# Convolution Neural Networks For Object Classification

**Doğukan Duman**

**Abstract**

In this project, Convolution Neural Network **[1]** algorithm has implemented with various kind of Neural Network Architectures. Project is targeting CIFAR-10**[2]** dataset. Dataset contains 60,000 images with 10 classes which consist of different kind objects such as car, dog, cat etc. Main objective was finding an optimum combination of neural networks to classify objects in CIFAR dataset.

# 1.Introduction

Object recognition and classification have always been a hot topic since very early days of image processing and computer vision. Studies on this area have given state of art solutions for this problem such as Eigenfaces **[3].**

The conventional approach to image processing and computer vision problems is finding most valuable or important key features in an image such as edges, corners and transitions. There are also some state of art studies which focus on small regions in image that can be robust to rotation and transition. SIFT **[4]** and SURF**[5]** can be given as example to this approach.

Creating this kind of features needs deep knowledge on this area and good understanding of complex mathematical operations. Even after creating state of art methods and features, they cannot be used for every image processing problem. Because nature of feature engineering, solutions are mostly domain specific.

Low prices and increasing power of hardware gave chance to researches try different kind of solutions such as NN **[6].** Neural Networks have been used widely in Artificial Intelligence field. But the complex nature of NN didn't allowed us to make deep layered Neural Networks. After crucial developments took place in hardware world such as multi-processor GPUs, NN experiments started to solve sophisticate image processing problem which could not solve by conventional methods.

# 2.Problem Statement

Main purpose of this project is classifying of objects in images. For achieving this purpose, we build a system which uses CNN (Convolution Neural Network). CNN is kind of NN and specialized for classification problems in images.

NNs can be used as a solution for general classification problems but when we try to overcome classification problems in images, we have to consider some restrictions. One of the most of important feature is location of pixels in images. Relation and adjacency between pixels can

give us very valuable insights about the type of image. So, reason behind the using CNN is considering location information pixels.

For handling such a classification problem, we build a CNN architecture which is combination of CNN, Max Pooling, Drop-out and Full Connected layers.

# 3.Metrics

In project, I used Mean Square Error (MSE) for loss function and Accuracy for explain how our model did well on classification. MSE gives us ability to measure model performance.

$$accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

$$MSE = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2$$

**Figure 1 MSE and Accuracy**

Accuracy could be very bad metric when we deal with unbalanced datasets such as credit card fraud cases. But in CIFAR-10 dataset, all categories have similar number of image samples. Thanks to balanced CIFAR-10 dataset, we can accept accuracy is a reliable metric.

# 4.Dataset and Data Exploration

CIFAR-10 is a well-known dataset for object recognition. It consists of 60,000 32x32x3 color images and each image belongs to one of 10 object categories. At each category, there are 6000 images.

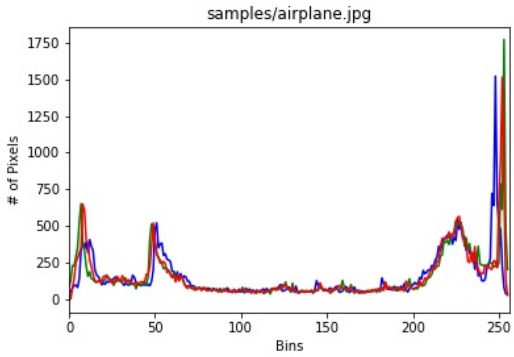
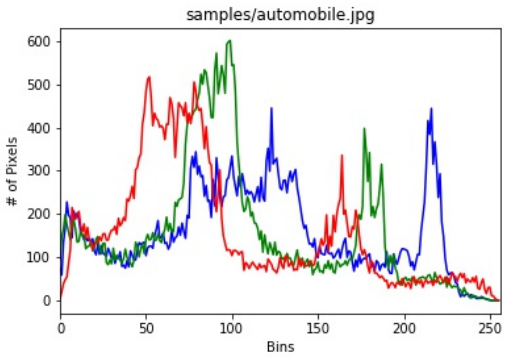
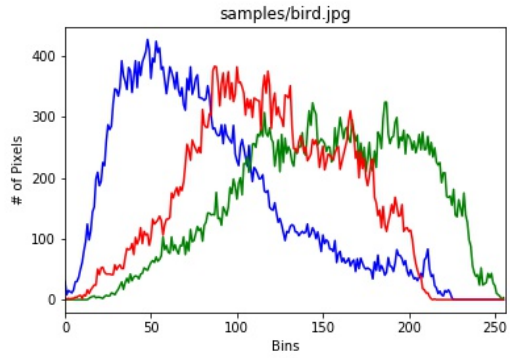
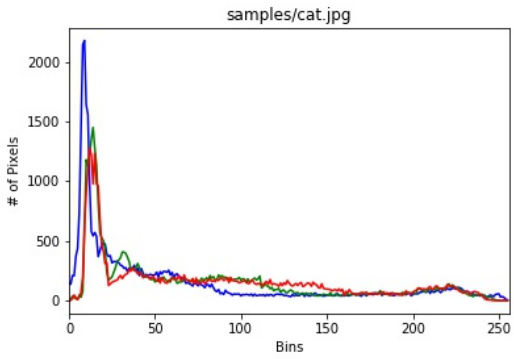
**Dataset Summary Info:**

