Capsule Network versus Convolutional Neural Network in Image Classification

Comparative Analysis

Ewa Juralewicz[1][0000−1111−2222−3333] and Urszula Markowska-Kaczmar[2][0000−0001−7606−3057]

[1] Wroclaw University of Science and Technology,Wroclaw, Poland
eva.juralewicz@gmail.com
[2] Wroclaw University of Science and Technology,Wroclaw, Poland
urszula.markowska-kaczmar@pwr.edu.pl

Abstract. Many concepts behind Capsule Networks cannot be proved due to limited research, performed so far. In the paper, we compare the CapsNet architecture with the most common implementations of convolutional networks (CNNs) for image classification. We also introduced Convolutional CapsNet - a network that mimics the original CapsNet architecture but remains a pure CNN - and compare it against CapsNet. The networks are tested using popular benchmark image data sets and additional test sets, specifically generated for the task. We show that for a group of data sets, usage of CapsNet-specific elements influences the network performance. Moreover, we indicate that the use of Capsule Network and CNN may be highly dependent on the particular data set in image classification.

Keywords: Capsule Network · Convolutional Network · image classification · comparative analysis

1 Introduction

Over past ten years, Deep Learning (DL) has introduced a huge progress in computer vision. This great success is a result of convolutional neural networks (CNNs) application [10]. They make predictions by checking if certain features are present in the image or not. They do not posses ability to check spatial relationship between features. The weakness of CNN is its need for a vast amount of data to train.

A new attractive neural network has been proposed by Hinton [6] in response to the convolutional neural network drawbacks. He indicated the pooling operation used to shrink the size and computation requirements of the network as the main reasons for the poor functionality of CNNs. To solve this problem Capsule Networks and dynamic routing algorithms have been proposed. Capsule Networks are characterized by translational equivariance instead of translational invariance as CNNs. This property enables them to generalize to a higher degree

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

from different viewpoints with less training data. Many successful alternatives have been proposed for the routing process [6, 1, 9] to increase representation interpretability and processing time, which proves there are possibilities for Capsule Networks to improve in this area. The experiments conducted in [14] provide an estimate on CapsNet capabilities while taking training time needed to achieve relevant results into account. The experiment setup, however, leaves much space for further improvement. The generalization capability, related to data efficiency, has also been analyzed by training a network to recognize a new category based on a network trained on a subset of initial categories [2]. By concept, capsules hold a more complex entity representation compared to a single neuron and allow for more significant viewpoint variations. This leads to a hypothesis that Capsules need less training data than CNNs to achieve a similar performance quality, which has been confronted in [13]. In the comparison, apart from previously mentioned CapsNet architecture presented in [12] (CapsNet I), a variant with EM (*Expectation-Maximization*) routing algorithm, which was introduced in [6], is compared (CapsNet II). The new version of routing algorithm based on EM is proposed in [3].

Most cited above publications aim to provide an improvement rather than a comparison for the task they consider. This way, it is impossible to directly move observations made in such experiments to contemplate comparative performance, as the experimental setup is not valid. Our work tries to fulfill this gap. In opposite to [8], we focus on a comparison of both specifically designed for image processing neural networks – CNN and Capsule Networks with comparable number of parameters. Neural Networks and Capsule Networks in areas and applications relevant to conceptual differences between the two in a highly comparable environment. Notably, the research is aimed to verify whether Capsule Networks introduce advances in the areas in which they are supposed to be superior to CNNs by intuition and following the core concept differences. In the experiments, we verify the influence of the routing process's utility, increased viewpoint variance coverage, and robustness of Capsule Network to randomly shuffled images about Convolutional Network. The comparison covers the training process of the networks, their features emerging directly from architecture design, and their performance under specified conditions. It allows an understanding of their performance better.

The paper consists of four sections. The next one presents details of compared neural models. Section 3 describes conducted experiments and analysis of obtained results. Conclusions finish the paper.

2 Methodology

In this section, we describe details of models evaluated in the experimental part.

