

DeepCaps+: A Light Variant of DeepCaps

Pouya Shiri^a and Amirali Baniasadi
University of Victoria, BC, Canada

Keywords: Capsule Networks, DeepCaps, Deep CapsNet, Fast CapsNet.

Abstract: Image classification is one of the fundamental problems in the field of computer vision. Convolutional Neural Networks (CNN) are complex feed-forward neural networks that represent outstanding solutions for this problem. Capsule Network (CapsNet) is considered as the next generation of classifiers based on Convolutional Neural Networks. Despite its advantages including higher robustness to affine transformations, CapsNet does not perform well on complex data. Several works have tried to realize the true potential of CapsNet to provide better performance. DeepCaps is one of such networks with significantly improved performance. Despite its better performance on complex datasets such as CIFAR-10, DeepCaps fails to work on more complex datasets with a higher number of categories such as CIFAR-100. In this network, we introduce DeepCaps+ as an optimized variant of DeepCaps which includes fewer parameters and higher accuracy. Using a 7-ensemble model on the CIFAR-10 dataset, DeepCaps+ obtains an accuracy of 91.63% while performing the inference 2.51x faster than DeepCaps. DeepCaps+ also obtains 67.56% test accuracy on the CIFAR-100 dataset, proving this network to be capable of handling complex datasets.

1 INTRODUCTION


Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs. Modern solutions rely on deep learning and convolutional neural network (CNN) to derive this information.

CNNs have made significant progress in solving different computer vision problems. Despite their high performance, CNNs offer limited translational invariance and lose important information (due to the max-pooling layer). In addition, they are unable to understand the spatial relationship among the features existing in an image. Sabour et al. proposed Capsule Network (CapsNet) (Sabour et al., 2017) to address these issues. The base units in CapsNet are capsules (versus neurons in CNNs), which are created by reshaping the output of a low-level feature extractor consisting of two convolutional layers. CapsNet does not include the pooling layers and considers the relationship between features of different levels in an image by developing a part-to-whole relationship. CapsNet provides higher robustness to affine transformations compared to CNNs. This is due to the affine matrix multiplication stage which is performed right after the capsules are created. CapsNet also performs

better than CNNs for images including overlapping categories.

CapsNet implements classification by employing only two convolutional layers for extracting the features and creating capsules. The first layer of capsules is called Primary Capsules (PCs). CapsNet also includes one fully-connected capsule layer to reduce the number of capsules. CapsNet performs well on small-scale datasets such as MNIST (LECUN and Y.,) and Fashion-MNIST (Xiao et al., 2017). As datasets become more complex, (e.g. CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009)), CapsNet falls behind CNNs. There are certain methods for improving the performance of a neural network. One common approach is increasing the depth of the networks by adding more layers, making them complex enough to handle more complex datasets. However, there are several issues with stacking up several fully-connected layers (Rajasegaran et al., 2019). The main issue stems from the fact that the fully-connected capsule layer consists of an iterative computationally intensive algorithm referred to as Dynamic Routing (DR). Therefore stacking up these layers results in a huge number of parameters. Rajasegaran et al. (Rajasegaran et al., 2019) introduced DeepCaps in an attempt to make CapsNet deeper.

DeepCaps consists of several capsule layers referred to as Capsule Cells (CapsCells). These layers

^a  <https://orcid.org/0000-0002-8037-9481>

include skip-connections to improve the gradient flow and implement the DR algorithm. To reduce the complexity resulting from the added several layers, DR is eliminated from the initial layers. Instead, a new routing algorithm inspired by the 3D convolution is introduced in one of the middle layers. This 3D convolution method is referred to as 3D Dynamic Routing (3DR). 3DR reduces the number of parameters by parameter sharing. In this work, we refer to a CapsCell including a 3DR algorithm as 3DR-CapsCell. Caps-cells play an important role in creating robust PCs. In DeepCaps, the primary capsules are a combination of low-level and high-level capsules (the input and the output of the 3DR-CapsCell).

