Matrix Capsules with EM-Routing

Heidelberg Collaboratory for Image Processing (HCI)

Seminar report by
Carsten Lüth

Supervisor: PD. Dr. Ullrich Köthe

SS 2018

Contents

1 Introduction **3**

2 Theory of capsules **5**

3 Routing by agreement based on EM **6**

4 Capsule Architecture **7**
4.1 Spread Loss 8

5 Results **9**
5.1 Results on smallNORB 9
5.2 Generalization to novel viewpoints on smallNORB 9
5.3 Results on other datasets 10

6 Capacity of the architecture **11**

7 Sources **12**

1 Introduction

A convolutional neural network is a class of deep, feed-forward that is mostly applied to analyze visual imagery. They use a variation of multilayer perceptron that is designed to minimize preprocessing, since they have a shared-weight architecture which is inspired by the biological vision system that uses the same knowledge on all locations in the image.
They have achieved remarkable progress on numerous practical vision tasks. The stacking of convolutional filters and non-linearity units in combination with other techniques such as max-pooling imply a difficulty of understanding the internal organization of neural networks. Thereby current neural networks are widely regarded as "black box".
The human vision system shows a much higher interpretability during the recognition since humans can describe how a system is designed of many parts and how they unite to a whole and thereby describe the part-whole relationship as well as name cues such as the shape, color and texture for their final decision.
The capsule network is inspired by the way a rendering system works which describes a specific entity with a pose. For a 3 dimensional space 4 dimensional matrices are used in a rendering system and from this pose matrix it is possible to get the positional arrangement or pose matrix of all smaller parts it consists of by a simple matrix multiplication with a transformation matrix which describes the viewpoint invariant part-whole relationship as can be seen in figure 1.

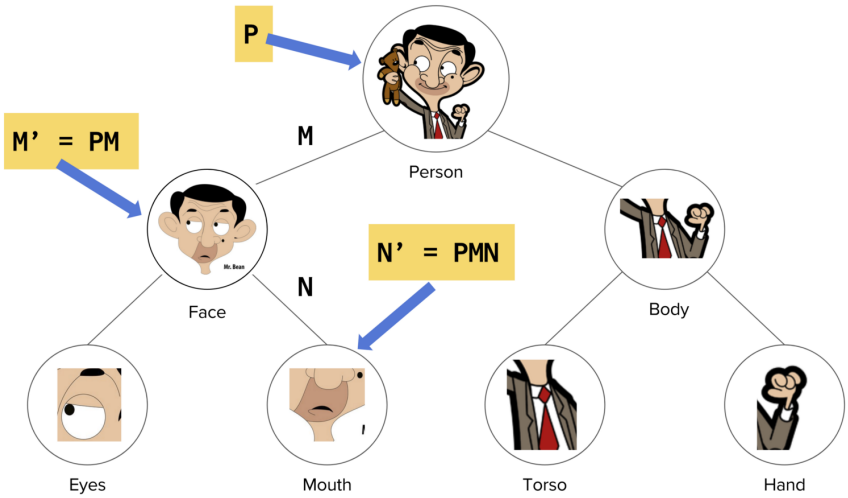

Figure 1: Pose matrices representing the hierarchical relationship.
P: pose matrix of the person.
M: spatial relationship between face and person
N: represents the relationship between mouth and face.
M' and N': pose matrices for the face and the mouth

A capsule uses grouped scalars as pose to represent the instantiation parameters (or properties) of the entity or part the capsule is describing. Based on

this pose of the capsule it gives a vote for the capsules in the higher layer which is computed by matrix multiplication of the pose with a learned transformation matrix. When viewpoint or the position of the object changes the pose matrices of the parts and wholes will change in a coordinated way and the agreement between the votes of the parts for wholes will persist.
The network then uses high dimensional coincidence filtering to find tight clusters of votes from the parts (lower level capsules) for the wholes (higher level capsules) thereby comes the name of this algorithm: "routing by agreement". This procedure is using weights or probabilities which describes which part is assigned to which whole based on the similarity of the votes from other parts assigned to the same whole. If there is a substantial amount of votes that have a high agreement for a certain capsule this capsule will be activated.
This activation of a capsule thereby is very similar to a measurement of agreement of the votes, thereby capsules with low agreement have a low activation and will be neglected in the routing procedure for the next level of capsules. Thereby the routing procedure filters out irrelevant information while it keeps the spatial relation of the parts.
Thereby the capsule network structure works analogously to the already understood rendering algorithm which makes it more interpretable than a standard neural network. Also the structure of the activations is very similar to a parse tree where each activation corresponds to a a node of a parse tree. By using the dynamic routing algorithm each active capsule will choose a capsule in the higher layer to be its parent in the tree. This way the dynamic routing procedure solves the problem of assigning parts to wholes.
This leads to an important difference of capsules and standard neural networks which is that the activation of a capsule is based on comparing multiple incoming pose predictions in difference to the standard neural network where the activation is based on a comparision between a single incoming activity vector and a learned weight vector.