2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) is especially suitable for image processing because of its structure and the way of information processing. A simple CNN

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

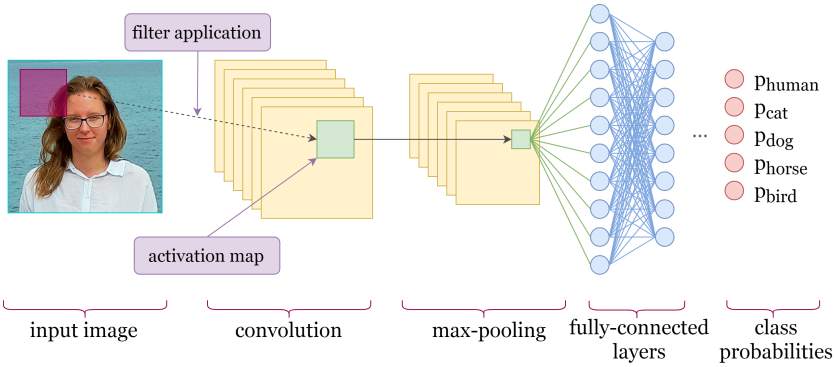


Fig. 1: Simple schema of processing information in CNNs with one convolutional layer, one pooling layer and two fully-connected hidden layers

model with one convolutional and one pooling layer is presented in Fig.1. It is composed of three different layer types: convolutional, pooling, and fully-connected. Convolutional layers are composed of several feature maps, which are two-dimensional matrices of neurons. Each feature map has its *convolutional filter* applied to the input. Convolutional filters (also called kernels) are applied locally sliding over its input. For each kernel position the convolution operation is performed. Pooling layers typically follow convolutional layers. This operation locally subsamples the output of the preceding convolutional layer. The most commonly used is max-pooling, which is performed by sliding a window of a specified width and height and extracting the current pool's maximum value. The fully connected layers end a processing pipeline in a convolutional network (Fig. 1). The preceding layer output is transmitted to the fully-connected layer after being flattened to form a vector.

Training a Convolutional Neural Network is made with the backpropagation algorithm based on minibatch SGD. It iteratively searches the set of weights W that minimizes the *loss function* for data D. For classification problems, the network is trained using the categorical cross-entropy loss function.

It is worth noting that the pooling operation significantly reduces the number of trainable parameters in the following layer allowing for minor input variation while preserving the level of neural activation. In this way, they enrich CNNs with a property referred to as *translational invariance*. It exists only in a limited scope. It makes it necessary for CNNs to be fed with augmented data in terms of rotation, scale, and varying perspective, causing a giant data volume needed for training, which covers as many viewpoints as possible.

There are many convolutional neural networks architectures proposed so far. Currently, the most common CNNs in use are: VGG-16 [14], DenseNet [7] and Residual Neural Network (ResNet) [4].

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

2.2 Capsule Network

A capsule network is a type of neural network in which the basic low-level node is a *capsule* [5]. Capsule's inputs and output are vectors. Each vector encodes a representation of an entity. The vector's direction indicates the pose of the object, (position and orientation in a specified coordinate system), and the vector's norm indicates the network confidence of this representation. Capsules take vectors $\underline{u}_i$ from a lower capsule layer as input. Then, they multiply them by weight matrices W_{ij} and a scalar *coupling coefficients* c_{ij}, where i is the number of a capsule in the lower layer, and j is the number of a capsule in a successive higher layer, as in eq. 1.

$$s_j = \sum_{i=1}^{N} W_{ij} \underline{u}_i c_{ij} \tag{1}$$

The multiplication vector result s_j is summed element-wise and normalised using a *squashing function*, producing an output vector v_j, (eq. 2).

$$v_j = squash(s_j) = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||} \tag{2}$$

The squashing function reduces the original vector length to the range of $[0, 1]$ while maintaining vector's orientation in space. It also brings a non-linearity to the result.The coupling coefficients are modified proportionally to the level of accordance a_{ij} between prediction vector $\underline{\hat{u}}_{j|i}$ and capsule output v_j. Accordance a_{ij} is simply a scalar value acquired by cosine similarity, which is a product of vectors $\underline{\hat{u}}_{j|i}$ and v_j (eq. 3).

$$a_{ij} = \underline{\hat{u}}_{j|i} v_j \tag{3}$$

The flow control is obtained by performing *dynamic routing*. Its version called routing-by-agreement is presented in [12]. Routing-by-agreement is an iterative process in which coupling coefficients c_{ij} are calculated. They are interpreted as the probability that a part from the lower-level capsule i contributes to the whole which capsule j should detect.

CapsNet is the first fully-trainable architecture following the ideas behind Capsule Networks. It consists of two parts - an **encoder** and a **decoder**. The encoder is the part that is responsible for building the vector representation of the input. The decoder's role is to achieve the input image's best reconstruction quality based on its representation vector from the encoder.

