An overview over Capsule Networks

Luca Alessandro Dombetzki
Advisor: Marton Kajo
Seminar Innovative Internet Technologies and Mobile Communications
Chair of Network Architectures and Services
Department of Informatics, Technical University of Munich
Email: luca.dombetzki@tum.de

ABSTRACT

Hinton et. al recently published the paper "Dynamic Routing Between Capsules" [20], proposing a novel neural network architecture. This Capsule Network (CapsNet) outperforms state-of-the-art Convolutional Neural Networks on simple challenges like MNIST [13], MultiMNIST [20] or smallNORB [6]. In this paper, we describe multiple aspects of the current research in Capsule Networks. This includes explaining the shortcomings of CNNs, the idea and architecture of Capsule Networks and the evaluation on multiple challenges. Furthermore, we give an overview of current research, improvements and real world applications, as well as advantages and disadvantages of the CapsNet.

Keywords

capsule networks, overview, computer vision, convolutional neural networks, pooling

1. INTRODUCTION

In 1986 Geoffrey E. Hinton revolutionized artificial neural networks with the use of the backpropagation algorithm. Since then artificial intelligence has made a big leap forwards, especially in computer vision. Deep Learning gained in popularity, when deep convolutional networks performed extraordinarily well on the ImageNet challenge in 2012 [10]. It has since been a very active field of research, with companies like Google and Microsoft being dedicated to it. This brought forth ideas like Inception [23] and Residual blocks [5], boosting the performance of CNNs. However, all of these advancements build upon the basic structure of a CNN.

Based on his research in human vision, Geoffrey E. Hinton stated that there is something fundamentally wrong with CNNs [7]. By trying to replicate the human visual cortex, he came up with the idea of a capsule as a group of neurons. In "Dynamic routing between capsules" [20] Hinton et. al developed a first working implementation, proving this theory. In computer graphics, a scene is built by putting known parts into relation, forming a more complex object. Inverse graphics does the exact opposite, the scene is deconstructed into parts and their relationships. The main goal of the Capsule Network is to be capable of performing inverse graphics [7]. To achieve this, Hinton proposed to encode the idea of an entity inside a neural network, a capsule [7].

In the following, we first explain the basic structure of a Convolutional Neural Network and its possible shortcomings in Section 2. In Section 3.1 and 3.2, we describe the architecture of a Capsule Network by comparing it to a general CNN. Furthermore, we explain the idea and implementation of routing-by-agreement algorithm, as well as the loss functions and the training of the network (Section 3.3-3.4). Section 3.5 shows the network's performance on multiple challenges, including an overview over novel matrix capsules in Section 3.6. After summarizing the main advantages and disadvantages in Section 3.7 and 3.8, we look at improvements to the CapsNet (Section 3.9). Before drawing a conclusion in Section 4, we outline some real world use cases of capsule networks in Section 3.10.

2. CONVOLUTIONAL NEURAL NETWORKS

2.1 Basic CNN architecture

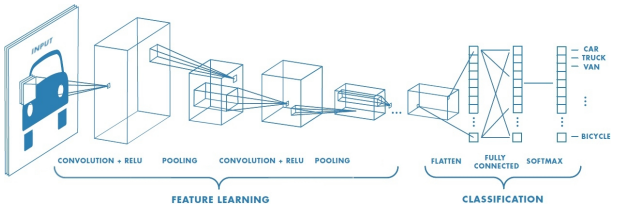


Figure 1: Basic architecture of a CNN; figure from [15]

Fig. 1 shows the basic architecture of a Convolutional Neural Network. The network first extracts learned features, which are then fed through a fully connected neural network, that produces a classification. The network can learn features by chaining together convolutional blocks. Such a block consists of a convolutional layer, an activation function and a pooling layer. The convolutional layer learns multiple simple features, also called kernels. To learn more complex, non-linear, problems, the output is fed through a non-linear activation function (e.g. ReLU). To connect the blocks together, the

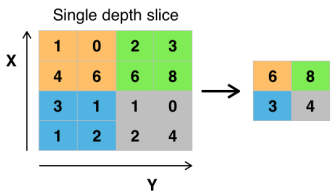


Figure 2: Max pooling example; figure from [2]

outputs of the previous block need to be routed to the next block's inputs. The most commonly used routing algorithm

doi: 10.2313/NET-2018-11-1_12

is pooling. Max pooling can be seen in Fig. 2 To improve the classification significantly, it discards unimportant activations and only propagates the most prominent ones. This allows the next convolutional layer to work only on "important" data and makes the classifier robust against small transformations in the input data.

