Learning Vector Quantization Capsules

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

A capsule is a grouping of neurons with the goal to perform a quite complicated computation on tensors and the passing of a highly informative tensor to the next layer. Several capsules form a capsule layer. We present a capsule architecture based on Learning Vector Quantization (LVQ) called LVQ-Capsule. Each LVQ-Capsule corresponds to a certain class and is equipped with prototypes and a respective dissimilarity measure. Our capsule architecture receives two inputs from each capsule in the parent layer: the feature tensors and the dissimilarity value for each feature tensor. A LVQ-Capsule measures the dissimilarity between the prototypes and the given tensors and updates the dissimilarity for each input tensor. Via a dissimilarity routing process without an iteration, we estimate the expected input tensor and the expected dissimilarity value as output quantities of the capsule. If the expected dissimilarity value is small, the feature tensors are potentially close to the prototypes. Thus, a LVQ-Capsule gives a potentially high vote to the output if all the input tensors are similar to the prototypes. In experiments on MNIST, affNIST, CIFAR-10 and smallNORB we show exemplarily the application of LVQ-Capsules and that their performance is competitive to other capsule architectures. Furthermore, we show that the use of dissimilarities and prototypes provide a new framework to visualize what is learned by a capsule.

1 Introduction

Averaging repeated experiments and measurements is a usual concept to estimate the true value quantity. For example, gene expression analysis usually is done applying the same or slightly varying conditions for genes. These changing conditions can also be seen as variations of the measurement. Frequently, a subsequent averaging of the response signals takes place before evaluation. Alternatively, more sophisticated data analysis methods can be applied like clustering, principal component analysis or advanced outlier detection based on expert knowledge, to name just a few. But the question that still always arises is: *How can parts be assigned to a whole or, moreover, parts to wholes? (*assignment problem*)* [1].

In [2] HINTON ET AL. introduced the capsule concept. The idea is to split a layer into small groups of neurons, denoted as capsules, and thus forming a capsule layer. Thereby, the connection between two capsules in different layers is realized as a tensor[1] pipe. Considering a tensor flow between the processing units of a network (the capsules), one can easily think about more complex layer types, which allow a broader range of mathematical methods like numerical analysis, statistics, etc. Equal to the neurons in basic neural networks, the capsules have to follow the rule: *several in – one out*. Thus, the capsule processes the several input tensors to just one output tensor. This internal processing is equivalent to modeling the assignment problem.

[1]Like usual in neural networks, we denote a multidimensional object as a tensor. Thus, a tensor can be a vector, matrix, etc., or even a scalar.

Learning Vector Quantization (LVQ) is a supervised classification method based on prototype vectors distributed in the data space and a dissimilarity measure between data points and prototypes, whereas vector quantization[2] (VQ) refers to the unsupervised counterpart [3]. The transformation of Hebbian maximum excitation for perceptrons to the minimum distance principle, as it was proposed by KOHONEN already in [4], ensures a maximum closeness in a natural way. For distances (metrics) as dissimilarity this implies immediately the identity of indiscernibles. Unfortunately, from our point of view, LVQ is not used for modern neural network architectures and pure LVQ frequently cannot compete with those. One reason might be that the assignment problem is not considered by LVQ-researchers so far. Yet, the respective answer is required for an efficient coupling of LVQ to well-known neural network layer concepts.

Our contribution in this work is that we bring together modern neural network architectures, LVQ and capsules defining LVQ-Capsules. Thus, we are not aiming to define a special network architecture in this paper. We rather want to see LVQ-Capsules as a new layer type which is, in accordance to the previous discussion, just a nice way to group neurons to work on tensors. Compared to previous capsule definitions, LVQ-Capsules work without an iteration for solving the assignment problem. Furthermore, the use of dissimilarities, or more specially metrics, give access to a new framework to model the probability transformation of a neural network at the final layer. Additionally, the prototypes could be useful to define new interpretation techniques.

Accompanying the paper, we publish a software package[3] called $\vec{anysma}$ (greek: vector) to provide a framework for easy capsule definitions in KERAS [5]. All architectures and results of the experiments will be available there.

2 The LVQ-Capsule architecture

2.1 Capsules - previous work

In [2], a capsule was defined as a group of neurons together with complex operations for the first time. It is stated there that "[...] artificial neural networks should use local capsules that perform some quite complicated internal computations on their inputs and then encapsulate the results of these computations into a small vector of highly informative outputs". They applied the concept to define a transforming auto-encoder.

SABOUR ET AL. [1] captured this idea and enhanced the concept. Their capsule network operates on vectors where the probability that an entity exists is coded by the length of the vector and the instantiation parameters are mapped to the vector orientation. Furthermore, an iterative routing-by-agreement process called Dynamic Routing is introduced. The goal is to solve the problem how to assign multiple input vectors to one output vector. The output vector is a squashed linear combination of the linear transformed input vectors of the capsule. The combination coefficients are computed iteratively via transforming assigned vector weights with the softmax function and updating the vector weights with the dot product (Euclidean inner product) between the current output vector and the respective linear transformed input vectors. The authors also introduced a probability transformation which can deal with multi-class label assignments. For this purpose, the authors used again the length of the parameter vector squashed by an appropriate transformation as probabilistic output (activation) of the network. Since these transformations are independent of each other, the resulting output values do not necessarily sum up to one and should be interpreted as possibility values as known from fuzzy approaches [6]. To optimize these possibilities a "margin loss" was defined to keep these possibilities as close as possible to the optimum values.

HINTON ET AL. refined the proposed capsule architecture for vectors [7]. In this approach they used matrices as parameter storages instead of vectors and tracked the activations on separate channels. In the final layer of the network the authors transformed the activations via a sigmoid function to create the possibility vector[4]. More precisely, they used 4×4 pose matrices as instantiation parameters and defined a routing procedure based on an instance of the Expectation-Maximization (EM) algorithm.

[2]If it is not confusing, we will always talk about LVQ instead of differentiating all the time between the supervised and unsupervised version.

[3]`https://github.com/AnysmaForBlindReview/anysma`; anonymised for double blind review process

[4]A possibility vector or unnormalized probability vector is a vector of probability values, which do not sum up to one. [8]

The probability model of the EM-algorithm is a Gaussian model. The authors denote this iterative procedure as EM-routing and state: “The non-linearity implemented by a whole capsule layer is a form of *cluster finding* using the EM-algorithm [...]”. Furthermore, instead of using margin loss they defined a loss function called “spread loss”.

Beside the three fundamental contributions above, many papers were published since SABOUR ET AL. published the CapsNet architecture in 2017. Most of these contributions deal with a wider range of applications and just few present some slight modifications or analysis of the capsule structure, e.g. see [9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. But in general, their basic capsule concept is quite similar to [1] or [7].

2.2 Learning Vector Quantization

Learning vector quantization (LVQ) as proposed by T. KOHONEN is an intuitive prototype based classifier model [19]. Prototype vectors $\mathcal{W} = \{\mathbf{w}_{1,}\mathbf{w}_{2,...,}\mathbf{w}_m\}$ with $\mathbf{w}_j \in \mathbb{R}^n$ are distributed in the data space $\mathbb{R}^n$ according to a simple scheme of attraction and repulsion depending on their class assignments $c(\mathbf{w}_j) \in \mathcal{C} = \{1, 2, ..., l\}$, the class $c(\mathbf{v})$ of a given training sample $\mathbf{v} \in \mathbb{R}^n$ and the dissimilarity $d(\mathbf{v}, \mathbf{w}_j)$. The dissimilarity measure $d(\cdot, \cdot)$ is used to determine the similarity between the prototypes $\mathbf{w}_j$ and the data point $\mathbf{v}$. A data point $\mathbf{v}$ is assigned to the class $c(\mathbf{w}_{j^*})$ where

$$j^* = \arg\min \{d(\mathbf{v}, \mathbf{w}_j) \,|j \in \{1, 2, ..., m\}\}$$

is the index of the best matching prototype. This assignment realizes a winner-takes-all decision.

