LightLayers: Parameter Efficient Dense and Convolutional Layers for Image Classification

Debesh Jha[1,2], Anis Yazidi[3], Michael A. Riegler[1],
Dag Johansen[2], Håvard D. Johansen[2], and Pål Halvorsen[1,3]

[1] SimulaMet, Norway
[2] UIT The Arctic University of Norway
[3] Oslo Metropolitan University, Norway
debesh@simula.no

Abstract. Deep Neural Networks (DNNs) have become the de-facto standard in computer vision, as well as in many other pattern recognition tasks. A key drawback of DNNs is that the training phase can be very computationally expensive. Organizations or individuals that cannot afford purchasing state-of-the-art hardware or tapping into cloud hosted infrastructures may face a long waiting time before the training completes or might not be able to train a model at all. Investigating novel ways to reduce the training time could be a potential solution to alleviate this drawback, and thus enabling more rapid development of new algorithms and models. In this paper, we propose LightLayers, a method for reducing the number of trainable parameters in deep neural networks (DNN). The proposed LightLayers consists of LightDense and LightConv2D layer that are as efficient as regular Conv2D and Dense layers, but uses less parameters. We resort to Matrix Factorization to reduce the complexity of the DNN models resulting into lightweight DNN models that require less computational power, without much loss in the accuracy. We have tested LightLayers on MNIST, Fashion MNIST, CIFAR 10, and CIFAR 100 datasets. Promising results are obtained for MNIST, Fashion MNIST, CIFAR-10 datasets whereas CIFAR 100 shows acceptable performance by using fewer parameters.

Keywords: Deep Learning, Lightweight model, Convolutional neural network, MNIST, Fashion MNIST, CIFAR-10, CIFAR 100, Weight decomposition

1 Introduction

Deep learning techniques have revolutionized the field of Machine Learning (ML) and gained immense research attention during the last decade. Deep neural network provides state-of-the-art solution in several domains such as image recognition, speech recognition, and text processing [20]. One of the most popular techniques within deep learning is Convolutional Neural Network (CNN), which possesses a structure that is well-suitable specially for image and video processing. A CNN [16] comprises a convolution layer and dense layer. CNN has

emerged as powerful techniques for solving many classification [14] and regression [12] tasks. Additionally, CNN has produced promising results in various applications areas, including in the medical domain, with applicability in diabetic retinopathy prediction [3], endoscopic disease detection [23], and breast cancer detection [19].

Recently, developing deeper and larger architectures has been a common trend in the development of state-of-the-art methods [4]. Most of the time, we can observe that deeper networks especially with large and complex datasets lead to better performance. One of the major drawbacks of CNNs are that they often require an immense amount of training time compared to other classical ML algorithms. Hyperparameter optimization for fine-tuning the model is another challenging task that increases dramatically the overall training time to achieve optimum results from any model. CNN models often require powerful Graphical Processing Units (GPUs) for training, which can span over days, weeks, and even months, with no guarantee that the model will produce satisfactory results. A long training process also consumes a lot of energy and is not considered environmental friendly. Furthermore, long training is demanding in terms of resources as a large amount of memory is required which renders it difficult to deploy it into low-power devices [11]. The requirements for the expensive hardware and high training time complicate the use of models with large number of trainable parameters to be deployed it into portable devices or conventional desktops [20].

A potential way to address these issues is the introduction of lightweight models. A lightweight model can potentially be built by reducing the number of trainable parameters within the layers. In an effort towards reducing the training time and complexity of CNN models, we propose LightLayers, which is a combination of LightDense and LightConv2D layers, that focuses on CNNs and more particularly on creating both a lightweight convolutional layer and a lightweight dense layer that are both easy to train. Lightweight CNN models are computationally cheap and can be deployed various applications for carrying out online estimation. Therefore, the main goal of the paper is to present a general model to reduce the number of parameters in a CNN model so that it can be used in various image processing or other applicable tasks in the future.