DeepCaps performs well on the CIFAR-10 dataset (92.74% test accuracy). On the negative side, being a dense network (13.4M parameters) results in slow inference (2.86ms for a single image on a 2080Ti GPU). Moreover, DeepCaps includes a high number of PCs. Note that the number of PCs affects the network time and the number of parameters (Shiri et al., 2020). Consequently, the network loses accuracy if we reduce the number of PCs without proper investigation. In this work, we propose DeepCaps+ by designing the Capsule Assembler (CA) module to carefully combine the PCs of DeepCaps and reduce them into fewer capsules. The CA module provides a linear combination of PCs and produces fewer capsules capable of obtaining a very robust representation. DeepCaps+ reduces the number of parameters by 45.5% and to only 7.4M for the CIFAR-10 dataset. In addition, the inference is 2.51x faster than DeepCaps. In summary, our contributions in this paper are:

- We reduce the number of parameters significantly. DeepCaps+ includes smaller weight matrices in the affine multiplication stage. Since a separate matrix is considered for each category, this stage includes a high fraction of parameters in the network. DeepCaps+ includes only 7.3M parameters on the CIFAR10 dataset (compared to 13.4M in DeepCaps).
- We optimize the network inference speed. Our implementation helps DR finish computations faster. DeepCaps+ is 2.51x faster in inference for the CIFAR-10 dataset compared to DeepCaps.
- We carefully consider the inference accuracy in our implementation. Even though there are fewer capsules in DeepCaps+ compared to DeepCaps, DeepCaps+ obtains an accuracy of 91.63% using a 7-ensemble model for the CIFAR-10 dataset (compared to 92.74% in DeepCaps).

The rest of this paper is organized as follows. Section 2 presents the related works. The background of

this work is explained in detail in Section 3. Section 4 presents our solution. Section 5 reports the experimental results and finally, the paper is concluded in Section 6.

2 RELATED WORKS

In this section, we review some of the recent studies on optimizing CapsNet’s performance.

Yang et al. (Yang et al., 2020) introduce an alternative to the capsule network referred to as RS-CapsNet. This network takes advantage of residual blocks for multi-scale feature extraction. In addition, the Squeeze-and-Excitation (SE) block is used to weigh the features based on their importance. RS-CapsNet creates a linear combination of capsules improving their representation ability.

Huang et al. introduce another variant of CapsNet referred to as Dual Attention mechanism Capsule Network (DA-CapsNet) (Huang and Zhou, 2020). DA-CapsNet takes advantage of the attention mechanism both after the primary capsules and the convolutional layers. DA-CapsNet outperforms CapsNet in image reconstruction and obtains state-of-the-art accuracy on small-scale datasets.

Shiri et al. (Shiri and Baniasad, 2021) propose Convolutional Fully-Connected CapsNet (CFC-CapsNet). They design a new layer which creates PCs from the extracted features. The application of this layer results in reducing capsules while obtaining higher network test accuracy. The reduction of capsules results in fewer parameters, as well as reducing the network training and inference times.

Deliege et al. (Deli, 2018) propose HitNet in which a capsule layer referred to the Hit-or-Miss layer is used. HitNet consists of a central capsule, which is hit or missed by input capsules in the training process. HitNet also consists of a decoder which synthesizes samples of the input images. The decoder can be utilized as an augmentation tool for avoiding overfitting. This network also includes a new type of capsule referred to as the ghost capsule. This new capsule is used to detect mislabeled data in the training set.

He et al. (He et al., 2019) propose a capsule network based on complex values (CV-CapsNet). CV-CapsNet contains a multi-scale feature extractor based on complex values. PCs which are created from the extracted feature, also consist of complex values. CV-CapsNet modifies the Dynamic Routing algorithm to comply with the complex domain.

Chen et al. (Chen and Liu, 2020) introduce a deep variant of CapsNet (DCN-UN) which contains a U-Net module as the feature extractor to improve the

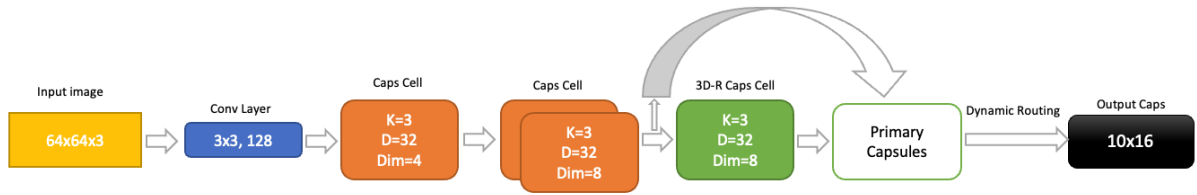


Figure 1: DeepCaps architecture. This network contains multiple CapsCell layers followed by a 3D CapsCell layer to generate the primary capsules. The output capsules are inferred from the primary capsules using the Dynamic Routing algorithm.

performance of CapsNet on complex datasets. DCN-UN includes a convolutional capsule layer which uses weight-sharing to reduce the number of parameters of the network. DCN-UN also introduces an alternative method for dynamic routing referred to as Mask Dynamic Routing (MDR).

