A Complete Study on Image Classification on CIFAR-10 Dataset Using Deep Learning

Shreyanshu Bhushan Heechul Jeong
Graduate School of Artificial Intelligence, Kyungpook National University
{bhushan,heechul}@knu.ac.kr

Abstract

Deep convolutional neural networks (DCNNs) have shown their promising achievement in image classification. There are several pre-existing models that already achieves excellent performance on CIFAR-10 dataset. Due to this it becomes very difficult for someone who wants to work on image classification, to choose the right model with correct parameters. In this paper, we present a full detailed report on image classification on CIFAR-10 dataset. We build our own convolutional neural network with optimized parameters and also use 10 different DCNNs. Afterwards, we perform several experiments on it and compare with each other. In addition, we also use some traditional and modern techniques in order to increase the performance accuracy of the models. Extensive experiments and analysis shows that the best accuracy is achieved by ResNet model by combing the traditional and modern techniques, having an score of ***97.95%****. Code and outputs are available at https://github.com/shreyanshu09/CIFAR-10.*

1. Introduction

Recently, image classification is developing and turning into a pattern among innovation designers particularly with the development of information in various sectors of industry, for example, web-based business, automotive, medical services, and gaming. The clearest illustration of this innovation is applied to Facebook. Facebook currently can recognize up to 98% exactness to distinguish your face with a couple of labeled pictures and group it into your Facebook collection. The actual innovation nearly beats the capacity of humans in image classification or recognition. It can also be utilize to recognize various regions by the type of land use. Land-use information is utilized broadly for metropolitan preparation. High-resolution imagery is additionally utilized during natural disasters events like floods, volcanoes, and serious dry seasons to check out effects and harm.

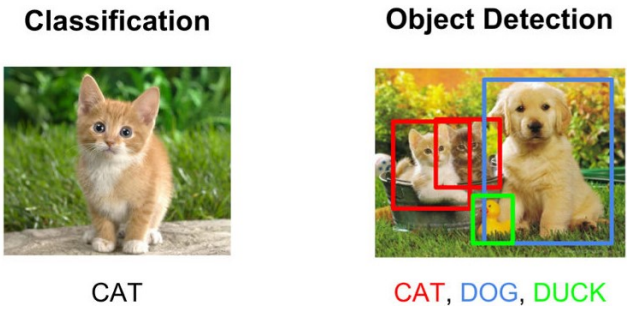

Figure 1. Difference between image classification and object detection. Image taken from google image.

One of the pre-dominant methodologies for this innovation is deep learning and it falls under the classification of Artificial Knowledge where it can act or adopt the thought process of a human. Regularly, the actual framework will be set with hundreds or perhaps large numbers of information to make the training session to be more productive and quick. The major task of image classification is to ensure every one of the pictures is sorted by its particular areas or groups. Classification is simple for people yet it has ended up being a serious issue for machines. Image classification and object detection are two things as shown in Fig. 1. Image classification incorporates predicting the class of one object in a picture. Object localization alludes to recognizing the area of no less than one thing in an image and drawing a bounding box around their area. Object detection joins these two undertakings and localizes and classifies at least one object in a picture. It comprises unidentified examples contrasted with distinguishing an item as it ought to be arranged to the appropriate classifications. The different applications like vehicle navigation, robot navigation, and remote detecting by utilizing picture arrangement innovation. It is as yet going through testing work and restricted assets are expected to further develop it. Image classification has turned into a significant test in machine vision and has a long history with it. The test incorporates a wide intra-class scope of pictures brought about by shading, size, eco-

logical conditions, and shape. It requires large information of labeled training images and to set up this huge information, devours a ton of time and cost.