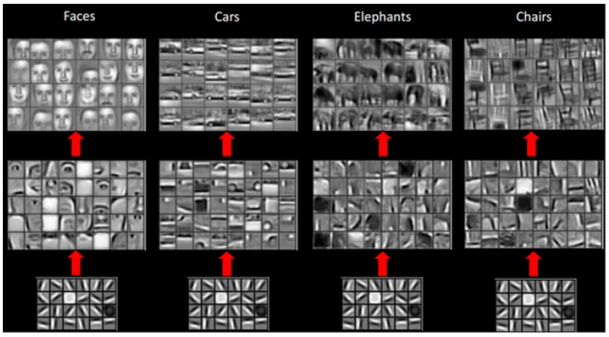


Figure 3: Feature detections of a CNN from [14]

Adding more blocks allows later blocks to extract features from results of the previous block. Thereby the network can learn more and more complex features. Like in Fig. 3 this means that the first block of a CNN might learn edges, the next block learns parts of an object parts and ultimately the last block learns complete objects like a dog, a plane, etc. With enough pooling layers, the network can become location invariant. Hence a car in the top left corner of the image is detected as well as one in the lower right.

2.2 Problems with pooling for routing

In his talk "What is wrong with convolutional nets?"[7], Hinton stresses his belief in convolution, however he brought forth four main arguments against using pooling for routing. They are explained in the following.

2.2.1 *Pooling is unnatural*

Hinton states that pooling is a "bad fit to the psychology of shape perception". Human vision detects the object instantaneously. Based on the information this object holds, we route it to the area in the brain that best handles that information. Max pooling on the other hand routes the most active information to all subsequent neurons. This is unnatural and prevents the network from learning small details.

2.2.2 *Invariance vs. equivariance*

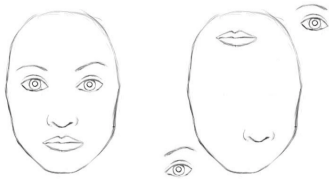


Figure 4: Both images are being classified as "face" by a CNN; figure from [9]

CNNs try to make the neural activities invariant to small changes in the viewpoint, pooling them together [7]. This is helpful for classification tasks, since the label should be the same, no matter where the object is (spacial invariance). However changes in viewpoint should ideally lead to changes in neural activities [7] (spacial equivariance). This is especially important for segmentation and detection tasks, but is also needed in classification. Fig. 4 shows that CNNs can only detect features, but cannot relate them. This means that, in an extreme case, both images are perfect representations of a face for a CNN.

2.2.3 *Not using the linear structure of vision*

A single transformation, such as rotation, can change a significant number of pixels in the image. Thereby the viewpoint is the largest source of variance in images [7]. In computer graphics, composing the scene of multiple parts is a linear problem, since we know all the relations between the parts. Without using this linear structure in computer vision, the problem of detecting an object is not linear anymore, but far more complex. As a result, a normal CNN has to be exponentially increased in size and trained with a similarly exponential amount of data [20].

2.2.4 *Dynamic instead of static routing*

Pooling collects the most prominent activations, but transports them to the same neurons of the following layer. This is like broadcasting an important information. Thereby, if the input image is translated, a completely different set of neurons is responsible for handling different kinds of activations. Our human vision however is smart enough to understand that the image is translated and activates the same neurons. This happens at runtime, hence is dynamic, and not statically preconnected like pooling. This is like letting the neurons from the next layer choose, what is most interesting to them. In a nutshell, pooling chooses the best input, while dynamic routing chooses the best worker neuron.

