R-MnasNet: Reduced MnasNet for Computer Vision

Prasham Shah
IoT Collaboratory IUPUI
Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology
Indianapolis, USA
pashah@purdue.edu

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

***Abstract*—In Deep Learning, Convolutional Neural Networks (CNNs) are widely used for Computer Vision applications. With the advent of new technology, there is an inevitable necessity for CNNs to be computationally less expensive. It has become a key factor in determining its competence. CNN models must be compact in size and work efficiently when deployed on embedded systems. In order to achieve this goal, researchers have invented new algorithms which make CNNs lightweight yet accurate enough to be used for applications like object detection. In this paper, we have tried to do the same by modifying an architecture to make it compact with a fair trade-off between model size and accuracy. A new architecture, R-MnasNet (Reduced MnasNet), has been introduced which has a model size of 3 MB. It is trained on CIFAR-10 [4] and has a validation accuracy of 91.13%. Whereas the baseline architecture, MnasNet [1], has a model size of 12.7 MB with a validation accuracy of 80.8% when trained with CIFAR-10 dataset. R-MnasNet can be used on resource-constrained devices. It can be deployed on embedded systems for vision applications.**

***Index Terms*—Convolutional Neural Networks (CNNs), Computer Vision, R-MnasNet, CIFAR-10, MnasNet.**

I. Introduction

Convolutional neural networks (CNNs) have made noteworthy progress in computer vision applications like image classification [19], object detection [21] etc. Deep Neural Networks become slow because of large number of computations. It difficult t o d eploy s uch s tate-of-the-art C NN m odels on resource-constrained platforms.

As there are limited resources on mobile devices, researchers have been working on designing compact models. By reducing the depth of the network and utilizing less expensive convolutions, the models become more compact. Designing such models is very challenging as there must be a fair trade-off between model size and accuracy.

In this paper, a new architecture, R-MnasNet, is introduced which is designed to work efficiently w hen d eployed on such resource constrained platforms like mobile or embedded devices. This architecture has been derived from its baseline architecture MnasNet. Some modifications were m ade t o this baseline architecture to make it compact with a fair trade-off between model size and accuracy.

The baseline architecture is discussed in section 2. Section 3 introduces the new architecture and discusses the modifications made to the baseline architecture. Section 4 will acknowledge the hardware and software requirements. Section 5 shows the experimental results achieved while training the networks with its specifications and comparison between the models. Section 6 concludes the paper by discussing the future scope and utility of R-MnasNet.

II. Prior Work

The efficiency of CNNs can be improved by designing hardware aware networks which have less computational cost and more accuracy. These models have performance like state-

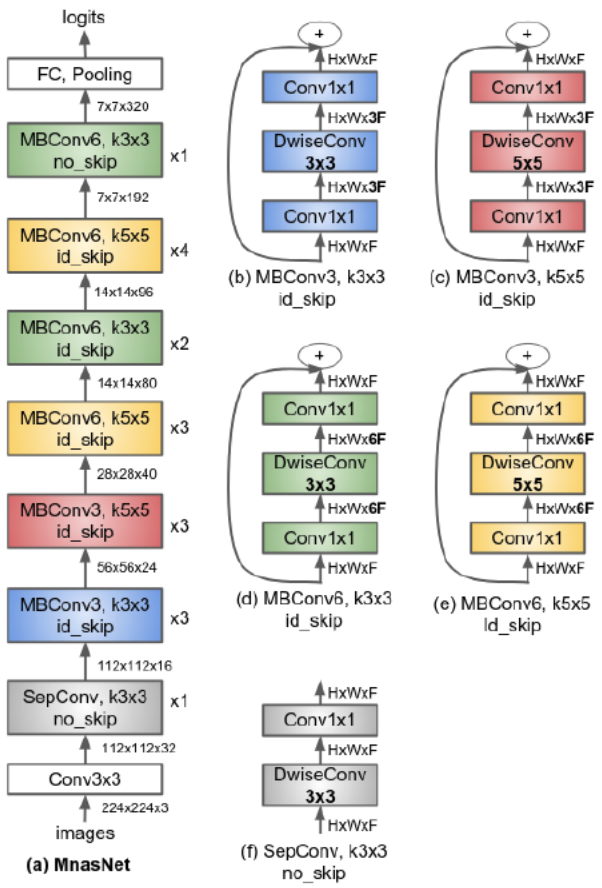

Fig. 1. MnasNet architecture

978-1-7281-9615-2/20/$31.00 ©2020 IEEE

of-the-art CNN models. They are extensively used for mobile vision applications. MnasNet is such architecture which was designed to accomplish such goals.