In this paper, we use 10 different DCNNs such as AlexNet, VGG, ResNet, DenseNet, Inception v3, GoogLeNet, ShuffleNet v2, MobileNet v2, ResNeXt, and Wide ResNet for image classification on CIFAR-10 dataset. Furthermore, we use several techniques in order to improve the accuracy of the pre-existing model such as traditional and modern data augmentation, different activation function, different optimizers, learning rate scheduling, shake-shake regularization and many more. Also, we try to build our own convolutional neural network by considering all important aspects such as number of layers, best activation function, and regularization method. The main contributions of this paper are as follows:

- We build our own convolutional neural network with optimal number of layers and parameters.
- We perform extensive experiments with several DCNNs models for image classification on CIFAR-10 dataset and present a full detailed comparison between them.
- We also use different techniques on several DCNNs for further enhancement of the performance.

2. Related Work

There are many works already done on image classification on CIFAR-10 dataset. In this section, we will describe a brief introduction about all the DCNN models used in this study.

2.1. AlexNet

AlexNet [5] was the first convolutional neural network that utilizes GPU to enhance the performance. AlexNet architecture comprises of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 softmax layer. Each convolutional layer comprises convolutional channels and a nonlinear activation function ReLU. The pooling layers are utilized to perform max pooling.It consist of around 60 million parameters.

2.2. VGGNet

Visual Geometry Group (VGG) [8] is a standard deep Convolutional Neural Network (CNN) with various layers. The "deep" represents the quantity of layers with VGG-16 or VGG-19 comprising of 16 and 19 convolutional layers. It replace the large kernel-size filters used in AlexNet with various 3x3 filters, which makes it better than AlexNet. But the problem with this model is that it is very slow to train because of deep layers and their architecture weights make themselves quite large.

2.3. GoogLeNet

GoogLeNet [9] is a kind of DCNN dependent on Inception architecture. It uses Inception modules, which permit the network to pick between different convolutional channel sizes in each block. An Inception network stacks these modules on top of one another, with infrequent max-pooling layers with stride of 2 to split the resolution of the grid. It is a deep 22-layer convolutional neural network (CNN) developed by researchers at Google.

2.4. ResNet

A residual neural network (ResNet) [2] is an artificial neural network (ANN) that work according to the pyramidal cells in the cerebral cortex. The architecture of ResNet is inspired from VGG-19 which consist of 34-layer plain network with the addition of skip connections. These skip connections convert the model into residual network. There are different variants of ResNet: ResNet-18, ResNet-50, ResNet-101, ResNet-152. Here, the numbers along with the ResNet represent the number of layers like ResNet-50 consist of 50 layers deep.

2.5. ResNeXt

ResNeXt [11] is a straightforward, exceptionally modularized network design for image classification. This architecture is developed by repeating a structure block that totals a bunch of transformations with the same topology. This simple design results in a homogeneous, multi-branch model that has a couple of hyper-parameters to set. Acquired from ResNet, VGG, and Inception, ResNeXt incorporates alternate ways from the previous to next block, stacking layers and adjusting split-transform-merge procedure.

2.6. DenseNet

A DenseNet [3] is a kind of CNN that uses dense associations between layers, through Dense Blocks, where they connect all layers straightforwardly with one another. To save the feed forward nature, each layer acquires extra contributions from all preceding layers and passes on its own feature-maps to all to every resulting layer. The fundamental parts that compose a DenseNet are dense blocks and transition layers.

2.7. ShuffleNet v2

ShuffleNet v2 [6] is a CNN which improved for an immediate measurement (speed) rather than indirect measurements like FLOPs which measure the performance of a network, in terms of its computations. It expands upon ShuffleNet v1, which uses point-wise group convolutions, bottleneck-like designs, and a channel shuffle activity. Based on number of parameters, ShuffleNet v2 has