Frequently, the (squared) Euclidean distance is used as dissimilarity measure being a special case of the Minkowski distance

$$d_p(\mathbf{v}, \mathbf{w}) = \left(\sum_{i=1}^{n} |v_i - w_i|^p\right)^{1/p} \tag{1}$$

with $p \geq 1$. Yet, adaptive parametrized dissimilarities are promising alternatives to the Minkowski approach [20]. For example, the tangent distance (TD)

$$d_{TD}(\mathbf{v}, \mathbf{w}(\boldsymbol{\theta})) = \min_{\boldsymbol{\theta} \in \mathbb{R}^r} d_{p=2}(\mathbf{v}, \mathbf{w}(\boldsymbol{\theta})) \tag{2}$$

is used for handling variations in data and transfer learning [21, 22, 23]. Thereby, $\mathbf{w}(\boldsymbol{\theta}) \in \mathbb{R}^n$ is a vector $\mathbf{w}(\boldsymbol{\theta}) = \mathbf{w} + \mathbf{W}\boldsymbol{\theta}$ with $\boldsymbol{\theta} \in \mathbb{R}^r$ of an affine subspace approximating the unknown data manifold by the tangents $\mathbf{W} \in \mathbb{R}^{n \times r}$ [24]. Another variant is the rectangular restricted tangent distance (rTD) where the prototypes are r-dimensional orthotopes [25]. For a general, differentiable dissimilarity measure $d(\mathbf{v}, \mathbf{w})$ the prototypes are updated during the learning process regarding

$$\Delta\mathbf{w} \propto \frac{\partial d(\mathbf{v}, \mathbf{w})}{\partial \mathbf{w}}$$

which simplifies to vector shifts $\pm(\mathbf{v} - \mathbf{w})$ for the attraction and repulsion scheme in case of the Euclidean distance.

A stochastic gradient descent learning (SGDL) approach was established by SATO and YAMADA taking an approximated classification accuracy as cost function [26] realizing a margin optimizer for the hypothesis margin [27]. After training, the prototype positions reflect the class distribution in the data space. Probabilistic variants of LVQ were proposed in [28, 29]. Particularly, the Robust Soft LVQ (RSLVQ) is a LVQ model based on Gaussian mixtures to describe the data class distributions [28]. Here, the centers of the Gaussians are taken as the prototypes of the RSLVQ. The class assignments are realized as probabilistic decisions using the softmax function regarding the responsibilities of the Gaussians for a given data sample. During the learning process, the centers of the Gaussian models are adapted by SGDL.

As discussed in [30], LVQ can be seen as a special type of a multi-layer feedforward network. In addition, if the data space consists of interpretable objects like images, the prototype vector quantization principle leads to an interpretable model [31].

2.3 Learning Vector Quantization Capsules

A LVQ-Capsule layer $\mathbf{C}^h$ contains several LVQ-Capsules $\mathbf{C}^h_k$, each responsible for exactly one class $k \in \mathcal{C}$. Every capsule $\mathbf{C}^h_k$ contains a set $\mathcal{W}^h_k = \{\mathbf{w}_j | c(\mathbf{w}_j) = k\}$ of class responsible prototypes

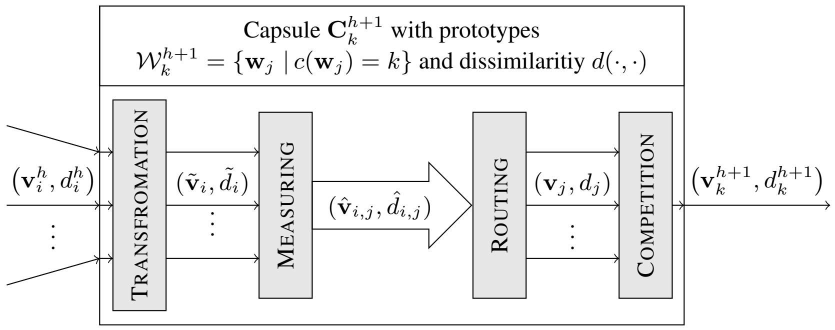
Capsule $\mathbf{C}_k^{h+1}$ with prototypes
$\mathcal{W}_k^{h+1} = \{\mathbf{w}_j \mid c(\mathbf{w}_j) = k\}$ and dissimilaritiy $d(\cdot,\cdot)$
$(\mathbf{v}_i^h, d_i^h)$
Transfromation
$(\tilde{\mathbf{v}}_i, \tilde{d}_i)$
Measuring
$(\hat{\mathbf{v}}_{i,j}, \hat{d}_{i,j})$
Routing
$(\mathbf{v}_j, d_j)$
Competition
$(\mathbf{v}_k^{h+1}, d_k^{h+1})$

Figure 1: The LVQ-Capsule architecture. To avoid confusion with indices we drop the layer index $h+1$ at the prototypes $\mathbf{w}_j$, the class k and the internal signals of the capsule and keep in mind that these definitions are layer dependent. Furthermore, we also neglect the capsule index k at the internal signals of a capsule.

Algorithm 1 LVQ-Capsule $\mathbf{C}_k^{h+1}$. The set I_k is defined as $I_k = \{j \mid \mathbf{w}_j \in \mathcal{W}_k^{h+1}\}$.

```
1: procedure LVQ-Capsule($\mathbf{v}_i^h, d_i^h, \alpha$)
2:     $\forall i : \tilde{\mathbf{v}}_i, \tilde{d}_i \leftarrow$ Transformation($\mathbf{v}_i^h, d_i^h$)
3:     for all $j \in I_k$ do                                  ▷ for each prototype $\mathbf{w}_j$ in the capsule
4:         $\forall i : \hat{\mathbf{v}}_{i,j}, \hat{d}_{i,j} \leftarrow$ Measuring($\tilde{\mathbf{v}}_i, \tilde{d}_i, \mathbf{w}_j, \alpha$)
5:         $\mathbf{v}_j, d_j \leftarrow$ Routing($\{\hat{\mathbf{v}}_{i,j} \mid \forall i\}, \{\hat{d}_{i,j} \mid \forall i\}$)
6:     $\mathbf{v}_k^{h+1}, d_k^{h+1} \leftarrow$ Competition($\{\mathbf{v}_j \mid j \in I_k\}, \{d_j \mid j \in I_k\}$)
7:     return $\mathbf{v}_k^{h+1}, d_k^{h+1}$

1: procedure Measuring($\tilde{\mathbf{v}}_i, \tilde{d}_i, \mathbf{w}_j, \alpha$)
2:     $\hat{d}_{i,j} \leftarrow (1-\alpha) \cdot \tilde{d}_i + \alpha \cdot d(\tilde{\mathbf{v}}_i, \mathbf{w}_j)$     ▷ update dissimilarities
3:     $\hat{\mathbf{v}}_{i,j} \leftarrow$ DissimilarityTransformation($\tilde{\mathbf{v}}_i, \mathbf{w}_j$)
4:     return $\hat{\mathbf{v}}_{i,j}, \hat{d}_{i,j}$

1: procedure Routing($\{\hat{\mathbf{v}}_{i,j} \mid \forall i\}, \{\hat{d}_{i,j} \mid \forall i\}$)
2:     $\forall i : p_{i,j} \leftarrow$ NegSoftmax($\{\hat{d}_{\hat{i},j} \mid \forall \hat{i}\}, i$)     ▷ routing probabilities, see Eq. (3)
3:     $d_j \leftarrow \sum_i p_{i,j} \cdot \hat{d}_{i,j}$                         ▷ dissimilarity routing
4:     $\mathbf{v}_j \leftarrow \sum_i p_{i,j} \cdot \hat{\mathbf{v}}_{i,j}$                         ▷ signal routing
5:     return $\mathbf{v}_j, d_j$
```

together with four information processing modules: transformation, measuring, routing and competition, see Fig. 1.

