An Improvement for Capsule Networks Using Depthwise Separable Convolution

Nguyen Huu Phong[(✉)] and Bernardete Ribeiro

CISUC, Department of Informatics Engineering,
University of Coimbra, Coimbra, Portugal
{phong,bribeiro}@dei.uc.pt

Abstract. Capsule Networks face a critical problem in computer vision in the sense that the image background can challenge its performance, although they learn very well on training data. In this work, we propose to improve Capsule Networks' architecture by replacing the Standard Convolution with a Depthwise Separable Convolution. This new design significantly reduces the model's total parameters while increases stability and offers competitive accuracy. In addition, the proposed model on 64×64 pixel images outperforms standard models on 32×32 and 64×64 pixel images. Moreover, we empirically evaluate these models with Deep Learning architectures using state-of-the-art Transfer Learning networks such as Inception V3 and MobileNet V1. The results show that Capsule Networks can perform comparably against Deep Learning models. To the best of our knowledge, we believe that this is the first work on the integration of Depthwise Separable Convolution into Capsule Networks.

Keywords: Capsule Networks · Depthwise Separable Convolution · Deep Learning · Transfer Learning

1 Introduction

In our previous research, we performed experiments to compare accuracy and speed of Capsule Networks versus Deep Learning models. We found that even though Capsule Networks have a fewer number of layers than the best Deep Learning model using MobileNet V1 [2], the network performs just slightly faster than its counterpart.

We first explored details of MobileNet V1's architecture and observed that the model utilizes Depthwise Separable Convolution for the speed improvement. The layer comprises of a Depthwise Convolution and Pointwise Convolution which sufficiently reduces the model size and computation. Since Capsule Networks also integrate a Convolution layer in its architecture, we propose to replace this layer with the faster Convolution.

At the time this article is being written, there are 439 articles citing the original Capsule Networks paper [5]. Among these articles, a couple of works attempt to improve speed and accuracy of Capsule Networks. For example, the

authors [1] propose Spectral Capsule Networks which is composed by a voting mechanism based on the alignment of extracted features into a one-dimensional vector space. In addition, other authors claim that by using a Convolutional Decoder in the Reconstruction layer could decrease the restoration error and increase the classification accuracy [4].

Capsule Networks illustrated its effectiveness on MNIST dataset, though much variations of background to models e.g. in CIFAR-10 probably causes the poorer performance [5]. To solve this problem, we argue that primary filters to eliminate such backgrounds should be as important as other parts of Capsule Networks' architecture. Just, improvements on speed and accuracy of these Convolution layers can be bonus points for the network.

After a thorough search of relevant literature, we believe that this is the first work on the integration of Depthwise Separable Convolution into Capsule Networks.

The rest of this article is organized as follows. In Sect. 2, we highlight our main contributions. Next, we analyse the proposed design in Sect. 3. Following, the design of Deep Learning models for the purpose of comparison is illustrated in Sect. 4. Experiments and results are discussed in Sect. 5 accordingly. Finally, we conclude this work in Sect. 6.

2   Contribution

Our main contributions are twofold: first, on the replacement of Standard Convolution with Depthwise Separable Convolution in Capsule Networks' Architecture and, second, on the empirical evaluations of the proposed Capsule Networks against Deep Learning models.

Regarding the design of Capsule Networks, we found that the integration of Depthwise Separable Convolution can significantly reduce the model size, increase stability and yield higher accuracy than its counterpart.

With respect to experimental evaluations of Capsule Networks versus Deep Learning models, the Capsule models perform competitively both on model size and accuracy.

3   Integration of Depthwise Separable Convolution and Capsule Networks

As mentioned in the previous Section, we propose to substitute a Standard Convolution (SC) of Capsule Networks with a Depthwise Separable Convolution (DW). Figure 1 illustrates the architecture of the proposed model. Additionally, we apply this architecture for an American Sign Language (ASL) dataset with 29 signs. We discuss about this dataset more details in Sect. 5.1.