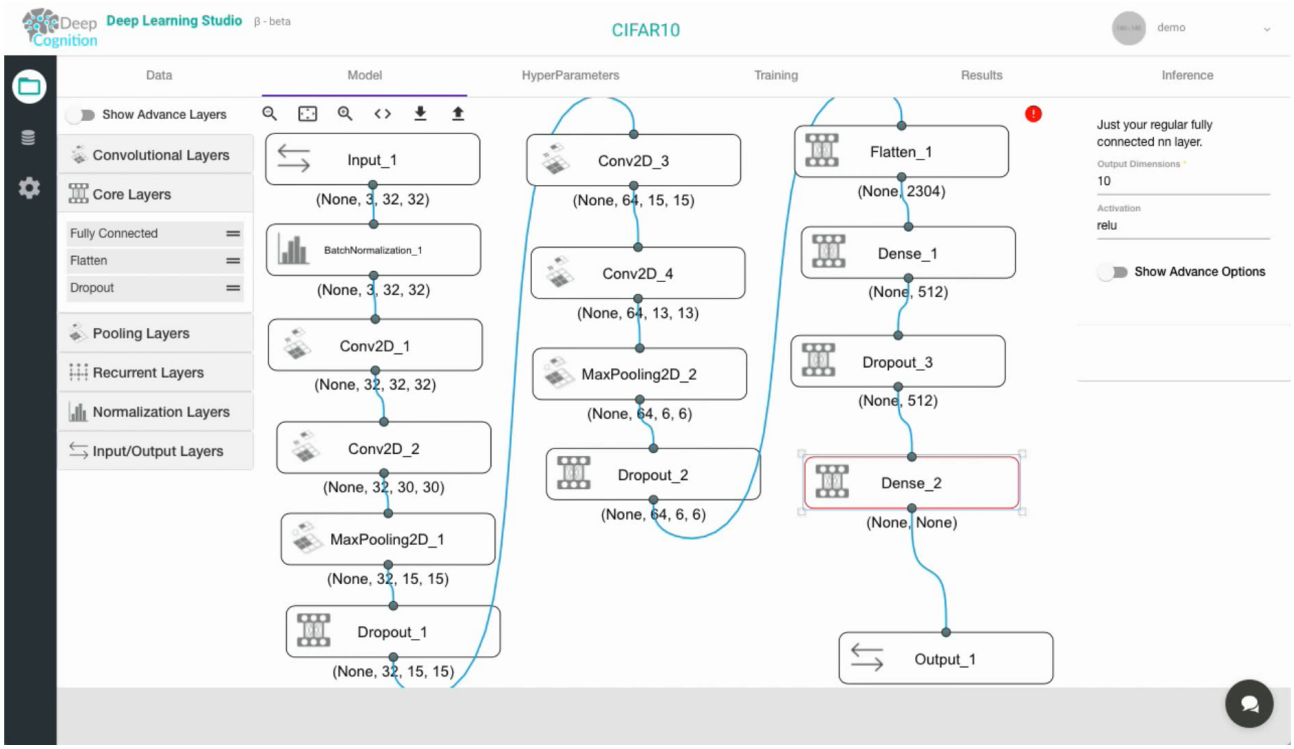

Figure 2. Sample Image of Deep Cognition Software Tool

two variant: ShuffleNet V2 x1.0 and ShuffleNet V2 x0.5 respectively.

2.8. MobileNet v2

As the name applied, the MobileNet v2 [7] model is intended to be utilized in mobile applications. MobileNet utilizes depth-wise detachable convolutions. It altogether lessens the quantity of parameters when compared to the network with standard convolutions with similar depth in the nets. This outcomes in lightweight deep neural networks. A depth-wise distinguishable convolution is produced using two tasks: Depth-wise convolution and Point-wise convolution.

2.9. Inception v3

Inception v3 [10] is a convolutional neural network for aiding image classification and object identification, and started out as a module for GoogLeNet. It is the third version of Google's Inception convolutional neural network, initially presented during the ImageNet Recognition Challenge. In contrast with VGGNet, Inception Networks have ended up being the best model for computationally proficient, both as far as the quantity of parameters generated by the network and also the affordable expense (memory and different assets).