The main contributions of the paper are:

- LightLayers, a combination of LightConv2D and LightDense layers, is proposed. Both layers are based on matrix decomposition for reducing the number of trainable parameters of the layers.
- We have investigated and tested the proposed model with four different publicly available datasets: MNIST [16], Fashion MNIST [26], CIFAR10 [13], CIFAR100 [13], and showed that the proposed method is competitive in terms of both accuracy and efficiency when the number of training parameters used are taken into consideration.
- We experimentally show that good accuracy can be achieved by using a relatively small number of trainable parameters with MNIST, Fashion MNIST,

and CIFAR 10 dataset. Moreover, we found there was a significant reduction in the number of trainable parameters as compared to Conv2D.

2 Related Work

In the context of reducing the cost of network model training, several approaches have been presented. For example, Xue et al. [27] presented a Deep Neural Network (DNN) technique for reducing the model size while maintaining the accuracy. For achieving this goal, they used singular value decomposition (SVD) on the weight matrix in DNN, and reconstructed the model based on inherent sparseness of the original matrices. The application of DNN for mobile applications has become increasingly popular. The computational and storage limitation should be taken into account while deploying DNN into such devices.

To address this need, Li et al. [17] proposed two techniques for effectively learning from DNNs with a smaller number of hidden nodes and smaller number of senones set. The details about both the techniques can be found in the literature [17]. Similarly, Xue et al. [28] introduced two SVD based techniques to solve the issue related to DNN personalization and adaptation. Garipov et al. [8] developed a tensor factorization framework for compressing fully-connected layer. The focus of their work was to compress convolutional layers which would potentially excel in image recognition tasks by reducing the memory complexity and high computational cost. Later, Kim et al. [10] proposed energy-efficient kernel decomposition architecture for binary-weight CNNs.

Ding et al. [7] proposed CIRCNN, an approach for representing the weights and processing neural networks by the use of block-circulant matrices. CIRCNN utilizes Fast Fourier Transform based fast multiplication operation which simultaneously reduces the computational and storage complexity causing negligible loss in accuracy. Chai et al. [25] proposed a model for reducing the parameters in deep neural networks via product-of-sums matrix decomposition. They obtained good accuracy on the MNIST and Fashion MNIST datasets with a smaller number of trainable parameters. Another similar work is by Agrawal et al. [2], where they designed a lightweight deep learning model for human activity recognition that is sufficiently computationally efficient to be deployed in edge devices. For more recent works on matrix and tensor decomposition, we refer the reader to [6, 15].

Kim et al. [11] proposed a method for compressing CNN to be deployed into a mobile application. Mariet et al. [18] proposed another efficient neural network architecture that reduces the size of neural network without hurting the overall performance. Novikov et al. [20] converted dense weight matrices of fully-connected layers to Tensor Train [21] format such that the number of parameters are reduced by huge factor by preserving the expressive power of the layer.

Lightweighted networks have gained attention in computer vision (for instance, in the area of real-time image segmentation [9, 22, 24, 29]). Real-time applications are growing because the lightweight models can be an efficient so-

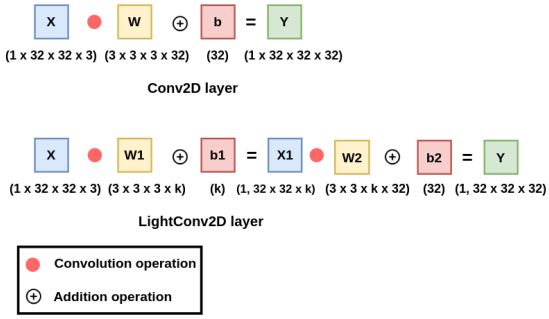

Fig. 1: Comparative diagram of Conv2D layer and LightConv2D layer

lution for resource constraints and mobile devices. Only a lightweight model demands lower memory that leads to a lower computation and faster speed. Therefore, developing a lightweight model can be a good idea for achieving real-time solutions, and it can also be used for other applications too.