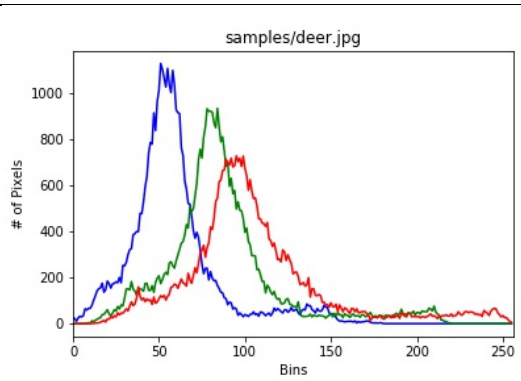
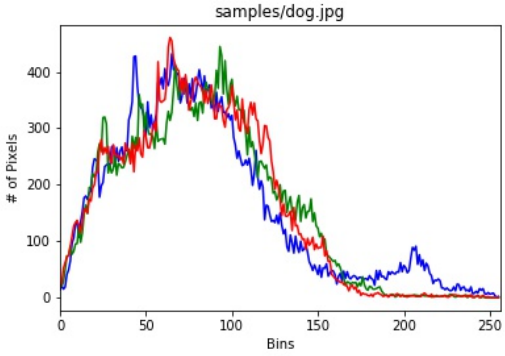
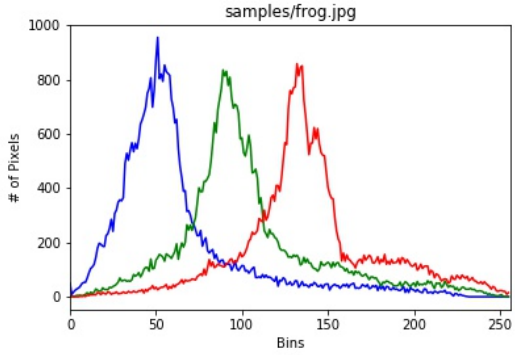
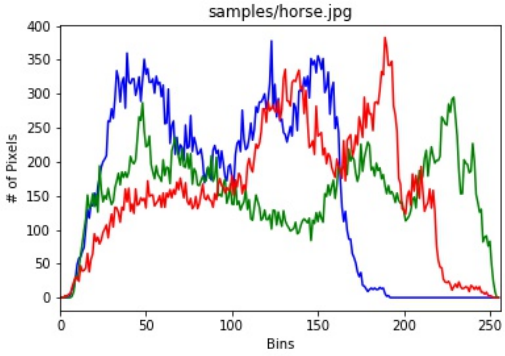
- # data instances: 60000
- # data attributes: 3072 (32x32x3)
- # target instances: 60000
- # target attributes: 10 (0 0 0 0 0 0 0 0 0 1, 0 0 0 0 0 0 0 0 1 0 , .. )

**Class Labels:**

- Airplane
- Automobile
- Bird
- Cat
- Deer
- Dog
- Frog
- Horse
- Ship
- Truck

## Colour Channel Intensity for Per Class

| | |
|---|---|
| **Airplane**  |   |
| **Automobile**  |   |
| **Bird**  |   |
| **Cat**  |   |

| | |
|---|---|
| **Deer**  |   |
| **Dog**  |   |
| **Frog**  |   |
| **Horse**  |   |

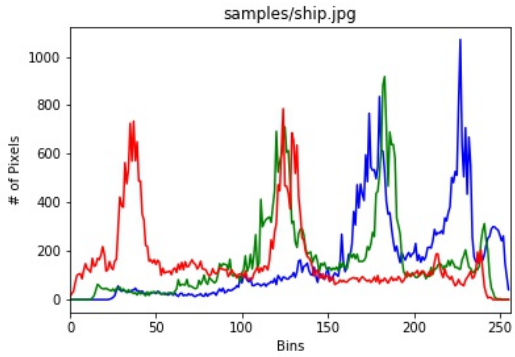
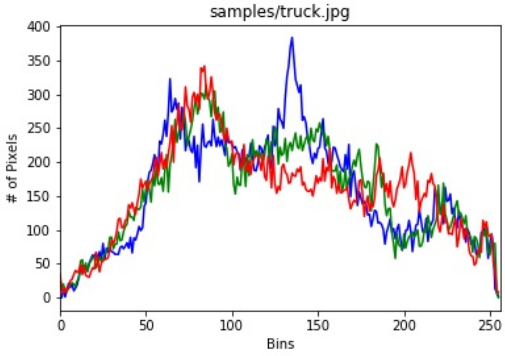
| | |
|---|---|
| **Ship**  |   |
| **Truck**  |   |