2.10. Wide ResNet

The Wide Residual Network [12] is a later enhancement for the first Deep Residual Networks. Rather than depending on expanding the depth of a network to work on its precision, it was shown that a network could be made shallower and more extensive without undermining its performance. Wide Residual networks just have expanded the number of channels contrasted with ResNet. Otherwise the architecture is mostly similar.

3. Method

In this section, we have describe about our own CNN model and all those techniques used along with the pre-existing models in order to increase their accuracy.

3.1. CNN Model

We build our own CNN model for image classification. To choose the correct layers and parameters, we try a lot of networks on deep cognition software (https://deepcognition.ai/) that helps in the creation of deep learning models. Fig. 2 shows the sample of deep cognition software tool. This tool also helps in hyper-parameter tuning such as changing the loss function or optimizer and to select the best one. With this tool anyone can build their own neural network easily without any programming skills

because it comes with the feature of drag and drop GUI which is very user friendly.

Architecture: The overall summary of our model is depicted in Fig. 3. The model starts with a convolutional layer which take the inputs and run convolutional filters on them. The size of the filter is 3X3 and we use relu and elu activation functions. Also, the size of input is same means we didn't use padding. Then we make a dropout layer which is generally used for preventing overfitting problems. The dropout function randomly eliminated some of the connections between the layers. we set the dropout to 0.2 which means that it will drops 20% of the existing connections. To normalize the inputs going to the next layer, we use batch normalization, so that the network always creates activation with the same distribution. After that we create another convolutional layer, but this time, we increase the filter size so that the network can learn more non-linear representations. Then, at that point, utilization of the pooling layer helps in making the image classifier more powerful so it can learn relevant examples with again dropout and batch normalization. Again, repeating the same convolutional layer, activation, max pooling, dropout, and batch normalization. After that we need to flatten the data with Flatten function along with the dropout. Then we create the first densely connected layer with the help of Dense import. In the last layer, we pass the complete number of classes for the quantity of neurons. Every neuron addresses a class, and the result of this layer will be a 0 neuron vector with every neuron putting away some likelihood that the picture being referred to has a place with the class it addresses. At last, the softmax activation function chooses the neuron with the most relevant likelihood as its result.

3.2. Techniques

Data Augmentation: It is a technique use to increase the amount of data by by adding somewhat altered duplicates of previously existing data. There are various methods to do that but in our work, we have focus on few such methods such as random crop which make a random subset of an original picture, horizontal flips implies rotating a picture on a vertical axis, normalization which is used to carry the mathematical data to a typical scale without distorting its shape, color jittering which means means to randomly change the lightness, hue,and saturation of the picture and we also use some modern data augmentation such as cutout and mixup. Fig. 4 shows some of the data augmentation techniques result. Data Augmentation isn't simply used to prevent overfitting. As a general rule, having an enormous dataset is crucial for the performance of both machine learning (ML) and Deep Learning (DL) models. We can work on improving the performance of the model by enlarging the data. It implies that Data Augmentation is additionally useful for upgrading the model's performance and plays an im-

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 32, 32, 32)	896
activation_21 (Activation)	(None, 32, 32, 32)	0
dropout_21 (Dropout)	(None, 32, 32, 32)	0
batch_normalization_18 (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_13 (Conv2D)	(None, 32, 32, 64)	18496
activation_22 (Activation)	(None, 32, 32, 64)	0
max_pooling2d_6 (MaxPooling2D)	(None, 16, 16, 64)	0
dropout_22 (Dropout)	(None, 16, 16, 64)	0
batch_normalization_19 (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_14 (Conv2D)	(None, 16, 16, 64)	36928
activation_23 (Activation)	(None, 16, 16, 64)	0
max_pooling2d_7 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_23 (Dropout)	(None, 8, 8, 64)	0
batch_normalization_20 (BatchNormalization)	(None, 8, 8, 64)	256
conv2d_15 (Conv2D)	(None, 8, 8, 128)	73856
activation_24 (Activation)	(None, 8, 8, 128)	0
dropout_24 (Dropout)	(None, 8, 8, 128)	0
batch_normalization_21 (BatchNormalization)	(None, 8, 8, 128)	512
flatten_3 (Flatten)	(None, 8192)	0
dropout_25 (Dropout)	(None, 8192)	0
dense_9 (Dense)	(None, 256)	2097408
activation_25 (Activation)	(None, 256)	0
dropout_26 (Dropout)	(None, 256)	0
batch_normalization_22 (BatchNormalization)	(None, 256)	1024
dense_10 (Dense)	(None, 128)	32896
activation_26 (Activation)	(None, 128)	0
dropout_27 (Dropout)	(None, 128)	0
batch_normalization_23 (BatchNormalization)	(None, 128)	512
dense_11 (Dense)	(None, 10)	1290
activation_27 (Activation)	(None, 10)	0