The encoder is visualised in Fig. 2. It is composed of 3 layers: a purely convolutional layer with ReLU activation function (ReLU Conv1), a PrimaryCaps layer, which is a combination of a convolutional and a capsule layer, and a DigitCaps layer - a pure capsule layer. The PrimaryCaps layer is very similar to the standard convolutional layer with an output extension to a vector and vector-wise application of the activation function. The DigitCaps layer is a capsule layer responsible for producing a vector representation for each of the categories available for the considered task (10 classes in MNIST). The routing operation is applied only between the PrimaryCaps and the DigitCaps layers. The weights

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

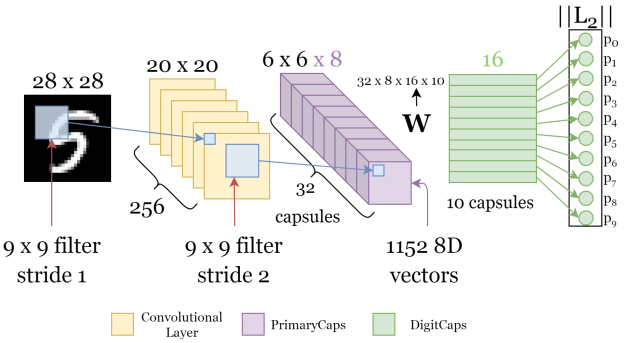


Fig. 2: Architecture of CapsNet encoder with an image from MNIST dataset

in other layers are modified exclusively during the backpropagation algorithm based on minibatch SGD and remain constant in the network's forward pass.

The decoder in CapsNet is presented in Fig. 3. The decoder's input is a 16-dimensional vector, which is the output from a capsule from the DigitCaps layer responsible for encoding the true label of the image. The remaining outputs are masked by inserting a value of 0. This masking mechanism and the reconstruction loss incorporation to the entire network's loss function force the separation of label responsibility between capsules in the last layer of the encoder.

The output from the last layer is interpreted as pixel intensities of the reconstruction image. The reconstruction loss is calculated as a distance between the input image and the reconstruction obtained from the capsule representation vector. The loss function for the network training L_c is calculated as in eq. 4.

$$L_c = L_e + \alpha L_d \tag{4}$$

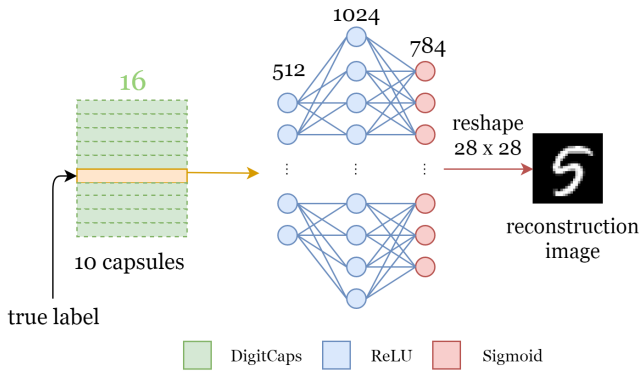


Fig. 3: Architecture of CapsNet decoder with an image from MNIST dataset

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

where L_e is a margin loss function for the classification task and L_d is the loss from the decoder. The classification task loss function is expressed by eq. 5:

$$L_e = \sum_{k=1}^{K} T_k max(0, m^+ - ||v_k||)^2 + \lambda(1 - T_k) max(0, ||v_k|| - m^-)^2 \quad (5)$$

where: K is the number of considered categories; T_k is a logic value – 1 when the correct label of the sample corresponds with capsule k and 0 – otherwise; λ down-weights the loss function value for categories other than the true label. It has a constant value of 0.5; v_k is the squashed vector representation output by capsule k; m^+ is a margin for positive classification outcome and is equal to 0.9, m^- is a margin for negative classification outcome and is equal to 0.1.

The component L_d in eq. 4 refers to the decoder training. It is based on the euclidean distance between the target input image and the output from the decoder. It is scaled down by α hyperparameter that is set to 0.0005 as its initial value is much higher than that coming from the margin loss of L_e.

2.3 Convolutional CapsNet

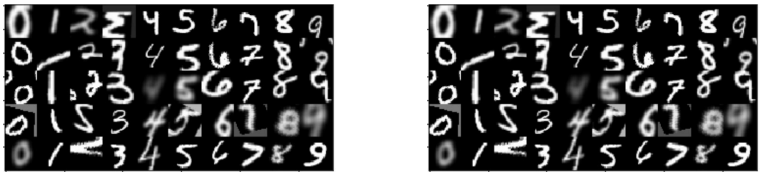
Convolutional CapsNet is a self-designed architecture that aims to keep maximum possible similarity to the CapsNet architecture, while – opposite to [15] – remaining a pure Convolutional Neural Network. In reference to Convolutional CapsNet, a term capsule indicates groups of neurons and their similarity to capsules in Capsule Networks. Routing Operation is entirely ignored. By intuition, such maneuver removes the control of preserving the relationships between parts and wholes in situations where they do exist in the image but are spatially disturbed. In the Convolutional CapsNet, the capsule-specific vector-wise operation is replaced with an activation function applied neuron-wise. The activation function can be set as ReLU or Sigmoid.