Nonetheless, pooling still works, as long as the network is big enough and trained with enough data, that neurons of the following layers can handle every input type. As a result, CNNs perform very well, when classifying images with only one kind of object. But especially in detection and segmentation, they lack in performance, since the information of multiple objects has to stay separated.

2.3 Solution

To solve these shortcomings, Hinton proposes to propagate not only the probability of a feature's existence, but also the spacial relationships, i.e. the pose of the feature[7]. By adopting biological research he tries to tackle all of the problems stated above. His implementation of this solution is known as Capsule Networks.

3. CAPSULE NETWORKS

Hinton's basic idea was to create a neural network capable of inverse graphics. In other words the network should be able to deconstruct a scene into co-related parts. To achieve this, the architecture of a neural network needs to be changed to reflect the idea of an entity. Every entity gets its own part of the network, encapsulating a number of neurons. This entity is called a capsule.

3.1 The capsule

A normal layer of neurons will be divided into many capsules, which in turn contain the neurons [20] (see Sec. 3.2).

doi: 10.2313/NET-2018-11-1_12

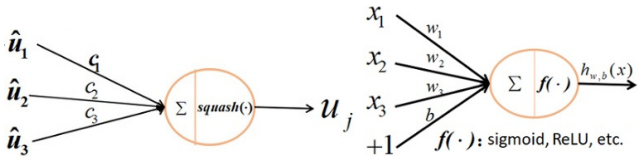


Figure 5: An capsule and neuron in comparison [8]

Therefore a capsule is a wrapper around a dedicated group of neurons. Fig. 5 shows a simplified comparison between a capsule and a neuron. A neuron computes a scalar value from a list of scalar values. Since a capsule essentially wraps a group of neurons, it computes a vector from a list of input vectors. It is now able to encode entity parameters like location, skew, etc. [20]. However this also means, that it does not represent the probability for the existence of a feature anymore. Instead the length of the vector can be used as the probability for feature existence, while not losing the important pose information. Furthermore this also enables the network to learn the parameters by itself, removing the need for crafting them by hand. This means that a n-dimensional (nD) Capsule can learn n parameters and outputs a n-dimensional vector.

For the output vector to model a probability, it's length has to stay between 0 and 1. Normal activation functions like ReLU only work on scalar values, hence a novel non-linear squashing function Eq. 1 was introduced.

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||} \tag{1}$$

To understand how the inputs of the capsule are combined to s_j, we will now look into the architecture of the Capsule Network presented in [20].

3.2 Architecture

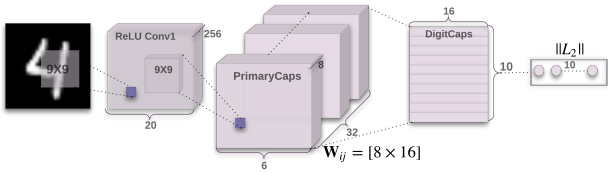


Figure 6: Capsule Network Architecture as described in [20]

The architecture in Fig. 6 shows a Capsule Network for classifying images from the MNIST dataset [13]. An input image is transformed into 10 scalar values, representing the probability for each number 0-9.

3.2.1 Conv1

The first layer applies a normal convolution with a $9 \times 9 \times 1$ kernel to the $28 \times 28 \times 1$ image over 256 channels.

3.2.2 PrimaryCaps

The next layer consists of 32 channels, each channel a 6×6 grid of so called primary capsules. They serve as a transition between the scalar values of the convolution to 8D vector outputs. The primary capsules can be seen as another convolutional layer with a $9 \times 9 \times 256$ kernel, just with squashing as their activation function. This means that the weights, i.e. the kernel, are shared between all capsules in each 6×6 grid.

3.2.3 DigitCaps

Following is the DigitCaps layer, fully connected to the primary capsules. These are now pure 16D capsules getting their inputs from the previous primary capsules. The weight matrix W_{ij} transforms the 8D output of primary capsule i to a 16D vector as input for digit capsule j ($\hat{u}_{j|i}$) Eq. 2.