A LVQ-Capsule $\mathbf{C}_k^{h+1}$ in layer $h+1$ receives input vectors or messages $\mathbf{v}_i^h$ together with corresponding dissimilarity values d_i^h from *all* capsules of the parent capsule layer h. The dissimilarity value d_i^h is interpreted as the certainty of the capsule i of the layer h regarding the transmission of the message $\mathbf{v}_i^h$. In LVQ-Capsules, the d_i^h-values are *non-negative* and the smaller the value, the higher is the certainty, with optimum value zero indicating absolute certainty.

Inside a LVQ-Capsule $\mathbf{C}_k^{h+1}$, the first processing module is a transformation of all inputs $\mathbf{v}_i^h$ and dissimilarity values d_i^h consistently chosen to the capsule architecture, see Algo. 1. Hence, the transformation can be chosen to be more general than a simple linear transformation as proposed in [1]. The subsequent measuring module calculates the dissimilarities $\hat{d}_{i,j}$ as a linear combination of the dissimilarity measure between the transformed input $\tilde{\mathbf{v}}_i$ and the corresponding prototypes $\mathbf{w}_j$ and the transformed dissimilarity values $\tilde{d}_i$. The linear factor α keeps the historic information and hence realizes a decay of old dissimilarities over several capsule layers. If required, a corresponding dissimilarity transformation, e.g. the tangent space projection, can be applied to $\tilde{\mathbf{v}}_i$ resulting in $\hat{\mathbf{v}}_{i,j}$.

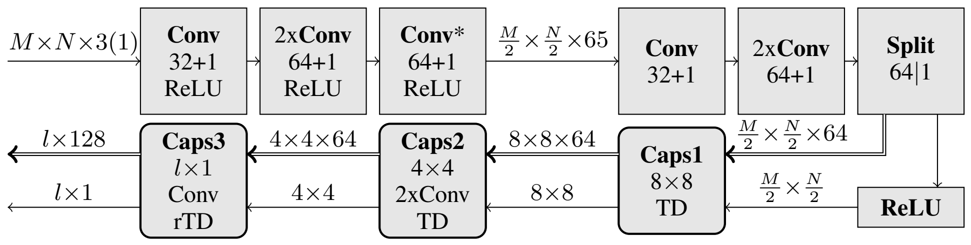

Figure 2: The network architecture for the experiments. See text for further explanations.

In this module, the prototypes can be seen as *adjustable parameters* for the dissimilarity calculation, which are adjusted during the training accordingly.

The routing module realizes an adaptive estimation of the class distribution centers from the perspective of each prototype $\mathbf{w}_j$. Here, the negative softmax

$$p_{i,j} = \textsc{NegSoftmax}\left(\left\{\hat{d}_{\hat{i},j} \mid \forall \hat{i}\right\}, i\right) = \frac{\exp\left(-\hat{d}_{i,j}\right)}{\sum_{\hat{i}} \exp\left(-\hat{d}_{\hat{i},j}\right)} \tag{3}$$

is used to compute the routing probabilities. Thus, the calculation of these quantities is equivalent to the classification decision procedure in RSLVQ in the case of just one prototype per class. Note that in contrary to the previous work on capsules we are not computing the routing probabilities over j.[5] Finally, the routing process is realized by computing the expected input vectors $\mathbf{v}_j$ and the expected dissimilarity values d_j. An expected dissimilarity value d_j is small if the vectors $\hat{\mathbf{v}}_{i,j}$ are dense around the prototype $\mathbf{w}_j$ and large otherwise. If more than one prototype is used in a capsule, a competition procedure is applied to resolve the redundancies. A general concept in LVQ is to consider just the signal of the best matching prototype (minimal dissimilarity value d_j) [32] or to apply a similar routing process as before.

The idea behind this capsule structure is to group several simple entities $\mathbf{v}_i^h$ into one more complex entity $\mathbf{v}_k^{h+1}$. Hence, each capsule receives all incoming simple entities and forms a more complex entity. During training the prototypes learn their responsibility for a certain entity. The communication between the capsules in different layers is realized via the dissimilarities d_i^h: if a capsule can not form a proper higher entity from the simple entities a high vote is delivered to the dissimilarity channel. Thus, the respective entity might be ignored in the subsequent upper capsules.

3 Experiments

3.1 The LVQ-Capsule network architecture

The architecture of the LVQ-Capsule network (LVQ-CN) used in the experiments is inspired by the slim SimpleNet architecture [33]. It consists of a couple of convolutional layers followed by three LVQ-Capsule layers, see Fig. 2. All convolutional layers have a kernel size of 3×3, stride $(1, 1)$ and use zero padding, except Conv*, which is a convolution with kernel size 5×5 and stride $(2, 2)$. The number of filters varies between 33 and 65. As non-linearities we use ReLUs for the first layers.

After the last convolutional layer we split the filtered image stack into the input vectors $\mathbf{v}_i^h$ and dissimilarity values d_i^h for the first LVQ-Capsule layer Caps1 where $i \in \{1, 2, ..., {}^{M}\!/_{2} \cdot {}^{N}\!/_{2}\}$ and M, N are the image dimensions. More precisely, we use 64 filters to form $\mathbf{v}_i^h$ and one to form d_i^h. Each i corresponds to a pixel position in the last filtered image stack. To get valid non-negative dissimilarity values d_i^h we apply ReLU activation. Caps1 has no internal transformation. The measuring block consists of $8 \times 8 = 64$ prototypes and no multiple prototypes per class/cluster. Hence, a competition module is not required. The dissimilarity measure is the TD with a subspace dimension $r = 16$, see Eq. (2) without any additional transformation.

[5] If we would compute the routing probabilities regarding the prototypes $\mathbf{w}_j$ we would obtain an unnormalized probability vector and therefore, the computation of the expectation values would be invalid.

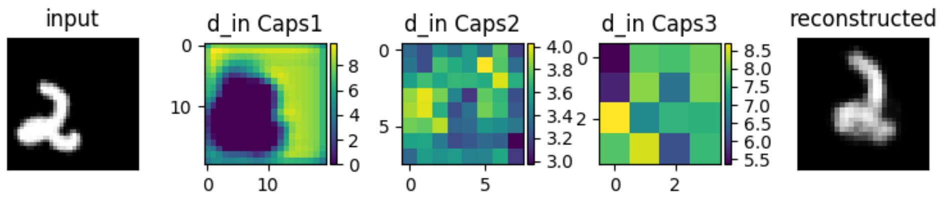

Figure 3: Visualization of dissimilarity maps and the reconstruction on affNIST. The heatmaps d_in are the inputs to the corresponding capsule layers.

The Caps2 layer is similar to Caps1 with a few slight changes: the input transformation module consists of two convolutions applied over the set of input vectors. We consider the vector dimension as feature dimension, the 8×8 shape as image size and filter over the $8\times8\times64$ structure. The convolutions are equivalent to the input convolutional layers with two times 64 filters and no non-linearities.

The final Caps3 is a LVQ-Capsule with a rTD of subspace dimension $r = 16$. Again, the input transformation is a convolution applied over the set of input vectors with 128 filters and settings equivalent to the transformation of Caps2. The number of prototypes is simply the number of classes l. For the training we transform the final dissimilarities of Caps3 via the negative softmax into probabilities, see Eq. (3). Instead of a heuristic loss function we use the generalized Kullback-Leibler divergence for the training [8]. This loss can be applied to possibility vectors, too.

Depending on the number of classes for the experiments, the LVQ-CN has a total number of 498K parameters for MNIST, affNIST and CIFAR-10 and of 442K for smallNORB, which is only slightly bigger than the network architecture in [7] but much smaller than in [1].

