Thin MobileNet: An Enhanced MobileNet Architecture

Debjyoti Sinha
Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, USA
debsinha@iupui.edu

Mohamed El-Sharkawy
Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, USA
melshark@iupui.edu

***Abstract*—In the field of computer, mobile and embedded vision Convolutional Neural Networks (CNNs) are deep learning models which play a significant role in object detection and recognition. MobileNet is one such efficient, light-weighted model for this purpose, but there are many constraints or challenges for the hardware deployment of such architectures into resource-constrained micro-controller units due to limited memory, energy and power. Also, the overall accuracy of the model generally decreases when the size and the total number of parameters are reduced by any method such as pruning or deep compression. The paper proposes three hybrid MobileNet architectures which has improved accuracy along-with reduced size, lesser number of layers, lower average computation time and very less overfitting as compared to the baseline MobileNet v1. The reason behind developing these models is to have a variant of the existing MobileNet model which will be easily deployable in memory constrained MCUs. We name the model having the smallest size (9.9 MB) as Thin MobileNet. We achieve an increase in accuracy by replacing the standard non-linear activation function ReLU with Drop Activation and introducing Random erasing regularization technique in place of drop out. The model size is reduced by using Separable Convolutions instead of the Depth-wise separable convolutions used in the baseline MobileNet. Later on, we make our model shallow by eliminating a few unnecessary layers without a drop in the accuracy. The experimental results are based on training the model on CIFAR-10 dataset.**

***Index Terms*—Convolutional Neural Network (CNN), MobileNet, Depthwise Separable Convolutions, Separable Convolutions, Drop Activation, Random Erasing, Keras, Tensorflow, CIFAR-10, Blue-box 2.0.**

I. Introduction

Deep Convolutional neural networks gained popularity when AlexNet [1] won the ImageNet Challenge in the year 2012. Since then, the domain of deep learning expanded exponentially. Many standard algorithms for computer vision like Canny's algorithm or HOG have been replaced by deep learning models like SqueezeNet [2], SqueezeNext [3], ResNet [4], Inception [5], etc. There also had been significant developments in new optimization techniques, non-linear activation functions, training methods, etc. In order to get higher accuracies, the models are made deeper and complex. The advent of deeper and complicated models [1, 2, 3, 4, 6] has led to the development of a wide number of hardware architectures like GPUs, Bluebox 2.0, S32V234 MCU, etc to increase the speed of the training process and deploy the models for various computer vision applications. But increasing the depth and complexity of a model increases the size and computation cost making it less efficient for hardware deployment, especially in resource- constrained mobile and embedded platforms. In real-time applications such as image classification [8], captioning [9], object detection [10,23,33] and semantic segmentation [34] in autonomous driving, the inference time and accuracy are very important factors to one's safety. So, it becomes necessary to have a model which is very accurate, requires less memory complexity and has less computation time for its utilization in real-time scenarios. To solve the above-mentioned problem, a lighter model called MobileNet [6] with a fewer number of parameters, and less computation time was invented. MobileNet does not use standard convolutions, instead, it uses Depthwise separable convolutions [6, 7] which requires only one-eighth of the computation cost. To solve the above-mentioned problem, a lighter model called MobileNet [6] with lesser number of parameters, and less computation time was invented. MobileNet does not use standard convolutions, instead it uses Depthwise separable convolutions [6, 7] which requires only one-eighth of the computation cost. MobileNet has got two hyperparameters: width multiplier and resolution multiplier [6]. When we use them to reduce the size of the network, the accuracy drops by a good margin. In this paper, we propose a new MobileNet architecture called the Thin MobileNet in which we have increased the accuracy and simultaneously reduced the model size, the computation time, and the overfitting problem in the baseline model. We also have two other models which perform better than the baseline model. The paper is divided into five sections: Section I. is an introduction to the work which we have done. Section II. is a discussion about the existing baseline MobileNet model. Section III. describes the modifications which we have introduced on the existing model. Section IV. lists the hardware and software requirements. Section V. shows the experimental results obtained after training the baseline MobileNet v1 model and the modified architectures on the CIFAR-10 dataset. Section VI. concludes the paper by discussing the advantages of the proposed Thin MobileNet model, further improvements which can be made on it, and its application.