Total params: 2,264,458
Trainable params: 2,263,114
Non-trainable params: 1,344

None

Figure 3. overall summary of our own CNN model

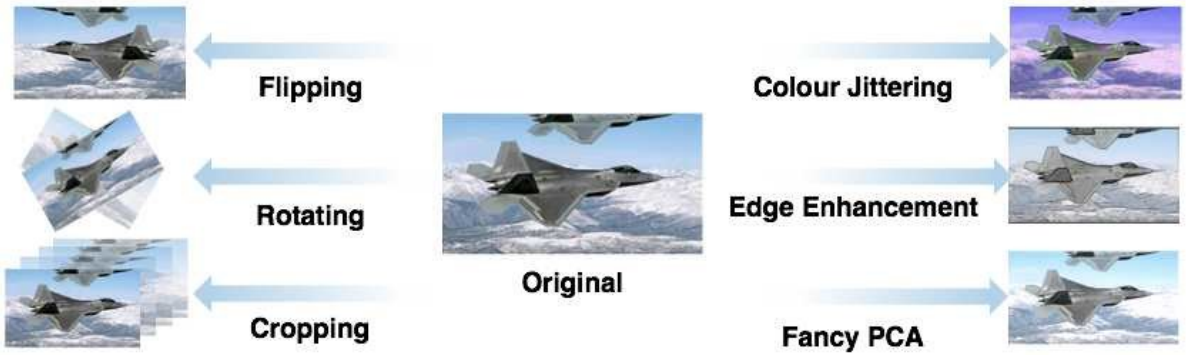

Figure 4. Some basic data augmentation techniques result. Image taken from google images.

portant role for better accuracy of the models. In this study, we use all the mentioned data augmentation techniques in order to further boost the performances of the model.

Shake-Shake Regularization: Shake-shake regularization [1] is a technique where gradient noise is replaced by a form of gradient augmentation. Generalization of complex neural networks is done by adding noise to the gradient during training and shake-shake regularization is just an extension of this concept.

Activation Functions: An activation function is a function that is added into an artificial neural network to assist the network with learning complex patterns in the data. When compared with a neuron-based model that is in the human brain, the activation function is toward the end concluding what is to be fired to the following neuron. Pre-existing architectures uses a certain type of activation function. We also try to replace those activation functions with different one to see the effect on the performance of the model.

Optimizers: Optimizers integrate with the loss function and model parameters by refreshing the model in response of the result of the loss function. In less difficult terms, optimizers shape and mold the model into its most precise possible structure by futzing with the loads. The loss function is the manual for the terrain, telling the optimizer when it's moving in the right direction or wrong direction. It is used to maximize the efficiency or to minimize the loss. Different architecture uses different optimizers such as some uses SGD optimizer and some uses Adam optimizer. To see the effect on the performance, we also change the optimizer and report the results.

Batch Normalization: Normalization is a data prehandling method used to carry the mathematical data to a typical scale without distorting its shape. Generally, when we input the information to a machine or deep learning calculation, we will quite often change the values to a reasonable scale. The reason we normalize is somewhat to guarantee that our model can sum up appropriately. It makes the Optimization quicker in light of the fact that normalization doesn't permit the weights to explode everywhere and confines them to a specific range. Batch normalization add extra layers to deep neural networks which makes it faster to process and more stable. We also use this technique in our work to process normalization in batches of data and not as a single input.