$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \ , \qquad \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij} \mathbf{u}_i \tag{2}$$

Therefore each digit capsule has a weighted sum of $32 \times 6 \times 6$ 8D vectors as input (s_j). Instead of using pooling, the new technique of routing-by-agreement is used to focus on the most important inputs. This is discussed in section 3.3.

3.2.4 Class predictions

The 10 16D vectors correspond to the numbers 0-9 (10 classes). Because of the squashing function, the length of each of the vectors can be directly used as a probability for each class. Hence there are no more fully connected layers needed for classification (compare to Fig. 1).

3.3 Routing-by-agreement

Routing-by-agreement is a novel dynamic routing technique. In contrast to pooling, the routing happens at runtime. The goal of this technique is to redirect previous capsule outputs to a following capsule where it agrees with other inputs. In the scope of inverse graphics this can be compared to routing a detected nose to the face-capsule and not the car capsule. A detected nose, eye and mouth agree together in the face-capsule, while the nose would not agree with a wheel and door in the car capsule.

This works because of "coincidence filtering". In a high dimensional space - in this case the parameter dimension - it is very unlikely for agreements to lie close to another. So a cluster of agreements can not be, in a probabilistic way, a coincidence.

As an implementation, Hinton et. al chose an iterative clustering algorithm, see Alg. 1.

Algorithm 1 Routing algorithm. (from [20])

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}$, r, l)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
return $\mathbf{v}_j$

In simple terms, the algorithm finds the mean vector of the cluster (s_j), and weighs all inputs based on their distance to this mean (b_{ij}) and normalizes the weights with the "routing softmax" (c_i) 3.

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{3}$$

doi: 10.2313/NET-2018-11-1_12

The number of iterations r is a hyperparameter for the network and is empirically found to produce the best results around 3 to 5 iterations. When the routing is finished the input vectors now have an associated weight c_{ij} [20].

3.4 Training

A Capsule Network produces vectors as outputs. Hinton et. al proposed multiple loss functions to be used at the same time for training [20]. Reconstruction loss is used to train the capsule parameters, while margin loss is used to optimize digit classification. Both are explained below.

3.4.1 Reconstruction loss

This loss is normally used to train a neural network unsupervised. It is mostly used in autoencoders [7] to force the network to find a different representation of the data. Therefore it makes sense to train Capsule Networks the same way, to force the capsules to find adequate pose parameters.

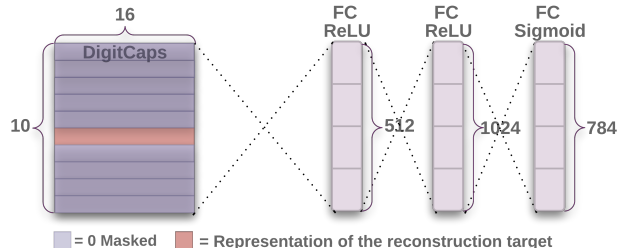


Figure 7: Decoder network, building on top of the DigitCaps layer, as described in [20]

To implement this loss, the previous architecture of Fig. 6 is extended with a decoder network (Fig. 7) to form the typical architecture of an autoencoder. The decoder network consists of two fully connected layers with ReLU activation and one sigmoid activated layer. The output of 784 values represents the pixel intensities of a 28×28 image, the same size as the images from the MNIST dataset.

As the actual reconstruction loss Hinton et. al [20] used the euclidean distance between the actual image and the sigmoid layer output. Additionally, the DigitCaps layer is masked to exactly one capsule as input for the decoder. The masked capsule corresponds to the ground thruth label, e.g. the numbers 0 to 9 as in [20]. This forces the DigitCaps to encode the actual digits [20].