The above studies show that there is great potential for lightweight networks for computer-vision tasks. With large amounts of training data, it is likely that a model with huge numbers of trainable parameters will outperform the smaller models—if one can afford the high training costs and resource demands at inference time. However, there is a need for models with low-cost computational power and small memory footprints [11], especially for mobile applications [11] and portable devices. In this repsect, we propose LightLayers that are based on the concept of matrix decomposition. LightLayers uses fewer trainable parameters and shows the state-of-the-art tradeoff between parameter size and accuracy.

3 Methodology

In this section, we introduce the proposed layers. Figure 1 shows the comparison of a Conv2D and a LightConv2D layer. In the LightConv2D layers, we decompose the weight matrix W into $W1$ and $W2$ on the basis of hyperparameter k, which leads to a reduction of the total number of trainable parameters in the network. We follow the same strategy for the LightDense layer. The block diagram of the LightDense layer is shown in Figure 2.

The main objective of building the model is to compare our LightLayers (i.e., the combination of LightConv2D and LightDense layers) with the conventional Conv2D and SeparableConv2D layers. For comparing the performance of the various layers, we have built a simple model from scratch. The block diagram of the proposed model is shown in Figure 3. We used the same hyperparameters and setting for all the experiments. For the LightLayers experiments, we used LightConv2D and LightDense layers (see Figure 3). For the other experiments,

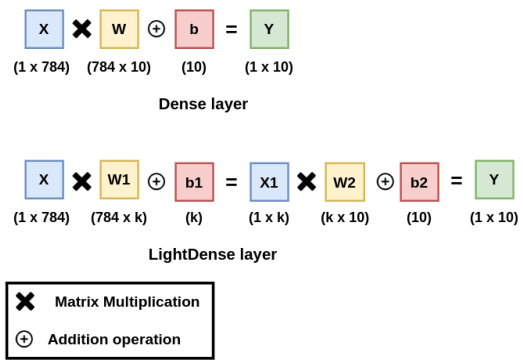

Fig. 2: Comparative diagram of Dense layer and LightDense layer

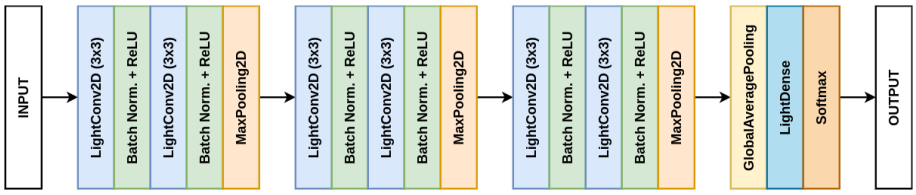

Fig. 3: Block diagram of the architecture used for comparison of the proposed Lightlayers with regular convolution and dense layers. In the case of regular layers, we use regular convolution and dense layers instead of Lightlayers.

we replaced LightConv2D with Conv2D or SeparableConv2D and LightDense with a regular Dense layer.

The model architecture used for experimentation (see Figure 3) comprises two 3×3 convolution layers, each followed by a batch-normalization and ReLU non-linearity as the activation function. We have introduced 2×2 max-pooling, which reduces the spatial dimension of the feature map. We have used three similar blocks of layers in the model followed by the GlobalAveragePooling, LightDense layers with $k = 8$, and a softmax activation function for classifying the input image.

3.1 Description of convolution layers

Conv2D A convolution layer is the most common layer used in any computer vision task and is applied extensively. This layer uses a multidimensional kernel as the weight, which is used to perform convolution operation on the input to produce an output. If the bias is used, then a $1D$ vector is added to the output. Finally, the activation is applied to introduce the non-linearity into the neural network. In this paper, we worked on a $2D$ convolution layer, which uses a $4D$ tensor as the weight.

$$Output = Activation((Input \otimes Weight) + Bias) \tag{1}$$

In the above equation, $\otimes$ represents the convolution operation, and weight represents the kernel.

Dense layer A dense layer is the regular, deeply connected neural-network layer. It is the most common and frequently used layer. It is also known as a fully-connected layer as each neuron receives input from the previous layer.

$$Output = Activation((Input \oplus Weight) + Bias) \tag{2}$$

In the above equation, $\oplus$ represents the matrix multiplication instead of convolution operation as above.