We divide this architecture into three main layers including Initial Filter, Depthwise Separable Convolution Layer and Capsules Layer. In principle, we can change the first Standard Convolution with a Depth Separable Convolution.
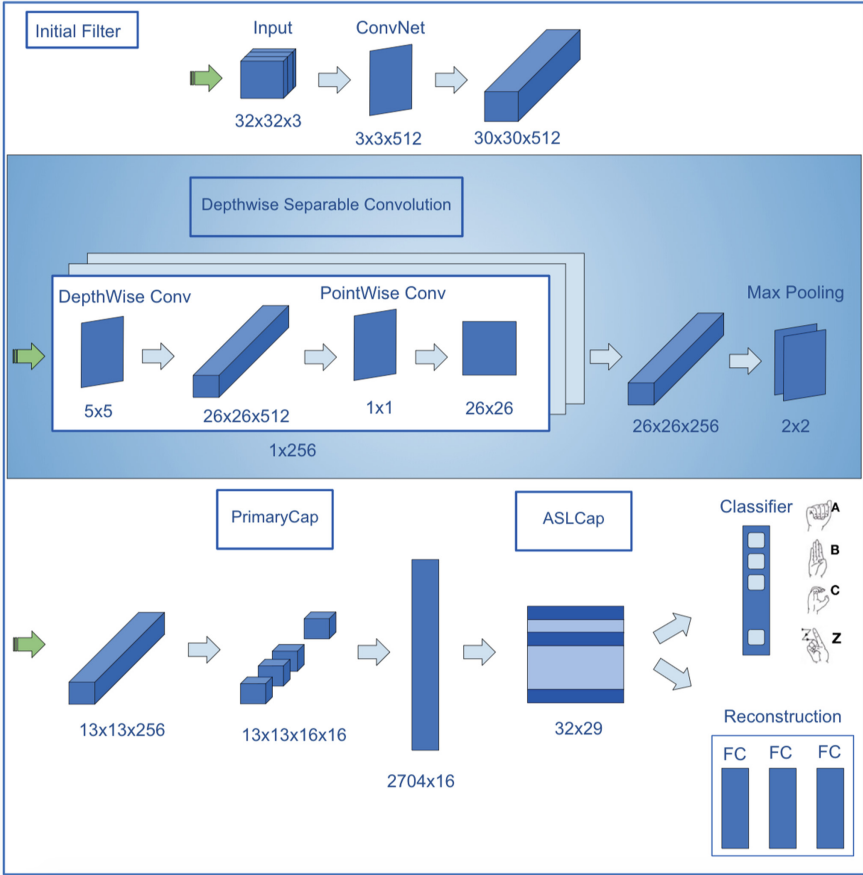
Fig. 1. Depthwise separable convolution capsule architecture.

However, the Input images have only three depth channels which require fewer computations than in the second layer so that we keep the first Convolution intact. The reduction of computation cost using DW is formulated as follows.

An SC takes an input $D_F \times D_F \times M$ feature map F and generates $D_G \times D_G \times N$ feature map G where D_F is the width and height of the input, M is the number of input channels, D_G is the width and height of the output, and N is the number of output channels.

The SC is equipped with a kernel of size $D_K \times D_K \times M \times N$ where D_K is assumed to be a spacial square. M and N are the number of input and output channels as mentioned above.

The output of the feature map G using an SC with kernel stride is 1 and same padding (or padding in short):

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \qquad (1)$$

The computation cost of the SC can be written as:

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \qquad (2)$$

For one channel, the output of feature map Ĝ after Depthwise Convolution:

$$\hat{G}_{k,l,m} = \sum_{i,j,m} \hat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m} \qquad (3)$$

where $\hat{K}$ is the denotation of a Depthwise Convolution with kernel size $D_K \times D_K \times M$ and this Convolution applies m_{th} filter to m_{th} channel of F yields the m_{th} channel in the output.

The computation cost of the Depthwise Convolution is written as:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F \qquad (4)$$

Pointwise Convolution uses 1×1 Convolution, therefore, the total cost after Pointwise Convolution:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \qquad (5)$$

The computation cost of Depthwise Separable Convolution after two convolutions is reduced as:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$
$$= \frac{1}{N} + \frac{1}{D_K^2} \qquad (6)$$

For more details of these computations, please refer to the MobileNet V1 article [5].

In Fig. 1, we show a sample where colour images have a size of $32 \times 32 \times 3$ and filtered by a ConvNet which is equipped with a 3×3 kernel and 512 filters. If the DW Conv is applied in this layer, then the cost of computation can be reduced between 8 and 9 times. Though the input image has a depth of only 3, the reduction of computation cost in this layer is unmatched with hundreds of channels in the second layer.