2 Theory of capsules

The routing procedure of the capsules is the replacement of a simple non-linearity such as ReLU that standard neural networks apply to the activation. The routing procedure computes the activation and pose of the capsules in the next layer based on the activations and poses of the lower layer which makes it much more complicated.
Each capsule consists out of an activation probability denoted as a and a 4×4 pose matrix M. The activations of the capsules are very similar to the activations in a standard neural net with the exception that they belong to a capsule which has also the pose to describe the entity which means that a capsule describes much more than the simple occurance of a certain muster in the activations of the lower layer. The capsule network is build with several layers of capsules and all capsules in layer L are denoted as Ω_L. Between to adjacent layers of capsules L and $L+1$ there is a learned transformation matrix $W_{ij} \in \mathbb{R}^{4\times4}$ for every pair of capsules $i \in \Omega_L$ and $i \in \Omega_{L+1}$. The vote from capsule i to capsule j is computed with a simple matrix multiplication of pose and transformation matrix:

$$V_{ij} = M_i W_{ij} \tag{1}$$

The activations a_i and votes V_{ij} for all capsules $i \in \Omega_L$ and $j \in \Omega_{L+1}$ are then used in the routing algorithm to compute the activations a_j and poses M_j in Ω_{L+1}. So the routing algorithm decides where the output of the capsule goes which is in standard neural networks learned with weights.
The dynamic routing by agreement algorithm uses a **routing coefficient** that is dependent on the input and gets computed dynamically. Thereby capsule i gets assigned to a capsule j that has the best fitting cluster of votes for its pose. An example of this can be seen in Figure 2.

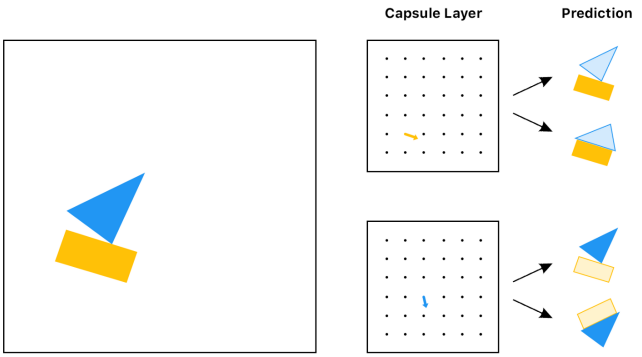

Figure 2: Example of the dynamic routing procedure with a simple boat. With the different parts of the boat it is possible to arrange both a house and the boat. Based on the different predictions from the parts the wholes look very different and the parts get assigned to the boat with the routing coefficient since the votes for its pose have the highest agreement

The routing algorithm that gets proposed in the paper [**HSF18**] is based on the Expectation-Maximization procedure. It computes means, variances and

activation of the capsules in Layer L probabilities iteratively as well as the routing coefficients between all capsules $i \in \Omega_L$ and $j \in \Omega_{L+1}$.

3 Routing by agreement based on EM

When given the activations and poses of all capsules in Ω_L the routing algorithm decides which capsules to activate in Ω_{L+1} and how to assign every capsule in Ω_L to one capsule in Ω_{L+1}. This is what leads to the parsing tree like structure of the network since to activate capsule j in Ω_{L+1} many capsules in Ω_L have to be assigned to capsule j and thereby leads to a compression of information which is still well described due to the fact that capsule j describes an entity which consists out of the parts of the capsules that get assigned to it.