**Figure 2 Samples and Their colour Intensities for per class in from Cıfar-10**

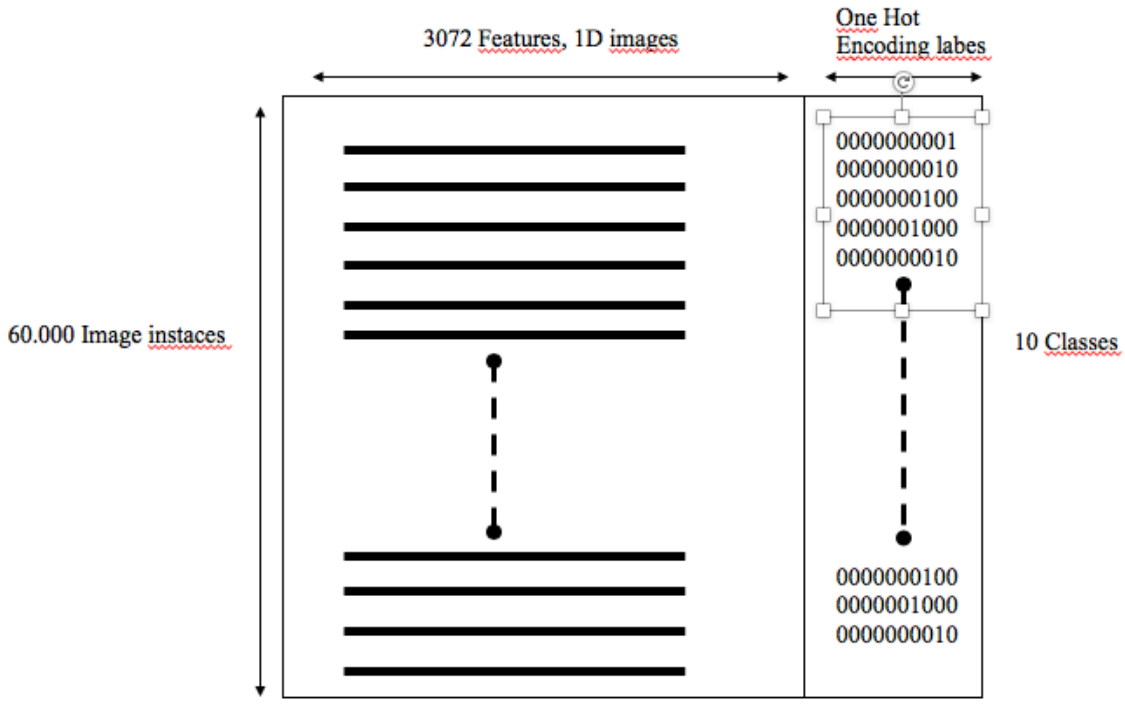



**Figure 3 Cifar-10 Data Set Structure**

# 5.Pre-processing

CIFAR-10 dataset was divided randomly as training (60%), validation (20%) and test (20%) sets. In CIFAR-10 dataset, every image is kept as one-dimensional vector. Due to nature of implementation CNNs, we need images as matrices which has valuable information about neighbours of pixels.

3072 attributes are converted to 32x32 image with 3 color channels.

# 6.CNN Algorithm

As we mentioned before, CNN is very powerful tool to classify images when we create an architecture with correct combination of operations and layers such as VGGNet, ResNet, Inception, and Xception**[7].**

Convolution filters have been used almost every problem in image processing. They can be used for smoothing images, detecting edges, corners or finding something valuable. **[Fig 4]**

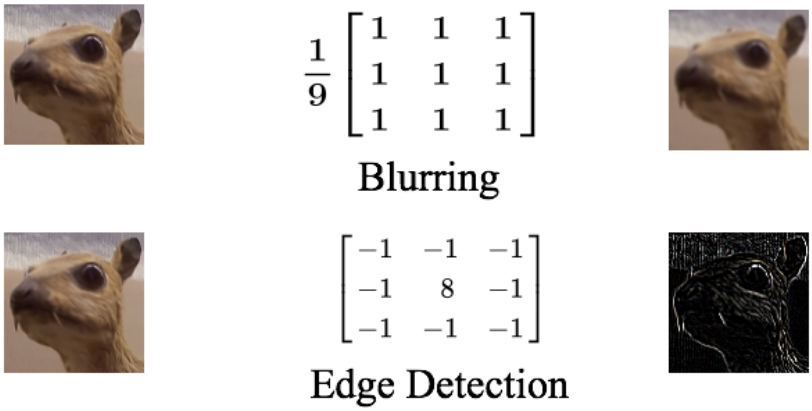



**Fig 4 Blurring and Edge Detection**

The main idea behind the using edge or corner features for classification is assumption of being uniqueness of these features for every object. On the other hand, these features may be meaningless too. For example, if we want to classify a truck, edge features can be very handy but at the same time we cannot use edge features for classify a sheep. Due to this reason, it is very hard to make a general object classifier with conventional methods.

CNN still depends on convolution filters but instead of engineered filters, it gives responsibility of generating filters to neurons. Thus, CNN decides which features would be the most valuable for classifying objects in dataset.

After pre-processing of images, we give the images as input to convolution layer which consists of randomly generated three dimensional filters. Size of the convolution layer can be various. It depends on how we build the architecture. For reducing complexity and prevent over-fitting, we can apply max pooling before giving outputs of previous layer as input of the next convolution layer. Number of convolution layers is related to complexity of problem and your architecture. Throughout the journey between convolution layers, heights and widths of outputs tend to be smaller and z-dimension of the outputs tend to be deeper. **[Fig 5]**

For classification, number of neurons in last layer must be same size with the number of categories in dataset and input data must be one dimensional for giving it to last layer for classification. So, using outputs for prediction has to be flatten. Last layer is named read out layer.

Some architectures have additional neural network layers between convolution neural network layers and read out layer. Additional layers may increase or decrease accuracy of the prediction model for different kind classification problems.