A. *MnasNet Architecture*

They have introduced a neural architecture search approach which optimized accuracy and latency on mobile devices using reinforcement learning. By using this approach, they propose architectures based on real world latency and accuracy trade-offs. They show that diversity of layers in such resource-constrained models yield better trade-offs between accuracy and latency of the model. They have shown results confirming better performance of their model then other models like MobileNet [1], SqueezeNext [16], ShuffleNet [23], MobileNetV2 [7], NASNet [30] and many other models.

Figure 1 shows the architecture of MnasNet. It uses Convolution Ops, depthwise separable convolution, mobile inverted bottleneck layers to extract features. It uses RMSProp optimizer [31], Batch Normalization [27] and Dropout regularization [32]. Table I shows the MnasNet architecture which is trained with CIFAR-10 dataset.

MnasNet Architecture					
Layers	Convolutions	t	c	n	s
$32^2 \times 3$	Conv2d 3×3	-	32	1	1
$112^2 \times 32$	SepConv 3×3	1	16	1	2
$112^2 \times 16$	MBConv3 3×3	3	24	3	2
$56^2 \times 24$	MBConv3 5×5	3	40	3	2
$28^2 \times 40$	MBConv6 5×5	6	80	3	2
$14^2 \times 80$	MBConv6 3×3	6	96	2	1
$14^2 \times 96$	MBConv6 5×5	6	192	4	1
$7^2 \times 192$	MBConv6 3×3	6	320	1	1
$7^2 \times 320$	FC,Pooling			10	

TABLE I
WHERE T: EXPANSION FACTOR, C: NUMBER OF OUTPUT CHANNELS, N: NUMBER OF BLOCKS AND S: STRIDE

III. R-MNASNET ARCHITECTURE

This section will explain the proposed R-MnasNet architecture and the modifications made to the baseline architecture (MnasNet) to reduce the model size and increase the accuracy. Table II shows the R-MnasNet architecture.

A. *Convolution Layers*

To increase the accuracy of the model, the convolution layers must extract features in an efficient way. MnasNet uses Depthwise Separable Convolutions to extract features. Depthwise separable convolutions have two layers of convolution. First, a depthwise convolution layer which extracts the features along the spatial dimensions of the input tensor. Second, a 1x1 pointwise convolution layer which covers the depth of the input tensor. In these convolutional layer, the spatial dimensions remain constant.

In order to extract features more efficiently, new convolutional layers were added in the baseline model. These layers are known as Harmonious Bottleneck Layers [2]. Along with channel transformations, these layers take spatial dimensions

R-MnasNet Architecture					
Layers	Convolutions	t	c	n	s
$32^2 \times 3$	Conv2d 3×3	-	32	1	1
$112^2 \times 32$	SepConv 3×3	1	16	1	2
$112^2 \times 16$	MBConv3 3×3	3	24	3	2
$112^2 \times 4$	Harmonious Bottleneck	2	36	1	1
$56^2 \times 36$	MBConv3 5×5	3	40	3	2
$112^2 \times 40$	Harmonious Bottleneck	2	72	1	2
$28^2 \times 72$	MBConv6 5×5	6	80	3	2
$112^2 \times 80$	Harmonious Bottleneck	2	96	4	2
$14^2 \times 96$	MBConv6 3×3	6	96	2	1
$112^2 \times 80$	Harmonious Bottleneck	2	192	1	2
$112^2 \times 80$	Harmonious Bottleneck	2	96	4	2
$14^2 \times 96$	MBConv6 5×5	6	192	4	1
$112^2 \times 80$	Harmonious Bottleneck	2	288	1	1
$7^2 \times 192$	MBConv6 3×3	6	320	1	1
$7^2 \times 320$	FC,Pooling			10	