3.2 MNIST and affNIST

To relate our work to already introduced capsule techniques, we applied our LVQ-CN to both, MNIST [34] and affNIST[6] . Just like SABOUR ET AL. in [1], we connected the output vector of our final LVQ-Capsule layer to a reconstruction network (RN) to reconstruct an image. Since our output vector is of size 10×128 instead of 10×16, we have a different input size for the RN compared to them. Additionally, we trained the reconstruction network to reconstruct the *original* input image and not the augmented training image. All other settings were identical to SABOUR ET AL. For example, we trained on 28×28 MNIST digit images that have been randomly shifted up to two pixels in each direction. In contrast to [1], we observed that the regularization effect of the RN is negligible for our architecture. After the training the RN was used to extract insight information about what the capsules have learned.

The lowest test error for MNIST achieved in our experiments with early stopping is 0.28% and 0.3%, otherwise. SABOUR ET AL. achieved a test error rate of 0.25% on MNIST with the CapsNet architecture and HINTON ET AL. 0.44% with the matrix capsule network [7]. Applying the resulted network with early stopping to the affNIST test dataset *without* any further training on affNIST, results in an acceptable test error rate of 2.4%. This result shows the good generalization ability and the robustness of the LVQ-CN. Furthermore, we improved the error rate in a more difficult experimental setting from 21% [1] down to 2.4%. Fig. 3 shows a visualization of a reconstructed affNIST image and the corresponding dissimilarity heatmaps for each capsule layer of the LVQ-CN. The heatmaps are a direct visualization of the input dissimilarities d_i^h of each capsule layer. Feature vectors $\mathbf{v}_i^h$ that might be preferred during the routing[7] have a small d_i^h value. For the heatmaps, this means that a small value (dark pixel) corresponds to a more important feature vector. The RN returns a 28×28 centered digit image since it was trained accordingly. Although the network was never trained on affNIST, the reconstruction ability is acceptable.

In particular, the input dissimilarity heatmap of Caps1 reflects the position of the digit in the image. It is obvious that the most important feature vectors are related to image regions where the digit

[6]In affNIST each sample is a MNIST digit image with a random small affine transformation on 40×40 black background. Available at `https://www.cs.toronto.edu/%7Etijmen/affNIST/`

[7]Keep in mind that each capsule adds its own dissimilarity measure to d_i^h. Hence, a capsule can also reject supposedly important features.

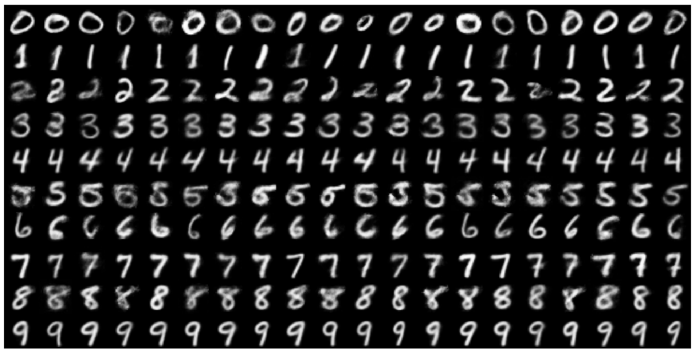

Figure 4: Visualization of 20 random points for each prototype in the LVQ-Capsule Layer (Caps3) via the reconstruction network. Each prototype corresponds to a class and learns to capture the most important transformations.

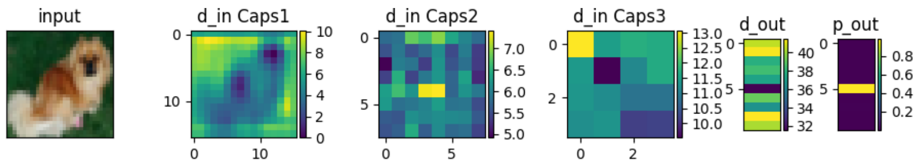

Figure 5: Visualization of dissimilarity maps on CIFAR-10. The heatmaps d_in are the inputs to the corresponding capsule layers. p_out is the transformation of d_out to a probability vector containing probability values obtained by the negative softmax function.

is located, which shows the strong localization ability of the LVQ-CN. Moreover, it seems that the network "looks" directly onto the digit to classify the given image but ignores the surroundings.

For the capsule layers Caps2 and Caps3 the image representations are more abstract. Because each capsule is connected to all feature vectors of the parent capsule layer, we are not observing an object localization as for the first capsule layer. Nevertheless, comparing several images, we observe that the patterns of the input dissimilarity heatmaps of Caps2 and Caps3 are similar for similar input images. That means, inputs of the same class usually activate the same capsules in the layer. Moreover, if objects of different classes share common features, the same regions are activated. See the supplementary material A for visualizations and an extended discussion.

To visualize the learned information of a certain LVQ-Capsule, we sample points from the learned prototypes of the last LVQ-Capsule layer Caps3 and process them through the RN. Since the capsules are equipped with the rTD, all the prototypes are r-orthotopes. Furthermore, it is well known that the prototypes tend to approximate the hidden data manifold structure to capture the most important invariances of the dataset [35]. In Fig. 4 we see what type of transformations are captured for each class by the corresponding prototype. This visualization procedure is a direct "random walk" inside the r-orthotope of a capsule. The procedure is possible due to the fact that the prototypes are living in the respective data space (here, the final 128 dimensional parameter space). Therefore, we can feed sampled points from a prototype (a r-orthotope) into the RN to obtain an image. The same visualization technique can be applied to the hidden layers Caps1 and Caps2. We refer to the supplementary material A for an illustration and an extended discussion. Additionally, we see parallels in the image generation ability to Generative Adversarial Neural Networks (GANs) [36].

3.3 CIFAR-10 and smallNORB

We conducted experiments as before to the more complex CIFAR-10 [37] and smallNORB datasets [38]. The LVQ-CN architecture was the same as in the previously described experiments. However, the RN was omitted here, because the simple RN was not appropriate to reconstruct such complex objects. Furthermore, we want to show that the localization behavior of the network is not the result of the reconstructor.

The best results that we observed on CIFAR-10 are test error rates of 9.5% and 9.7% with and without early stopping, respectively. During the training, we applied slight random rotations and horizontal flips in addition to the random shifts. These results are slightly better than the result of 11.9% in [7] and 10.6% in [1]. A visualization of the heatmaps of a CIFAR-10 test sample is depicted in Fig. 5. Again, the LVQ-CN shows a good localization ability regarding object position insight the image even tough it is not as strong as on the previous experiment. Furthermore, we observed that more prototypes are activated to classify an object in comparison to the LVQ-CN for the MNIST data. This might be seen as an indicator that the CIFAR-10 dataset is more complex than MNIST. Moreover, considering several inputs, repeating patterns in Caps3 cannot be identified. See the supplementary material B for respective visualizations.

For smallNORB we achieved a test error rate of 5.6%, which is comparable to the base line model but not competitive with the result of 1.8% of the special matrix capsule structure of HINTON ET AL. in [7]. The network was trained with the same augmentation techniques as in [7]. To capture the structure of the stereo images in an appropriate manner, we changed our initial dissimilarity map learning from "+1" to "+2" for all initial convolutions. Thus, we increased the number of filters by one for the first convolutional layers. Thereafter, we split the last filtered image stack twice into $32|1$ and changed the vector dimension of Caps1 to 32, accordingly. Hence, instead of learning a 64-dimensional vector regarding the first convolutions we learned a 32-dimensional vector for each image of the stereo pair. This helps to capture the structure of the $M \times N \times 2$ stereo image pairs. Furthermore, we repeated the experiments of the generalization ability to novel viewpoints as in [7]. We were not able to train our network to a comparable test error rate of 3.7% on familiar viewpoints for azimuth. Our network converges to 5.7%. Nevertheless, our generalization ability on novel azimuth viewpoints is 10.8%, which is better than the matrix capsule network. The performance of our network in the elevation experiment on novel viewpoints is slightly worse (error rate 15.4% compared to 12.3%). In the supplementary material C we present some illustrations of the heatmaps.

