Comparison of Data Efficiency in Dynamic Routing for Capsule Networks

Kenny Schlegel [1] Peer Neubert [1] Peter Protzel [1]

Abstract

Capsule Networks are an alternative to the conventional CNN structure for object recognition. They replace max pooling with a dynamic routing of capsule activation. The goal is to better exploit the spatial relationships of the learned features, not only to increase recognition performance, but also improve generalization capability and sample-efficiency. Recently, two algorithms for dynamic routing of capsules have been proposed. Although they received a lot of interest and they are from the same group, an experimental comparison of both is still missing. In this work we compare these two routing algorithms and provide experimental results on data efficiency and generalization to increased input images. Although the experiments are limited to variants of the MNIST dataset, they indicate that the approach of Sabour et al. (2017) is better at learning from few training samples and the EM routing of Hinton et al. (2018) is better at generalizing to changed image sizes.

1. Indroduction

Convolutional neuronal networks (CNN) showed very good results in the fields of object recognition and classification. For instance, in 2012, the well-known *AlexNet* classification network (Krizhevsky et al., 2012) was able to distinguish 1,000 classes of the *ImageNet* dataset (Jia Deng et al., 2009). Due to their convolutional character, CNNs can reuse learned weights across the whole image. This increases sample efficiency and creates a translation equivariance of the responses on a single layer. However, the combination of these responses on higher CNN layers doesn't retain this property: changing the relative positions of parts of objects might result in completely different representations at the higher layers. This highly increases the required amount of training samples. Typically, CNNs address this issue by using max-pooling after convolutional layers to obtain invariance towards small spatial distortions of the input. Unfortunately, this invariance property of max pooling causes a loss of spacial information about the recognized parts. Furthermore, non maximal responses are suppressed without incorporation of feedback from higher layers. To address both issues, Hinton et al. (2011) proposed Transforming Autoencoders to explicitly estimate the relative transformation of object parts instead of ignoring this information using pooling. The extension of this inverse-graphics idea of estimating parts and their mutual relation to a deep-learned hierarchy is called a Capsule Network. Implementing this novel approach poses the question how information of parts is combined to form the next layer in the hierarchy. In Capsule Networks, this is called dynamic routing. Recently, Sabour et al. (2017) and Hinton et al. (2018) proposed two approaches to address this task.

This paper compares these two approaches. In particular, we evaluate their data efficiency in comparison to two baseline CNNs using two tasks: (1) Classification with increasing amount of training data and (2) generalization to larger images based on training on small patches. In the following section 2 we start with a short theoretical comparison of both routing approaches and present experimental results in section 3.

2. Routing Procedures

This section describes the Capsule Network approaches from Sabour et al. (2017) and Hinton et al. (2018). A summary of the comparison can be found in the table 1. For both, we describe two groups of properties:

- **Capsules** sections describe the representation, interpretation of instantiation parameters, and computation of the activation of capsules.
- **Routing** sections provide properties of the routing procedures: computation of agreement, transformation, and regularization.

2.1. Dynamic Routing I

The first dynamic routing procedure by Sabour et al. (2017) demonstrates the idea of routing by agreement. It contains groups of neurons, named Capsules after (Hinton et al., 2011), and the routing algorithm is based on cosine-similarity of vectors.

[1]Dept. of Electrical Engineering and Information Technology TU Chemnitz, Germany. Correspondence to: Kenny Schlegel <scken@hrz.tu-chemnitz.de>.

Capsules represent a visual entity in an image with particular properties, encoded by a set of instantiation parameters. The interpretation of these parameter vectors by Sabour et al. (2017) is quite different from Hinton et al. (2011), since the latter contains an explicit definition of pose parameters in the transforming auto-encoder. In contrast, the instantiation parameters in the first routing variant by Sabour et al. (2017) are rather abstract and, for example, lack the interpretability of the position due to the application of non-linear *squashing*. The propagated activation of a low-level capsule to a higher capsule is define by computing the length of the representation vector in combination with the squashing function. Every capsule has an explicit receptive field to recognize a visual entity.

Routing by agreement is implemented by iteratively updating couple-coefficients between low- and high-level capsules. The couple-coefficients are based on the cosine similarity of a transformed low-level capsule and its corresponding higher-level capsule output. If there is a high similarity, the couple-coefficient is increasing iteratively in the routing algorithm. This feedback loop is similar to the explaining away in (George et al., 2017). All capsules in a layer receive predictions of all capsules in the layer below and send feedback to them. Routing with softmax is applied to select the most likely object hypothesis. For network regularization, Sabour et al. (2017) combine the routing with an image reconstruction during training.