The differences between CapsNet and Convolutional CapsNet are summarised in Table 1. Any element not mentioned in the Table is exactly the same in Convolutional CapsNet as it is in CapsNet.

Table 1: Characteristics of CapsNet and proposed Convolutional CapsNet architectures

Factor	**CapsNet**	**Convolutional CapsNet**
Routing algorithm	Applied	Not applied
Activation funtion after PrimaryCaps and DigitCaps layers	Squashing	Sigmoid or ReLU
Level of activation funtion application after PrimaryCaps and DigitCaps layers	Capsule-wise	Neuron-wise

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2



(a) Samples from *Light VarMNIST* (b) Samples from *Strong VarMNIST*

Fig. 4: Samples from *Light and Strong VarMNIST* test set for each category (by columns)

3 Experiments

Experiments aim to explore the role of capsule network-specific elements on the performance of CapsNet to understand better the mechanisms applied in CapsNet and compare its quality of image classification to the results commonly achieved by CNNs. The compared network architectures need to satisfy the following criteria: maximum architectural similarity, similar number of trainable parameters, comparable performance in terms of classification accuracy.

3.1 Datasets Description

In the experiments, we want to check the robustness of compared networks to input translations and sensibility to spatial relationships between parts and wholes. We prepared two modified datasets based on the original MNIST and CIFAR-10 datasets, to achieve our goal. Both datasets contain images of size 32x32 pixels, and in each case, the training dataset includes 50000, and the test dataset contains 10000 samples.

In the first modification - *VarMNIST* and *VarCIFAR-10* datasets arose as the results of operation similar to those applicable in data augmentation. There are two variants of each test set: *Light VarMNIST* (*Light VarCIFAR*) and *Strong VarMNIST*(*Strong VarCIFAR*). They differ by the extent and amount of input alternations performed, applied with usage of *imgaug* library [11]. Image samples from both data sets for the MNIST dataset are visualized in Fig. 4.

The second group of test sets is *N-Shattered MNIST* and *N-Shattered CIFAR*sets where $N \in \{2, 4, 8, 16\}$. These datasets are obtained by shattering the original test image sample into N slices and combining them into one image in random order. Samples from N-Shattered CIFAR data sets are visualized in Fig. 5. For VarMNIST and VarCIFAR test sets, the desirable outcome is maximum classification accuracy. For N-Shattered MNIST and N-Shattered CIFAR sets it is its minimum as samples in most cases do not preserve the spatial relationships between parts and wholes.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

3.2 Experiment1. Study on impact of capsule-specific elements on performance quality

The experiment aims at checking the role of the routing by agreement procedure and activation function in CapsNet. To perform such comparison, the CapsNet architecture is compared to its sibling network - Convolutional CapsNet. The test is run using *Light Var* and *Strong Var* sets, and finally on N -Shattered test sets for MNIST and CIFAR-10 datasets.
CapsNet and Convolutional CapsNet setup: ReLU Conv1 Layer applies 256 filters of shape 9×9 with a stride 1. Its output consists of 256 feature maps of shape 20×20 with ReLU activation function; PrimaryCaps consists of 32 Capsules. Each Capsule applies 8 filters of size $9 \times 9 \times 256$ with stride of 2. Each filter produces a 6×6 feature map. In total there are 8 such feature maps per each capsule. The dimension of the output vector from a capsule is 8. The weight matrix between the PrimaryCaps and the DigitCaps layer is thus of shape $6 \times 6 \times 32 \times 8 \times 16$, resulting in a single 8×16 matrix between every capsule in the PrimaryCaps and the DigitCaps layers. The encoder network consists of 3 fully-connected layers: 512 ReLU units, 1024 ReLu units and 784 sigmoidal units. Convolutional CapsNet (CCapsNet) has identical architecture.

The hyperparameters value assumed based on tuning CapsNet architecture are presented in Table 2. They were also applied to the Convolutional CapsNet (assigned as CCapsNet). Performance of the model in terms of the number of parameters and processing time for the assumed hyperparameter values for MNIST and CIFAR-10 datasets is presented in Table 3. Note that the value of decoder loss is averaged over a number of pixels.

