Big Data Systems and Intelligence Analytics

Anusha Jain| jain.anus@husky.neu.edu || Anurag Singh | singh.anura@husky.neu.edu

Pruning Convolution Neural Network

Abstract:

When we build a Convolution Neural Network with large number of neurons it can achieve the state-of-the-art performance for variety of computer vision tasks, but how large or deep a neural network can be? Can we make them small without affecting the accuracy? Can they run on less computational power machines like CPU's or mobile devices? As a part of this research we are trying to find the optimum number of neurons in a network so that it can retain its accuracy even when we reduce the network size. Initially the model accuracy was around 71.87% (before pruning) and it dropped to 69.01% after pruning last layer in the network, we have removed around 80,000 parameters from the model.

Introduction:

We are trying to reduce the network architecture and parameters of a CNN model that has been created by us and it is trained on CIFAR-10 dataset. For this project we are pruning only the last layer i.e. the dense layer which has large number of neurons and try to maintain the accuracy that the original model has achieved.

Our focus will be to study the behavior of each neuron once it is trained, we will try to find the neurons that are inactive or not participating in image classification and remove them from the network. Since, we are creating a CNN model with 2 convolution layers and 3 fully connected layers, the network size is around 5MB which is already small, can we make it smaller without making it worse that is the biggest question?

Modern deep CNNs are composed of a variety of layer types, runtime during prediction is dominated by the evaluation of convolutional layers. With the goal of speeding up inference, we prune entire feature maps, so the resulting networks may be run efficiently even on embedded devices. We interleave greedy criteria-based pruning with fine-tuning by backpropagation, a computationally efficient procedure that maintains good generalization in the pruned network.

The success of this research will be measured by comparing the Initial Accuracy vs Accuracy after the pruning and Initial Network Size vs Network Size after pruning. It is still an open

question in machine learning to find the optimum number of neurons that is required to train any model for image classification.

Datasets:

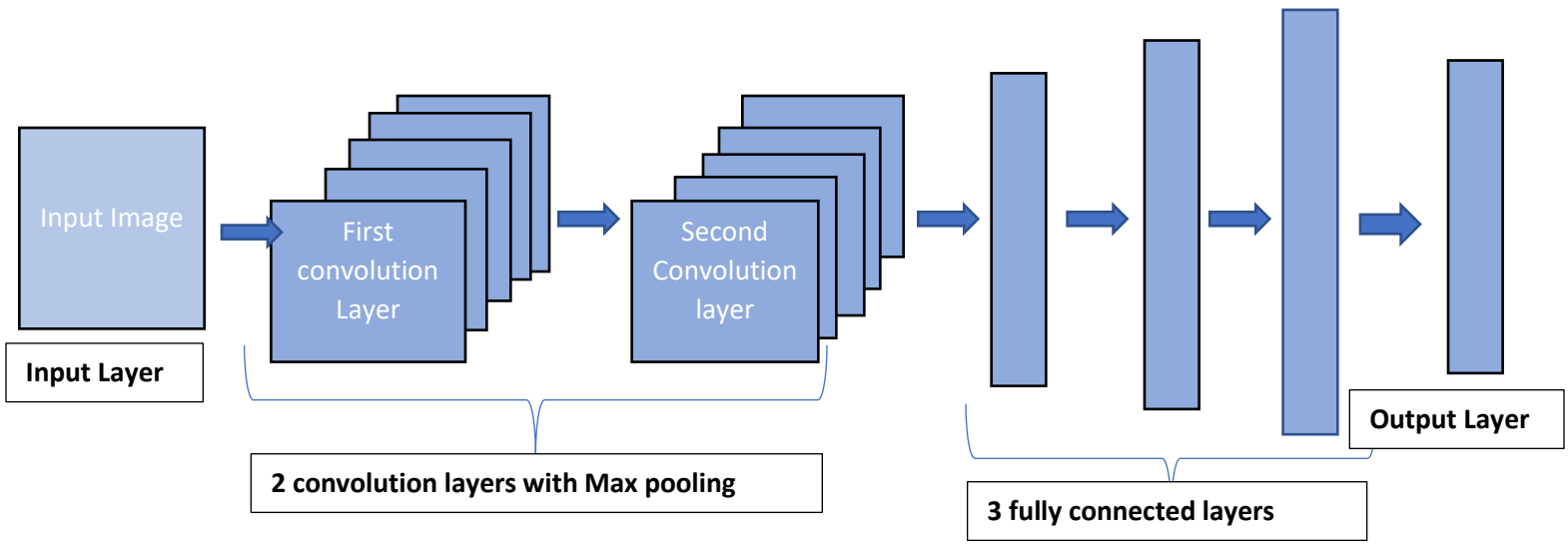
We have used CIFAR-10 dataset for training and validating the accuracy for CNN model. The dataset can be downloaded from the below link-

