vector. Instead of using the sum of all weighted predictions for one upper level capsule we calculate the average of a subset P_j of weighted lower level predictions. The calculation of the activity vector is shown in line 6 and in line 7.

The subset P_j that is used to calculate the activity vector contains only predictions that agree with each other such that wrong predictions can not influence the activity vector negatively. More precisely all prediction vectors $\hat{u}_{j|i}$ which have a coupling $c_{ij} \geq \frac{1}{J}$ are added to P_j. So in the first iteration the activity vector for one upper level capsule j is the average of all predictions because $c_{ij} = \frac{1}{J}$. After the first iteration the coupling from lower level capsules to upper level capsules with large distances will be decreased and increased with smaller distances. Therefore c_{ij} becomes larger when predictions agree with other lower level capsule predictions and c_{ij} decreases if it disagrees with other predictions. So in the next iteration the set P_j contains only prediction that agree with each other and therefore the activation vector v_j is calculated without the influence of wrong predictions.

The combination of the euclidean distance to calculate the agreement and the clustering to calculate the activity vector ensures that the distance from the activation vector to parent predictions become smaller as the routing proceeds. Therefore the coupling to parent capsules become larger and ensure a parse tree structure. Also the activity vector changes between different training examples and so the routing algorithm is dynamic too.

5 Experimental

Using the three defined metrics we will first evaluate both routing algorithms on the original CapsNet model. We

Table 2: Comparison of the metrics defined in section 3 for CapsNet using routing-by-agreement and dynamic deep routing.

Algorithm	Avg. Max. c_{ij}	Max. Batch Std.
RBA	0.11	0.008
D2R	0.73	0.23

will then evaluate those metrics on a model with two hidden layers. Afterwards we show that deeper models can be trained on MNIST when the dynamic deep routing is used. At the end of the chapter we evaluate the robustness against adversarial attacks.

5.1 Architecture

Different architectures are used to execute the experiments. The architectures differ in the number of hidden capsule layers that are used (we do not count the primary caps layer and the digit caps layer as hidden layers). So a network with two hidden layers consists of four capsule layers. A network with 0 hidden capsule layers is exactly the same as the CapsNet introduced by Sabour et al. (2017) and contains two capsule layers. An overview is shown in Figure 1.

5.2 Setup

We used the original implementation of CapsNet provided by Sabour et al. (2017) which is available at GitHub[1] for all experiments in this paper. They ran with TensorFlow 1.12.0 from Abadi et al. (2015) on a workstation with a single *Nvidia-Titan XP* GPU. Unless stated explicitly we used the hyperparameters from the version provided in the repository: A batch size of 128, the Adam optimizer with a decay rate of 0.96, a learning rate of 0.001 and 3 routing iterations.

5.3 Experiment 1 - Comparison routing-by-agreement and dynamic deep routing

In this experiment we compare the metrics we defined in section 3 between the routing-by-agreement algorithm and the dynamic deep routing algorithm.

First we used the 0 hidden layer model that was trained by Sabour et al. (2017) on the MNIST dataset. The same network architecture was trained for 50k iterations with the dynamic deep routing algorithm.

[1]Sabour S. (2018, September 11). *Original GitHub Repository of CapsNet TensorFlow implementation.* From `https://github.com/Sarasra/models/tree/984fbc754943c849c55a57923f4223099a1ff88c`

Table 3: Comparison of the metrics defined in section 3 for a two hidden layer network using routing-by-agreement and dynamic deep routing.

Algorithm	Avg. Max. c_{ij}	Max. Batch Std.
RBA-3	0.19	0.10
RBA-5	0.32	0.17
RBA-7	0.42	0.23
RBA-9	0.50	0.27
RBA-11	0.53	0.26
D2R	0.63	0.31

Considering that the uniform coupling for 10 output capsules is 0.1, we can see in Table 2 that the tested network with routing-by-agreement is much closer to a uniform routing than to the actually desired routing to a single parent. Also, the batch std. of c_{ij} is very small, which shows that a capsule couples always with the same parent capsules or with all capsules in the case of uniform coupling.

In contrast dynamic deep routing has a strong coupling to a single parent of 0.73 and therefore produces a parse tree. A standard deviation of c_{ij} between the training examples of 0.23 indicates that the coupling is much more dynamic than the routing-by-agreement.

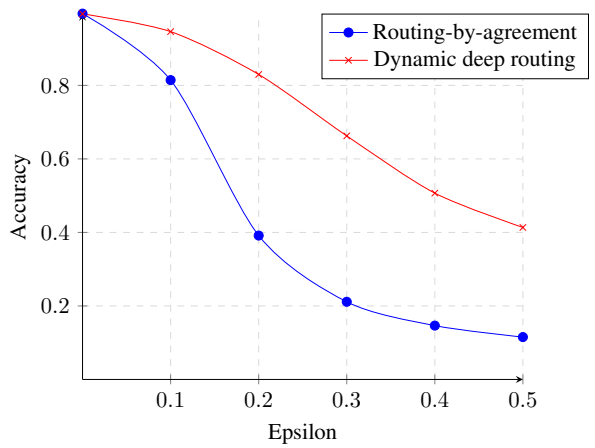
We next compare the routing-by-agreement and the dynamic deep routing algorithm for a deeper capsule network. We used a network with two hidden capsule layers and tested the routing-by-agreement algorithm with different numbers of routing iterations and the dynamic deep routing algorithm with 3 iterations. We consider the metrics after 5k training iterations. Since the behavior between all capsule layers is almost the same, we calculate the metrics between the first and second hidden capsule layer.