No hyperparameter tuning has been performed in this step. The training process curves of CapsNet and CCapsNet on the MNIST dataset almost overlap. Therefore we skip these charts here. Table 4 shows the results. Using MNIST the CCapsNet network managed to outperform CapsNet in terms of accuracy with $0.9921_{\pm 0.0011}$ compared to $0.9917_{\pm 0.0007}$ achieved by CapsNet. However, the decoder loss on the validation set is on average higher for CCapsNet ($0.0302_{\pm 0.0016}$) than for CapsNet ($0.0244_{\pm 0.0012}$). An additional observation is a slight tendency to overfit the training data observed for CCapsNet compared to CapsNet. This tendency is more visible in the case of CIFAR-10 datasets. In this case CCapsNet

(a) 2-Shattered CIFAR (b) 4-Shattered CIFAR

Fig. 5: Visualisations of N-Shattered CIFAR test sets for each category, where *airplane* is the leftmost column and *truck* is the rightmost column.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

Table 2: Hyperparameters of CapsNet –baseline configuration; PC no – the number of capsules in PrimaryCaps layer (DC for DigitCaps), PC dim – the output vector dimension of each capsule in PrimaryCaps layer (DC in DigitCaps)

Hyperparameter	**Value**
Optimizer	Adaptive Moment Estimation (Adam)
Learning rate	0.001
Learning rate decay	0.9
Data batch size	100
Epochs	50
Decoder loss coefficient	0.0005
No. of routings	3
Capsule number and dimension	PC no(32); PC dim(8); DC no(10); DC dim(16)

also outperformed CapsNet ($0.7005_{\pm 0.0018}$ vs $0.6629_{\pm 0.0007}$). However, the difference between the training set and test set accuracy and loss value is far greater for CCapsNet. Moreover, overfitting is visible on loss function value curves.

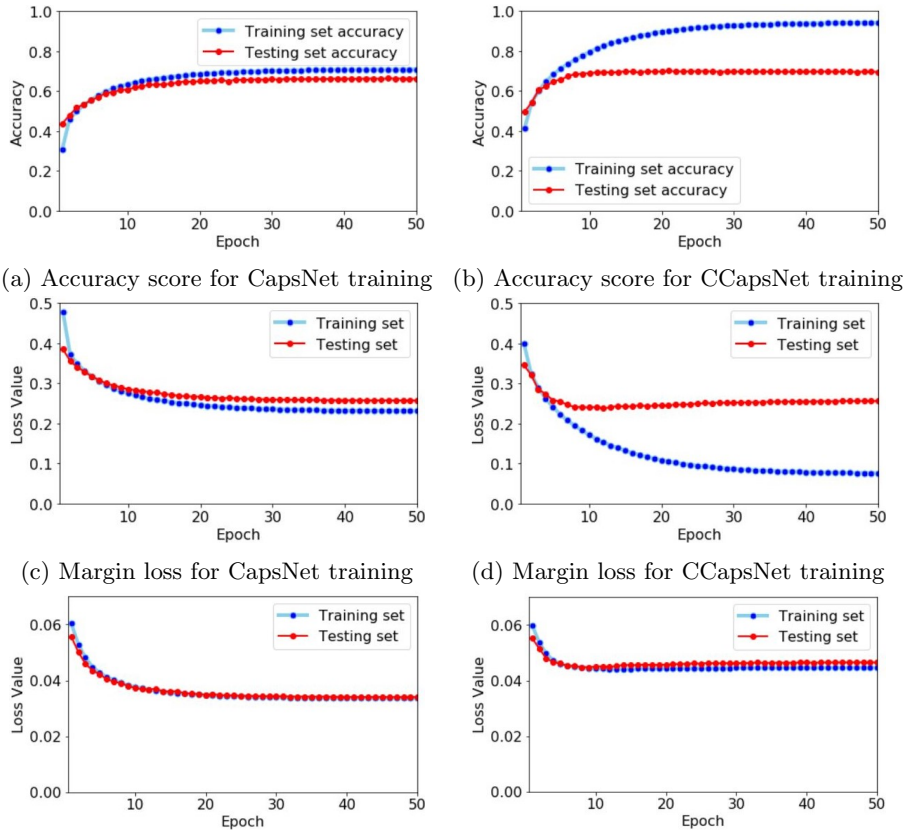
Fig. 6 displays charts for CIFAR-10 data set. Each column contains accuracy score, classifier loss function value (margin loss) and decoder loss function value (euclidean distance averaged over pixels) for consecutive epochs. To make the comparison of performance as reliable as possible, the Convolutional CapsNet has been juxtaposed with CapsNet in two variants *CCapsNet* - model with the best achieved accuracy on the test set in 50 epochs - (*best*) , *CCapsNet Matching* - model from the epoch when the test accuracy was the closest to that achieved by CapsNet (*comparable*). Accuracy for all considered test sets and models is shown in Table 4.