3.4.2 Margin loss

$$L_k = T_k \ \max(0, m^+ - ||\mathbf{v}_k||)^2 + \lambda\,(1 - T_k)\ \max(0, ||\mathbf{v}_k|| - m^-)^2 \quad (4)$$

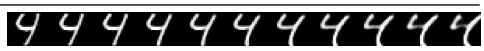
Eq. 4 shows the mathematical definition of the margin loss. In [20], the constants were chosen as $m^+ = 0.9$ and $m^- = 0.1$. T_k acts as a switch between two cases. $T_k = 1$ if a digit of class k is present $T_k = 0$ if not. This loss ensures that the output vectors of the digit capsules are at least m^+ long, when the class is detected, and at most m^- long when that class is not detected. $\lambda = 0.5$ is used to prevent the loss from shrinking all vectors in the initial learning phase.

This loss function is applied to each digit capsule individually. The total loss is simply the sum of the losses of all digit capsules [20].

Table 1: CapsNet classification accuracy. The MNIST average and standard deviation results are reported from 3 trials. Methods used were B=Baseline=CNN and C=CapsNet. [20]

Method	Routing Iterations	Rec. Loss	MNIST (%)	MultiMNIST (%)
B	-	-	0.39	8.1
C	1	no	$0.34_{\pm 0.032}$	-
C	1	yes	$0.29_{\pm 0.011}$	7.5
C	3	no	$0.35_{\pm 0.036}$	-
C	3	yes	$\mathbf{0.25}_{\pm 0.005}$	**5.2**

Figure 8: Resulting reconstructions if one of the 16D in a digit capsule is alterered marginally [20].

Scale and thickness	
Localized part	
Stroke thickness	
Localized skew	
Width and translation	
Localized part	

3.4.3 Hyperparameters

As stated in Sec. 3.4.2, the margin loss parameters were defined in [20] as $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$.

The total loss to be optimized is a weighted combination of both losses. To prevent the reconstruction loss from dominating the margin loss, the reconstruction loss is scaled down by 0.0005 [20].

Hinton et. al [20] experimented with the number of iterations in the routing algorithm 1. They empirically found 3 iterations, combined with the reconstruction loss, to produce the best results. This can be seen in table 1.

3.5 Evaluation

For evaluation, Hinton et. al generated a new dataset called MultiMNIST. For each sample two digits from the MNIST dataset were overlapped by 80% [20].

The proposed network has been tested on the MNIST and MultiMNIST dataset. The results are depicted in table 1. This shows that Capsule Networks are able to outperform the baseline CNN in both challenges, achieving significantly better results in the MultiMNIST dataset.

3.5.1 Representation of the pose parameters

To prove their goal of encoding transformation parameters in the capsule, Hinton et. al fed a capsule prediction to the decoder network (see 3.4.1. Fig. 8 displays how small changes to some of the 16D parameters affect the reconstructions. The results suggest that their goal has been reached.

When decoding the two predicted capsules, this leads to very accurate reconstructions, see Fig. 9.

doi: 10.2313/NET-2018-11-1_12

Figure 9: Correct reconstructions (lower image) of the CapsNet on MultiMNIST test dataset (upper image). L:(l_1, l_2) represents the label for the two digits in the image. R:(r_1, r_2) represents the two digits used for reconstruction. [20]

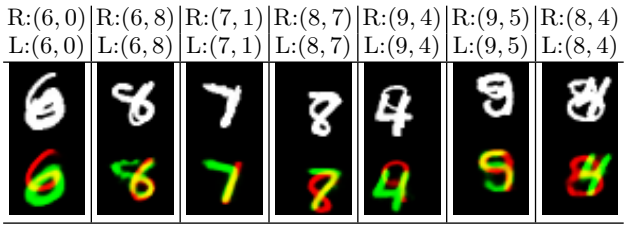
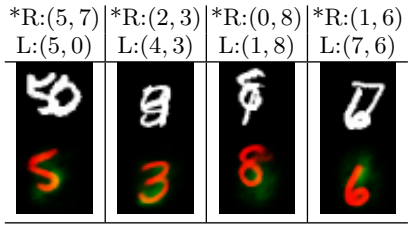


Figure 10: Capsule Network forced to reconstruct on false lables (marked with *) on the MultiMNIST dataset. [20].