4   Deep Learning Models

For the purpose of comparison, we also briefly discuss the Deep Learning's architecture as shown in Fig. 2. As we can see from the Figure, this architecture comprises of several crucial layers including Transfer Learning, Multilayer Perceptron (MLP) and Long Short Term Memory (LSTM) layers. In Transfer Learning
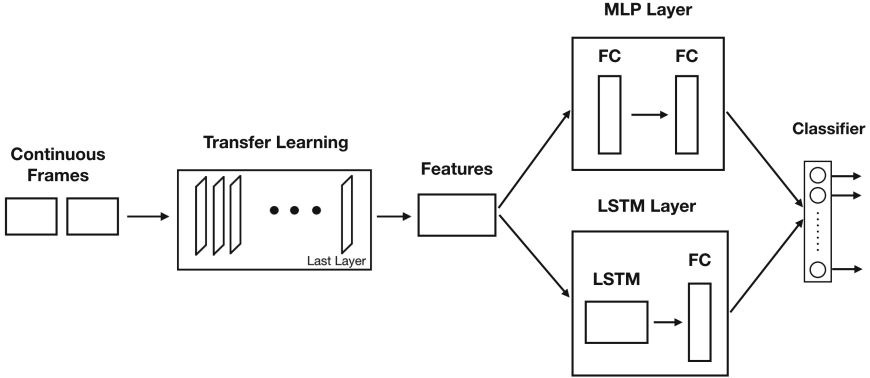
Fig. 2. Deep Learning architecture

layer, we utilize one of the models including Inception V3 [7], DenseNet V201 [3], NASNetMobile [8], MobileNet V1 [2] and MobileNet V2 [6] to extract features from input ASL signs. We then use MLP Layer as a baseline to compare with LSTM Layer. The MLP Layer includes two Fully Connected (FC) Neural Networks each with 512 neurons whereas the LSTM Layer contains one LSTM with 2048 units and one FC. More detail of the architecture can be referred to our earlier work.

5   Experiments and Results

In this section, we first discuss an ASL Dataset that will be used as our testbed. Then we setup experiments to compare our proposed Capsule's architecture using Depthwise Separable Convolution against typical Convolution Networks. Next, we analyse performances of Transfer Learning models including Inception V3, DenseNet V201, NASNet, MobileNet V1 and MobileNet V2 using MLP and LSTM. Finally, we pickup the best of Capsule Networks and challenge the best of Deep Learning models.

5.1   ASL Dataset

For the purpose of comparison with our previous work, we use the same ASL dataset for fingerspelling. The dataset was obtained from Kaggle website and includes 26 signs for letters A to Z with 3 additional signs using in other cases. Each sign contains 3000 samples (200×200 pixels), totally 87000 samples for all signs. Figure 3 shows 10 random samples from this dataset. In these experiments, we use half of the number of samples since the accuracy are similar to that of using all dataset and this also reduces the training time by half. We divide the data into a train set and a test set with the ratio 70 and 30.
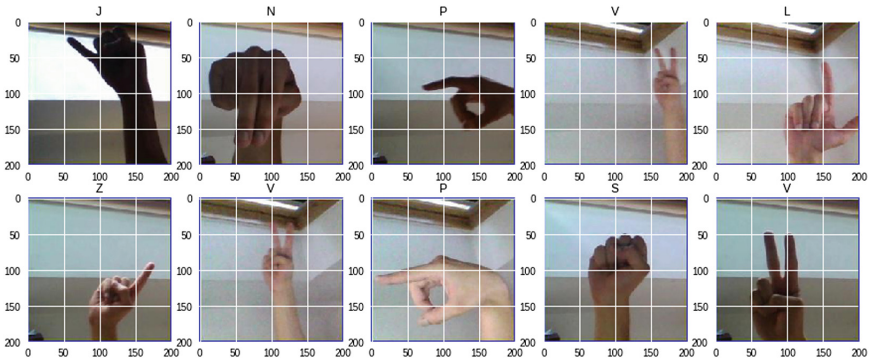
Fig. 3. Random samples from the ASL dataset

5.2   Experiment 1: DW Capsules vs SC Capsules

In this experiment, we perform experimental evaluations of DW Capsules against SC Capsules. First, we vary the size of Convolution's kernel including 9×9, 7×7, 5×5 and 3×3 using the Input image's size of 32×32. Then we add one more ConvNet in the first layer in the Capsule Networks's architecture as shown in Fig. 1. We also provide two variations of Capsule Networks, one for scaling down the total number of model parameters by adding a Max Pooling after the second ConvNet (Mini version) and, another, which increases Input image sizes to 64×64 (Max version).