978-1-7281-3885-5/19/$31.00©2019 IEEE

TABLE I
MOBILENET V1 TRAINED ON CIFAR-10 DATASET FOR 200 EPOCHS

Model name	MobileNet v1 baseline
Accuracy	84.30%
Model size	39.1 MB
Computation time per epoch	31s
Total number of parameters	3,239,114

II. PREVIOUS WORK

The unique characteristic of MobileNet is that it uses depth- wise separable convolutions which can be thought of standard convolutions split into depthwise convolution and 1 by 1 pointwise convolution. In depthwise convolution, each input channel is filtered separately. This is followed by 1 by 1 pointwise convolution which linearly combines all the depthwise convolution outputs. The drastic reduction in model size and computation cost is the result of this factorization. The computation cost is reduced by about 8 to 9 times as compared to the usage of standard convolutions. There are two hyper-parameters [6] present in the network: 1. Width multiplier and 2. Resolution multiplier. The function of the width multiplier is to slim the network at each layer uniformly by reducing the number of filters, while the resolution multiplier can be used to decrease the image resolution. We can obtain some smaller models by playing with different values of the width multiplier and resolution multiplier, but it always leads to a reduction in the overall accuracy. Both Batch normalization [11] and non–linear activation function ReLU [26] have been used after each convolution layer in the model. Downsampling is done with the use of strided convolutions in the first convolution as well as in the depthwise convolution layers. These are followed by an Average Pooling layer, Fully Connected layer and Softmax classifier. In total, there are 28 layers in the baseline MobileNet, if the Depthwise convolution and Pointwise convolution layers are considered separately. Fig. 1 shows a schematic diagram of the MobileNet v1 baseline architecture with all the layers. TABLE I depicts the accuracy, the model size, computation time per epoch and the total number of parameters of the baseline MobileNet architecture trained on the CIFAR-10 dataset.

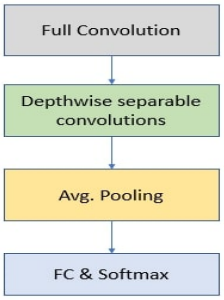

Fig. 1. MobileNet v1 baseline model [6]

III. THE PROPOSED MODIFICATONS

In this part, we introduce some modifications like using Separable Convolutions instead of Depthwise Separable Convolution to reduce the size of the network. Also, Drop Activation and Random Erasing methods are introduced to improve the overall performance of the model. After introducing these modifications, we make the MobileNet shallower by eliminating some layers from the network to reduce the number of parameters and computation overhead without compromising on the accuracy.

A. Modification 1 - MobileNet architecture with Separable Convolutions [7] instead of Depthwise Separable convolutions: A depthwise separable convolution is implemented by first performing channel-wise convolution (filtering each input channel separately) and then linearly integrating those outputs with the help of pointwise convolutions. In the baseline model, the depthwise convolution layers and the pointwise convolution layers are defined separately. In our model we use separable convolutions [7] instead of depthwise separable convolutions which combines the depthwise layer and the pointwise layer into one layer and there is no need to define them separately. We have used the Keras framework in which Separable Convolution 2D is defined. Here, the pointwise initializer, pointwise regularizer and pointwise constraint for the pointwise convolution are defined inside the same init() function as the depthwise initializer, regularizer and constraints. This reduces the network to 14 layers, keeping the basic functionality of the depthwise separable convolutions intact, but does not do much in increasing the accuracy of the network. The model size becomes 26.9 MB (12.2 MB less than the baseline) and the total number of parameters becomes 2,158,826. Fig 2. shows the difference between the core layers of the original network and the modified network. The computation time per epoch is now reduced to 21s.

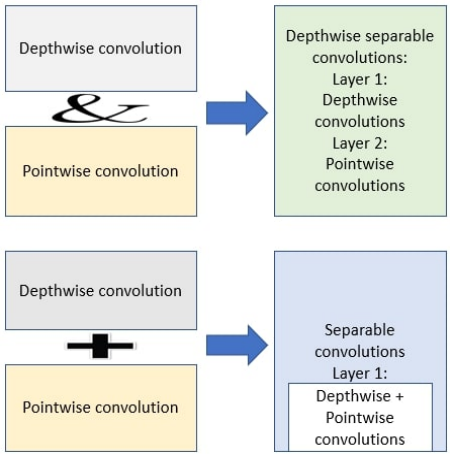

Fig. 2. Comparison between Depthwise separable convolutons and Separable convolutions- In Depthwise separable convolutons, the DW and PW convolutions are defined in separate layers that is Layer 1 and Layer 2 respectively. Separable convolutions consists of DW and PW convolutions integrated in a single layer (Layer 1).