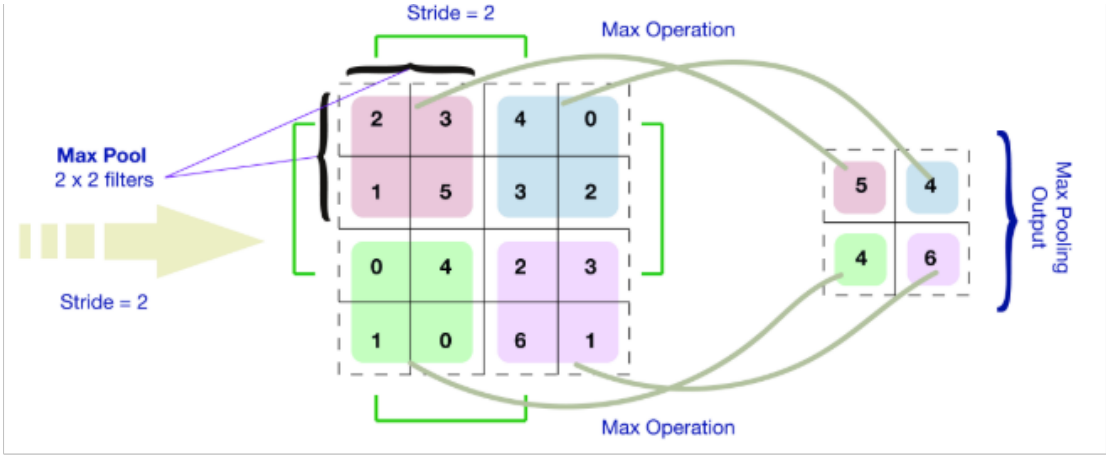



**Fig 5 Max Pooling**

# 7.Trials

As we mentioned before, CNN was used this project which has achieved very high success rates on multi-class classification problems in recent years. Three trials with different architectures were executed achieve to the best test performance. Same size of batches and number of epochs were used for both three trials.

For implementation and testing architectures below, Keras**[8]** is used which is a high-level neural networks API. For every trail, number of epochs are set to 20. Most of the CNNs achieve accuracy results in 10 epochs which is generally highest accuracy values.

## 7.1 Trial 1

In this trial one convolution layer (32) and one fully connected layer (100) were used. Pool sizing process was executed only once before the fully connected layer. Before read out layer dropout layer added with probability of 0.5. Adam optimizer was used as optimizer and relu was used as activation function. **[Fig 7]**

**Result;**

| Adam Opt. Epoch:20 | | Activation Function |
|---|---|---|
| | | Relu |
| Accuracy | Train | 61.41% |
| | Test | 61.22% |
| Exec Time | Train | 480s |
| | Test | 8s |

**Fig 6**

**Architecture;**

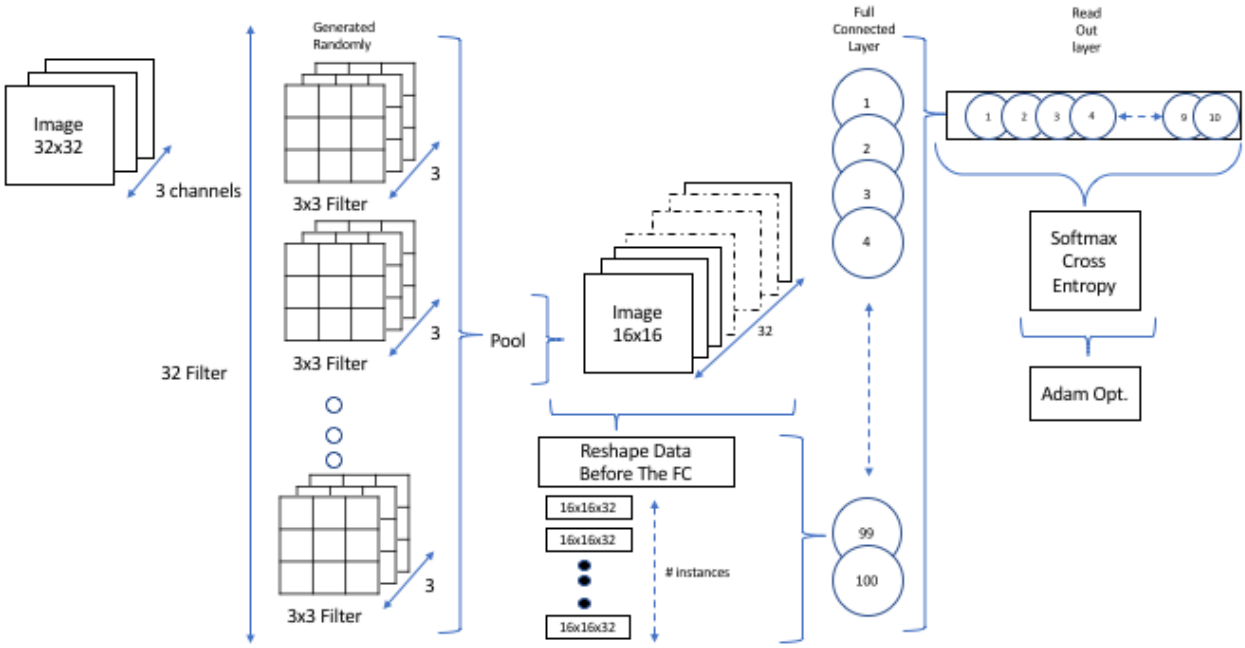



**Fig 7**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_2 (Conv2D) | (None, 32, 32, 32) | 896 |
| max_pooling2d_2 (MaxPooling2 | (None, 16, 16, 32) | 0 |
| flatten_2 (Flatten) | (None, 8192) | 0 |
| dense_3 (Dense) | (None, 100) | 819300 |
| dropout_2 (Dropout) | (None, 100) | 0 |
| dense_4 (Dense) | (None, 10) | 1010 |