4 Conclusion

The LVQ-Capsule approach presented in this paper provides an alternative capsule approach compared to the methods proposed by SABOUR ET AL. and HINTON ET AL. One of the key ideas is to replace the computationally expensive and crucial iteration parameters by incorporation of LVQ and VQ machine learning techniques. At the same time we provide an easy fusion of LVQ methods and neural networks. Additionally, the proposed routing method is of great interest for various vector quantization methods to solve the problem of handling several input signals and to reduce the curse of dimensionality in dissimilarity measuring. Moreover, dissimilarity routing offers the advantage of an obvious optimum in terms of the lower zero bound for dissimilarities compared to frequently unbounded similarities or inner products.

In experiments on MNIST, affNIST, smallNORB and CIFAR-10 we showed the competing performance of LVQ-CN in comparison to the other capsule approaches. As a side effect we discovered that the input heatmap to the first LVQ-Capsule layer is strongly related to saliency maps [39] and gives an idea which regions of the image are important for the network, see Fig. 3 and 5. In contrast to saliency maps, the visualization is a product of the forward propagation of an input image and is obtained without any computational overhead. Moreover, this visualization is independent of the output. With a different visualization technique we showed how the learned prototypes could be visualized to discover the learned information, see Fig. 4. We hope that these initial findings are attractive to trigger the study of the visualization/interpretation abilities of prototype based vector quantization.

Finally, the LVQ-Capsule layers are a powerful alternative to final fully-connected layers for classification tasks. Future work should consider well-known outlier detection concepts of LVQ like in [40, 41], which could be useful in the area of novelty detection. Also in this scope, the dissimilarity measure provides a mathematically well-founded quantity, frequently easier to handle than similarities or inner products. Furthermore, we want to study LVQ-Capsules on larger datasets like ImageNet, the applicability for regression problems and if it is possible to use LVQ-Capsules in a convolutional manner.

References

[1] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3856–3866. Curran Associates, Inc., 2017.

[2] G.E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In T. Honkela, W. Duch, M. Girolami, and S. Kaski, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN), Espoo Finland*, volume LNCS 6791 of *Lecture Notes in Computer Science*, pages 44–51, Berlin Heidelberg, 2011. Springer.

[3] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

[4] Teuvo Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1984. 3rd ed. 1989.

[5] François Chollet et al. Keras. `https://keras.io`, 2015.

[6] N.R. Pal, K. Pal, J.M. Keller, and J.C. Bezdek. A possibilistic fuzzy c-means clustering algorithm. *IEEE Transactions on Fuzzy Systems*, 13(4):517–530, 2005.

[7] G. Hinton, S. Sabour, and N. Frosst. Matrix capsules with EM routing. In *Proc. of the Sixth International Conference on Learning Representations*, 2018.

[8] A. Cichocki, R. Zdunek, A.H. Phan, and S.-I. Amari. *Nonnegative Matrix and Tensor Factorizations*. Wiley, Chichester, 2009.

[9] Ayush Jaiswal, Wael AbdAlmageed, and Premkumar Natarajan. CapsuleGAN: Generative adversarial capsule network. *arXiv preprint arXiv:1802.06167*, 2018.

[10] Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. Investigating capsule networks with dynamic routing for text classification. *arXiv preprint arXiv:1804.00538*, 2018.

[11] Rodney LaLonde and Ulas Bagci. Capsules for object segmentation. *arXiv preprint arXiv:1804.04241*, 2018.

[12] David Rawlinson, Abdelrahman Ahmed, and Gideon Kowadlo. Sparse unsupervised capsules generalize better. *arXiv preprint arXiv:1804.06094*, 2018.

[13] Tomas Iesmantas and Robertas Alzbutas. Convolutional capsule network for classification of breast cancer histology images. *arXiv preprint arXiv:1804.08376*, 2018.

[14] Mohammad Taha Bahadori. Spectral capsule networks. 2018.

[15] Dilin Wang and Qiang Liu. An optimization view on dynamic routing between capsules. 2018.

[16] Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. Zero-shot user intent detection via capsule neural networks. 2018.

[17] Yequan Wang, Aixin Sun, Jialong Han, Ying Liu, and Xiaoyan Zhu. Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1165–1174. International World Wide Web Conferences Steering Committee, 2018.

[18] Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti Bathula. Dense and diverse capsule networks: Making the capsules learn better. *arXiv preprint arXiv:1805.04001*, 2018.

[19] Teuvo Kohonen. Learning Vector Quantization. *Neural Networks*, 1(Supplement 1):303, 1988.

[20] P. Schneider, B. Hammer, and M. Biehl. Distance learning in discriminative vector quantization. *Neural Computation*, 21:2942–2969, 2009.

[21] P. Simard, Y. LeCun, and J.S. Denker. Efficient pattern recognition using a new transformation distance. In S.J. Hanson, J.D. Cowan, and C.L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 50–58. Morgan-Kaufmann, 1993.

[22] T. Hastie, P. Simard, and E. Säckinger. Learning prototype models for tangent distance. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 999–1006. MIT Press, 1995.

[23] S. Saralajew and T. Villmann. Transfer learning in classification based on manifold models and its relation to tangent metric learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN), Anchorage*, pages 1756–1765. IEEE Computer Society Press, 2017.

[24] D. Keysers, W. Macherey, H. Ney, and J. Dahmen. Adaptation in statistical pattern recognition using tangent vectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):269–274, 2004.

[25] S. Saralajew and T. Villmann. Restricted tangent metrics for local data dissimilarities - mathematical treatment of the corresponding constrained optimization problem. *Machine Learning Reports*, 11(MLR-01-2017), 2017. ISSN:1865-3960, `http://www.techfak.uni-bielefeld.de/%7Efschleif/mlr/mlr%5f01%5f2017.pdf`.

[26] A. Sato and K. Yamada. Generalized learning vector quantization. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference*, pages 423–9. MIT Press, Cambridge, MA, USA, 1996.

[27] K. Crammer, R. Gilad-Bachrach, A. Navot, and A.Tishby. Margin analysis of the LVQ algorithm. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing (Proc. NIPS 2002)*, volume 15, pages 462–469, Cambridge, MA, 2003. MIT Press.

[28] S. Seo and K. Obermayer. Soft learning vector quantization. *Neural Computation*, 15:1589–1604, 2003.

[29] S. Seo, M. Bode, and K. Obermayer. Soft nearest prototype classification. *IEEE Transaction on Neural Networks*, 14:390–398, 2003.

[30] T. Villmann, M. Biehl, A. Villmann, and S. Saralajew. Fusion of deep learning architectures, multilayer feedforward networks and learning vector quantizers for deep classification learning. In *Proceedings of the 12th Workshop on Self-Organizing Maps and Learning Vector Quantization (WSOM2017+)*, pages 248–255. IEEE Press, 2017.

[31] M. Biehl, B. Hammer, and T. Villmann. Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 7(2):92–111, 2016.

[32] J. C. Bezdek and L. I. Kuncheva. Nearest prototype classifier designs: An experimental study. *International Journal of Intelligent Systems*, 16(12):1445–1473, December 2001.

[33] Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*, 2016.

[34] Y. LeCun, C. Cortes, and C. Burges. The MNIST database, 1998. `http://yann.lecun.com/exdb/mnist/`.

[35] S. Saralajew and T. Villmann. Adaptive tangent metrics in generalized learning vector quantization for transformation and distortion invariant classification learning. In *Proceedings of the International Joint Conference on Neural networks (IJCNN) , Vancouver*, pages 2672–2679. IEEE Computer Society Press, 2016.