B. Modification 2 - MobileNet architecture with Drop-Activation layers instead of ReLU : Regularization [22, 31, 32] has been an important part of Deep learning networks. Sometimes, regularizations individually work quite well but when they are combined, they do not enhance the overall performance of the network. For example, if we are using Batch normalization and Dropout [20, 28, 29] in our model, the performance drops as Batch normalization requires the statistical variance should be same in both training and testing scenarios. Dropout changes the variance of the layers output when the model is in testing phase after its training phase.

To make our model more robust, accurate and compatible to other regularization techniques, the non-linear activation function ReLU is replaced by Drop-Activation [18] layers. The non-linear function is randomly deactivated and activated during training i.e. randomness is introduced into the activation function. The nonlinearity in the activation function is kept with a probability P and dropped with a probability of (1-P). Here, the non-linear activation function considered is ReLU and the way of applying ReLU to the network is modified by using Drop-Activation layer. Suppose, f(x) is the non-linear operator. If x is the input and if the activation is ReLU then, f(x)=0, when $x \leq 0$ and f(x)=x, when, $x \geq 0$.

Fig. 3 illustrates that if the input is negative, then in case of standard ReLU, the output is zero (the graph does not go to the third quadrant). Fig. 4 illustrates that in case of drop-activation, suppose if the input is negative having the probability P=0.75, the output is zero as in normal ReLU, but the identity function I is also used with a probability of 0.25. That means, the graph may go towards the third quadrant with a probability of 0.25. Thus, we switch between standard ReLU (75%) and Identity mapping function (25%). The value of drop-activation probability P should be somewhere in between 0 and 1, since P=1 means all the non-linearities have been kept and P=0 means all the non-linearities have been dropped.

In the testing phase, we average the realizations of P and get a deterministic non-linear function as a result. We use this function for testing. Mathematically speaking, we calculate the expectation of the equation of the standard non-linear function we are using during training, for example, ReLU in this case and get Leaky ReLU [30] with slope (1-P) as our deterministic non-linear function for the testing phase.

The advantages are that, Drop-activation technique increases the accuracy of the model to 85.14% and reduces the over-fitting problem present in the baseline architecture. The difference between the training and testing accuracy is only 0.2 now. It is also compatible with other training methods like Batch normalization and regularization techniques like data augmentation. Fig. 5 shows that all the Separable convolution layers are followed by Batch normalization and Drop-Activation layers. TABLE II depicts the output shape produced after each layer and the number of parameters involved in each layer.

C. Modification 3- Use of Random erasing in the network: Random erasing [19] is a kind of data augmentation [21, 27] method where we select rectangular regions in an image I

Fig. 3. Graph of Standard ReLU function [18]

Fig. 4. Graph of Drop-Activation function [18]

in a mini-batch randomly, and erase the pixels of that region and substitutes it with random values. It enhances the ability of generalization of a convolutional neural network. When some regions of an object in an image are occluded, a CNN model can be unsuccessful in recognizing the object from its global structure due to poor generalization power. To curb this problem, Random erasing [19] technique was established. Random erasing has the following input parameter values (base setting) [19] in our model:

1. Probability of erasing p = 0.5.
2. Maximum erasing area ratio S_h = 0.4.
3. Minimum erasing area ratio S_l = 0.02.
4. Erasing aspect area ratio r_e = 0.3.
5. Erasing aspect ratio range = [0.3,3.33].

The Erasing area ratio is equal to S_e/S, where S is the area of the original image and S_e is the erased area. It helps us to further enhance the accuracy to 85.21% and to reduce the overfitting problem in our model. Now, the difference between the validation loss and the training loss is roughly 0.1 which is an acceptable value in any CNN object recognition model. There is a little increase in computation time per epoch which is currently 23s but still, it is much less than that of the baseline architecture which has this value as 31s.