Total params: 821,206
Trainable params: 821,206
Non-trainable params: 0

**Fig 8**

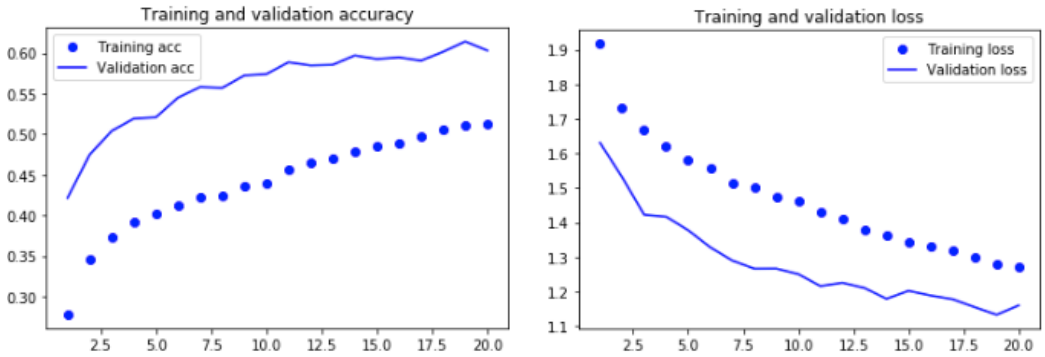



**Fig 9**

When we look at Figure 9, trend of the accuracy seems to increase. For clarifying this situation, number of epochs increased from 20 to 40.

**Results:**

| Adam Opt. Epoch:40 | | Activation Function |
|---|---|---|
| | | Relu |
| Accuracy | Train | 64.70% |
| | Test | 63.89% |
| Exec Time | Train | 1173s |
| | Test | 8s |

**Fig 10**

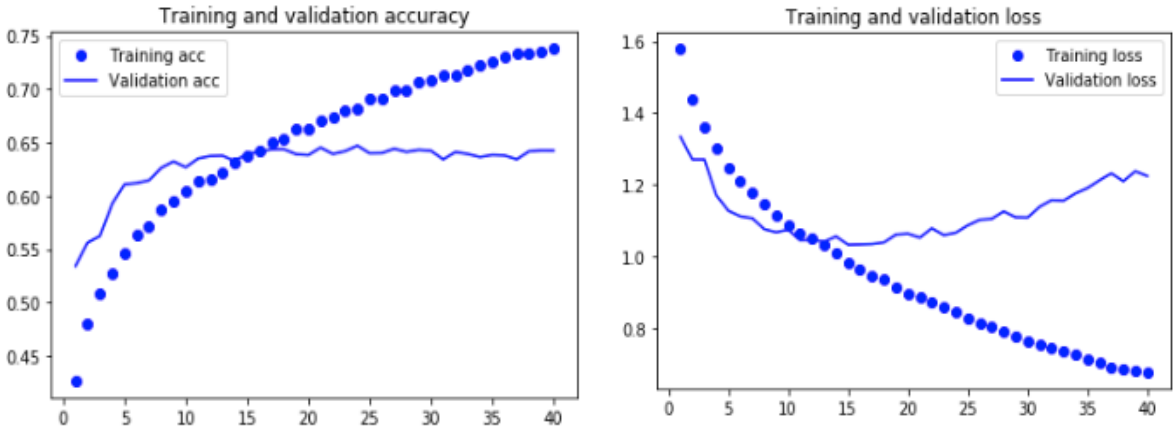



**Fig 11**

After iteration 15, model seems to not improve until last iteration **[Fig 11]**. Accuracy of model gained very little improvement from 61.22% to 63.89%. In spite of very little gain, train execution time increase triple. Furthermore, validation loss started to increase around iteration 15 which indicates to overfitting. In the next trails number of epoch will be 20 which is same as first trail.

Composed CNN architecture is not enough to achieve success rate and insufficient to solve such a complex problem.

## 7.2 Trial 2

In this trial two convolution layers (32-64) and one fully connected layer (512) were used. Fully connected layer size is bigger than previous trial. Pool sizing process was executed twice between convolution layers. Before read out layer dropout layer added with probability of 0.5. Adam optimizer was used as optimizer and relu was used as activation function. **[Fig 13]**

**Result;**

| Adam Opt. Epoch:20 | | Activation Function |
|---|---|---|
| | | Relu |
| Accuracy | Train | 93.75% |
| | Test | 72.99% |
| Exec Time | Train | 1280s |
| | Test | 11s |