[36] I.J. Goodfellow, Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[37] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. https://www.cs.toronto.edu/%7Ekriz/cifar.html.

[38] 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, pages II–104. IEEE, 2004. https://cs.nyu.edu/%7Eylclab/data/norb-v1.0-small/.

[39] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. https://distill.pub/2018/building-blocks.

[40] T. Villmann, M. Kaden, D. Nebel, and M. Biehl. Learning vector quantization with adaptive cost-based outlier-rejection. In G. Azzopardi and N. Petkov, editors, *Proceedings of 16th International Conference on Computer Analysis of Images and Pattern, CAIP 2015, Valetta - Malta*, volume Part II of *LNCS 9257*, pages 772 – 782, Berlin-Heidelberg, 2015. Springer.

[41] L. Fischer, B. Hammer, and H. Wersing. Efficient rejection strategies for prototype-based classification. *Neurocomputing*, 169:334–342, 2015.

Supplementary material

A Additional Visualizations for MNIST and affNIST

In Fig. 6 we depict some random samples from the MNIST test dataset and show the respective reconstructed image. As one can see, the reconstruction is really powerful in decoding the 128 dimensional representations of a given digit image. In the following, we use the reconstruction network (RN) to visualize the learned information of the capsules. We emphasize that the prototypes of the capsules are living in the data space in which they interact. Thus, we can use the prototypes itself to observe what is learned by a capsule. In the case of the Minkowski distance Eq. (1) the prototypes are points of the data space. Hence, if our final LVQ-Capsule layer would be equipped with this distance we could easily push the parameters of the learned point prototype through the RN to get an understanding of what the prototype is and what is modeled by the prototype.

In our network implementation we are dealing with (restricted) tangent distances Eq. (2) as dissimilarity measure. Thus, the prototypes are affine subspaces. This means that a prototype has learned an infinite number of particular representations of the data space, because each point of the affine subspace is an instantiation for what is modeled by the prototype. Therefore, we can start a random walk in this subspace to see what transformations are learned.

The prototype based classification is realized using the winner-takes-all rule. Hence, if the LVQ-CN is trained properly we can assume that the prototypes are similar to the data points that they represent. For the tangent distance case, it is known that the prototypes are trying to model the most important transformations: the invariances of the dataset. In the following discussion we show that the "function" of each capsule could be interpreted by these principles and the use of the RN.

In Fig. 7 we present some additional digit images from the affNIST dataset as an extension of the visualization of Fig. 3. Although neither the capsule network nor the decoder was trained on affNIST, the reconstruction ability is still acceptable. It is interesting how the network tries to remove the affine transformation of the digit images. For example in the second image from above the scaling is removed.

If we consider several images of the heatmaps we observe repeating patterns. Obviously, if digits belong to the same class, the same prototypes are marked as important, indicated by small dissimilarities. This property becomes easily visible if we plot the dissimilarities d_i^h for several samples of one class as lines over the number of prototypes i. In Fig. 8 the respective line plots of the dissimilarity input of Caps3 are depicted. Note that these d_i^h's are the output values of Caps2.

We can see that the important prototype patterns are unique for a given class. Moreover, we discover that digits which share common structural features also share the same prototype. For example, the prototype one (x-coordinate one) is important for the digits 0, 2, 3, 5, 6 and 8. A common structure of these digits are circular segments. Hence, the abstract representation of the prototype one could be interpreted as the modeling of arcs. Another example is the prototype 16, which is important for the digits 1, 4 and 7. Here, the common structure is a straight line. This observed behavior is an indicator for our claim that the prototypes of our capsule network learn to model more and more abstract representations over increasing depth of the network.

Fig. 4 shows a random walk inside the 16-dimensional orthotope of the final LVQ-Capsule layer. We can detect the most important transformations for the class discrimination. The same concept is used to visualize the behavior of some trained LVQ-Capsules in the layer Caps2. Since Caps2 is a hidden layer, we cannot visualize the random walk inside the affine subspace directly, because this requires to proceed the signal through the final capsule layer too. The trick works as follows: we feed the learned center of the 16-dimensional orthotope of Caps3 through the RN and obtain a visualization of a more or less ideal digit image of the corresponding class. Then, we use this virtual image to feed it through the capsule network until the hidden capsule layer Caps2. This is done to keep the modeling effect of the final layer as small as possible. At the layer Caps2 we substitute the original vector output of the prototype of interest with a random vector of the affine subspace. After that we push the manipulated signals through the capsule network and the subsequent RN to receive a visualization of what is modeled by the intermediate prototype.

From the previous discussion we assume that the prototype one in Caps2 is sensitive for circular segments. If this statement is true the visualization via the random walk should show the modeling

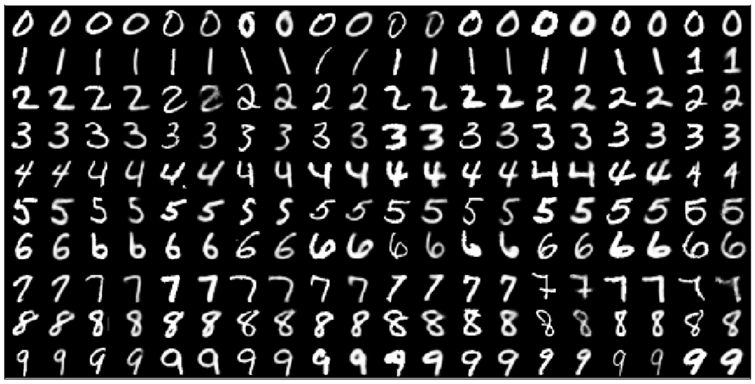
Figure 6: Reconstructed digits from the MNIST test dataset. From left to right: the digits are pairs of (original image, reconstructed image). For each digit we visualize 10 randomly chosen digits.

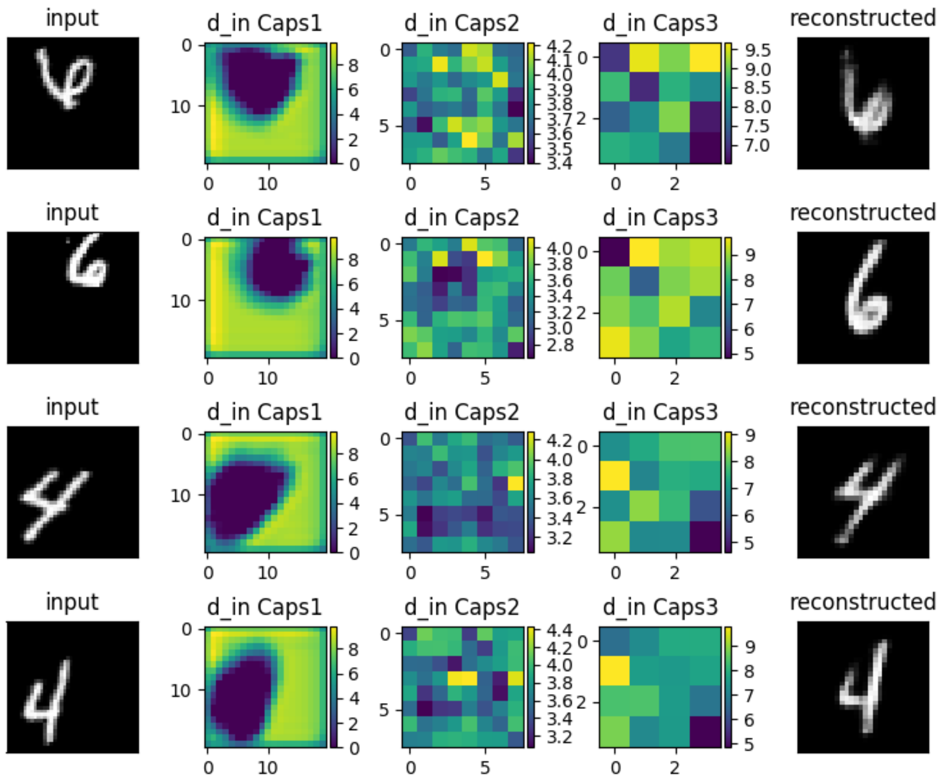

Figure 7: Visualization of dissimilarity maps and the reconstruction on affNIST digits. The heatmaps d_in are the inputs to the corresponding capsule layers.