For the EM-Algorithm each capsule in Ω_{L+1} corresponds to a Gaussian where the pose of each active capsule corresponds to a data-point. The choice of activating capsule j is made differentiable by introducing a cost of explaining data point i by using capsule j where h is the h^{th} dimension of the vectorized vote V_{ij}:

$$\begin{aligned} cost_{j|i} &= -\sum_h ln(P^h_{i|j}) \\ P^h_{i|j} &= \frac{1}{\sqrt{2\pi(\sigma^h_j)^2}} \cdot exp(-\frac{(V^h_{ij} - \mu_j)^2}{2(\sigma^h_j)^2}) \end{aligned} \tag{2}$$

V^h_{ij} : h^{th} dimension of the vectorized vote V_{ij}

μ_j : mean of the fitted gaussian

σ_j : variance of the fitted gaussian under the assumption of an axis aligned covariance matrix

How μ_j and σ_j are computed is shown in figure 3 alongside the routing algorithm. The routing coefficient r_{ij} has the following property which makes it a possible way to view how information flows through the network:

$$a_i = \sum_j r_{ij} \tag{3}$$

This is the reason why the cost is weighted with the routing coefficient since a capsule with a low activation has no big impact on the routing procedure. The activation of the higher level capsule then is computed with the logistic function of the cost:

$$\begin{aligned} a_j &= logistic(\lambda(\beta_a - \beta_u \sum_i r_{ij} - cost_j)) \\ cost_j &= \sum_i r_{ij} cost_{j|i} = \sum_h (ln(\sigma^h_j) + \frac{1 + ln(2\pi)}{2}) \sum_i r_{ij} \end{aligned} \tag{4}$$

β_a : trained parameter for all capsules

β_u : trained parameter for one capsule

$\sum_i r_{ij}$: amount of data assigned to capsule j

Capsule j thereby only is activated when σ_j is small enough and $\sum_i r_i j$ is big enough this corresponds to a tight cluster of votes with sufficient capsules from Ω_L inside the cluster with very importantly relatively high activations.
The final pose and activation for all capsules in Ω_{L+1} are computed by running the EM algorithm for a set amount of iterations with set poses and activations in Ω_L. So it takes not only the likelihood but also the pose/spatial arrangement of the parts into the decision and by this the weighted mean μ_j which is the pose of M_J after the final iteration has a very small deviation from the votes that have a high routing coefficient. The algorithm is a form of cluster finding which is very similar to the traditional EM algorithm.

Procedure 1 Routing algorithm returns **activation** and **pose** of the capsules in layer $L+1$ given the **activations** and **votes** of capsules in layer L. V_{ij}^h is the h^{th} dimension of the vote from capsule i with activation a_i in layer L to capsule j in layer $L+1$. β_a, β_v are learned discriminatively and the inverse temperature λ increases at each iteration with a fixed schedule.

procedure EM ROUTING($\boldsymbol{a}$, V)
  $\forall i \in \Omega_L, j \in \Omega_{L+1}$: $R_{ij} \leftarrow 1/|\Omega_{L+1}|$
  for t iterations **do**
    $\forall j \in \Omega_{L+1}$: M-STEP($\boldsymbol{a}$, R, V, j)
    $\forall i \in \Omega_L$: E-STEP(μ, σ, $\boldsymbol{a}$, V, i)
  return $\boldsymbol{a}$, M

procedure M-STEP($\boldsymbol{a}$, R, V, j) ▷ for one higher-level capsule
  $\forall i \in \Omega_L$: $R_{ij} \leftarrow R_{ij} * \boldsymbol{a}_i$
  $\forall h$: $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$
  $\forall h$: $(\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$
  $cost^h \leftarrow (\beta_v + log(\sigma_j^h)) \sum_i R_{ij}$
  $a_j \leftarrow sigmoid(\lambda(\beta_a - \sum_h cost^h))$

procedure E-STEP(μ, σ, $\boldsymbol{a}$, V, i) ▷ for one lower-level capsule
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{p}_j \leftarrow \frac{1}{\sqrt{\prod_h^H 2\pi(\sigma_j^h)^2}} e^{-\sum_h^H \frac{(V_{ij}^h - \boldsymbol{\mu}_j^h)^2}{2(\sigma_j^h)^2}}$
  $\forall j \in \Omega_{L+1}$: $\boldsymbol{R}_{ij} \leftarrow \frac{\boldsymbol{a}_j p_j}{\sum_{u \in \Omega_{L+1}} \boldsymbol{a}_u p_u}$

Figure 3: EM-Routing

4 Capsule Architecture

When using a capsule network one of the most important tasks is it to have good low level features in the low capsule layers and how to get those low level capsule features. This is why the first few layers of a capsule network are composed of convolutional layers which extract the low level features. Behind the convolutional layers the primary capsules are used to turn the channel-wise output of the convolutional layers into capsules. The method used is a simple convolutional layer with a 1×1 kernel supposed it receives A input channels from the previous layer and should create B different capsule types it will have $B \cdot (4 \cdot 4 + 1)$ output channels. The $b \cdot 1$ additional channels are used to describe the activation of the different capsules after applying the logistic function on them.