Figure 4b and d show that DW Capsules perform equivalently or even better on accuracy than SC Capsules. Additionally, SC Capsules seems to be unstable and fluctuating based on kernel's size. Figure 4a shows a similar trend when SC Capsules are more likely to volatile when a half of SC Capsules are outperformed by DW Capsules. Only in Fig. 4c, all SC Capsules achieve higher accuracy than that of DW Capsules. Though this can be a trade of between training epoch and training speed.

5.3   Experiment 2: Deep Learning Models Using MLP and LSTM

In this section, we perform experiments of Deep Learning models using MLP and LSTM on variations of the ASL dataset including $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$ and the whole dataset.

Due to the limited number of pages allowed, we excerpt results from most of all Deep Learning models and preserve only one model for mobile devices and one model for powerful computers i.e. MobileNet V1 and Inception V3. We select MobileNet V1 because of its best accuracy and Inception V3 since the model is faster than DenseNet V201.

The Fig. 5a shows that Inception V3 Transfer Learning model when integrated with LSTM outperforms its version on MLP in all sets of data. Similarly, MobileNet V1 LSTM achieves a higher accuracy than MobileNet V1 MLP.
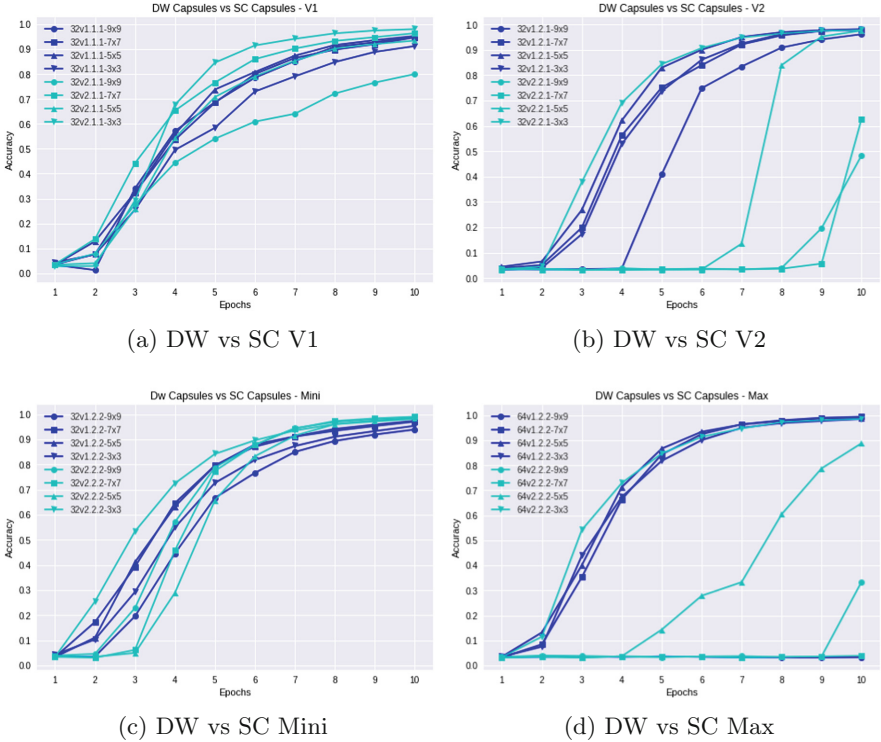
(a) DW vs SC V1

(b) DW vs SC V2

(c) DW vs SC Mini

(d) DW vs SC Max

Fig. 4. DW Capsules (blue) vs SC Capsules (cyan). The numbers 32 and 64 denote Input image sizes 32×32 and 64×64, respectively; v1 stands for DW Capsule whereas v2 stands for SC Capsule. The next number expresses the amount of ConvNets. The last number denotes whether the ConvNet is followed by a Max Pooling (1 if not, 2 if followed). All ConvNets in the primary layer have the kernel sizes vary from 9×9, 7×7, 5×5 to 3×3. (Color figure online)