Ayidzoe et al. (Ayidzoe et al., 2021) propose Gabor-CapsNet containing an improved feature extractor and fewer parameters. A Gabor filter and other customized blocks are used in the feature extractor, which results in improved activation diagrams and better learning of the hierarchical information. Tao et al. (Tao et al., 2022) introduce Adaptive CapsNet (AC-CapsNet), which replaces PCs with an adaptive layer of capsules. The adaptive values hold both the spatial information of capsules and the local relationship of each dimension of capsules.

In this work, we build on deep networks (DeepCaps) and improve performance. We do so by proposing an alternative network with fewer parameters and faster training and inference.

3 BACKGROUND

In this section, we present the required background for this work. We slightly reformulate and simplify DeepCaps and explain the different structures used in it.

3.1 DeepCaps Architecture

Figure 1 shows the architecture of DeepCaps (excluding the decoder). This network consists of several CapsCells. These units consist of several Convolutional Capsule (ConvCaps) layers and a skip connection (shown in Figure 2). CapsCells have three parameters (K, D and Dim) that correspond to the ConvCaps layers used in them.

The ConvCaps layer is defined as a convolutional layer with its outputs reshaped to vectors. There are different methods to reshape a convolutional layer including transforming 128 kernels to vectors. For example, we can build 32 vectors each with 4 elements

(4D vectors). Alternatively, we can reshape the kernels to 16 vectors (8D). Each ConvCaps layer has 3 parameters: K or the kernel size, Dim or the number of elements for each output vector, and D or the number of vectors per spatial location of the output feature map.

The architecture of a CapsCell is shown in figure 2. As the figure shows, this unit consists of consecutive ConvCaps layers and a skip-connected ConvCaps layer. For 3D-R CapsCells, the skip-connection performs the 3D routing operation.

DeepCaps also employs a decoder on top of the output capsules. The decoder used in this network is explained later in this section.

DeepCaps includes three consecutive normal CapsCells followed by a CapsCell with 3D-Routing (3DR CapsCell). The output of the 3DR CapsCell creates a fraction of PCs. The rest of the PCs are created by the consecutive normal CapsCells only. This is achieved using a skip-connection from the output of the third normal CapsCell. The skip connection helps avoid vanishing gradients while training. It also routes low-level capsules to high-level capsules.

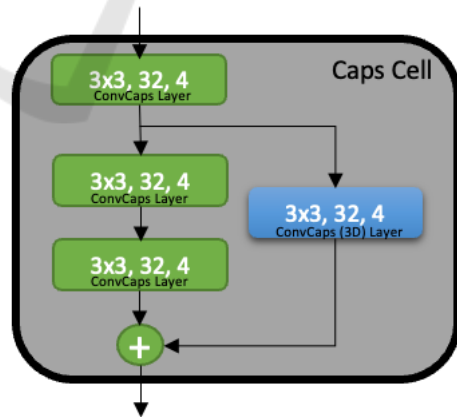


Figure 2: The architecture of a CapsCell with (K=3, D=32 and Dim=4). This unit contains several ConvCaps layers and a skip-connection. For the 3D-R CapsCells, the skip connection performs the 3D dynamic routing operation.

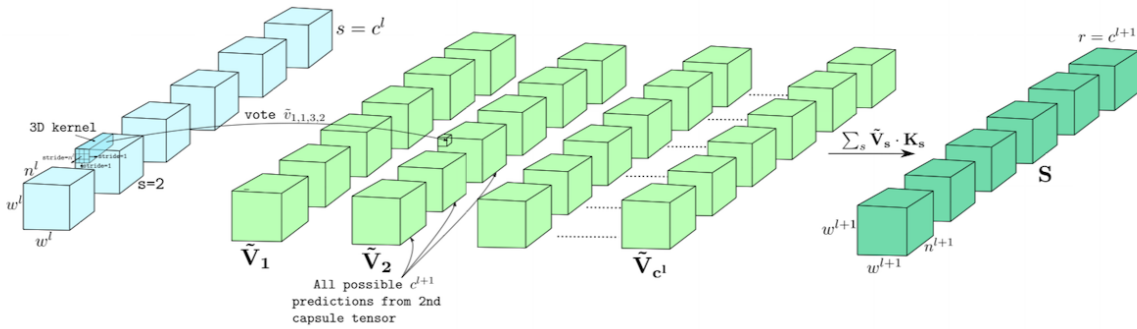


Figure 3: 3D-Routing method. Each capsule in layer l , predicts c^{l+1} capsules. As a result, there are c^l predictions for a capsule in layer $l+1$ (Rajasegaran et al., 2019).