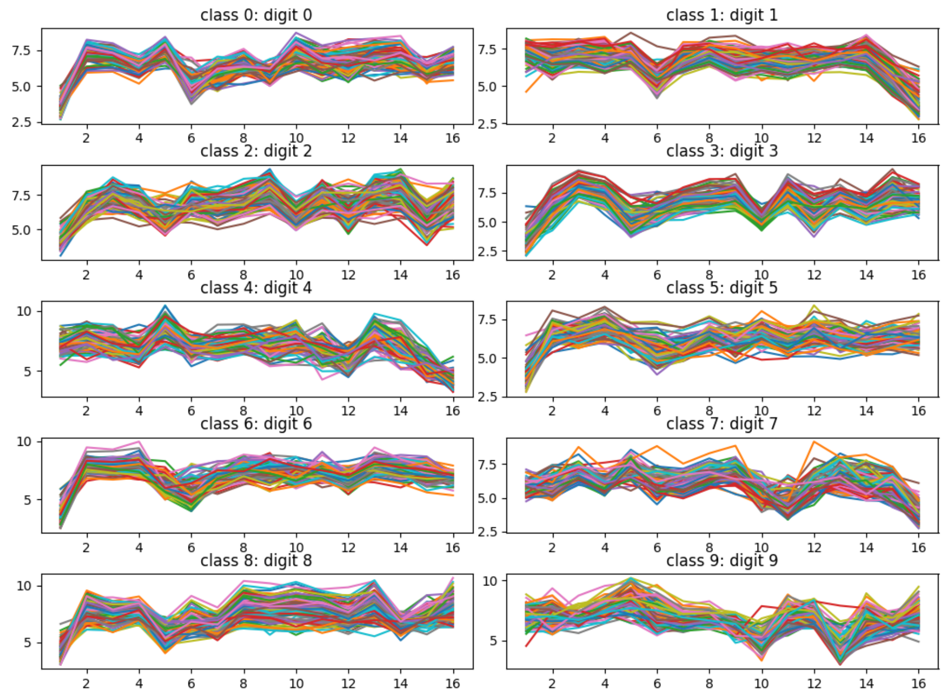

Figure 8: Line plot of the dissimilarity inputs of Caps3 over 100 samples per class of the MNIST test dataset.

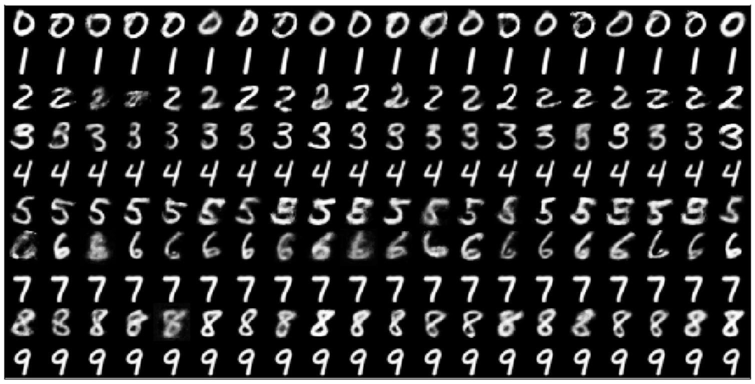
Figure 9: Visualization of 20 random points per class of prototype one in the LVQ-Capsule Layer (Caps2) via the reconstruction network.

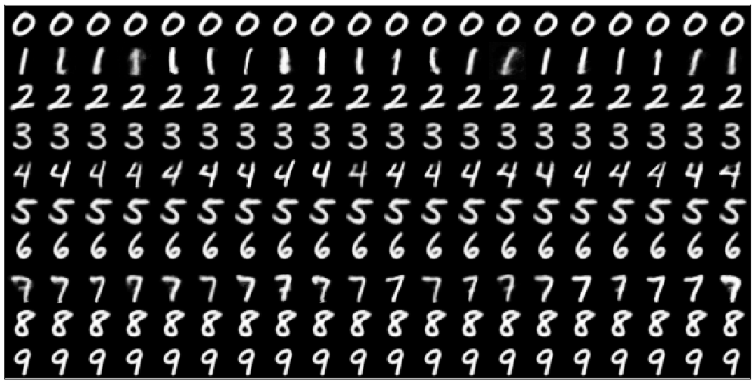
Figure 10: Visualization of 20 random points per class of prototype 16 in the LVQ-Capsule Layer (Caps2) via the reconstruction network.

of an arc, because the prototype tries to model the most important invariances. In Fig. 9 we show the respective illustration. As described before, we push each ideal digit image through the network and manipulate it to see the influence of the intermediate prototype for each class. It is visible that only the assumed digits are affected by that prototype. Further, it seems that the prototype is sensitive for circular segments. For example the closed loop of the digit 9 is mostly not affected. Otherwise, for the digits 0, 2, 3, 5, 6 and 8 the prototype is able to model an opening of the arc segment as well as heavy variations.

In Fig. 10 we see the equivalent visualization for the prototype 16. As claimed before, this prototype is sensitive to straight lines. In Fig. 10 it is visible how the vertical strokes of the digits 1, 4 and 7 are affected. The prototype is able to model a thickening, a snake style, etc. Looking at the digit 4, it is interesting that the small strokes are not really affected.

B Additional Visualizations for CIFAR-10

As in the previous section we want to detect visually if there are repeating patterns in the heatmaps. In Fig. 11 we depict the dissimilarity line plots regarding the 16 prototypes of Caps3. In contrast to MNIST, repeating patterns cannot be detected. It looks more chaotic, which is an indicator that all the feature vectors are important to compute a classification decision. Due to the missing reconstruction network we cannot visualize the prototypes like in the previous section.

In Fig. 12 we show four heatmap plots of CIFAR-10 samples. Despite that the network was not equipped with a RN the localization ability is still good. The network detects the most important features in the object region. For example, for the class 'automobile' we observe that the network has a strong tendency to detect the wheels of the car, if they are visible. For the class 'ship' the presence of water seems to be important. Nevertheless, the localization ability is not as strong as on datasets without background noise like smallNORB.

C Additional Visualizations for smallNORB

In contrast to CIFAR-10, we achieved a network behavior for smallNORB similar to that for MNIST. The network uses the same prototypes again and again for one class, see Fig. 13. Furthermore, the line plot has some irregularities regarding class 3 'trucks' and class 4 'cars'. If we consider the accuracy of the network for the wrongly classified classes, we can see that the confusion between 'cars' and 'trucks' causes trouble. Particularly, if those objects are presented under a high elevation angle, misclassifications are significantly increased. This observation is also mirrored in the results of the generalization ability to novel viewpoints.

Fig. 14 provides the heatmap plots over four samples. Similar to MNIST, the object localization sensitivity of the network is really high. Obviously, the object localization is strongly correlated to the amount of background noise. As for CIFAR-10, a visualization of prototypes is not possible due to the missing RN.