So to sum this up a primary capsule that receives an input with A channels and size 15×15 creates with a linear transformation B different capsule types every one of them consisting of $4 \cdot 4 + 1$ channels and every capsule type exists 15×15 times which preserves the spatial arrangement. Important to note here is that all capsules that belong to the same capsule type detect what the network learned to be the same entity. Every single capsule thereby has a receptive field on which its activation and pose is based.
This fact gets used by the next type of capsule the convolutional capsules which utilize a kernel with size K just like a normal convolutional layer, but they work just as described in 2 but as votes every capsule just gets the input from $K \times K$ spatial arranged capsules per capsule type in the lower layer. Convolutional capsules allow the model to create more complex features for the capsule types very similar how a normal convolutional layer does it which allows the model to make predictions based on better features. The last layer is the class capsule layer where the are only D different capsules for D different classes. Every capsule in this layer receives input from all capsules in the layer below. To lower the amount of parameters every capsule type in the lower layer has the same weights to one capsule in the class capsule layer. By adding the scaled coordinate to the vote of every capsule according to its receptive field that the model is able to distinguish between the same entity with the same pose but in different places, this technique is called **Coordinate Addition**. The model with some parameters can be seen in figure 4.

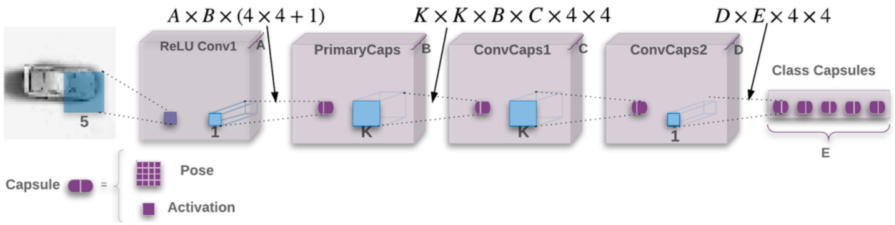

Figure 4: Capsule network architecture with the amount of parameters noted between capsule layers.

4.1 Spread Loss

When training a wide network like the capsule network it can occur that only a part of the width of the network actually is used, this would show itself in this case when the network only uses a s very small amount of capsules. To counteract this building of dead capsules in lower layers and make the training less sensitive to initialization the the "spread loss" is introduced to maximize the gap between the activation of the target class a_t and the activation of the other classes. It uses a margin m where if the distance squared distance between them is smaller than m the loss of this pair is set to 0:

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

This margin gets linearly increased from 0.2 to 0.9 during the training.

5 Results

Since the capsule network is designed to be good at using the simple linear pose changes for different viewing angles it is tested mostly on shape detection tasks to showcase its advantages over traditional convolutional neural networks.

5.1 Results on smallNORB

The smallNORB dataset has 5 different classes of toy models: airplanes, trucks, cars, humans and animals. Every class has 5 different toys photographed in different positions and lightings as test and training data. The dataset consists of 24,300 images with a resolution of 96×96 for the network those images get downsampled to 48×48 and normalized. For the training they additionally randomly crop 32×32 patches with random added brightness and contrast the test images are a 32×32 crop from the center. A for this task optimized tradition CNN is trained in exactly the same fashion, it is referred to as Baseline CNN. The results can be seen in figure 5

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

Figure 5: Results on the smallNORB dataset with varying amount of iterations and comparision of the best models.

Based on these results it is safe to assume that the capsule network is a superior method for pure shape recognition tasks compared to a traditional CNN with a much lower amount of parameters. This capsule network also beats the capsule network proposed in the paper [SFE17] which achieved a test error rate of 2.7%. Also it shows that too many iterations of the routing algorithm hurt the systems performance.

5.2 Generalization to novel viewpoints on smallNORB

To better test the generalization capacity of the capsule network it is only trained on a small subset of different azimuths and elevations of the training data and then tested on the unfamiliar viewpoints. The baseline CNN again is trained in exactly the same way. Since the capsule network surpassed the best accuracy on familiar viewpoints by a big margin which would make a comparision very hard the capsule net is only trained to match the accuracy of

the baseline CNN. The results can be seen in figure 6. This should show how good the model learns the internal structure of the objects it should detect very similar to how a human can generalize from a picture seen to many different objects of the same class in different viewing positions which shows that he understands the arrangement of the parts it consists of and has a model of the object in his head.

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

Figure 6: Results of the generalization to novel viewpoints test on the small-NORB dataset.