3.2 Routing Capsules

One of the important novelties of DeepCaps, is the 3D-Routing method applied to partially generate the PCs. The conventional DR algorithm routes capsules globally (all input capsules take part in generating all output capsules). However, the 3D-Routing method routes the capsules locally. In other words, only spatially correlated capsules take part in generating a specific output capsule.

Figure 3 shows how the 3D-Routing method infers the output capsules from the input capsules. Each capsule in layer l , predicts c^{l+1} capsules. As a result, there are c^l predictions for a capsule in layer $l+1$. Initially, all the input capsules are weighted equally and summed to produce the output capsules. Afterwards, there are several iterations in which the coefficients are updated based on the agreement between the votes and the output capsules.

3.3 Class-Independent Decoder

The output capsules are fed to a decoder sub-network for reconstructing the input images. In order to regularize training and tackle the problem of over-fitting, the decoded images are compared to the input images. This comparison is used in the loss function (reconstruction loss). The decoder used in the DeepCaps network has two advantages over the original one. First, it includes deconvolution layers, which capture spatial relationships better than Fully-Connected (FC) layers using fewer parameters.

The second feature of this decoder is that it is class-independent. To achieve this it disallows incorrect capsules from taking part in the reconstruction process. As a result, only the correct capsule is kept and the rest of the output capsules need to be discarded. In the original version of CapsNet, Sara Sabour et al. (Sabour et al., 2017) masked the incorrect capsules with zeros. However, the decoder used

in DeepCaps, discards the incorrect capsules completely. Different categories (classes) keep a fixed vector of data and use it for reconstruction. This decoder is class-independent as all classes are treated equally. This results in a more powerful decoder as earlier work shows (Rajasegaran et al., 2019).

3.4 Loss function

The loss function of DeepCaps is similar to the one introduced by Sara Sabour et al. (Sabour et al., 2017) (margin loss). This loss function considers penalties for incorrect predictions and disregards predictions with a very high or very low probability:

$$L_k = T_k \max(0, m^+ - \|V_k\|)^2 + \lambda(1 - T_k) \max(0, \|V_k\| - m^-)^2$$

In this equation, L_k is the loss term for capsule k , T_k is 0 for incorrect class and 1 otherwise, m^+ and m^- are used to disregard high or low probabilities, and lambda is used for controlling the gradient at the start of the training.

4 DeepCaps+

The number of PCs used in capsule networks is a key factor in determining the network's number of parameters and also the network training and testing speed. The primary capsules are multiplied by several weight matrices. There is a different matrix considered for each category in the classification task. For this reason, reducing the number of PCs results in using fewer matrices leading to fewer overall parameters. In addition, the DR algorithm becomes more expensive by increasing the number of PCs in the network. This is explained by the fact that the DR algorithm iterates over PCs multiple times to find an agreement between those capsules.