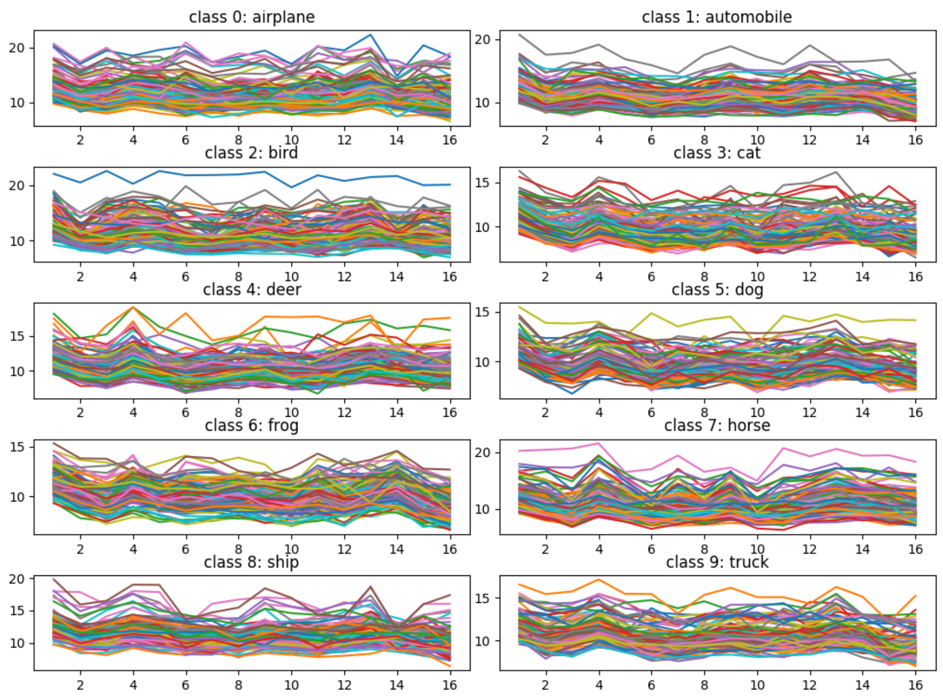

Figure 11: Line plot of the dissimilarity inputs of Caps3 over 100 samples per class of the CIFAR-10 test dataset.

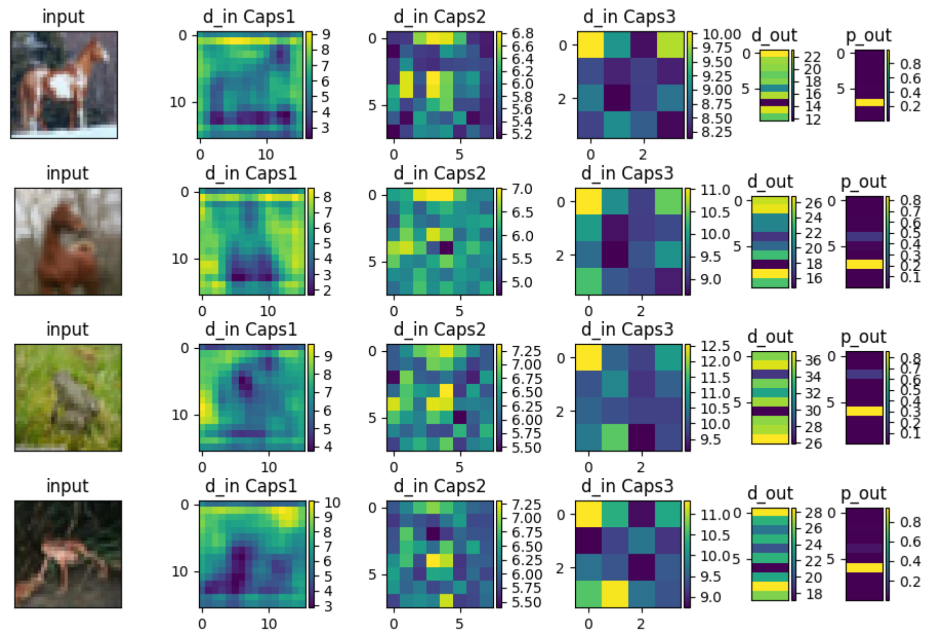

Figure 12: Visualization of dissimilarity maps of CIFAR-10. The heatmaps d_in are the inputs to the corresponding capsule layers. p_out is the transformation of d_out to a probability vector by means of the negative softmax function. The upper two samples are from the class horses and the lower two are from the class frogs.

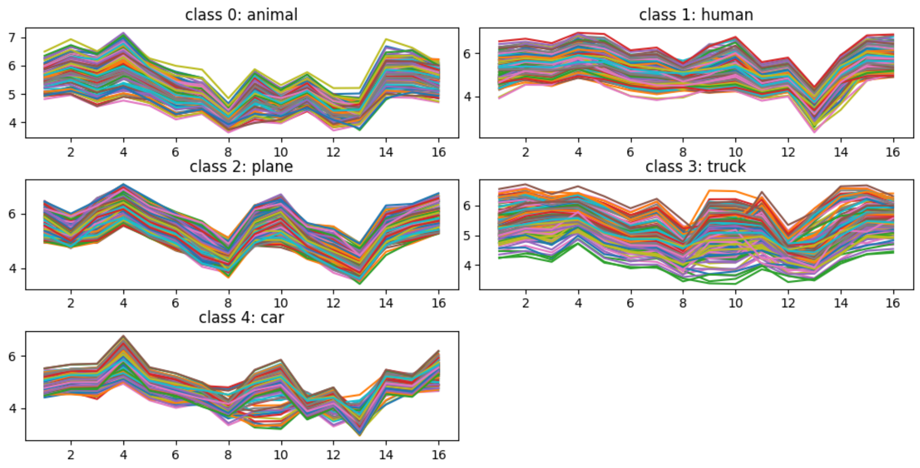

Figure 13: Line plot of the dissimilarity inputs of Caps3 over 100 digits per class of the smallNORB test dataset.

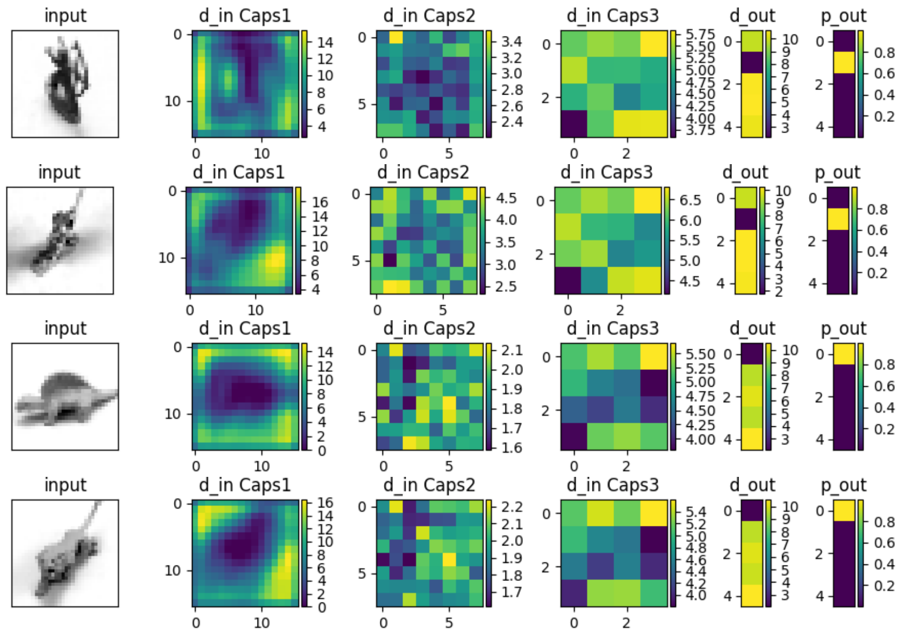

Figure 14: Visualization of dissimilarity maps of smallNORB. The heatmaps d_in are the inputs to the corresponding capsule layers. p_out is the transformation of d_out to a probability vector by means of the negative softmax function. The upper two samples are from the class humans and the lower two are from the class animals.