3.5.2 *Smart reconstructions*

As another experiment they forced the decoder network to reconstruct non-predicted capsules (Fig. 10). It can be observed, that the CapsNet only reconstructed digits that it also detected. Hinton et. al proposed [20], that the model is not just finding the best fit for all the digits in the image. Instead it also includes the ones that do not exist. Hinton et. al suggest, that "in case of $(8, 1)$ the loop of 8 has not triggered 0 because it is already accounted for by 8. Therefore it will not assign one pixel to two digits if one of them does not have any other support"[20].

3.6 Matrix Capsules with EM routing

Hinton et. al published another paper, currently under open review, called "Matrix Capsules with EM routing" [6]. They propose to use a EM [6] algorithm instead of the current routing algorithm 1 from [20]. Additionally they changed the capsules to use a 4×4 pose matrix instead of a vector. Such a matrix is used in computer graphics to compute the scene, like it would be seen through a virtual camera. This is called the viewport. Since the network is able to learn this matrix, it is able to become viewport invariant [6].

They tested this new network on smallNORB dataset (Fig. 11) and outperformed the current state of the art CNN by 45%, reducing the error percentage from 2.56% to 1.4% [6]. Furthermore, they conducted an experiment, training the network only on specific viewpoints and testing it on unseen viewpoints [6]. Both networks were trained to the same error of 3.7% on seen viewpoints. While the baseline CNN's error increased to 20% on unseen viewpoints, the Capsule Network still achieved 13.5%. Based on these results, CapsNets seem to be able to generalize better than CNNs, being able to adapt to 3D viewports in 2D images.

Figure 11: Example images from the smallNORB dataset [6]. Multiple object classes in different viewports.

3.7 Advantages of Capsule Networks

Capsule Networks show multiple advantages compared to classic Convolutional Neural Networks. The following list with explanations is adapted and extended from [4].

3.7.1 *Viewpoint invariance*

The use of parameter vectors, or pose matrices [6], allows Capsule Networks to recognize objects regardless of the viewpoint from which they are viewed. Furthermore Capsule Networks are moderately robust to small affine transformations of the data [20].

3.7.2 *Fewer parameters*

The connections between layers require fewer parameters, since only neuron groups are fully connected, not the neurons themselves. The CNN trained for MultiMNIST consisted of 24.56M parameters, while the CapsNet only needed 11.36M parameters [20]. This is close to half as many as before. Matrix capsules with EM-routing required even less [6]. This also means that the model can generalize better.

3.7.3 *Better generalization to new viewpoints*

CNNs memorize, that an object can be viewed from different viewpoints. This requires the network to "see" all different transformations possible. Capsule Networks however generalize better to new viewpoints, because parameter information of a capsule can capture these viewpoints as mere linear transformations [7]. Therefore CapsNets are not as prone to misclassification of unseen data, as shown in Sec. 3.6.

3.7.4 *Defense against white-box adversarial attacks*

Common attacks on CNNs use the Fast Gradient Sign Method. It evaluates the gradient of each pixel against the loss of the network. The pixels are then changed marginally to maximize the loss without distorting the original image. This method can drop the accuracy of CNNs to below 20%. Capsule Networks however maintain an accuracy over 70% [4].

3.7.5 *Validatable*

A problem for industry usage of CNNs is their black box behaviour. It is neither predictable how a CNN will perform on new data, nor can its performance be properly analyzed and understood. Because Capsule Networks build upon the concept of inverse graphics, the network's reasoning can be explained considerably better than CNNs. Shahroudnejad

doi: 10.2313/NET-2018-11-1_12

et. al [21] proposed an explainability method building naturally upon capsules and their structure. This suggests, that Capsule Networks are superior to CNNs in validatability.

3.7.6 Less amount of training data

Through unsupervised learning and the dynamic routing procedure, Capsule Networks converge in fewer iterations than CNNs. Furthermore, CNNs need exponentially more training data to understand affine transformations [20].

3.8 Challenges for Capsule Networks

The CapsNet's early development stage, brings not only common problems with it, but also reveals unique challenges. In the following both kinds are listed in more detail.