**Fig 12**

**Architecture;**

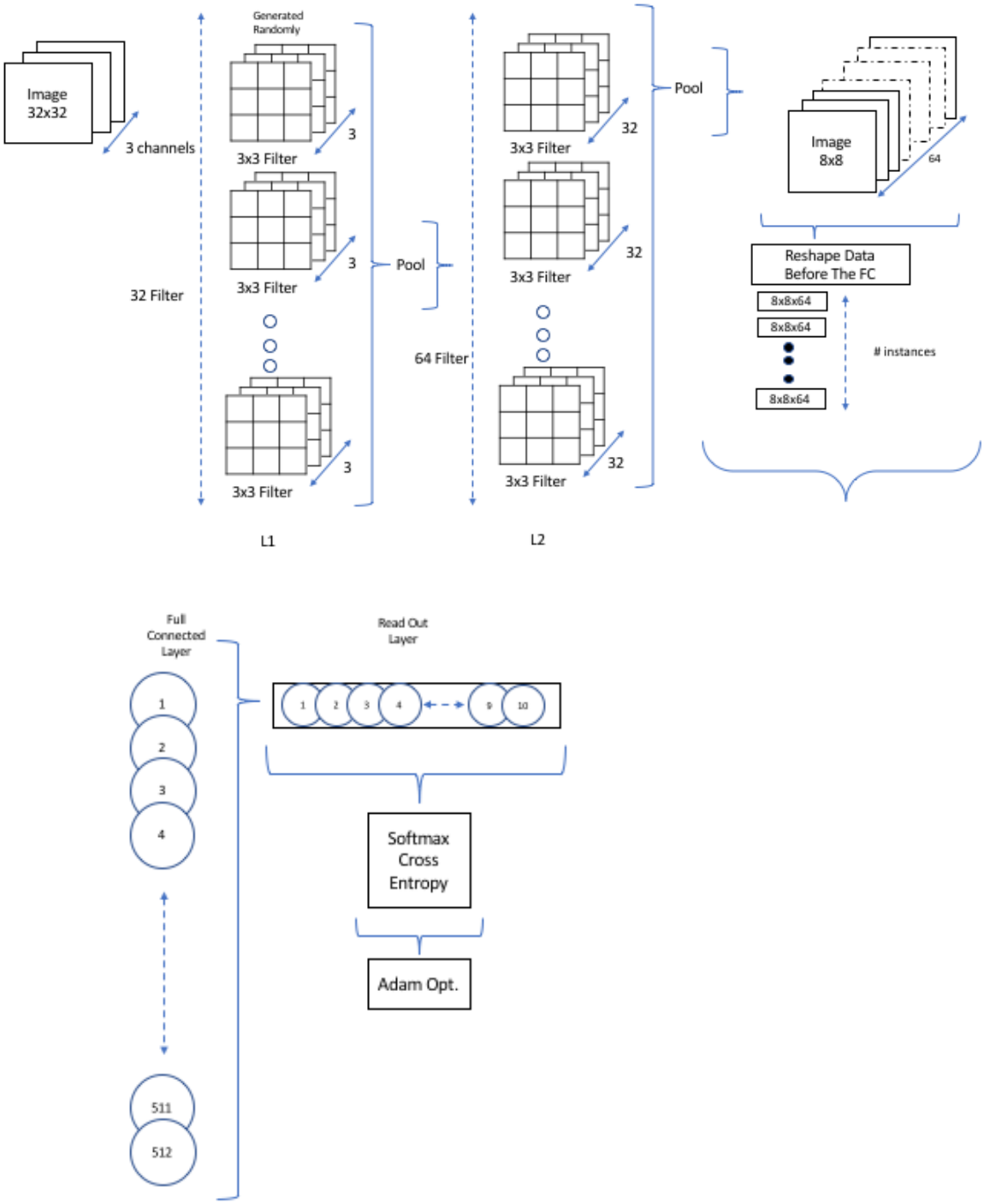



**Fig 13**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_4 (Conv2D) | (None, 32, 32, 32) | 896 |
| max_pooling2d_4 (MaxPooling2 | (None, 16, 16, 32) | 0 |
| conv2d_5 (Conv2D) | (None, 16, 16, 64) | 18496 |
| max_pooling2d_5 (MaxPooling2 | (None, 8, 8, 64) | 0 |
| flatten_4 (Flatten) | (None, 4096) | 0 |
| dense_7 (Dense) | (None, 512) | 2097664 |
| dropout_4 (Dropout) | (None, 512) | 0 |
| dense_8 (Dense) | (None, 10) | 5130 |

Total params: 2,122,186
Trainable params: 2,122,186
Non-trainable params: 0

**Fig 14**

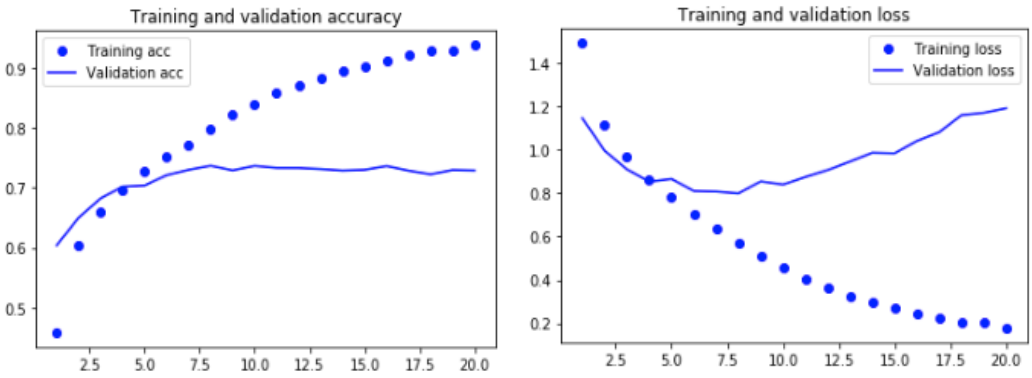



**Fig 15**

Second composed CNN architecture is more promising than previous trail and did very well on train set but still not enough to achieve promising result on test set. It stuck around %70-72. After iteration around 6, trend of the increasing accuracy seems to be losing its momentum. Furthermore, validation loss started to increase around iteration 5 which indicates to overfitting. We can say training time relatively longer than first trail.