CIFAR-10: https://www.cs.toronto.edu/~kriz/cifar.html

Model Architecture Design:

We are building the CNN model with 2 convolution layers and 3 dense layers and training it on CIFAR-10 dataset to classify an Image.

Initial network architecture:



Model Summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               524416
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_3 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 713,546
Trainable params: 713,546
Non-trainable params: 0
_________________________________________________________________
```

It has total 713,546 parameters out of which around 18% of parameters are coming from dense3 layer which is the final fully connected layer and we are trying to reduce the size of that particular layer.

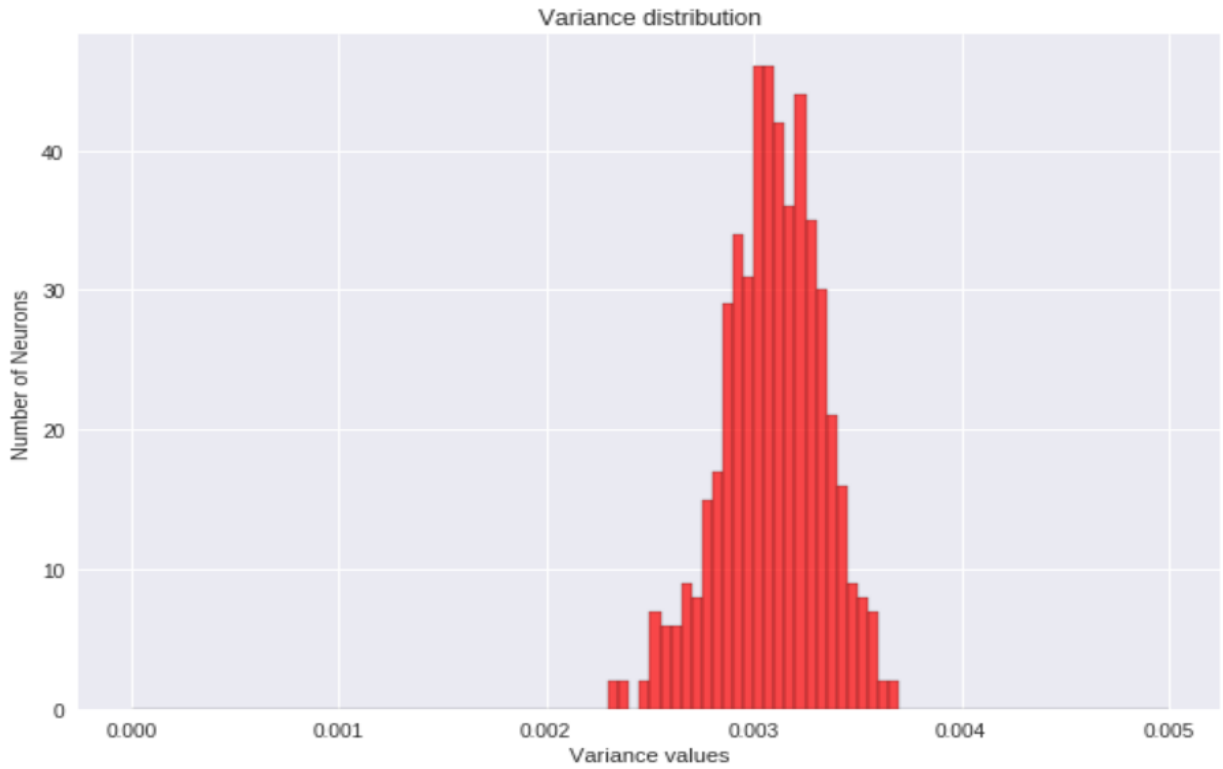
There are different techniques to prune a neural network, we are going to try our hand on few of them:

1. Pruning the network by removing low variance neurons
2. Pruning the network by removing low weighted neurons

1. Pruning the network by removing low variance neurons:

We will be calculating the variance of each neuron weights after every epoch and will remove the neuron that has low variance. The Idea behind this approach is to check whether the weight of a neuron is changing over time or not, if the weight is not changing after several epochs that implies that the neuron is not getting trained and it will not have any impact in classifying the images.

Variance distribution before pruning:



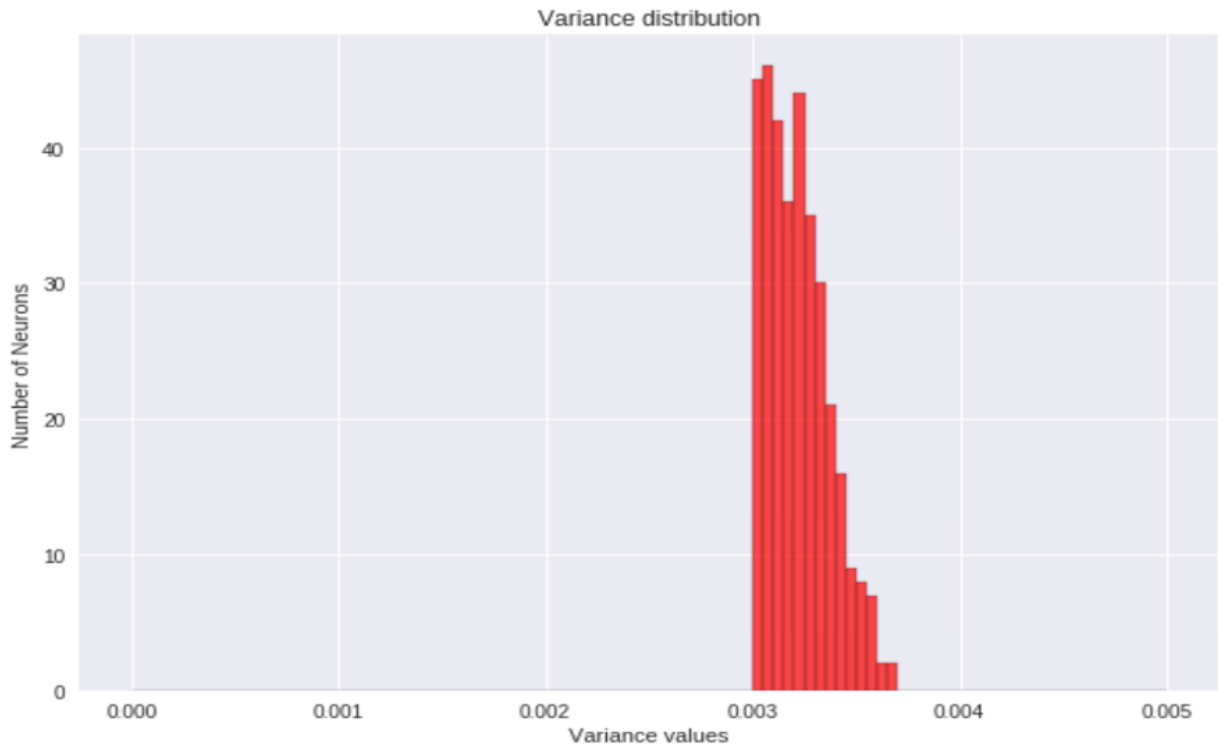
Extracting Variance: Initially we start with training the CNN model on CIFAR-10 dataset and check the accuracy of the model, by this time all the weights and bias values are adjusted to minimize the error. We have extracted the weights in a separate file by keeping the ModelCheckPoint's save_weights_only function as true. Once the weights are saved for all epochs we iterate through the files to extract the weights and using these values we are calculating the variance.

Threshold: Defining a threshold and removing all the weights which are below the threshold.

Threshold = (minimum Variance + Maximum Variance) / 2

Indexing of Neurons: Removing neurons directly from the network would not solve our problem, we need to remove the specific neurons with the zero variance and for that we have extracted the index of each neurons of the dense3 layer and using the index values we have removed the neurons from the network.

Variance distribution after pruning:



Retraining the model:

Removing the neurons makes the accuracy of the network worse as we are throwing away what the network has learnt, even if it did not have much importance. In order to compensate for this, we need to retrain the model.

Retraining is done by model.fit(). Here we retrain for less number of epochs and lower learning rate as the model is already trained, we only want to make small changes to tweak the results.

Change in model architecture after pruning:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               524416
_________________________________________________________________
dense_6 (Dense)              (None, 256)               33024
_________________________________________________________________
dense_7 (Dense)              (None, 363)               93291
_________________________________________________________________
dropout_6 (Dropout)          (None, 363)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                3640
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 673,763
Trainable params: 673,763
Non-trainable params: 0
_________________________________________________________________
```