In Table 3 the metrics of section 3 measured for the routing-by-agreement algorithm with different routing iterations (i.e. RBA-3 for 3 iterations, RBA-5 for 5 iterations and so on) and the dynamic deep routing for 3 routing iterations are shown. We want to emphasize the following:

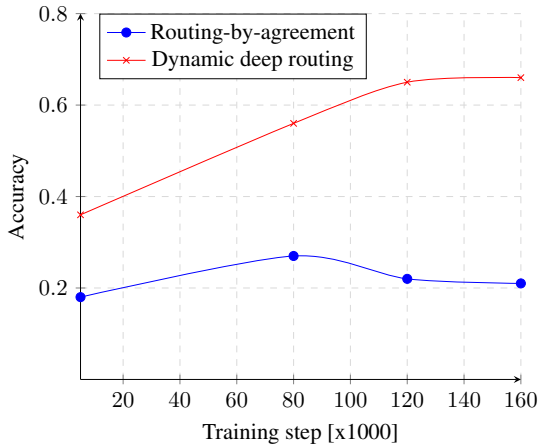
1. The routing-by-agreement with three iterations produces a coupling distribution that is close to uniform. Therefore the average maximum c_{ij} is small.

2. As the number of routing iterations increases using the routing-by-agreement, the average maximum c_{ij} and the coupling to a single parent becomes stronger. This was already reported by Ancheng et al. (2018). However, the average maximum c_{ij} is still relatively small even for 11 routing iterations.

3. We can see that dynamic deep routing has the largest



(a) Accuracy for different values of ϵ after an FGSM attack is applied on the MNIST test set for a two hidden layer capsule network.



(b) Accuracy for different training steps after an FGSM attack with $\epsilon = 0.3$ is applied on the MNIST test set for a two hidden layer capsule network.

coupling of 0.63 with only 3 routing iterations. This indicates that capsules form a parse tree. A maximum c_{ij} batch standard deviation of 0.31 shows that the produced parse tree is also very dynamic (i.e. the structure of the parse tree changes as the input changes).

5.4 Experiment 2 - Training deeper capsule networks

In the previous section we have seen that the dynamic deep routing algorithm enforces the properties we have formulated for a good routing algorithm. Rawlinson et al. (2018) argue that supervised capsule networks trained with the routing-by-agreement algorithm can not be very deep. To test whether this changes by using the dynamic deep routing algorithm, we created a network with four hidden layers and trained it either using routing-by-agreement or dynamic deep routing. We varied the learning rate between 0.001 and 0.00001 and evaluated the test accuracy after 5k iterations on MNIST.

The train and test accuracy never increased above 0.10 for all learning rates using routing-by-agreement with 3 routing iterations. With the default learning rate of 0.001, the test accuracy increased to 0.87 when we used the dynamic deep routing algorithm. We can confirm the hypothesis by Rawlinson et al. (2018) that deep capsule networks can not be trained with routing-by-agreement. In addition we conclude that the adaptations that lead to the dynamic deep routing algorithm also allow for the training of deeper models compared to the original routing algorithm.

5.5 Experiment 3 - Robustness to adversarial attacks

In this experiment we compare the adversarial robustness of the routing-by-agreement algorithm and the dynamic deep routing algorithm for a 2 hidden layer capsule network on MNIST. We trained the model for 160k steps and compared the accuracy of both networks after we applied the *fast sign gradient method* (FGSM) from Goodfellow et al. (2015) to the test set. This method changes the pixel intensity by ϵ into the direction of the sign of the gradient such that the loss is increased. The results for different values of ϵ are shown in figure 2(a) for the routing-by-agreement and the dynamic deep routing algorithm. The network that uses dynamic deep routing is less vulnerable than a model that uses the routing-by-agreement algorithm. For example with $\epsilon = 0.3$ the accuracy of the network that uses the dynamic deep routing algorithm is 45% larger.

In figure 2(b) the robustness of the network at different training steps is shown for $\epsilon = 0.3$. The robustness against adversarial attacks becomes larger as the training proceeds for both routing algorithms. One difference is that the routing-by-agreement already saturates after 80k steps and slightly decreases afterwards whereas the robustness of the dynamic deep routing saturates after about 120k steps at a much higher accuracy.

6 Conclusion

In this paper we have shown that the coupling coefficients calculated by the routing-by-agreement algorithm are close to a uniform coupling. Therefore we concluded

that capsules of such a network do not necessarily represent a parse tree structure.

We presented different properties that a routing algorithm should have to form a dynamic parse tree. We defined metrics that can be used to measure these properties and created a novel routing algorithm that improves these metrics compared to the routing-by-agreement algorithm.

We showed experimentally that the dynamic deep routing algorithm (in comparison to the routing-by-agreement algorithm) creates a dynamic parse-tree and that this property allows for the training of deeper capsule networks. We also showed that the algorithm presented here increases the robustness of the network against adversarial examples.

References

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

Lin Ancheng, Li Jun, and Ma Zhenyuan. On learning and learned representation with dynamic routing in capsule networks. abs/1805.04041, 2018.

Kevin Duarte, Yogesh Singh Rawat, and Mubarak Shah. Videocapsulenet: A simplified network for action detection. abs/1805.08162, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples., 2015.

Geoffrey Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. 2018.

Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming auto-encoders. In *ICANN*, 2011.

Aryan Mobiny and Hien Van Nguyen. Fast capsnet for lung cancer screening. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-Lpez, and Gabor Fichtinger, editors, *MICCAI (2)*, volume 11071 of *Lecture Notes in Computer Science*, pages 741–749. Springer, 2018. ISBN 978-3-030-00934-2.

David Rawlinson, Abdelrahman Ahmed, and Gideon Kowadlo. Sparse unsupervised capsules generalize better. abs/1804.06094, 2018.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NIPS*, pages 3859–3869, 2017.