(a) MLP vs LSTM on Inception V3
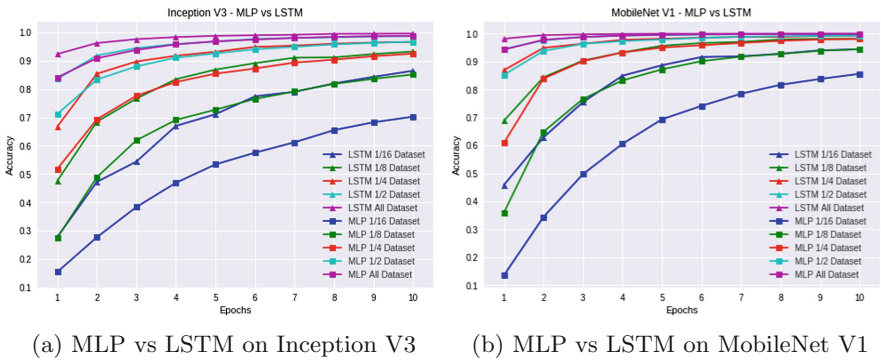
(b) MLP vs LSTM on MobileNet V1

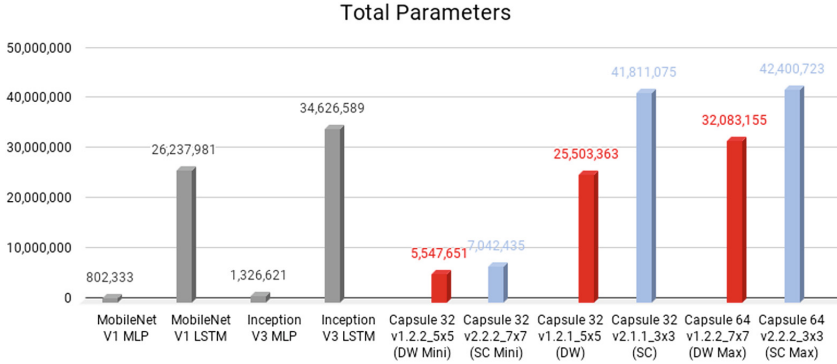**Fig. 5.** Comparisons of MLP and LSTM on Deep Learning models.

Fig. 6. Total parameters of Capsule Networks and Deep Learning models. The numbers 32 and 64 denote Input image sizes 32×32 and 64×64, respectively; v1 stands for DW Capsule whereas v2 stands for SC Capsule. The next number expresses the amount of ConvNets. The last number denotes whether the ConvNet is followed by a Max Pooling (1 if not, 2 if followed). All ConvNets in the primary layer have the kernel sizes vary from 9×9, 7×7, 5×5 to 3×3.

Remarkably, MobileNet V1 performs better than Inception V3 on both MLP and LSTM versions even though the model is mainly built for much smaller devices.

5.4 Capsule Networks vs Deep Learning Models on Model Size and Accuracy

In this Section, we analyse Capsule Networks and Deep Learning Models with respect to the total size of these models as well as accuracy. We select the best DW Capsules and SC Capsules including: (1) Input image size 32×32 pixel followed by two ConvNets and a Max Pooling (Fig. 4c), (2) Input image size 32×32 pixel one for the best accuracy SC (Fig. 4a) and one for the best accuracy DW (Fig. 4b), (3) Input image size 64×64 pixel integrated with two ConvNets and one Max Pooling (Fig. 4d). These Capsule Networks are denoted as DW Mini, SC Mini, DW, SC, DW Max and SC Max, respectively.

Generally, we can see from Fig. 6 that DW Capsules drastically decrease the models' size. More specifically, Capsule 32 DW Mini reduces the model size by 21% while Capsule 64 DW Max shrinks 25% of the total parameters. It can be noted that Capsule 32 DW reduces the number of parameters by 40% which is more than Capsule 32 DW Mini and Max due to one more Convolutions.

In comparison between Capsule Networks and Deep Learning models, we can observe that MobileNet V1 MLP has a smallest model size followed by Inception V3 MLP. Additionally, Capsule 32 DW and Capsule 64 DW Max have smaller sizes than MobileNet LSTM and Inception V3 LSTM.