## 7.3 Trial 3

In this trial two convolution layers (32-64) and two fully connected layers (1024-256) were used. Pool sizing process was executed twice between convolution layers. Before read out layer two dropout layers added between fully connected layers with probability of 0.5 and 0.4 [**Fig 17**] Adam optimizer was used as optimizer and relu was used as activation function. This trial very similar with previous trial, only differences between them added one more fully connected layer and changed number of neurons. Because of first part of the architecture completely same with [**Fig 13**]

**Result;**

| Adam Opt. Epoch:20 | | Activation Function |
|---|---|---|
| | | Relu |
| Accuracy | Train | 94.08% |
| | Test | 72.36% |
| Exec Time | Train | 1820s |
| | Test | 18s |

**Fig 16**

**Architecture;**

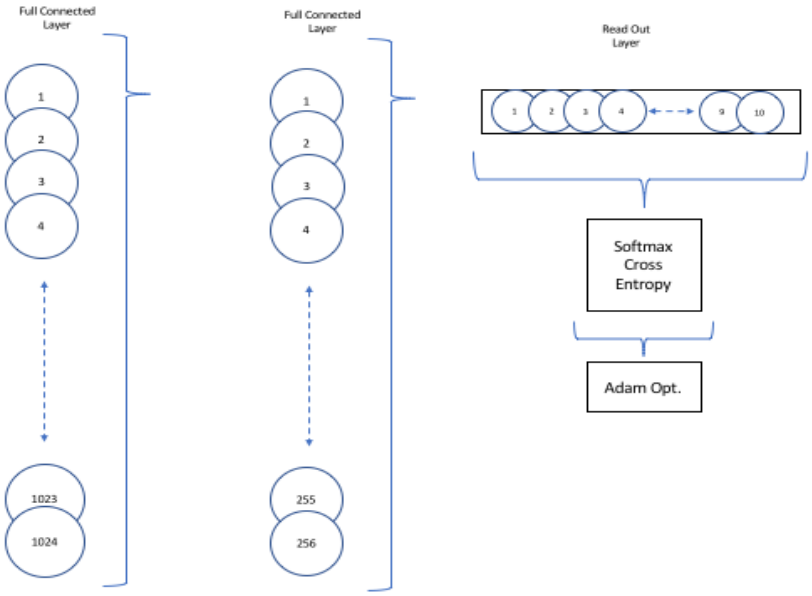



**Fig 17**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_6 (Conv2D) | (None, 32, 32, 32) | 896 |
| max_pooling2d_6 (MaxPooling2 | (None, 16, 16, 32) | 0 |
| conv2d_7 (Conv2D) | (None, 16, 16, 64) | 18496 |
| max_pooling2d_7 (MaxPooling2 | (None, 8, 8, 64) | 0 |
| flatten_5 (Flatten) | (None, 4096) | 0 |
| dense_9 (Dense) | (None, 1024) | 4195328 |
| dropout_5 (Dropout) | (None, 1024) | 0 |
| dense_10 (Dense) | (None, 256) | 262400 |
| dropout_6 (Dropout) | (None, 256) | 0 |
| dense_11 (Dense) | (None, 10) | 2570 |

Total params: 4,479,690
Trainable params: 4,479,690
Non-trainable params: 0

**Fig 18**

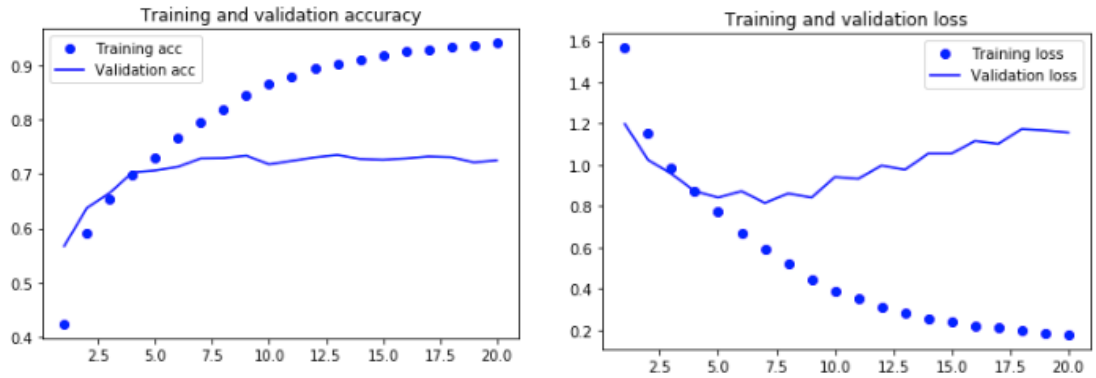



**Fig 19**

Third composed CNN architecture is not so different from previous trail. It did well on train set and it achieved %72 success. In spite of this, training time increase from 1280s to 1820 and testing time increased 11s to 18s. If we consider training accuracy, and training loss, there is obviously an overfitting problem. The architecture which is tried in this trial is not profitable when we compare it with previous one. More complex architectures do not always mean better results.