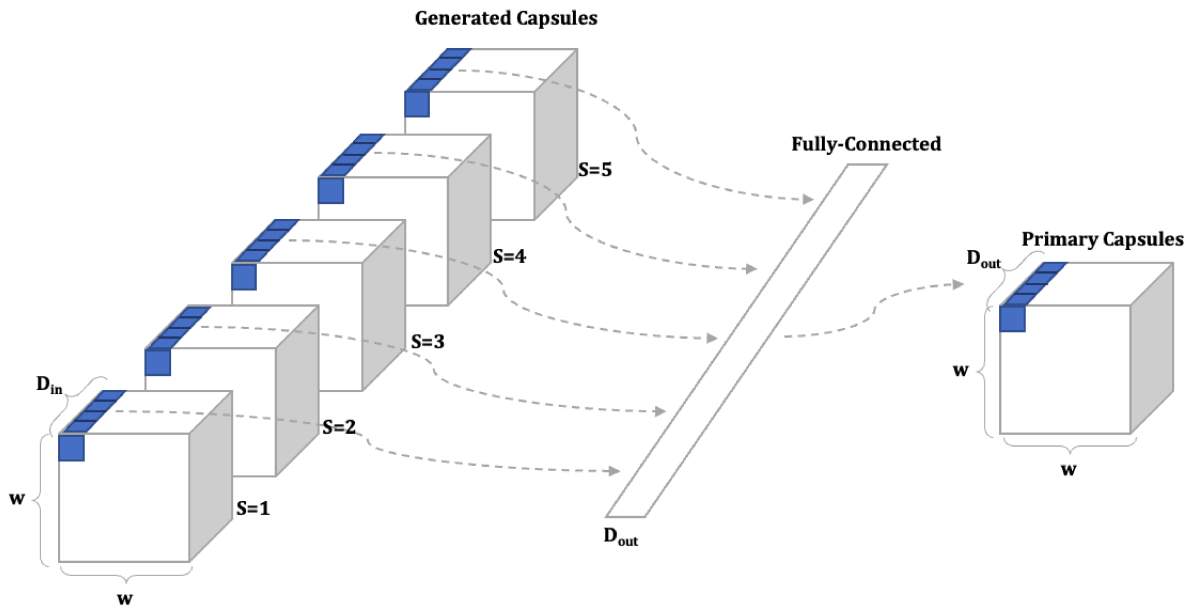


Figure 4: The capsule summarization layer. A total of $w \times w \times 5$ generated capsules are summarized into $w \times w \times D_{out}$ primary capsules using $w \times w$ Fully-Connected (FC) layers. The first FC layer is shown.

4.1 Capsule Summarization

In this work, we propose a method to summarize the capsules generated by DeepCaps to produce a new set of PCs. To this end, we introduce a capsule summarization layer. Figure 4 shows how this layer works. As the figure shows, the generated capsules at a specific spatial location are all flattened, and fed to a fully-connected layer to produce a single capsule. This procedure is repeated for all spatial locations in the generated capsules. There are a total of the $S \times w \times w$ generated capsules each with D_{in} elements. For each spatial location (i, j) where $i, j \in [1, w]$, a total of S capsules with D_{in} dimensionality are collected, flattened and fed to a single fully-connected layer to produce a single capsule with a different dimensionality D_{out} . It is noteworthy that the figure shows the procedure for the first spatial location $(1, 1)$, and there are $w \times w$ fully-connected layers to summarize the whole set of generated capsules to the primary capsules. The proposed layer reduces the number of capsules S times.

Intuitively, each location in the generated capsules corresponds to a set of spatially correlated capsules. We reduce all those correlated capsules to a single capsule using a fully-connected layer. The reduction process includes weights (contained in the fully-connected layer) that are improved over the training process. This results in producing a powerful summary of capsules.

4.2 Capsule Assembler

As mentioned earlier in the background section, the primary capsules are a combination of capsules generated by the 3D-R CapsCell (high-level capsules) and those generated by the normal CapsCells (low-level capsules). The capsules generated by 3DR are high-level, as they carry information regarding the agreement between locally correlated capsules. However, the low-level capsules are transmitted to the PCs' layer through a skip connection. This separation is done for all datasets.

We designed the Capsule Assembler (CA) to create an improved representation from the high-level and low-level capsules. In this work, as figure 5 shows, we summarize the generated capsules at different levels separately instead of concatenating capsules and forming the PCs. CapsSum provides an opportunity to learn the relationship between each capsule's neurons while reducing the number of these capsules. In fact, we produce lower capsules with more robust neurons in terms of representation. We concatenate the output capsules produced by the CapsSum layer to form the alternative PCs. Originally, on the CIFAR10 dataset there are $8 \times 8 \times 32 = 2048$ capsules generated by the 3D-R CapsCell, and $4 \times 4 \times 32 = 512$ capsules generated by the normal CapsCells each of which includes 8 neurons. We employ CA and reduce the 2048 and 512 generated capsules to $8 \times 8 = 64$ and $4 \times 4 = 16$ primary capsules respectively. We implement this by flattening all the spatially correlated capsules of each high-level or

Table 1: Classification accuracy of some state-of-the-art CNNs (shown on top) and the state-of-the-art CapsNet variants (shown in the middle) compared to DeepCaps+ (shown on the bottom).

Model	CIFAR-100	CIFAR-10	SVHN	FMNIST
DenseNet (Huang et al., 2016)	82.4%	96.40%	98.41%	95.40%
RS-CNN (Yang et al., 2020)	61.14%	90.15%	95.56%	93.34%
BiT-M (Kolesnikov et al., 2019)	92.17%	98.91%	-	-
DA-CapsNet (Huang and Zhou, 2020)	-	85.47%	94.82%	93.98%
CapsNet (7-ens) (Sabour et al., 2017)	-	89.40%	95.70%	-
Cv-CapsNet++ (He et al., 2019)	-	86.70%	-	94.40%
CFC-CapsNet (Shiri and Baniasadi, 2021)	-	73.15%	93.29%	92.86%
HitNet (Deli, 2018)	-	73.30%	94.50%	92.30%
DCN-UN MDR (Chen and Liu, 2020)	60.56%	90.42%	-	93.33%
Gabor CapsNet (Ayidzoe et al., 2021)	68.17%	85.24%	-	94.78%
AC-CapsNet (Tao et al., 2022)	66.09%	92.02%	96.86%	-
RS-CapsNet (Yang et al., 2020)	64.14%	91.32%	97.08%	94.08%
DeepCaps (7-ens) (Rajasegaran et al., 2019)	-	92.74%	97.56%	94.73%
DeepCaps+	61.88%	89.63%	96.14%	94.52%
DeepCaps+ (7-ens)	67.56%	91.63%	96.82%	94.90%