In terms of accuracy, MobileNet V1 LSTM outperforms all other models. In spite of that, Capsule 64 DW Max reaches the second position and performs better than MobileNet V1 MLP after 10 epochs. In addition, Capsule 64 DW
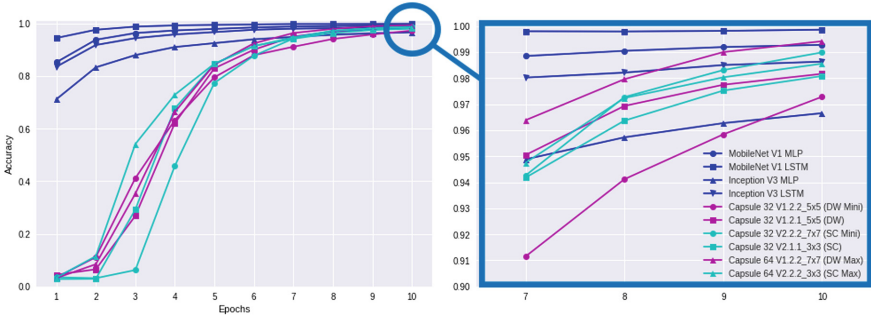
Fig. 7. Capsule Networks vs Deep Learning models

Max outperforms both versions of Inception V3 LSTM and MLP by a large extent. Though Capsule 64 DW has 40 times larger model size than MobileNet V1 MLP and 20% more than MobileNet V1 LSTM, its total parameter is less than that of Inception V3 LSTM Fig. 7.

It can be noticed that Capsule 32 DW Mini though having the smallest size among all Capsule Networks but is outperformed by MobileNet V1 MLP with regards to accuracy. However, when we perform these experiments with more epochs, this gap can be eliminated as Capsule 32 DW Mini yields the accuracy of MobileNet V1 MLP (0.9928) after 17 epochs. A similar trend also occurs to Capsule 32 SC Mini as it reaches the accuracy of MobileNet V1 LSTM (0.9986) after 18 epochs (approximately 1 h training on Tesla K80).

6 Conclusions

In this research, we first propose to replace Standard Convolution in Capsule Networks' Architecture with Depthwise Separable Convolution. Then we perform empirical comparisons of the best Capsule Networks with the best Deep Learning models.

The results show that our proposed DW Capsules remarkably decrease the size of models. Among the chosen Capsule Networks, the total parameters had shrunk roughly by an amount between 21%–25%.

In terms of accuracy, Capsule 64 DW Max performs better than other Capsule models. Though Capsule 32 SC Mini can be a trade-off between the accuracy and the number of parameters.

In comparison with Deep Learning models, Capsule 32 DW Mini has a larger number of parameters, yet it achieves the accuracy of MobileNet V1 after a few more epochs. Meanwhile, Capsule 32 SC Mini can attain that of MobileNet V1 LSTM's accuracy with 3 to 4 times smaller in the number of parameters.

Capsule 32 DW Mini and 64 DW Max outperform Inception V3 MLP and LSTM on accuracy. Moreover, Capsule 64 DW Max occupies 5% less the number of parameters than Inception V3 LSTM though Capsule 32 DW Mini has 4 times larger size than Inception V3 MLP.

After a thorough literature search, we believe that this is the first work that proposes the integration of Depthwise Separable Convolution into Capsule Networks. Additionally, we provide empirical evaluations of the proposed Capsule Networks versus the best Deep Learning models.

In future work, we will apply the proposed Capsule Networks on different datasets and will develop these networks for mobile platforms.

References

1. Bahadori, M.T.: Spectral capsule networks. In: 6th International Conference on Learning Representations (2018)
2. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv Preprint arXiv:1704.04861 (2017)
3. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261–2269. IEEE (2017)
4. Mobiny, A., Van Nguyen, H.: Fast CapsNet for lung cancer screening. arXiv Preprint arXiv:1806.07416 (2018)
5. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems, pp. 3856–3866 (2017)
6. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: inverted residuals and linear bottlenecks. arXiv Preprint arXiv:1801.04381 (2018)
7. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
8. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8697–8710 (2018)