2.2. Dynamic Routing II

The second dynamic routing approach by Hinton et al. (2018) is based on a maximum-likelihood estimator, the Expectation Maximization algorithm.

Capsules represent visual entities similar to the first routing (Sabour et al., 2017) but provide more explicit interpretations of the parameters. A capsule contains an activation unit and a 4×4 pose matrix which is intended to contain interpretable pose parameters. However, Hinton et al. (2018) do not explicitly ensure the pose meaning of the matrix entries during training. Only a coordinate addition of the shift parameters in the last capsule layer enables the interpretation of the position parameters.

Routing by agreement is implemented with a expectation-maximization clustering. It iteratively fits a Gaussian distribution to the votes from lower capsules to the higher layers. The couple-coefficients to higher capsules are computed from the probabilities of capsules within the estimated clusters and their activations. A major difference to routing algorithm I is the receptive field of transformation and routing. In (Hinton et al., 2018) transformations are like convolutions: votes and also routing are computed within a specific receptive field. This significantly reduces the number of parameters compared to routing procedure I. In contrast to the first dynamic routing variant, no image reconstruction regulation is applied at the end of capsule network.

Table 1. Comparison of dynamic routing approaches

	dynamic routing I (Sabour et al., 2017)	**dynamic routing II (Hinton et al., 2018)**
Capsules		
representation	vector	matrix
interpretation of instantation parameters	abstract, not defined	pose
activation	squashing length of vectors	activation unit
Routing		
transformation and routing range	each lower to each higher capsule	only within receptive fields
kind of agreement	cosine-similarity	Gaussian probability
reconstruction regulation	yes	no

3. Experiments

The following experiments demonstrate the data efficiency of networks and generalization of a learned capsule network (with routing I and II) to few variations of a given dataset.

3.1. Implementations

We implemented the two dynamic routing approaches in two networks CapsNet I and CapsNet II. The implementation of CapsNet I follows Sabour et al. (2017) and builds upon an open source third-party implementation available on GitHub[1]. The used implementation of CapsNet II also builds upon an open source third-party implementation available on GitHub[2] and follows Hinton et al. (2018) up to the schedule for the inverse temperature, which is not provided in the paper. We used $[1, 2, 3]$ in the experiments.

We also used a modified version of the latter CapsNet, which we call CapsNet II Small. Different to Hinton et al. (2018), it contains 64 filters with kernels of size 9x9 in first the convolutional layer, 16 primary capsule channels, only one convolutional capsule layer with 16 channels and stride of 2, the number of iteration is 2, and the input values are normalized to range $[0, 1]$.

We compare the CapsNets with two standard CNNs from (Hinton et al., 2018) and (Sabour et al., 2017). The first CNN I has three convolutional layers with kernel size 5, stride 1 and depths of 256, 256 and 128. The convolutional layers

[1]Implementation based on [Xifeng Guo, Github, `https://github.com/XifengGuo/CapsNet-Keras`]

[2]Implementation based on [Jonathan Hui, Github, `https://github.com/jhui/machine_learning/tree/master/capsule_em`]

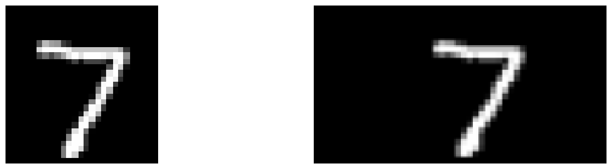

Figure 1. Illustration of L-MNIST dataset. The left image from the original MNIST test dataset is randomly placed in the bigger image on the right.

are followed by two fully connected layers with sizes 328 and 192. The second CNN II has two convolutional layers with kernel size 5, stride 1 and depths of 32 and 64. Max pooling with size of 2 is applied after each convolutional layer. Finally, there is a fully connected layer with size of 1024 at the end of the network. Both CNN networks use a softmax layer for classification of the 10 classes with a dropout of 0.5.

All network were trained using Adam optimizer and learning rate of 0.001. Cap sNet I involved a decaying rate as explained in (Sabour et al., 2017).

3.2. Datasets

In the experiments, we use variants of the MNIST dataset (LeCun et al., 1998). For the data efficiency experiments, the original MNSIT test set is used in combination with increasingly large fractions of the training set. For the evaluation of the pose generalization capabilities, we created a L-MNIST dataset, where the test images are randomly placed in images of larger size than the original MNIST images. Image width varies from 28 to 88 pixels. Background in L-MNIST images is black. An example can be seen in Fig. 1. For experiments on L-MNIST, the default training set is used, only the test images are enlarged.