Performance analysis Contrary to expectations, Convolutional CapsNet achieved better accuracy score in almost all cases. However, training with routing-by-agreement and the squashing function makes the training process more stable than without these components and far less prone to overfitting. The squashing function's application to the decoder's vector representation allows for a slight increase in the accompanying task of image reconstruction. The capsule-specific elements do not make the network react better to test samples modified in terms of translation and noise. It also does not provide considerable improvement in preserving the spatial relationship between capsules, which should be ensured by applying routing-by-agreement between capsules.

Table 3: Performance of CapsNet achieved on MNIST and CIFAR-10

Dataset	Train accuracy	Test accuracy	Decoder loss	No of parameters	Train time per epoch
MNIST	0.9996 ± 0.0005	0.9917 ± 0.0007	0.0244 ± 0.0012	8 215 568	208 ± 4.23
CIFAR-10	0.7001 ± 0.0018	0.6629 ± 0.0007	0.0302 ± 0.0016	11 749 120	383 ± 9.13

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2



(a) Accuracy score for CapsNet training (b) Accuracy score for CCapsNet training

(c) Margin loss for CapsNet training (d) Margin loss for CCapsNet training

(e) Decoder loss for CapsNet training (f) Decoder loss for CCapsNet training

Fig. 6: Accuracy (a, b), margin loss value for classification (c, d) and decoder loss value (e, f) in training process on CIFAR-10 data set. CapsNet the - left column and Convolutional CapsNet - the right one.

3.3 Experiment2. Comparative performance of CapsNet and various CNNs types on augmented test data

The experiment aims to check whether capsule networks learn a more complex entity representation. The datasets and CapsNet setup are the same as in the Experiment1. The CapsNet model with the best hyperparameter values from the Experiment1. is used as a baseline. We consider the most popular CNN architectures - VGG, DenseNet, ResNet and its smaller version ResNet II networks. Their characteristics selected based on tuning are defined in Table 5. All architectures used a batch size of 100.

The results for the MNIST data set are shown in Table 6. We can observe that VGG achieved the best scores for all test sets. The situation changes slightly in the case of the comparable models. The best model for *Light VarMNIST* is

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

Table 4: Accuracy of CapsNet and Convolutional CapsNet (CCapsNet) averaged over 5 runs (best results are bolded)

Data set	**MNIST**		**CIFAR-10**		
Architecture	**CapsNet**	**CCapsNet**	**CapsNet**	**CCapsNet**	**CCapsNet Matching**
Standard	0.9917	**0.9921**	0.6629	**0.7005**	0.6698
Light Var	0.6443	**0.6894**	**0.5274**	**0.5411**	0.5211
Strong Var	0.5234	**0.5833**	**0.4841**	0.4828	0.4705
2-Shattered	0.5778	**0.5577**	0.4746	0.4802	**0.4706**
4-Shattered	**0.2512**	0.2553	0.3253	**0.2993**	**0.2982**
8-Shattered	**0.2481**	0.2661	0.3599	**0.3511**	**0.3501**
16-Shattered	0.1324	**0.1247**	0.2590	**0.2423**	**0.2449**

ResNet I and VGG for *Strong VarMNIST* but the difference is extremely low. In the case of CIFAR-10., the best CNN models sometimes surpassed the baseline method, however relation between accuracy on the standard set and transformed sets falls in favor of CapsNet. The observation is further notable in Table 7. We can observe that CapsNet surpasses all CNN models on *Light VarCIFAR* and *Strong VarCIFAR* datasets by far.

Performance analysis In the case of the MNIST dataset, it is possible that deep architectures of CNNs managed to learn so many different features that they cover a large part of possible cases. In the case of CIFAR-10, the possible image domain is far greater, thus CNN could not have learnt enough general representations to work well for augmented data. It may also come from higher overfitting of the model for CIFAR-10 than for MNIST.

3.4 Experiment3. Comparison of CapsNet and CNN models performance on randomly shattered test data

This experiment aims to verify the statement that capsules encode the spatial relationship between parts and wholes as opposed to Convolutional Neural Networks. We used N-Shattered vesion of datasets. The experiment setup is the same as in Experiment2.