Separable Conv2D Separable convolution, also known as depth-wise convolution, is used in our experiment. We use depth-wise separable 2D convolution to compare the performance of our model. It first applies a depth-wise spatial convolution, i.e., performing convolution operation on each input channel independently. After that, it is followed by a point-wise convolution, i.e., a 1×1 convolution. Point-wise, convolution controls the number of filters in the output feature maps.

4 Experimental Setup

For the experiments, we use the same number of layers, filters, filter sizes, and activation functions in every model for the individual dataset. We have modified the existing Dense and Conv2D layer in such a way that the number of trainable parameters decreases with some decrease in the accuracy of the model. In particular, we use three types of layers for this experiment, i.e., Conv2D, SeparableConv2D, and LightLayers. First, we run the model using Conv2D layers. The Conv2D layer is replaced by SeperableConv2D and run again. Again, we replace SeperableConv2D with the LightLayers and run the model.

In the modified layers, we introduced the hyperparameter k to control the number of trainable parameters in the LightDense and LightConv2D layer. In the LightDense layer, we set k to 8. In the LightConv2D layer, k varies between 1 to 6, and more could be set depending on the requirement. The values of the k are chosen empirically. We only replace the Conv2D layer with the LightConv2D layer and Dense layer with the LightDense layer of the proposed lightweight model. The rest of the network architecture remains the same.

4.1 Implementation Details

We have implemented proposed layers using the Keras framework [5] and TensorFlow 2.2 [1] as backend. The implementation can be found at GitHub[1]. We performed all the experiments on an NVIDIA GEFORCE GTX 1080 system, which has 2560 NVIDIA CUDA Cores with 8 GB GDDR5X memory. The system was running on Ubuntu 18.04.3 LTS. We used a batch size of 64. All the experiments were run, keeping all the hyperparameters (i.e., learning rate, optimizer, batch size, number of filters, and filter size) the same. We have trained all the models for 20 epochs. After each convolution layer, batch normalization is used, which is activated by the Rectified linear unit (ReLU).

[1] `https://github.com/DebeshJha/LightLayers`

4.2 Datasets

To evaluate LightConv2D layer and LightDense layer, we have performed experiments using various datasets.

MNIST Database Modified National Institute of Standards and Technology (MNIST) [16] is the primary dataset for computer vision tasks introduced by LeCun et al. in 1998. MNIST comprises 10 classes of handwritten digits with $60,000$ training and $10,000$ testing images. The resolution of the images in the MNIST dataset is 28×28. There is a huge recent advancement in ML and Deep Learning (DL) algorithms. However, the MNIST remains a common choice for learners and beginners. The reason is that it is easy to deploy, test, and compare an algorithm on a publicly available dataset. The dataset can be downloaded from `http://yann.lecun.com/exdb/mnist/`.

Fashion MNIST Database Fashion MNIST [26] is a 10 class of $70,000$ grayscale images of size 28×28. Han et al. released a novel image dataset that could be used for benchmarking ML algorithms. Their goal was to replace the MNIST database with a new database. The images of the Fashion MNIST database are more challenging as compared to the MNIST database. It contains natural images such as t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. The database can be downloaded from `https://github.com/zalandoresearch/fashion-mnist`.

CIFAR-10 Database CIFAR-10 [13] is a commonly established dataset for computer-vision tasks. It is especially used for object recognition tasks. CIFAR-10 contains $60,000$ color images of size 32×32. It also has 10 classes of images. Each class contains 6000 images per class. The classes contain datasets of cars, birds, cats, deer, dogs, horses, and trucks. The dataset can be downloaded from `https://www.cs.toronto.edu/~kriz/cifar.html`.

CIFAR-100 Database CIFAR-100 [13] is also collected by the team of Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. This database is similar to the previous CIFAR-10 database. The 100 classes of the database consist of images such as beaver, dolphin, flatfish, roses, clock, computer keyboard, bee, forest, baby, pine, tank, etc. Each class of the database contains 600 images each. This dataset contains 500 training examples and 100 testing examples per class. The dataset can be found on the same webpage as CIFAR-10.