3.8.1 Scalability to complex data

Hinton et. al [20] evaluated the network experimentally on the CIFAR10 dataset, failing to perform as good as current CNNs. The results were comparable to the first CNNs tackling the challenge. Matrix capsules and other discussed approaches in section 3.9 try to tackle this problem but are still far from performing on the ImageNet challenge.

3.8.2 Capsules need to model everything

As described in [20], capsules share this problem with generative models. The network tries to account for everything in the image. This also means that it performs better, if it can model the clutter like background noise, instead of having an extra "not-classifiable" category. LaLonde et. al [12] try to solve this issue by reconstructing not the whole image, but only the segmentation. This removes the need for the network to model the background and allows it to concentrate only on the active class. For solving their challenge of segmenting medical images, this approach shows promising results.

3.8.3 Structure forcing representation of entities

The concept of entities was introduced to Capsule Networks to aid in computer vision and perform inverse graphics. This network architecture could therefore prevent the network from being applied to non-vision areas. However this has not yet been investigated thoroughly.

3.8.4 Loss functions

Since the network produces vector or matrix outputs, existing loss functions cannot be simply reused. However, they can often be adapted and sometimes leverage the additional data, as can be seen in the reconstruction loss. Still, using the CapsNet on a new dataset will often require a new loss function as well.

3.8.5 Crowding

Human Vision suffers from the "crowding" problem [16]. We cannot distinguish objects, when they are very close together. This can also be observed in Capsule Networks [20], since this concept was used to model the capsule in the first place [20]. Capsules are based upon the idea, that in each location in the image is at most one instance of the type of entity that the capsule represents [20]. While this enables capsules to efficiently encode the representation of the entity [20], this could also present itself as a problem for specific use cases.

3.8.6 Unoptimized implementation

The original, but not sole, implementation of the Capsule Network can be found at [22]. It shows, that Capsule Networks present multiple challenges for current deep learning frameworks. Neural networks are often represented as a graph. Therefore, the number of routing iterations must be defined empirically [20], allowing for the $for\ loop$ to be unfolded beforehand and chained r times together in the graph. Another problem for training neural networks is that the routing algorithm is dynamic and not easy to parallelize, preventing GPUs from leveraging their full computing power. Nevertheless it is very likely that these deep learning frameworks will adapt with time.

3.9 Further improvements

The performance and scalability of the original Capsule Network has been analyzed by Xi et. al in [26]. They came to the conclusion that, apart from minor improvements, the Capsule Network does not work very well on complex data. Capsule Networks initially showed a lot of problems similar to CNNs before 2012. These were problems like unoptimized algorithms, vanishing gradients, etc. This inspired further research in this area, producing multiple different approaches for a new routing algorithm, new losses, or even complete restructuring of the architecture. Some of these publications are presented below.

Phaye et. al [17] investigated using DenseNet-like skip-connections to increase the performance of the Capsule Network. The resulting DCNet was furthermore restructured in a hierarchical manner (DCNet++), outperforming the original CapsNet.

Rawlinson et. al [18] also proposed a change in the Capsule Net architecture. By removing the margin loss (section 3.4.2) and introducing sparsity to the capsules, the network achieved similar results and generalized very well. Furthermore the network could now be trained completely unsupervised.

Bahadori et. al [3] developed a novel routing procedure. The algorithm is based on the eigen-decomposition of the votes and converges faster than EM-routing, described in [6]. The connection between the proposed S-Capsules and EM-Capsules is analogous to the connection between Gaussian Mixture Models and Principal Component Analysis. This analogy suggests why S-Capsules are more robust during the training [3].

Wang et. al [24] optimized the routing algorithm by leveraging Kullback-Leibler divergence [11] in regularization. Their proposed routing outperforms the original routing procedure in accuracy 1.

3.10 Use in real world applications

Capsule Networks are currently evaluated on challenging tasks and in difficult environment, where typical CNNs fail to produce acceptable results. In the following, three example cases are presented.

Afshar et. al [1] use Capsule Networks for brain tumor type classification. Capsule Networks require less training data

doi: 10.2313/NET-2018-11-1_12

than CNNs. This is very important in the medical environment, resulting in CapsNets being the superior network for this task.