TABLE II
WHERE T: EXPANSION FACTOR, C: NUMBER OF OUTPUT CHANNELS, N: NUMBER OF BLOCKS AND S: STRIDE

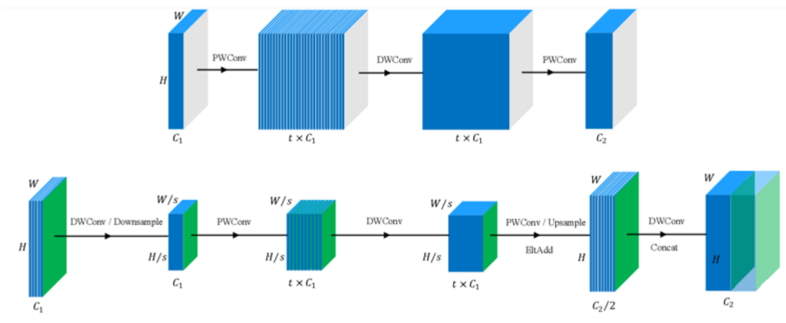

Fig. 2. Comparison of Depthwise Separable Convolution Layer and Harmonious Bottleneck Layer. [2]

into consideration as well. They change the spatial feature scale along composite layers of the network. As a result, the balance between capability of representation and computations is improved.

As shown in Figure 2, there is contraction-expansion of spatial dimensions and expansion-contraction of channel dimensions in the harmonious bottleneck layer. First, the channel dimensions remain constant and features get extracted from the spatial dimensions. Second, the spatial dimensions are kept constant and features are extracted from channel dimensions. This makes the convolution layer more efficient and increases the accuracy of the model. It also results in decreasing the computational cost of the model. The model size decreases and makes the model lightweight.

Figure 2 shows the comparison of depthwise separable convolutional block and harmonious bottleneck layer. The spatial size of input/output feature maps is (H x W), C1/C2 are input/output feature channels, (K x K) is the kernel size and s denotes stride.
The total cost of depthwise separable convolution is

$$(H \times W \times C1 \times K \times K) + (H \times W \times C1 \times C2) \quad (1)$$

The total cost of harmonious bottleneck layer is

$$B/s^2 + (H/s \times W/s \times C1 + H \times W \times C2) \times K^2 \tag{2}$$

where, B is the computational cost of the blocks inserted between the spatial contraction and expansion operations. It is evident that by squeezing the channel expansion-contraction component and using a pair of spatial transformations yields a slimmed spatial size of wide feature maps in each stage which reduces the computational cost.

In this paper, six harmonious bottleneck layers were added to the MnasNet architecture. The model is described in table II. After adding these layers, the accuracy of the model increased by 9.34% whereas the model size decreased from 12.7 MB to 3 MB. The comparison is shown in Table III.

B. Learning Rate Annealing or Scheduling

Different learning rates are used while training the network. There are various methods to choose the learning rate for particular epochs during training. Time-based decay, step decay and exponential decay are some of the methods which are used to determine learning rates during training. Figure 3 illustrates that step decay based learning rate performs better than other learning rate schedule methods. Adaptive learning rate methods are also used instead of manually scheduling the learning rates. In this paper, the step decay method proved to be more efficient. Therefore, this method is used for training R-MnasNet.

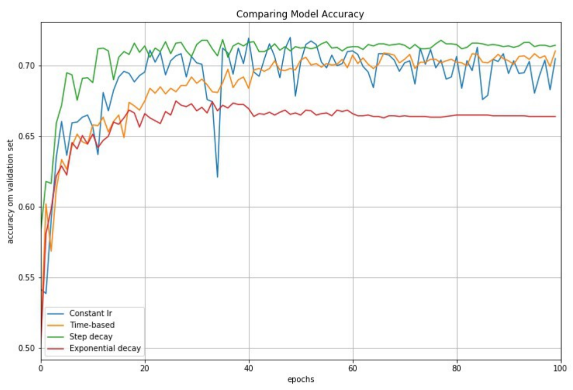

Fig. 3. Comparision of different LR scheduling methods.