3.3. Experiment 1: Increasing training size

According to the original papers (Sabour et al., 2017) and (Hinton et al., 2018), CapNet I performs better than CapsNet II on the standard MNIST classifictaion task (resulting error rates are 0.25% and 0.44%). In a first set of experiments, we investigate the dependency of the these results on the number of training samples similar to the experiments in (George et al., 2017). Therefore, we train all networks on increasingly large training sets and always evaluate on the full test dataset. The size of the training set increases from 1, 5, 10, 20, 30, 50 to 100 digits per class. All networks are trained using these sets until convergence.

The results of the efficiency test are shown in Fig. 2. From 1 to 20 training samples per class CNN II shows the best performance. Presumably, this is due to the smaller number of parameters compared to CNN I and also CapsNet I (cf. table 2). For higher training sample numbers, the CapsNet I achieves the highest accuracy, reaching 96.22 % using 100

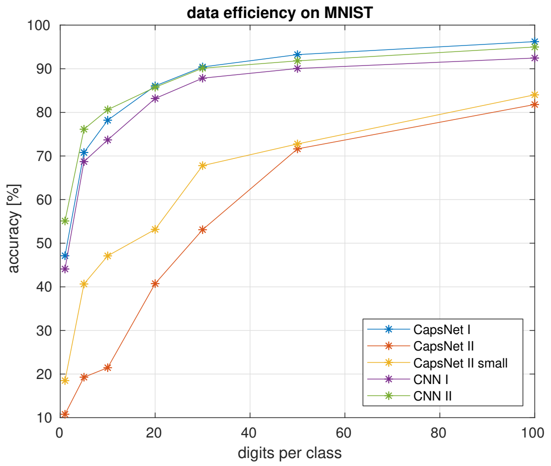

Figure 2. Results on training data efficiency of different networks.

Table 2. Number of trainable parameters of tested networks

Network	Number of parameter
CapsNet I	8,215,568
CapsNet II	318,964
CapsNet II small	62,404
CNN I	13,288,970
CNN II	1,111,946

samples per class. Although it has a much smaller number of parameters, results of CapsNet II are worse in this test. One reason could be the missing reconstruction regularization compared to CapsNet I (cf. (Sabour et al., 2017)), but tests with a similar decoder at the end of the network did not improve the results. Tests on the smaller CapsNet II Small showed somewhat improved results. The difference between the large and the small CapsNet II becomes smaller with increasing number of training samples. Presumably, the improvement is a result of the smaller number of parameters (cf. table 2). Potential differences in implementation compared to (Hinton et al., 2018) might influence the results, in particular since there are some missing details in the paper. However, based on these experiments, we can state that *for the available implementation*, learning transformation parameters between each layer in CapsNet II to get the correct spatial relationship required more training samples than the other approaches.

3.4. Experiment 2: Generalisation to larger images

The output of convolutional layers (without max pooling) is equivariant towards shift of the input. It is straightforward to train the kernel weights on a small image size and to build a convolutional layer for larger input images by using the learned kernel weights in an appropriate network. However, in practice, the convolutional layers are combined with other layers, e.g. fully connected layers, that prevent this direct approach. These fully connected layers would require additional retraining on large images and thus require additional training data.

This section evaluates the performance of the different CapsNet versions and baseline CNNs when trained on small images and inference is done on larger images - a capability that is very beneficial for sample-efficiency. The experiments use the L-MNSIST dataset from section 3.2.

CapsNet II can directly be used for this task. Due to its better results in the previous experiments, we use the CapsNet II Small. Since CapsNet I uses an explicit transformation matrix for each capsule, it has to be modified to share capsule transformation across each channel. The fully connected layers in the CNNs prevent a direct application to this task. Thus we replace them with a global maximum pooling in CNN GMP, and global average pooling in CNN GAP. The architecture of the used CNN is similar to CNN II (two convolutional layer, each followed by a max-pool layer). We increased the number of filters to 128 and 256 and inserted the global pooling layers before the fully connected layer to achieve independence from the image size. All networks are trained on the full default MNIST dataset.

Fig. 3 shows the results. The performance of CapsNet I with shared parameters rapidly decreases with increasing image size. A possible reason is the combination of softmax in the routing process and the lack of a background class. Too many background pixel create noise due to biases, the noise is scaled by the softmax and accumulated in the subsequent capsules. Presumably, an extra background class could be used to address this problem. The performance of the CNN with global average pooling slightly drops and goes down to 70 % for maximally increased images. The average pooling can compensate the larger images, but presumably, it is influenced by noise from the background similar to CapsNet I. In contrast, the CNN with global max pooling retains the high accuracy as expected. The absolute performance of the CapsNet II is somewhat worse, but it is only slightly affected by the changed image size as well, and retains more than 90 % accuracy even for the largest images.