D. Modification 4- Eliminating unnecessary layers: We can make our architecture shallower by eliminating some layers which are repetitive [6]. The five layers with output shape (2,2,512) that is layers, 9 to 13 as illustrated in TABLE II ,are eliminated as they contribute about 41% of the total number

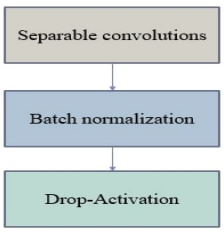

Fig. 5. Modified blocks containing Separable convolutions with Batch normalization and Drop-Activation

TABLE II
MODIFIED NETWORK ARCHITECTURE

Layer / Stride	Output Shape	Parameter
Input layer	32, 32, 3	0
Conv2d/s2	16, 16, 32	864
Separable conv2d/s1	16, 16, 32	1312
Separable conv2d/s2	8, 8, 64	2336
Separable conv2d/s1	8, 8, 128	8768
Separable conv2d/s2	4, 4, 128	17536
Separable conv2d/s1	4, 4, 256	33920
Separable conv2d /s2	2, 2, 256	67840
Separable conv2d /s1	2, 2, 512	133376
Separable conv2d /s1	2, 2, 512	133376
Separable conv2d /s1	2, 2, 512	266752
Separable conv2d /s1	2, 2, 512	266752
Separable conv2d /s1	2, 2, 512	266752
Separable conv2d/s2	1, 1, 512	266752
Separable conv2d /s1	1, 1, 1024	528896
Global average pool/s1	1, 1, 1024	0
FC and Softmax/s1	1,1,10	10250

of parameters. This drastically reduces the model size to 9.9 MB without a decrease in the accuracy.

IV. HARDWARE AND SOFTWARE REQUIREMENTS

(a) Python IDE- Spyder v3.6
(b) Anaconda Navigator 2.0
(c) Open-source API – Keras v2.2.0
(d) Backend framework – Tensorflow-gpu v1.11.0 [12]
(e) Livelossplot package from PyPI
(f) Intel i9 9th generation processor (32GB RAM)
(g) NVIDIA Geforce RTX 1080Ti GPU

We have used the CIFAR-10 dataset for our experiments. It contains 6000 images among which 5000 are training images and the remaining 1000 are testing images. There are ten exclusive classes of images in this dataset.

V. EXPERIMENTAL RESULTS

All the models have been trained from scratch using the CIFAR-10 dataset without using transfer learning technique. The parameters of each model are saved and loaded from a checkpoint file. Fig. 6 shows the plot of accuracy when the MobileNet baseline model is trained. We get an accuracy of 84.30%. We have trained the models with a batch-size of 32 and steps-per-epoch of 1563 for 200 epochs on CIFAR-10 dataset.The optimizer used is Nadam [13, 14] or Nesterov Adam. Nadam is an efficient optimization technique which combines Adam [14], RMSProp [15], and Nesterov momentum [16, 17]. Nadam performs better than SGD [24] and Momentum [25] as it does not overshoot suddenly around the minima.

Model 1: Modified MobileNet architecture after using Separable Convolutions, Drop-Activation and Random Erasing: This model is obtained as a result of using Separable Convolutions instead of Depthwise separable convolutions, Drop-Activation instead of ReLU and Random Erasing as a regularization technique i.e. introducing Modification 1,2 and 3 as mentioned above in section 3. The best accuracy is obtained when we set the Drop-Activation probability P = 0.75, with no