C. Optimization

There are various optimization techniques which are used to optimize CNNs. Some commonly used optimizers are Adaptive Learning Rate Method (Adadelta) [31], Adaptive Gradient Learning Algorithm (Adagrad) [31], Adaptive Moment Estimation (Adam) [31], Stochastic Gradient Descent (SGD) [31], Root Mean Square Propagation (RMSprop). RMSprop was used to optimize the MnasNet architecture. In this paper, we have used SGD with varying values of learning rates and momentum=0.9 for better optimization.

D. Data Augmentation

AutoAugment [3] was used for data augmentation. AutoAugment learns the best augmentation policies for a given dataset with the help of Reinforcement Learning (RL). A policy consists of 5 sub-policies and each sub-policy applies 2 image operations in sequence. Each of those image operations has two parameters: The probability of applying it and the magnitude of the operation (e.g. rotate 20 degrees in 65% of cases). There is a controller that decides the best data augmentation policy at that instant and tests the generalization ability of that policy by running a child model experiment on a small subset of a particular dataset. After the child experiment is finished the controller is updated with the validation accuracy as the reward signal, using a policy gradient method called Proximal Policy Optimization algorithm (PPO). In this paper, AutoAugment is used on CIFAR-10 dataset. The accuracy of R-MnasNet was 88.54% but after using AutoAugment the accuracy increased to 90.14%.

E. Mish Activation Function

Activation functions are used to introduce non-linearity in neural networks. They determine the correct non-linear relation between the input and output signals. In 2019, Mish [35] was introduced and it outperformed all other activation functions. It is a new type of gated softplus function. The softplus activation function can be represented as:

$$\varsigma(x) = \ln(1 + x) \tag{3}$$

Mathematically, Mish can be written as,

$$f(x) = x \cdot \tanh(\varsigma(x)) \tag{4}$$

Figure 6 shows the graphical representation of Mish. For comparison, Figure 7 shows commonly used activation functions along with the graph of Mish activation.

Mish [35] avoids saturation due to near zero gradients, strong regularization effects, preserves small negative gradients and has effective optimization and generalization. After implementing it in R-MnasNet, the acurracy of the model increased from 90.14% to 91.13%.

IV. Hardware and Software used

- Intel i9 9th generation processor with 32 GB RAM
- Aorus Geforce RTX 2080Ti GPU
- Python version 3.6.7.
- Pytorch version 1.0.
- Spyder version 3.6.
- Livelossplot

V. Results

The accuracy of baseline architecture is 80.8% when trained with CIFAR-10 [4] dataset with a model size of 12.7 MB. After introducing new layers, the accuracy became 88.84% and the model size reduced to 3 MB. AutoAugment was implemented to further incerase the accuracy. It was evident that the accuracy increased from 88.54% to 90.14%. Furthermore,