The CNN with global max pooling showed the best performance in this task. However, keeping only the single global maximum value resembles similar problems to the local max pooling after convolutional layers: all information about spatial relation lost. An example experiment is illustrated in Fig. 4. There are two images, the left showing a correct configuration of the parts of the number seven and the right image showing a wrong configuration of the same parts. In this experiment, the CNN with global maximum pooling correctly predicts the class “seven” for the correct configuration input with 100% confidence. For the wrong configuration it also outputs class “seven” with 99% confidence, although the parts in the image are not configured to show a seven. In contrast, the CapsNet II Small classifies the correct seven with 97.0% confidence to be of class “seven” and the wrong configuration is classified with 94.1% as class “one” - a much more reasonable result. This experiment resembles the main motivation for capsules by Hinton - Pooling looses spatial information: *”This method [...] is clearly incapable of dealing with recognition tasks, [...] that require knowledge of the precise spatial relationships between high-level parts ...”* (Hinton et al., 2011)

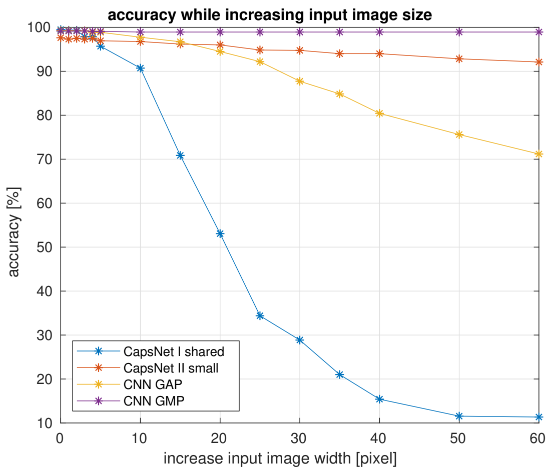

Figure 3. Results on generalization to enlarged images.

Figure 4. Illustration of a pitfall for global max pooling. The correct configuration of the parts of the number seven on the left image creates almost the same classification results as the wrong configuration in the right image. There, CapsNet II Small creates the more reasonable result “one” and returns lower confidence.

4. Conclusion

In the presented work we have examined two hypotheses: First, a capsule network with dynamic routing is able to learn with few training examples and second it can be applied to different input sizes during inference. The experiments on data efficiency have shown good results for the first routing process by Sabour et al. (2017), but no improvements compared to the compared standard CNNs. Despite its much smaller number of parameters, the second routing variant by Hinton et al. (2018) required more training samples than the other approaches. This could be due to the complex model structure of the EM routing or missing implementation details from the original paper. However, this capsule network variant showed promising results on the generalization to larger images during inference. Additionally, it retains the spatial relation of the learned objects. Training on small images and doing inference on arbitrarily sized images increases data efficiency. Future work includes the evaluation of these properties on more complex datasets. Particularly interesting is deeper investigation how the retained spatial relationship can be used to increase sample efficiency.

References

George, Dileep, Lehrach, Wolfgang, Kansky, Ken, Lázaro-Gredilla, Miguel, Laan, Christopher, Marthi, Bhaskara, Lou, Xinghua, Meng, Zhaoshi, Liu, Yi, Wang, Huayan, Lavin, Alex, and Phoenix, D Scott. A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science*, 358(6368), 2017. ISSN 10959203. doi: 10.1126/science.aag2612.

Hinton, Geoffrey, Sabour, Sara, and Frosst, Nicholas. Matrix capsules with EM routing. *International Conference on Learning Representations (ICLR)*, 2018.

Hinton, Geoffrey E, Krizhevsky, Alex, and Wang, Sida D. Transforming Auto-encoders. In *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'11, pp. 44–51, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21734-0.

Jia Deng, Wei Dong, Socher, Richard, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848.

Krizhevsky, Alex, Sutskever, Ilya, and Geoffrey E., Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pp. 1–9, 2012. ISSN 10495258. doi: 10.1109/5.726791.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998. ISSN 00189219. doi: 10.1109/5.726791.

Sabour, Sara, Frosst, Nicholas, and Hinton, Geoffrey E. Dynamic routing between capsules. In *Neural Information Processing Systems (NIPS)*, pp. 3859–3869, 2017.