5 Results

In this section, we present and compare the experimental results of the Conv2D, SeperableConv2D, and LightLayers models on the MNIST, Fashion MNIST, CIFAR-10, and CIFAR 100 datasets. Table 1 shows the summary of result comparison of Conv2D, SeperableConv2D, and LightLayers on MNIST dataset. Based on Conv2D and SeperableConv2D, we propose Layers and show improvement over both layers. The concept of LightLayers are based on weight matrix

Table 1: Results on **MNIST** test dataset (Number of epochs = 10, Batch size = 64, Learning rate = $1e-3$, Number of filters = $[8, 16, 32]$).

Method	**Parameters**	**Test Accuracy**	**Test Loss**
Conv2D	18,818	0.9887	0.018
SeparableConv2D	3,611	0.9338	0.2433
LightLayers ($K = 1$)	2,649	0.9418	0.1327
LightLayers ($K = 2$)	4,392	0.9749	0.0554
LightLayers ($K = 3$)	**6,135**	**0.9775**	**0.0513**
LightLayers ($K = 4$)	7,878	0.9720	0.0704

Table 2: Results on **Fashion MNIST** test dataset (Number of epochs = 10, Batch size = 64, Learning rate = $1e-3$, Number of filters = $[8, 16, 32]$).

Method	**Parameters**	**Test Accuracy**	**Test Loss**
Conv2D	18,818	0.9147	0.1468
SeparableConv2D	3,611	0.8725	0.3175
LightLayers ($K = 1$)	2,649	0.789	0.6752
LightLayers ($K = 2$)	4,392	0.8452	0.4247
LightLayers ($K = 3$)	6,135	0.8695	0.3708
LightLayers ($K = 4$)	7,878	0.8623	0.6184
LightLayers ($K = 5$)	**9,621**	**0.8820**	**0.2810**
LightLayers ($K = 6$)	11,364	0.8733	0.3986

decomposition. This is the main motivation behind comparison of the proposed layers with Conv2D and SeperableConv2D.

The hyperparameters used are described in the caption of the Table 1. We can see that the result of the proposed LightLayers is comparable to that of Conv2D and SeperableConv2D in terms of test accuracy. When we compare the LightLayers with Conv2D, in terms of the number of parameters used, it uses only $\frac{1}{3}$ of parameters of Conv2D, which is more efficient with only 1% drop in terms of test accuracy. LightLayers with hyperparameter $k = 3$ achieves the highest test accuracy. However, for the other values of k as well there is only minimal variation in test accuracy.

Table 2 shows the results for different layers for the model trained on the Fashion MNIST dataset. From the table, we can see that the proposed model (LightLayers) with hyperparameter $k = 5$ uses only half of the parameters with around drop 3% drop in terms of test accuracy with the Fashion MNIST dataset. However, when we compare the quantitative results with SeperableConv2D, our proposed LightLayers achieves better test accuracy with the trade-off in number of trainable parameters.

Table 3: Evaluation results on test set of **CIFAR10** dataset (Number of epochs = 20, Batch size = 64, Learning rate = $1e-4$, Number of filters = $[8, 16, 32, 64]$). The 'Params' in the bold represents total number of parameters.

Method	**Parameters**	**Test Accuracy**	**Test Loss**
Conv2D	76,794	0.6882	0.9701
SeparableConv2D	14,440	0.5953	1.3263
LightLayers ($K = 1$)	5,937	0.3686	1.6723
LightLayers ($K = 2$)	9,592	0.4596	1.5372
LightLayers ($K = 3$)	13,247	0.4937	1.5287
LightLayers ($K = 4$)	16,902	0.5319	1.3214
LightLayers ($K = 5$)	**20,557**	**0.5576**	**1.2122**

Table 4: Evaluation on **CIFAR100** test set (Number of epochs = 20, Batch size = 64, Learning rate = $1e-4$, Number of filters = $[8, 16, 32, 64]$).