Learning Rate Scheduling: Adjusting the learning rate according to a pre-defined schedule during training of a model. In this study, we use different learning rate to train our models and see the effect on the performances of the model.

Gradient Clipping: We also use gradient clipping in our work to avoid the exploding gradients which generally happens during backward propagation through the network. To solve this, we use the clipped gradient to update the weights.

Weight Decay: It is also a regularization technique which works by adding a penalty to the cost function of a neural network. It has the effect of shrinking the weights during backpropagation. It also helps in preventing the model from overfitting the training data as well as the exploding gradient problem.

4. Experiments

To validate the performance of our model and other DCNNs, we conduct extensive experiments on CIFAR-10 dataset. In this section, we first specify the experimental setup details which includes environment, dataset, models, hyper-parameters, and metrics used in this study, followed by the experimental results which shows the performance of each model and a full detailed comparison.

4.1. Experimental Setup

We perform our experiments on a single GPU (NVIDIA TITAN RTX) having a memory of 24GB with CUDA version 10.2. Our implementation is based on PyTorch and Keras with Python 3.6 as the programming language. Few programs, we run on colab while others on PyCharm environment.

Model	No Modification(%)	Traditional Techniques(%)	Modern Techniques(%)
AlexNet	75.39	-	-
VGG 19	90.01	93.31	-
ResNet	93.02	**96.92**	**97.95**
DenseNet	94.32	95.90	96.50
Inception v3	93.37	93.45	94.25
GoogLeNet	89.53	-	-
ShuffleNet v2	88.31	-	-
MobileNet v2	90.28	91.37	94.04
ResNeXt	93.69	95.19	96.12
Wide ResNet	**94.78**	96.13	97.06
CNN Model (Ours)	82.24	85.64	-

Table 1. Top-1 classification rate of each model on three different methods. Dash(-) means we didn't perform that test and bold numbers indicate the best result for each method.

Dataset: We use the standard CIFAR-10 dataset [4] which consists of 60,000 colour images of size 32X32 in 10 classes. It is a subset of 80 million tiny images dataset. The classes include: airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. We choose 50,000 as train images and 10,000 as test images from the dataset. This dataset is generally used to recognize objects. Since the size of images in CIFAR-10 dataset is 32X32, which is a low resolution image, It becomes easy to quickly try different algorithms and check which one works best.

Models: We use our own build CNN model as discussed in the above section along with 10 different DCNNs which includes AlexNet, VGG19, ResNet26_2x64d, DenseNet190bc, Inception v3, GoogLeNet, ShuffleNet v2, MobileNet v2, ResNeXt29_16x64d, and Wide ResNet28x10. These are the basic models we use in this study. Also, we combine these models with some regularization techniques.

Hyper-parameters: In this study, we use different parameters for different models to study the outcome from each of them. For AlexNet and VGG19, we use 250 epochs with batch size of 128 & test batch size of 200 and weight decay of 0.0001. For ResNet26_2x64d, we use 1800 epochs with batch size of 128 & test batch size of 200 and weight decay of 0.0001. Also we use cutout, mixup, and Shake regularization technique with this model. For DenseNet190bc, we use 300 epochs with batch size of 64 & test batch size of 200 and weight decay of 0.0001. For Inception v3, we use 250 epochs with batch size of 128 & test batch size of 200 and weight decay of 0.0001. For GoogLeNet, ShuffleNet v2, and MobileNet v2, we use 250 epochs with batch size of 128 & test batch size of 100 and weight decay of $5e^{-4}$. For ResNeXt29_16x64d, we use 300 epochs with batch size of 128 & test batch size of 200 and weight decay of 0.0005. For Wide ResNet28x10, we use 160 epochs. For our own CNN model, we use 25 epochs with both Adam and SGD optimizers having a batch size of 64. For more clarification, we have added the configurations for each model in the github repository whose link is mentioned in the abstract section.