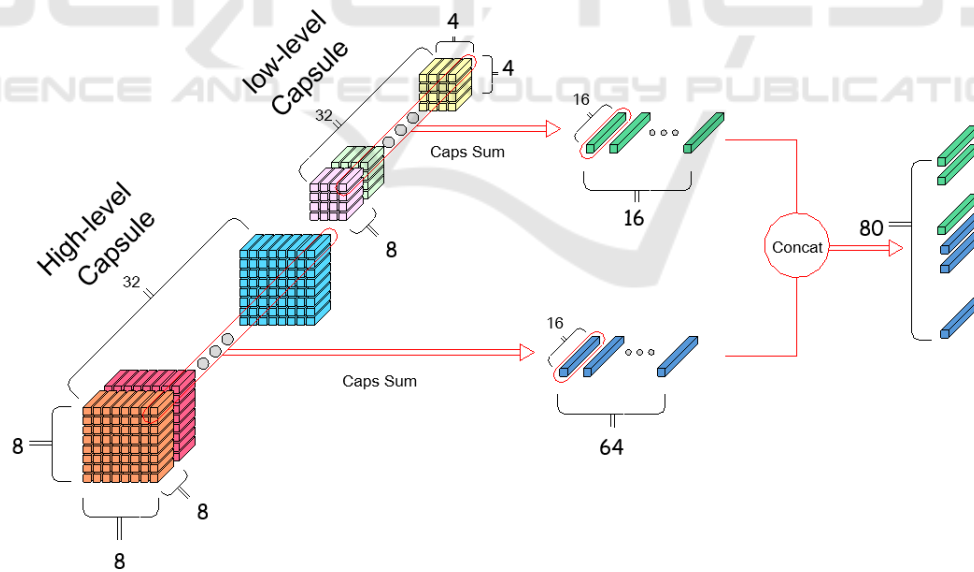


Figure 5: Capsule Assembler Module. Low-level and high-level capsules are summarized separately using CapsSum layers, and the outputs are concatenated.

low-level category and summarising them into a single more robust capsule. It is noteworthy that these spatially correlated capsules have the same position in the different 32 groups of the capsule-based feature maps. The Capsule Assembler improves accuracy as produces less but more robust high-level and low-level Capsules.

5 EXPERIMENTS AND RESULTS

In this section, we explain the experiments and the corresponding results.

5.1 Datasets

There are certain datasets used for testing a new image classifier. It is still not possible to test CapsNet on real-world high-scale datasets including a high number of categories such as ImageNet (Deng et al., 2009). This is due to the structure of capsule-based networks. These networks receive a tremendously high number of parameters and also become significantly slow as the number of categories and image size increases. We test DeepCaps+ against the regular datasets used for testing CapsNet and its variants: Fashion-MNIST (FMNIST), SVHN, CIFAR-10 and CIFAR-100. Note that testing CIFAR-100 is possible due to the significant reduction of the number of parameters achieved by using CA. For SVHN, CIFAR-100 and CIFAR-10 datasets, the input images are resized from $32 \times 32 \times 3$ to $64 \times 64 \times 3$ and for the rest of the datasets, the original images are used throughout the experiment. We do the resizing because the images in these datasets include rich features.

5.2 Experiment Settings

We implement DeepCaps+ on top of the Keras implementation of DeepCaps¹. We use an NVIDIA 2080Ti GPU for the experiments. We follow the "hard training" method implemented in DeepCaps. Specifically, after the network is trained for 100 epochs, the bounds of the loss function are restricted (different values m^+ and m^-), and the network is trained for another 100 epochs. The experiments are repeated 5 times and due to the little variation in the results, we report the average values. We used the Adam optimizer with starting learning rate of 0.001. We use an exponential decay ($\gamma = 0.96$) and batch size of 128.