Method	**Parameters**	**Test Accuracy**	**Test Loss**
Conv2D	82,644	0.3262	2.6576
SeparableConv2D	20,290	0.2207	3.2108
LightLayers ($K = 1$)	6,747	0.0275	4.2391
LightLayers ($K = 2$)	10,402	0.0398	4.1836
LightLayers ($K = 3$)	14,057	0.0559	4.0304
LightLayers ($K = 4$)	17,712	0.0551	3.9978
LightLayers ($K = 5$)	**21,367**	**0.0589**	**4.0009**

Table 3 shows the results on CIFAR 10 dataset. On this dataset as well, the proposed method is 3.75 times computationally efficient in terms of parameters it uses. However, there is a drop in accuracy of around 13%. Nevertheless, for some tasks the efficiency can be more important than the reduced accuracy.

Similarly, we have trained and tested the proposed model on the CIFAR 100 dataset, where the test accuracy of the proposed layers is much lower as compared to the Conv2D. This is obvious because CIFAR 100 consists of 100 classes of images that are difficult to generalize with such a small number of trainable parameters. However, the total number of parameters used is still around 4 times less than that of Conv2D. The total number of trainable parameter for Conv2D is 82,644, and for LightLayers, it is only 21,367. More details on the test accuracy and test loss can be found from Table 4.

From the experimental results, we can say that LightLayers has the following advantages:

- It requires less trainable parameters than Conv2D which is an important factor to implement it in different applications where heavy trainable parameters could not be beneficial.
- Due to less parameters, the space taken by the weight file is smaller which makes it more suitable to devices where storage space is limited.

6 Ablation Study

Let us consider that the input size is 784, and the number of output features is 10. Therefore, the weight matrix W is 784×10 resulting in 7840 trainable parameters. Now, in the LightDense layer, we decompose the weight matrix W into two smaller matrix $W1$ and $W2$ of lower dimension using the hyperparameter k.

Here, $W1 = [784, k]$ and $W2 = [k, 10]$ values from the above example, the total number of trainable parameters in the LightDense layer becomes $786 \times k + k \times 10$. Now, if $k = 1$, then trainable parameters are 796, and if $k = 2$ the number of trainable parameters becomes $1,588$, and so on.

Next, consider the weight decomposition in the LightConv2D layer. If the input is $32 \times 32 \times 3$, the number of filters is 32, and the kernel size is 3×3, then the filters size becomes $3 \times 3 \times 3 \times 32$. This means that the total number of trainable parameters is 864. Now, we will decompose the kernel W into $W1$ and $W2$ using hyperparameter k. Here, $W1$ is $3 \times 3 \times 3 \times k$ and $W2$ is $3 \times 3 \times k \times 32$. If k is 1, then the total number of trainable parameters becomes $27 + 288$, which is equals to 315.

From the ablation study, we see that the number of trainable parameters used is less in LightLayers compared to the Conv2D and Dense layers. Overall, we can argue that the proposed LightLayers approach has the potential to be a powerful solution to solve the problem of excessive parameter used by traditional deep-learning approaches. However, our LightLayers model needs further improvement for successfully implementing it on a larger datasets with high resolution images. We can conclude that further investigating matrix weight decomposition is important and other similar studies are necessary to reach the goal of lightweight models in the near future.

7 Conclusion

In this paper, we propose the LightLayers model, which uses matrix decomposition to help to reduce the complexity of the deep learning network. With the extensive experiments, we observed that changing the value of hyperparameter k yields a trade-off between model complexity in terms of the number of trainable parameters and performance. We compare the accuracy of the LightLayers model with Conv2D. An extensive evaluation shows the tradeoffs in terms of parameter uses, accuracy and computation. In the future, we want to train LightLayers on different publicly available datasets. We also aim to develop efficient techniques for finding the optimal value of k automatically. Further research will be required to find suitable algorithms and implementations that will scale this approach to a biomedical datasets.