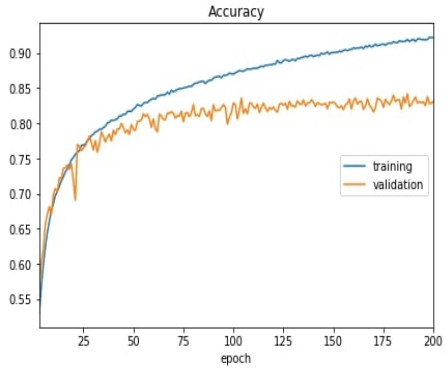

Fig. 6. MobileNet v1 baseline accuracy

TABLE III
THIN MOBILENET ARCHITECTURE

Layer / Stride	Output Shape	Parameter
Input layer	32, 32, 3	0
Conv2d/s2	16, 16, 32	864
Separable conv2d/s1	16, 16, 32	1312
Separable conv2d/s2	8, 8, 64	2336
Separable conv2d/s1	8, 8, 128	8768
Separable conv2d/s2	4, 4, 128	17536
Separable conv2d/s1	4, 4, 256	33920
Separable conv2d /s2	2, 2, 256	67840
Separable conv2d/s2	1, 1, 512	133376
Separable conv2d /s1	1, 1, 1024	528896
Global average pool/s1	1, 1, 1024	0
FC and Softmax/s1	1,1,10	10250

changes in the base setting of the Random erasing parameters. Fig. 7 shows the plot of accuracy when the above modifications are introduced.

TABLE III shows the features of the modified network. This network has improved accuracy, reduced model size, less computation time per epoch and fewer parameters as compared to the MobileNet v1 baseline model. The overfitting is reduced to a large extent, in this case.

Model 2: Modified MobileNet architecture using Drop-Activation and Random Erasing with Depthwise separable convolution: This model is obtained as a result of using Drop-Activation (P = 0.75) instead of ReLU and Random Erasing as in the case of Model 1 without the use of Separable convolution that is we preserve the Depthwise separable convolutions from the baseline architecture and only introduce Modification 2 and 3. modifications are introduced. This model is obtained as a result of using Drop-Activation (P = 0.75) instead of ReLU and Random Erasing as in the case of Model 1 without the use of Separable convolution that is we preserve the Depthwise separable convolutions from the baseline architecture and only introduce Modification 2 and 3. Fig. 8 shows the plot of accuracy for this model. The accuracy increases to 86.42%, while the computation time per epoch and the model size remains almost the same as the MobileNet baseline architecture. The overfitting is a bit more as compared to the Model 1 but it is much less than the baseline model.

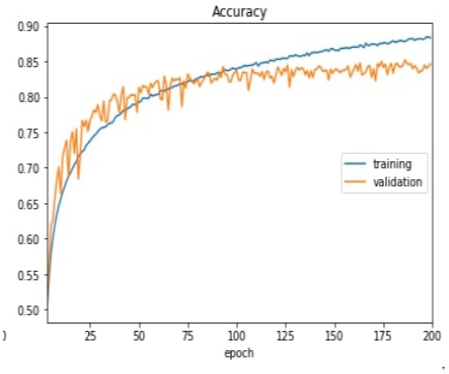

Fig. 7. Model 1 accuracy plot - Modified MobileNet architecture after using Separable Convolutions, Drop-Activation and Random Erasing

TABLE IV
MODEL 3-THIN MOBILENET FEATURES

Accuracy	85.61%
Model size	9.9 MB
Computation time per epoch	14s
Total no. of parameters	8,14,826

Model 3: Thin MobileNet architecture using Separable Convolutions, Drop-Activation and Random Erasing: Modification 4 that is eliminating some repetitive layers (Layers 9 to 13) is introduced along-with the other three modifications into the network to get a shallower version of the MobileNet. Fig. 9 shows the plot of accuracy after introducing these modifications.Fig. 10 shows the building blocks of the Thin MobileNet model. TABLE IV shows the output shape and the number of parameters linked with each layer of the model and TABLE V shows its features. The accuracy of the model improves with a drastic reduction in the model size, number of parameters and the computation time per epoch. The overfitting problem is almost negligible here. This model is suitable for deployment into resource-constrained processors due to its lightness and less computation overhead.