¹<https://github.com/brjathu/deepcaps>

5.3 Network Accuracy

Table 1 provides a comparison of the network classification accuracy of our proposed network with some state-of-the-art CNN networks (shown in the top part of the table), and some recent networks based on capsules (shown in the middle of the table). DeepCaps+ and other recent CapsNet variants are still behind powerful CNN networks such as BiT-M (Kolesnikov et al., 2019).

There are only a few networks based on CapsNet capable of supporting the CIFAR-100 dataset. For datasets with a high number of classes, the number of parameters can be very high and DR may take a long time to infer the output capsules out of the input capsules. We use a 7-ensemble model of DeepCaps+, where 7 instances of the network are trained and the classification results of the networks are averaged. With 67.56% accuracy for the 7-ensemble DeepCaps+, our proposed network stands in the second rank compared to state-of-the-art CapsNet-based networks.

5.4 Number of Parameters

Table 2 shows the number of parameters for the networks listed in table 1. Since the SVHN and the CIFAR-10 datasets share the same image format, they result in the same number of parameters and hence the SVHN dataset is removed from the table. DeepCaps+ is built on top of DeepCaps, and reduces the number of parameters by 85%, 45.5% and 18.8% for CIFAR-100, CIFAR-10 and F-MNIST datasets, respectively. Some recent and powerful CapsNet and CNN variants employ novel solutions for reducing the number of weights and achieve significantly fewer number of parameters.

5.5 Network Training and Inference Time

The CA module results in significant speedup, as it makes it easier for the DR algorithm to infer the output capsules. Rajasegaran et al. (Rajasegaran et al., 2019) have reported the network test time for DeepCaps on a NVIDIA V100 GPU. CapsNet takes 2.86ms for a $32 \times 32 \times 3$ image and DeepCaps takes 1.38ms for a $64 \times 64 \times 3$ image. Using a NVIDIA 2080Ti GPU, DeepCaps takes 4.30ms for a $64 \times 64 \times 3$ image while DeepCaps+ does the same job in 1.71ms resulting in a significant 2.51x speedup compared to DeepCaps.

Table 2: The number of network parameters for networks shown in Table 1. The SVHN dataset is removed because it gets the same number of parameters as the CIFAR-10 dataset.

Model	CIFAR-100	CIFAR-10	FMNIST
DenseNet (Huang et al., 2016)	15.3M	15.3M	15.3M
RS-CNN (Yang et al., 2020)	2.8M	2.7M	2.7M
BiT-M (Kolesnikov et al., 2019)	928M	928M	928M
DA-CapsNet (Huang and Zhou, 2020)	-	-	-
CapsNet (Sabour et al., 2017)	-	11.7M	8.2M
Cv-CapsNet++ (He et al., 2019)	-	2.7M	2.5M
CFC-CapsNet (Shiri and Baniasadi, 2021)	-	5.9M	5.7M
HitNet (Deli, 2018)	-	8.9M	8.9M
DCN-UN MDR (Chen and Liu, 2020)	4.8M	1.4M	-
Gabor CapsNet (Ayidzoe et al., 2021)	22.6M	10.4M	-
AC-CapsNet (Tao et al., 2022)	4.12M	1.26M	-
RS-CapsNet (Yang et al., 2020)	16.8M	5M	5M
DeepCaps (Rajasegaran et al., 2019)	72.4M	13.4M	8.5M
DeepCaps+	10.9M	7.3M	6.9M

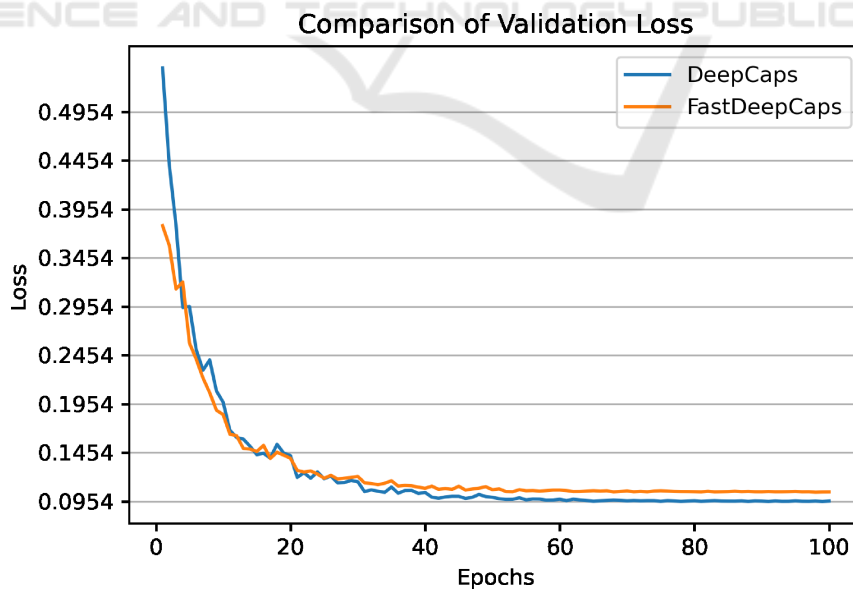


Figure 6: Comparing the validation loss between DeepCaps and DeepCaps+. DeepCaps+ results in a smoother convergence, and the final loss is slightly higher than DeepCaps.