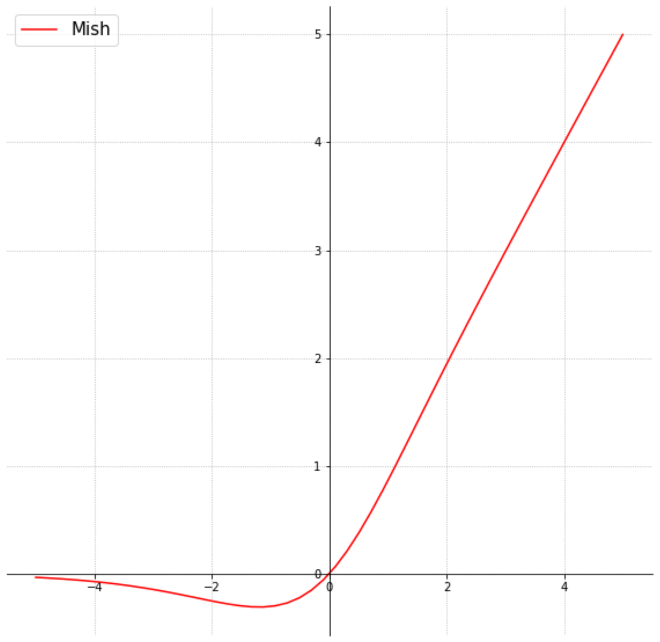

Fig. 4. Mish Activation Function

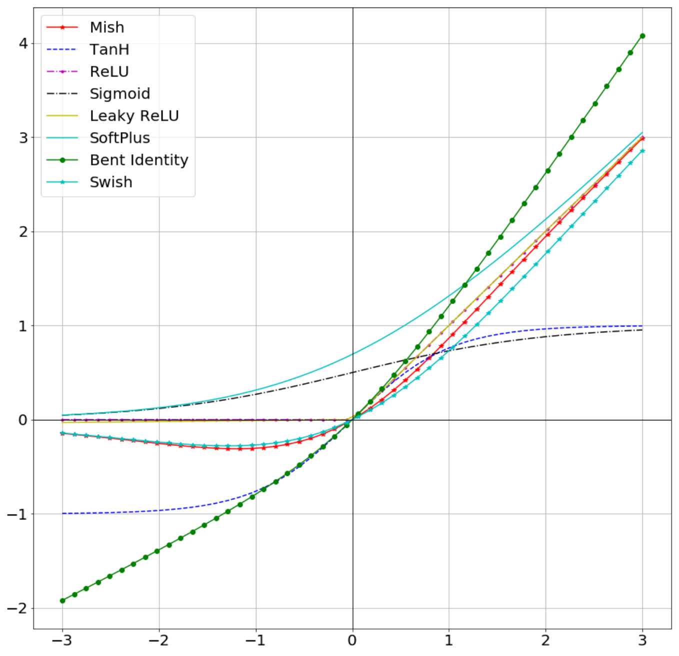

Fig. 5. Common Activation Functions

ReLU6 activation function was replaced with Mish activation function. This resulted with the model accuracy of 91.13%.

The aforementioned modifications were made to the baseline architecture. The model was trained with CIFAR-10 [4] dataset on Aorus Geforce RTX 2080Ti GPU using PyTorch framework for 200 epochs. The data was divided into batch size of 128 for training set and batch size of 64 for validation set.

Figure 6 and figure 7 shows the plots of log loss and accuracy of their respective models. The graphs were plotted using livelossplot visualization library.

The trade-off between model size and accuracy is shown in table III

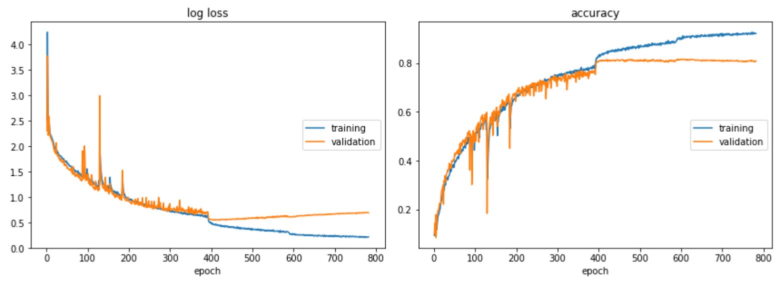

Fig. 6. Baseline

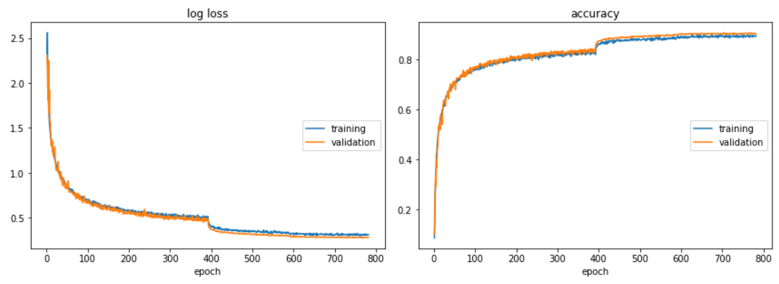

Fig. 7. R-MnasNet

Comparison of models		
Architecture	Model Accuracy	Model size (in MB)
MnasNet	80.8%	12.7
R-MnasNet	91.13%	3

TABLE III

Table IV shows the results obtained by scaling the model with different values of width multiplier.