Table 5: Characteristics of CNN architectures
*Comparable parameter space could not be reached due to computer memory limitations

Architecture	**Depth**	**MNIST**	**MNIST**	**CIFAR**	**CIFAR**
	(layers)	No. of params	epoch time [s]	No. of params	epoch time [s]
CapsNet	6	8 215 568	208	11 749 120	383
VGG	30	8 037 578	34	12 180 170	35
ResNet I	27	8 816 074	170	11 181 642	230
ResNet II	22	272 776	20	237 066	23
DenseNet	144	8 380 090	130	*9 012 202	145

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

Table 6: Accuracy on VarMNIST datasets for CapsNet and CNN models

	Dataset	CapsNet	VGG	ResNet I	ResNet II	DenseNet
comparable	Standard	**0.9917**	0.9908	**0.9917**	0.9903	0.9903
	Light VarMNIST	0.6443	0.7136	**0.7164**	0.7046	0.6717
	Strong VarMNIST	0.5234	**0.6417**	0.5860	0.6052	0.5738
best	Standard	0.9917	**0.9927**	0.9923	0.9903	0.9903
	Light VarMNIST	0.6443	**0.7623**	0.6412	0.7046	0.6717
	Strong VarMNIST	0.5234	**0.6923**	0.5380	0.6052	0.5738

The results for MNIST dataset are showed in Table 8 and for CIFAR-10 in Table 9. In the case of MNIST, CapsNet always managed to surpass all CNN models. In the case of CIFAR-10, the winner is ResNet.

Performance analysis Similar to previous experiments, observations made on the MNIST data sets are not transferable to CIFAR-10 test results. Ability to encode spatial relationships between parts and wholes by the baseline Capsule Network and compared CNNs seems to be dependent on the level of model under- or overfitting. In the case of the MNIST dataset, where no considerable overfitting is present, CapsNet surpasses all other architectures. There are different behaviours observed for different networks for CIFAR-10. VGG seems to have decreasing performance on N-Shattered test sets along with improvement on standard test set, which is opposite to what is partially observed for ResNet II. We observed that ResNet II highly overfits to training data, which is possibly the reason why it does not recognize shattered images correctly. However, considering only the aspect of spatial relationship encoding, data overfitting is not a disturbing occurrence as what we look for is encoding of parts and wholes and not the input variance covered by the network. Thus a conclusion can be drawn that applying skip connections in the model, like in the case of ResNet and DenseNet, improves network's ability to encode spatial relationships of input data.

4 Conclusions

Capsule Networks provide a framework for shallow architecture with mechanisms addressing common problems present in Convolutional Neural Networks. They have a higher computational complexity but may work better for data sets of

Table 7: Accuracy for *VarCIFAR* datasets for CapsNet and various CNN models

	Dataset	CapsNet	VGG	ResNet I	ResNet II	DenseNet
comparable	Standard	0.6629	**0.7019**	0.6602	0.6803	0.6363
	Light VarCIFAR	**0.5274**	0.4559	0.4826	0.4748	0.4731
	Strong VarCIFAR	**0.4841**	0.3862	0.4117	0.3938	0.3994
best	Standard	0.6629	**0.8519**	0.6602	0.7602	0.6363
	Light VarCIFAR	0.5274	**0.5609**	0.4826	0.5386	0.4731
	Strong VarCIFAR	**0.4841**	0.4608	0.4117	0.4503	0.3994

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

Table 8: Accuracy for N-Shattered MNIST for CapsNet and CNN models

	Dataset	CapsNet	VGG	ResNet I	ResNet II	DenseNet
compar-able	Standard	**0.9917**	0.9908	**0.9917**	0.9903	0.9903
	2-Shattered MNIST	**0.5778**	0.5974	0.7143	0.7892	0.6480
	4-Shattered MNIST	**0.2512**	0.3341	0.3913	0.4768	0.3685
	8-Shattered MNIST	**0.2481**	0.3220	0.3521	0.3384	0.3349
	16-Shattered MNIST	**0.1324**	0.1759	0.2218	0.1698	0.1882
best	Standard	0.9917	**0.9927**	0.9923	0.9903	0.9903
	2-Shattered MNIST	**0.5778**	0.5795	0.6839	0.7892	0.6480
	4-Shattered MNIST	**0.2512**	0.2922	0.3688	0.4768	0.3685
	8-Shattered MNIST	**0.2481**	0.3355	0.3457	0.3384	0.3349
	16-Shattered MNIST	**0.1324**	0.1443	0.1976	0.1698	0.1882

complex nature and with little training data. The experiments show that the training process of a CapsNet is very stable and not prone to overfitting, contrary to the tested CNNs. However, the specific cases in which CapsNet may be a better choice as compared to one of the popular deep CNN architectures cannot be clearly specified based on available research as the results and observations highly deviate for different data sets. We can observe that application of routing-by-agreement and the squashing function highly influences performance quality for medical and generated data sets, in their favor.