References

[1] Abadi, M., Barham, P., et al.: Tensorflow: A system for large-scale machine learning. In: Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI). pp. 265–283 (2016)

[2] Agarwal, P., Alam, M.: A lightweight deep learning model for human activity recognition on edge devices. arXiv preprint arXiv:1909.12917 (2019)

[3] Arcadu, F., Benmansour, F., Maunz, A., Willis, J., Haskova, Z., Prunotto, M.: Deep learning algorithm predicts diabetic retinopathy progression in individual patients. NPJ digital medicine 2(1), 1–9 (2019)

[4] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)

[5] Chollet, F., et al.: Keras (2015), `https://keras.io`

[6] Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems. pp. 1269–1277 (2014)

[7] Ding, C., Liao, S., Wang, Y., Li, Z., Liu, N., Zhuo, Y., Wang, C., Qian, X., Bai, Y., Yuan, G., et al.: Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In: Proceedings of the IEEE/ACM International Symposium on Microarchitecture. pp. 395–408 (2017)

[8] Garipov, T., Podoprikhin, D., Novikov, A., Vetrov, D.: Ultimate tensorization: compressing convolutional and fc layers alike. arXiv preprint arXiv:1611.03214 (2016)

[9] Jiang, W., Xie, Z., Li, Y., Liu, C., Lu, H.: Lrnnet: A light-weighted network with efficient reduced non-local operation for real-time semantic segmentation. In: Proceedings of International Conference on Multimedia & Expo Workshops (ICMEW). pp. 1–6 (2020)

[10] Kim, H., Sim, J., Choi, Y., Kim, L.S.: A kernel decomposition architecture for binary-weight convolutional neural networks. In: Proceedings of the Annual Design Automation Conference. pp. 1–6 (2017)

[11] Kim, Y.D., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv preprint arXiv:1511.06530 (2015)

[12] Kleinbaum, D.G., Dietz, K., Gail, M., Klein, M., Klein, M.: Logistic regression (2002)

[13] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[14] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of Advances in neural information processing systems. pp. 1097–1105 (2012)

[15] Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., Lempitsky, V.: Speeding-up convolutional neural networks using fine-tuned cp-decomposition. arXiv preprint arXiv:1412.6553 (2014)

[16] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
[17] Li, J., Zhao, R., Huang, J.T., Gong, Y.: Learning small-size dnn with output-distribution-based criteria. In: Proceedings of the conference of the international speech communication association (2014)
[18] Mariet, Z., Sra, S.: Diversity networks: Neural network compression using determinantal point processes. arXiv preprint arXiv:1511.05077 (2015)
[19] McKinney, S.M., Sieniek, M., Godbole, V., Godwin, J., Antropova, N., Ashrafian, H., Back, T., Chesus, M., Corrado, G.C., Darzi, A., et al.: International evaluation of an ai system for breast cancer screening. Nature 577(7788), 89–94 (2020)
[20] Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D.P.: Tensorizing neural networks. In: Advances in neural information processing systems. pp. 442–450 (2015)
[21] Oseledets, I.V.: Tensor-train decomposition. SIAM Journal on Scientific Computing 33(5), 2295–2317 (2011)
[22] Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147 (2016)
[23] Thambawita, V., Jha, D., Hammer, H.L., Johansen, H.D., Johansen, D., Halvorsen, P., Riegler, M.A.: An extensive study on cross-dataset bias and evaluation metrics interpretation for machine learning applied to gastrointestinal tract abnormality classification. ACM Transactions on Computing for Healthcare 1(3) (2020)
[24] Wang, Y., Zhou, Q., Liu, J., Xiong, J., Gao, G., Wu, X., Latecki, L.J.: Lednet: A lightweight encoder-decoder network for real-time semantic segmentation. In: Proceedings of International Conference on Image Processing (ICIP). pp. 1860–1864 (2019)
[25] Wu, C.W.: Prodsumnet: reducing model parameters in deep neural networks via product-of-sums matrix decompositions. arXiv preprint arXiv:1809.02209 (2018)
[26] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
[27] Xue, J., Li, J., Gong, Y.: Restructuring of deep neural network acoustic models with singular value decomposition. In: Interspeech. pp. 2365–2369 (2013)
[28] Xue, J., Li, J., Yu, D., Seltzer, M., Gong, Y.: Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In: Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6359–6363 (2014)
[29] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 325–341 (2018)