Wang et. al [25] employ the capsule concept in a Recurrent Neural Network. Thereby they achieved state-of-the-art performance in sentiment analysis tasks.

LaLonde et. al [12] designed a new network architecture similar to U-Nets [19]. Apart from outperforming the baseline U-Net model on large 512×512 images, they reduced the parameters needed by 95.4%.

4. CONCLUSION

Hinton et. al propose with Capsule Networks a completely new way for Convolutional Neural Networks to analyze data. The routing-by-agreement algorithm 1 tackles the problems of pooling (2.2) [7]. Together with the concept of capsules, this enables networks to be viewpoint invariant and robust against affine transformations. This eliminates the main reason for huge amounts of data being needed in CNN training. Overall the new architecture holds many advantages over the typical CNN. Current research shows that, with some alterations, Capsule Networks are able to perform even in complex scenarios [1]. However, Capsule Networks have to be developed further to outperform, or even replace CNNs in real world scenarios, especially when data is not a problem.

5. REFERENCES

[1] P. Afshar, A. Mohammadi, and K. N. Plataniotis. Brain tumor type classification via capsule networks. *CoRR*, abs/1802.10200, 2018.

[2] Aphex34. Convolutional neural network - max pooling. `https://en.wikipedia.org/wiki/Convolutional_neural_network#Max_pooling_shape`; last accessed on 2018/06/14.

[3] M. T. Bahadori. Spectral capsule networks. 2018.

[4] S. Garg. Demystifying "matrix capsules with em routing.". `https://towardsdatascience.com/demystifying-matrix-capsules-with-em-routing-part-1-overview-2126133a8457`; last accessed on 2018/06/14.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] G. Hinton, S. Sabour, and N. Frosst. Matrix capsules with em routing. 2018.

[7] G. E. Hinton. What is wrong with convolutional neural nets? Talk recorded on youtube, `https://youtu.be/rTawFwUvnLE`; last accessed on 2018/06/14.

[8] K.-Y. Ho. Capsules: Alternative to pooling. `https://datawarrior.wordpress.com/2017/11/14/capsules-alternative-to-pooling/`; last accessed on 2018/08/18.

[9] T. Kothari. Uncovering the intuition behind capsule networks and inverse graphics. `https://hackernoon.com/uncovering-the-intuition-behind-capsule-networks-and-inverse-graphics-part-i-7412d121798d`; last accessed on 2018/06/14.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[12] R. LaLonde and U. Bagci. Capsules for Object Segmentation. *ArXiv e-prints*, Apr. 2018.

[13] Y. LeCun, C. Cortes, and C. Burges. Mnist dataset, 1998.

[14] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[15] Mathworks. Convolutional neural network. `https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html`; last accessed on 2018/06/14.

[16] D. G. Pelli. Crowding: A cortical constraint on object recognition. *Current opinion in neurobiology*, 18(4):445–451, 2008.

[17] S. S. R. Phaye, A. Sikka, A. Dhall, and D. Bathula. Dense and diverse capsule networks: Making the capsules learn better. *arXiv preprint arXiv:1805.04001*, 2018.

[18] D. Rawlinson, A. Ahmed, and G. Kowadlo. Sparse unsupervised capsules generalize better. *CoRR*, abs/1804.06094, 2018.

[19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[20] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869, 2017.

[21] A. Shahroudnejad, A. Mohammadi, and K. N. Plataniotis. Improved explainability of capsule networks: Relevance path by agreement. *CoRR*, abs/1802.10204, 2018.

[22] soskek. Capsnets tensorflow implementation. `https://github.com/soskek/dynamic_routing_between_capsules`; last accessed on 2018/06/14.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[24] D. Wang and Q. Liu. An optimization view on dynamic routing between capsules. 2018.

[25] Y. Wang, A. Sun, J. Han, Y. Liu, and X. 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.

[26] E. Xi, S. Bing, and Y. Jin. Capsule Network Performance on Complex Data. *ArXiv e-prints*, Dec. 2017.

doi: 10.2313/NET-2018-11-1_12