Metrics: We use the Top-1 classification rate as the evaluation metric. The top-1 classification rate is a term used to describe the top-1 accuracy of a particular algorithm on a classification task. Similarly, the top-1 error rate means how many times the network has predicted the correct label with the highest probability.

4.2. Experimental Result

We conduct our experiments on three basis for each model. First, we run the model as it is, without adding any techniques. Second, for each model, we add some traditional modifications such as using different activation functions, different optimizers, gradient clipping, learning rate scheduling, weight decay, and also some traditional data augmentation like horizontal flips, random crops, colour jittering. And third, we add modern techniques along with traditional one such as shake-shake regularization, cutout, and mixup.

Tab. 1 shows the accuracy of each model on three different methods that is using no modification, with traditional modification and with modern modification. Talking about first method which is same as it is introduce in their paper (No Modification). The highest accuracy is achieved by Wide ResNet model having an score of 94.78%. However, DenseNet model also achieve an outstanding score of 94.32% which is slightly lower than Wide ResNet model.

After that ResNet, Inception v3, and ResNeXt, they all achieve a good score of 93.02%, 93.37%, and 93.69%. GoogLeNet and ShuffleNet achieves a descent score of 89.53% and 88.31% respectively. Our own CNN model achieves a score of 82.24% without using any additional techniques, which is a good score. The worst performance was shown by AlexNet here, which achieves only a score of only 75.39%. By adding some traditional techniques to the base models (Traditional Techniques), we can see the performance of every model gets increased. However, we didn't add this modification to some of the model such as AlexNet, GoogLeNet, and ShuffleNet v2. The reason behind not touching those models was their poor performance from their base model itself. In this category the best accuracy was scored by ResNet model of score 96.92%. Wide ResNet also achieves an outstanding score of 96.13%. DenseNet and ResNeXt model achieves a good score of 95.90%, and 95.19%. VGG 19, Inception v3, and MobileNet v2 achieves a score of 93.31%, 93.45%, and 91.37%. Our CNN model slightly improves from the previous method and achieves a descent score of 85.64%. And lastly, we perform experiments by adding more modern techniques to the model such as cutout, mixup, and shake-shake regularization. As we can see clearly from the table that the best accuracy is achieved by ResNet Model, having an score of 97.95%, which is an outstanding score for ResNet. DenseNet, ResNeXt, and wide ResNet model also achieves a good score of 96.50%, 96.12%, and 97.06% respectively. The inception v3 and MobileNet v2 model didn't perform well compare to other models but they perform better from their previous traditional techniques and achieves a score of 94.25% and 94.04% respectively.

Our CNN model only outperforms AlexNet model and was not powerful enough to outperform other models, but it achieve a descent score for image classification task. Overall, by considering all the aspects, the best accuracy is achieved by ResNet (ResNet26_2x64d) model with an outstanding score of 97.95%.

5. Conclusion

In this work, we build our own CNN model for image classification on CIFAR-10 dataset and try different techniques in order to improve that performance. We also presented a full detailed comparison between 10 different DCNNs run on three methods that is with no modification, with traditional modification and last is with modern modification techniques. We use different techniques in order to improve the pre-existing models. Also, we combine both tradition and modern techniques to check the performance of the models on that.

After comparing each model on every technique, we found that the combination of traditional and modern techniques gives the best result and in comparative with other models, ResNet achieves the highest score of **97.95**%.

6. Future Work

In future, we will try to conduct more experiments with different techniques. Due to limited resources such as number of GPUs and memory, it takes more time to train a single model. and because of short time period, we couldn't test all the techniques on each model.

References

[1] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 5

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 2

[6] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 2

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 3

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[10] Xiaoling Xia, Cui Xu, and Bing Nan. Inception-v3 for flower classification. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pages 783–787. IEEE, 2017. 3

[11] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 2

[12] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 3