Scaling R-MnasNet with width multiplier		
Width Multiplier	Model Accuracy	Model size
1.4	92.49%	5.6 MB
1.0	91.13%	3 MB
0.75	90.03%	2 MB
0.5	87.5%	1.3 MB
0.35	84.9%	837.6 KB

TABLE IV

VI. Conclusion

In this paper, a new architecture, R-MnasNet, is introduced by modifying its baseline architecture. The goal was to make the model more compact and having a fair trade-off between model size and accuracy. The accuracy of R-MnasNet is 91.14% with a size of 3 MB. It outperforms its baseline architecture MnasNet which has an accuracy of 80.8% and model size of 12.6 MB. New Harmonious Bottleneck layers were added to the baseline architecture. Mish activation was used to improve the optimization of the network. AutoAugment was used to further increase the accuracy of the model. This model can be deployed on mobile devices and various embedded systems. It can be used for embedded vision applications. Depending on the application and hardware, a particular variant of R-MnasNet can be deployed by scaling it with width multiplier to achieve specific model size and accuracy.

REFERENCES

[1] Mingxing Tan, Bo Chen, et al. "MnasNet: Platform-Aware Neural Architecture Search for Mobile" arXiv:1807.11626v3 [cs.CV] 29 May 2019

[2] Duo Li, Aojun Zhou, Anbang Yao. "HBONet: Harmonious Bottleneck on Two Orthogonal Dimensions" arXiv:1908.03888v1 [cs.CV] 11 August 2019

[3] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, QuocV. Le Google Brain, "AutoAugment: Learning Augmentation Strategies from Data" arXiv:1805.09501v3 (2019)

[4] https://www.cs.toronto.edu/ kriz/cifar.html

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, " Imagenet classification with deep convolutional neural networks" In Advances in neural information processing systems, pages 10971105, 2012.

[6] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv:1704.04861 (2017).

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." arXiv:1801.04381v4 (2019)

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z.Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. "Imagenet largescale visual recognition challenge", International Journal of Computer Vision, 2015.

[9] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. "Quantized convolutionalneural networks for mobile devices". arXiv preprint arXiv:1512.06473,2015.

[10] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y.Chen. "Compressing neural networks with the hashing trick". CoRR,abs/1504.04788, 2015

[11] S. Han, H. Mao, and W. J. Dally."Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding".CoRR, abs/1510.00149, 2, 2015.

[12] Karen Simonyan, Andrew Zisserman."VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION" arXiv preprint arXiv:1409.1556v6 (2015)

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision." arXiv preprint arXiv:1512.00567, 2015.

[14] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint-arXiv:1602.07261, 2016

[15] Sebastian Ruder. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747, 2017

[16] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K.Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 1MB model size. arXiv preprint arXiv:1602.07360, 2016.

[17] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, Kurt Keutzer "SqueezeNext: Hardware-Aware Neural Network Design." arXiv preprint arXiv:1803.10615v2

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

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

[20] 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.

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

[22] Abien Fred M. Agarap. "Deep Learning using Rectified Linear Units (ReLU)." arXiv preprint arXiv:1803.08375v2[cs.NE], 2019

[23] Xiangyu Zhang, Xinyu Zhou, Mengxiao LinJian, Sun. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile." arXiv preprint arXiv:1707.01083v2, 2017

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

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

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

[27] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." arXiv preprint arXiv: 1502.03167, 2015

[28] Timothy Dozat (2016). "INCORPORATING NESTEROV MOMENTUMINTO ADAM." Workshop track-ICLR 2016.

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

[30] Barret Zoph, Vijay Vasudevan, Jonathon Shlens and Quoc V. Le, "Learning Transferable Architectures for Scalable Image Recognition", arXiv:1707.07012v4 [cs.CV] 11 Apr 2018.

[31] Sebastian Ruder, "An overview of gradient descent optimization algorithms*",arXiv:1609.04747v2 [cs.LG] 15 Jun 2017

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research 15 (2014) 1929-1958, Submitted 11/13; Published 6/14.

[33] https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

[34] http://www.image-net.org/

[35] Diganta Misra, "Mish: A Self Regularized Non-Monotonic Neural Activation Function", arXiv preprint arxiv:1908.08681 (2019)