# 8.Benchmark

When we compare the results between trails, architecture 2 is slightly better than others. It has better prediction capability than architecture 2 and faster than architecture 3. [Fig 20]

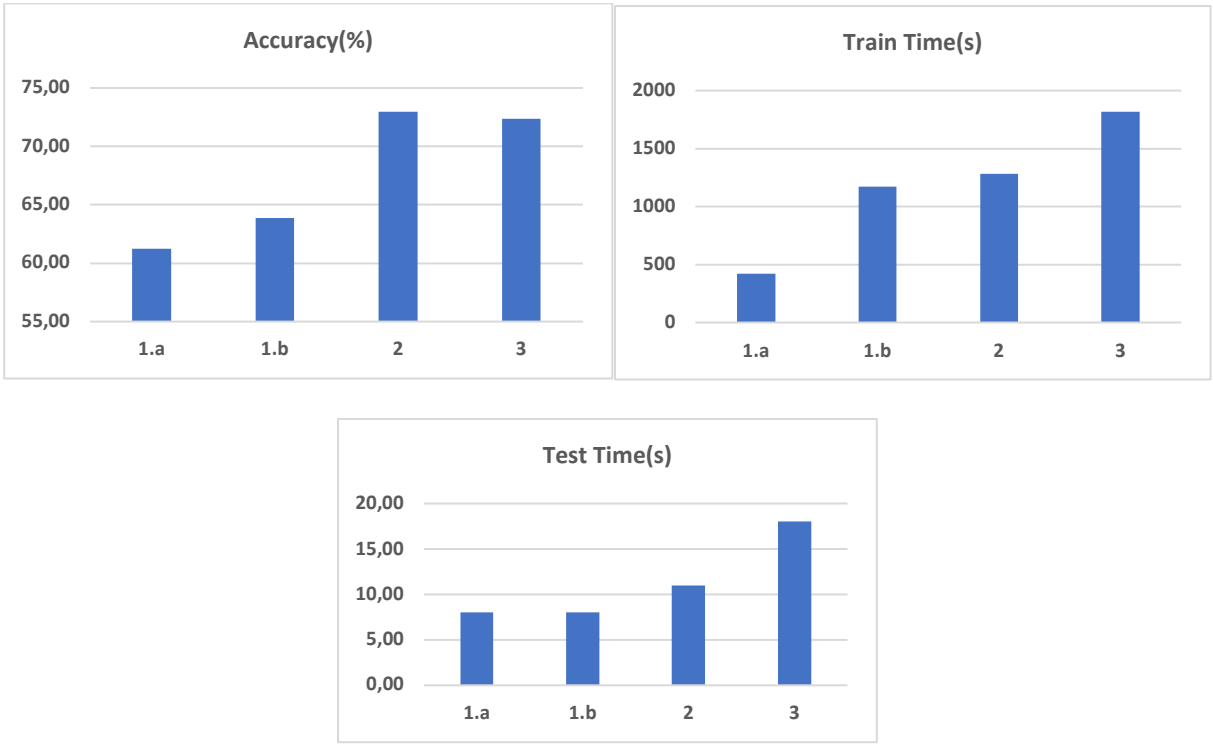



**Fig 20**

Even if architecture 2 achieved 72% accuracy and it is the best its class, we have to compare it with other kind of models. When we use pre-trained Resnet50**[9]** as a reference, we cannot say architecture 2 is successful. Pre-trained Resnet50 gives 90.8% accuracy and its means our model needs more work on it.

# 10.Conclusion

We cannot say second model is so successfully on test set. In despite of success on train set, it seems to be suffering from overfitting problem. It may occur because of big sized fully connected neuron layers.

In addition to this, doing pool process twice may not be good idea. Source images have 32x32 dimension and doing pool twice turn source images into 8x8 dimension. It may cause loss of valuable information in images.

In conclusion, creating a state of art convolutional neural network absolutely needs more works on it. It is impossible to train product ready prediction models in a relatively short time because of exponentially increasing calculation complexities.

# 10.1 Reflection

Even if we tried three different architectures, processes used in this project have similar main steps:

1. Getting data set and preparation
2. Exploration of dataset with presenting samples and histograms
3. Prepare dataset to use as input for CNN (reshaping)
4. Building Architecture with Layers (cnn, fc-nn, dropout, max pooling etc.)
5. Training model
6. Visualizing Training and Validation Accuracy
7. Testing model with test set

In addition to these steps, for benchmarking resnet50 pre-trained model implemented and test. Required steps

8. Importing resnet50 model
9. Resizing images in CIFAR dataset
10. Testing model with test set

During the implementation, I found steps 4 and 9. Building and finding model which predict with best accuracy needs so much trails and calculation time. Increasing number of layers used in model cause combinational complexity and it is so hard to predict how adding or removing layer will contribute to model without trying it. GPUs and distributed systems may help to solve this problem. Second most challenging step for me is 9. Before using Resnet50, you need to resize every image in data set from 32x32 to 200x200 because of pre-trained Resnet model only accept images larger than 197x197. Resizing 60000 images locked my computer nearly every trail. For solving this problem, instead of keeping whole data in memory, I wrote a method which saved data periodically chunk by chunk.

## 10.2 Improvements

For improving results, instead of trying combination of convolutional neural network layers blindly, researching and reading papers about experiments will be more helpful. Before adding or removing layers and extending size of neurons, reading papers would better which explain how these changes will affect our model.

Furthermore, using transfer learning may give us some insight about the convolutional neural networks and chance to improve it instead of reinventing wheel.

# 11.References


**[1] https://en.wikipedia.org/wiki/Convolutional_neural_network**
**[2] https://www.cs.toronto.edu/~kriz/cifar.html**
**[3] http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf**
**[4] http://www.merl.com/publications/docs/TR2004-043.pdf**
**[5] https://www.wikiwand.com/en/Scale-invariant_feature_transform**
**[6] https://www.wikiwand.com/en/Artificial_neural_network**
**[7] https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/**
**[8] https://keras.io/**
**[10] https://www.hackerearth.com/practice/machine-learning/transfer-learning/transfer-learning-intro/tutorial/**