Since the gap in performance from familiar to novel viewpoints is much lower for the capsule network it is safe to assume that it learns more about the internal structure of the object and is able to generalize better which is also supported by the fact that when fully trained on the smaller dataset surpasses the performance of the CNN by a significant margin.

5.3 Results on other datasets

The very familiar NORB dataset which has relatively natural backgrounds but still are grey images the capsule net achieves a 2.6% test error rate which also beats the current state of the art with 2.7% by a very small margin. When applied to MNIST it has a 0.44% teat error rate which is worse than the capsule net proposed by Sabour et al.[1] which has 0.25% but it was not trained regularized with a reconstruction loss which means the accuracy could get improved with this technique. On the dataset Cifar10 the capsule network achieved a test error rate of 11.9% which is relatively close to the test error rates of the first CNN which where applied to this dataset but nonetheless it is clearly not competitive with modern CNNs performance (with roughly $3-4\%$. This leads to one of the current drawbacks of the capsule network since it apparently does not handle background and too much information which has to be filtered out in an effective manner.

[1] SFE17.

6 Capacity of the architecture

The capsule architecture right now has a few problems which keep it from widespread application. The biggest disadvantage is the computational very expensive routing algorithm between the capsules which makes training these models very slow on the current generation of computational systems. Also the method on how to compute the primary capsules is very important since it is mandatory that they learn a good representation for the entity the represent so they need to be computed with good features that maybe even filter out background to a certain degree to make up for the problem current capsule networks have with it. Also they cannot detect two objects of the same class very near to each other. Also they are difficult to scale to datasets with more classes and higher dimensions due to the fact that capsules represent entities and therefore for very different classes the model has to be potentially the be very wide with a lot of different capsule types to construct very different classes out of these entities.

Nonetheless the first results on this structure are very promising and it has many advantages over a traditional CNN. Firstly it is for a human easier to understand how the model makes the prediction since its activations are very similar to a parse tree which is interpretable for a human and they can even be interpreted as saliency for a specific region since the capsules have a receptive field. Also the output of the pose for a capsule with a high activation is similar to the feature space of an autoencoder as shown in the paper [SFE17]. The training of a capsule network requires less training data since it learns internal structures faster and thereby can better generalize to unseen data which is also supported by the fact of a smaller amount of parameters which lowers the risk of over fitting.

In recent papers there was made a great progress to make capsule networks more efficient of effective for different tasks. For example was in the paper [**Sai18**] instead of standard convolutional layers before the capsules a dense layer net architecture implemented which boosted the performance of the system significantly on CIFAR10. They also added a skip connection like structure by having many layers of primary capsules. Thereby the feature learned in the primary capsules were more complex than from a standard convolutional layer.

Another interesting paper [**Bah18**] did use SVD instead of an iterative routing procedure which made the model less sensitive to initialization and hyperparameters as well as speeding up the convergence of the model.

In the paper [Ayu18] they achieved good results with a GAN that has a capsule network discriminator that only differentiates between synthesized and real images on MNIST with promising results.

And lastly a U-Net like structure for object segmentation was created based on capsule where they introduced deconvolutional capsules in the paper [**Rod18**]. They achieved state of the art results on the LUNA16 subset of the LIDC-IDRI database which consists of medical image data with a 512×512 input which means that their model is a first step towards capsule networks working with larger inputs.

7 Sources

References

[SFE17] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic Routing Between Capsules". In: *CoRR* abs/1710.09829 (2017). arXiv: 1710.09829. URL: https://arxiv.org/abs/1710.09829.

[Ayu18] et. AL Ayush Jaiswal Wael AbdAlmageed. "CapsuleGAN: Generative Adversarial Capsule Network". In: (2018). arXiv: 1802.06167. URL: https://arxiv.org/abs/1802.06167.

[Bah18] Mohammad Taha Bahadori. "Spectral Capsule Networks". In: 2018. URL: https://openreview.net/forum?id=HJuMvYPaM.

[HSF18] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. "Matrix capsules with EM routing". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=HJWLfGWRb.

[Rod18] Ulas Bagci Rodney LaLonde. "Capsules for Object Segmentation". In: (2018). arXiv: 1804.04241. URL: https://arxiv.org/abs/1804.04241.

[Sai18] et. AL Sai Samarth R Phaye Apoorva Sikka. "Dense and Diverse Capsule Networks: Making the Capsules Learn Better". In: (2018). arXiv: 1805.04001. URL: https://arxiv.org/abs/1805.04001.