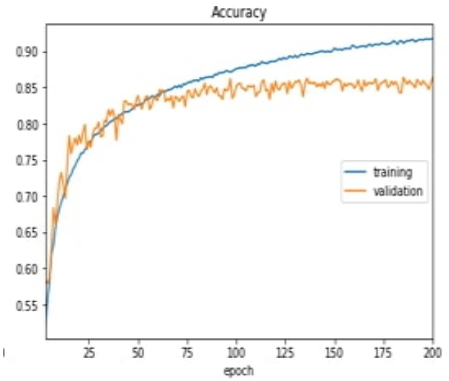

Fig. 8. Model 2 accuracy plot - Modified MobileNet architecture using Drop-Activation and Random Erasing with Depthwise separable convolution

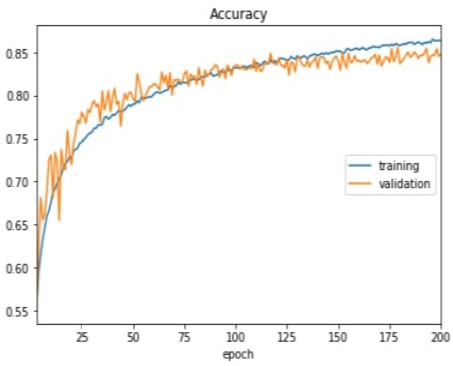

Fig. 9. Model 3 accuracy plot - Thin MobileNet

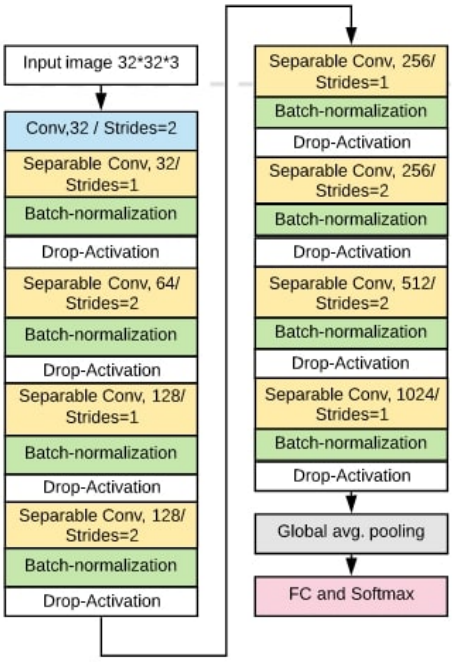

Fig. 10. Thin MobileNet architecture

VI. CONCLUSION

In this paper, we propose an enhanced version of the MobileNet called the Thin MobileNet. We have introduced four modifications to the existing MobileNet baseline model. This new CNN architecture is thinner, more accurate and faster than the MobileNet model and some other popular CNN architectures. Through our experiment, we have increased the accuracy and reduced the size of the network considerably,

TABLE V
MODEL 3-THIN MOBILENET FEATURES

Accuracy	85.61%
Model size	9.9 MB
Computation time per epoch	14s
Total no. of parameters	8,14,826

which makes it suitable for deployment in memory-constrained hardware. When we reduce the depth of the model by using Separable Convolutions instead of Depthwise separable convolutions, the accuracy does not decrease as Drop-activation and Random erasing play an important part in compensating for the accuracy. Generally, a shallow model is not as accurate as a deep model, but introducing randomness into the standard activation function ReLU combined with Random Erasing data augmentation has a good impact on keeping the accuracy of the model much above the baseline MobileNet model accuracy. Random Erasing also has a significant impact on removing the overfitting problem and enhancing the generalization ability of the model. The last modification is eliminating a few layers with the same output shape to make the model shallow, drastically reduces the total number of parameters resulting in a model size of 9.9 MB. Also, Nadam optimizer performs better compared to other optimizers like Adam, RMS Prop and SGD for training and testing the network on CIFAR-10 dataset from scratch. This model is also faster than many other CNN architectures and is safe for real-time embedded applications like object detection and recognition applications. All the proposed models perform better than the baseline MobileNet model trained from scratch on the CIFAR-10 dataset.The best model in terms of accuracy is Model 2 having an accuracy of 86.42% and the best model in terms of model size is Model 3 which we have named as the Thin MobileNet (9.9 MB). The future work involves trying out various combinations of the width multiplier and the resolution multiplier on the three models to achieve more efficient networks. Using optimum values for the width multiplier and the resolution multiplier can further reduce the model size, keeping the accuracy much above the accuracy of the baseline MobileNet model and at the same time beating other benchmark CNN models in terms of both model size and accuracy. Later on, the model will be deployed on to Bluebox 2.0, an autonomous hardware development platform produced by NXP Semiconductors for single-shot object detection [33] application. The Bluebox 2.0 incorporates the computer vision processor S32V234, the multi-core communication processor LS2084A and the radar. micro-controller unit S32R27.