5.6 Network Convergence

Figure 6 plots the network validation loss during the training process for DeepCaps and DeepCaps+. The validation loss is the sum of the decoder (reconstruction) loss and the margin loss. As the figure shows, both networks converge almost at the same epoch. In the meantime, training DeepCaps+ is more stable than DeepCaps. In addition, DeepCaps+ shows a slightly higher final loss value compared to DeepCaps.

6 CONCLUSION

In this work, we present DeepCaps+ as a powerful variant of Capsule-based Network, as an extension of DeepCaps. This network performs competitively compared to the state-of-the-art CapsNet-based networks. DeepCaps+ includes the 3D Dynamic routing and the class-independent decoder introduced by Rajasegaran et al. (Rajasegaran et al., 2019) and introduces the novel Capsule Assembler module in order to reduce the primary capsules to speed up the network and reduce the number of parameters while maintaining the test accuracy. Using a 7-ensemble model, DeepCaps+ obtains an accuracy of 91.63% for the CIFAR-10 dataset using 7.3M parameters while taking 1.71ms to test a single image. With 67.56% test accuracy on the CIFAR-100 dataset, DeepCaps+ is among the few variants of vector-based CapsNet that can process this dataset providing acceptable results.

ACKNOWLEDGEMENTS

This research has been funded in part or completely by the Computing Hardware for Emerging Intelligent Sensory Applications (COHESA) project. COHESA is financed under the National Sciences and Engineering Research Council of Canada (NSERC) Strategic Networks grant number NETGP485577-15.

REFERENCES

- Ayidzoe, M. A., Yu, Y., Mensah, P. K., Cai, J., Adu, K., and Tang, Y. (2021). Gabor capsule network with preprocessing blocks for the recognition of complex images. *Machine Vision and Applications*, 32(4).
- Chen, J. and Liu, Z. (2020). Mask dynamic routing to combined model of deep capsule network and u-net. *IEEE Transactions on Neural Networks and Learning Systems*, 31(7):2653–2664.
- Deli, A. (2018). HitNet : a neural network with capsules embedded in a Hit-or-Miss layer , extended with hybrid data augmentation and ghost capsules. pages 1–19.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.
- He, J., Cheng, X., He, J., and Xu, H. (2019). Cv-CapsNet: Complex-valued capsule network. *IEEE Access*, 7:85492–85499.
- Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2016). Densely Connected Convolutional Networks.
- Huang, W. and Zhou, F. (2020). DA-CapsNet: dual attention mechanism capsule network. *Scientific Reports*.
- Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2019). Big Transfer (BiT): General Visual Representation Learning.
- Krizhevsky, A., Nair, V., and Hinton, G. (2009). CIFAR-10 and CIFAR-100 datasets.
- LECUN and Y. THE MNIST DATABASE of handwritten digits. <http://yann.lecun.com/exdb/mnist/>.
- Rajasegaran, J., Jayasundara, V., Jayasekara, S., Jayasekara, H., Seneviratne, S., and Rodrigo, R. (2019). DeepCaps: Going Deeper with Capsule Networks.
- Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic Routing Between Capsules. (Nips).
- Shiri, P. and Baniasadi, A. (2021). Convolutional Fully-Connected Capsule Network (CFC-CapsNet). In *ACM International Conference Proceeding Series*.
- Shiri, P., Sharifi, R., and Baniasadi, A. (2020). Quick-CapsNet (QCN): A Fast Alternative to Capsule Networks. In *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, volume 2020-Novem.
- Tao, J., Zhang, X., Luo, X., Wang, Y., Song, C., and Sun, Y. (2022). Adaptive capsule network. *Computer Vision and Image Understanding*, 218:103405.
- Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.
- Yang, S., Lee, F., Miao, R., Cai, J., Chen, L., Yao, W., Kotani, K., and Chen, Q. (2020). RS-CapsNet: An Advanced Capsule Network. *IEEE Access*.