The experiments presented in this paper could be extended by further tuning of proposed architectures to specified problems. Capsule Networks can be further researched for more complex problems. Application of a Capsule Network to a larger task, like ImageNet classification task or COCO segmentation task would probably give better insight in their characteristics. Moreover, they can be tested in use as embedding builders for other tasks due to their natural way of encoding detected objects in forms of vectors, both for visual and textual data. Based on conducted experiments and available literature, Capsule Networks appear as an

Table 9: Accuracy scores for N-Shattered CIFAR data sets obtained by CapsNet and different CNN models

	Architecture	CapsNet	VGG	ResNet I	ResNet II	DenseNet
compar-able	Standard	0.6629	**0.7019**	0.6602	0.6803	0.6363
	2-Shattered CIFAR	0.4746	0.5172	**0.4549**	0.5527	0.4640
	4-Shattered CIFAR	0.3253	0.3764	**0.2954**	0.4119	0.3249
	8-Shattered CIFAR	0.3599	0.3635	**0.3241**	0.3886	0.3415
	16-Shattered CIFAR	0.2590	0.2836	**0.2249**	0.3050	0.2529
best	Standard	0.6629	**0.8519**	0.6602	0.7602	0.6363
	2-Shattered CIFAR	0.4746	0.6202	**0.4549**	0.5943	0.4640
	4-Shattered CIFAR	0.3253	0.4167	**0.2954**	0.4034	0.3249
	8-Shattered CIFAR	0.3599	0.4045	**0.3241**	0.3410	0.3415
	16-Shattered CIFAR	0.2590	0.2701	**0.2249**	0.2229	0.2529

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2

alternative model which may outperform other network types for specific problems, but does not show strong advantage over Convolutional Neural Networks, however, due to their consistent design, they remain an area worth exploring.

References

1. Duarte, K., Rawat, Y.S., Shah, M.: Videocapsulenet: A simplified network for action detection. CoRR **abs/1805.08162** (2018), http://arxiv.org/abs/1805.08162
2. Gritsevskiy, A., Korablyov, M.: Capsule networks for low-data transfer learning. CoRR **abs/1804.10172** (2018), http://arxiv.org/abs/1804.10172
3. Hasani, M., Saravi, A.N., Khotanlou, H.: An efficient approach for using expectation maximization algorithm in capsule networks. 2020 International Conference on Machine Vision and Image Processing (MVIP) pp. 1–5 (2020)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90, https://www.computer.org/csdl/proceedings-article/cvpr/2016/8851a770/12OmNxvwoXv
5. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I. pp. 44–51. ICANN'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2029556.2029562
6. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=HJWLfGWRb
7. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. CoRR **abs/1608.06993** (2016), http://arxiv.org/abs/1608.06993
8. Jiang, X., Wang, Y., Liu, W., Li, S., Liu, J.: Capsnet, cnn, fcn: Comparative performance evaluation for image classification. International Journal of Machine Learning and Computing **9**(6), 840–848 (2019). https://doi.org/10.18178/ijmlc.2019.9.6.881
9. Mobiny, A., Nguyen, H.V.: Fast capsnet for lung cancer screening. CoRR **abs/1806.07416** (2018), http://arxiv.org/abs/1806.07416
10. Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: A comprehensive review. Neural Computation **29**(9), 2352–2449 (2017). https://doi.org/10.1162/neco_a_00990
11. Revision, A.J.: imgaug: Image augmentation for machine learning experiments. https://github.com/aleju/imgaug (2019)
12. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. CoRR **abs/1710.09829** (2017), http://arxiv.org/abs/1710.09829
13. Schlegel Kenny, Neubert Peer, P.P.: Comparison of data efficiency in dynamic routing for capsule networks (2018), https://www.tu-chemnitz.de/etit/proaut/publications/schlegel_2018.pdf
14. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
15. Toraman, T., Alakusb, B., Turkoglu, I.: Convolutional capsnet: A novel artificial neural network approach to detect covid-19 disease from x-ray images using capsule networks. Chaos, Solitons & Fractals **140**, 110122 (2020). https://doi.org/https://doi.org/10.1016/j.chaos.2020.110122

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77977-1_2