REFERENCES

[1] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S. Awwal, Vijayan K. Asari (2018). The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. .arXiv preprint arXiv: 1803.01164.

[2] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. and Keutzer, K., (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5MB model size. arXiv preprint arXiv:1602.07360.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). Deep Residual Learning for Image Recognition. arXiv preprint arXiv: 1512. 03385

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, (2015). Deep Residual Learning for Image Recognition. arXiv preprint arXiv:1512.03385

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich (2014). Going Deeper with Convolutions. arXiv preprint arXiv:1409.4842

[6] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861

[7] Francois Chollet (2017). Xception: Deep Learning with Depthwise Separable Convolutions. arXiv preprint arXiv: 1610.02357

[8] Chen Wang, Yang Xi. Convolutional Neural Network for Image Classification.

[9] MD. ZAKIR HOSSAIN, FERDOUS SOHEL, MOHD FAIRUZ SHIRATUDDIN, HAMID LAGA (2018). A Comprehensive Survey of Deep Learning for Image Captioning. arXiv preprint arXiv: 1810.04020.

[10] Zhong-Qiu Zhao, Shou-tao Xu, and Xindong Wu (2019). Object Detection with Deep Learning: A Review. arXiv preprint arXiv: 1807.05511.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large scale machine learning on heterogeneous systems, 2015. Software available from tensorflow. org, 1, 2015.

[13] Timothy Dozat (2016). INCORPORATING NESTEROV MOMENTUM INTO ADAM. Workshop track - ICLR 2016. https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ.

[14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[15] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4(2),2012.

[16] Ilya Sutskever, James Martens, George Dahl, Georey Hinton (2013). On the importance of initialization and momentum in deep learning. http://proceedings.mlr.press/v28/sutskever13.pdf

[17] Aleksandar Botev, Guy Lever, David Barber (2016). Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent. arXiv preprint arXiv :1607.01981

[18] Senwei Liang, Yuehaw Khoo, Haizhao Yang (2018). Drop-Activation: Implicit Parameter Reduction and Harmonic Regularization. arXiv preprint arXiv : 1811.05850

[19] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, Yi Yang (2017). Random Erasing Data Augmentation. arXiv preprint arXiv: 1708.04896

[20] J. Ba and B. Frey. Adaptive dropout for training deep neural networks. In NIPS, 2013.

[21] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, Quoc V (2019). Le. Learning Data Augmentation Strategies for Object Detection. arXiv preprint arXiv : 1906.11172

[22] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In ICML,2013

[23] X. Wang, A. Shrivastava, and A. Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. 2017.

[24] Yi Zhou, Junjie Yang, Huishuai Zhang, Yingbin Liang, Vahid Tarokh. SGD CONVERGES TO GLOBAL MINIMUM IN DEEP LEARNING VIA STAR-CONVEXPATH. arXiv preprint arXiv :1901.00451, 2019

[25] Ilya Sutskever, James Martens, George Dahl, Georey Hinton (2013). On the importance of initialization and momentum in deep learning.

[26] Abien Fred M. Agarap. Deep Learning using Rectified Linear Units (ReLU). arXiv preprint arXiv :1803.08375, 2019

[27] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, Quoc V. Le. Unsupervised Data Augmentation for Consistency Training. arXiv preprint arXiv :1904.12848, 2019

[28] Haibing Wu, Xiaodong Gu. Towards Dropout Training for Convolutional Neural Networks. arXiv preprint arXiv : 1512.00242, 2015

[29] Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014) 1929-1958.

[30] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li. Empirical Evaluation of Rectified Activations in Convolutional Network. arXiv preprint arXiv : 1505.00853, 2015.

[31] Pirmin Lemberger. On Generalization and Regularization in Deep Learning. arXiv preprint arXiv : 1704.01312

[32] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for Deep Learning:A Taxonomy. arXiv preprint arXiv :1710.10686, 2017

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. SSD: Single Shot MultiBox Detector. arXiv preprint arXiv :1512.02325, 2016

[34] Xiaolong Liu, Zhidong Deng, Yuhan Yang2. Recent progress in semantic image segmentation. arXiv preprint arXiv :1809.10198, 2018.