Results:

	INITIAL MODEL	PRUNED MODEL
NETWORK SIZE	5616Kb	5061Kb
PARAMETERS	713,546	636,116
ACCURACY	71.47%	69.01%

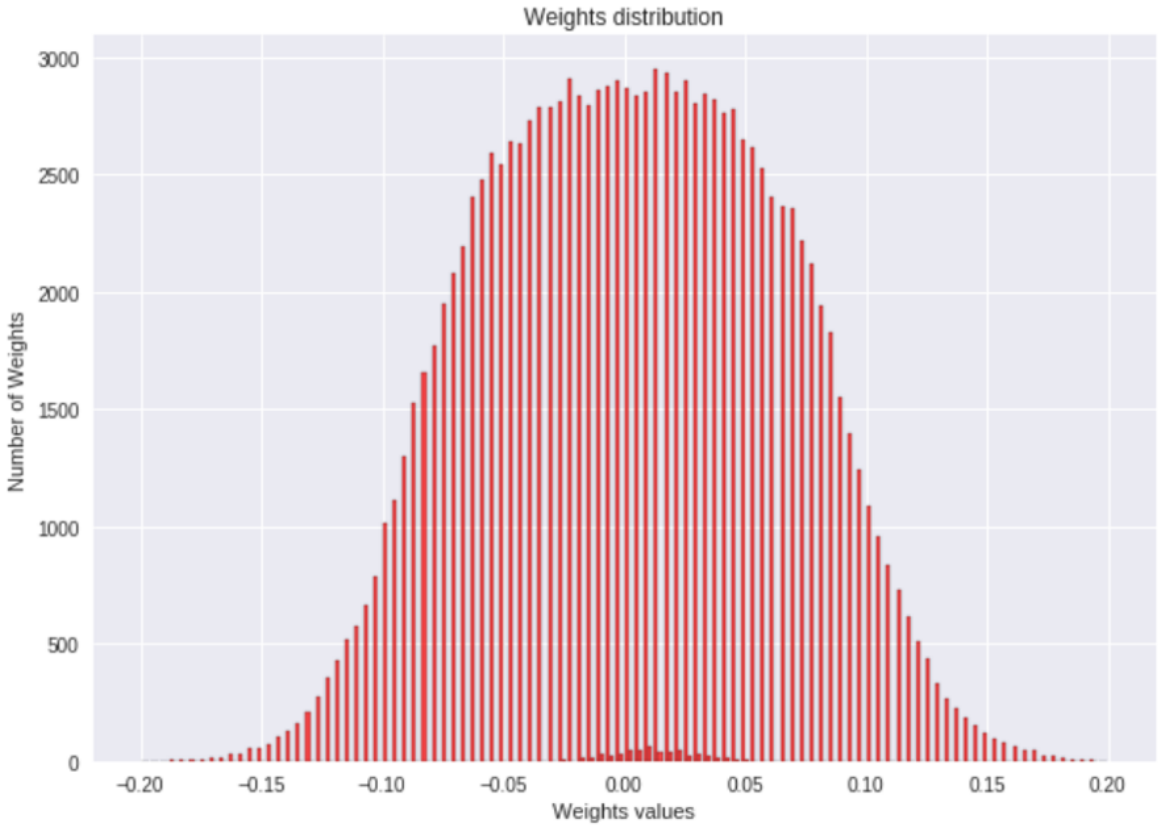
Summary:

Network size is reduced by **600kb**
Accuracy reduced by **2.5%**
New model has **80k less parameters**

2. Pruning the network by removing low weighted neurons:

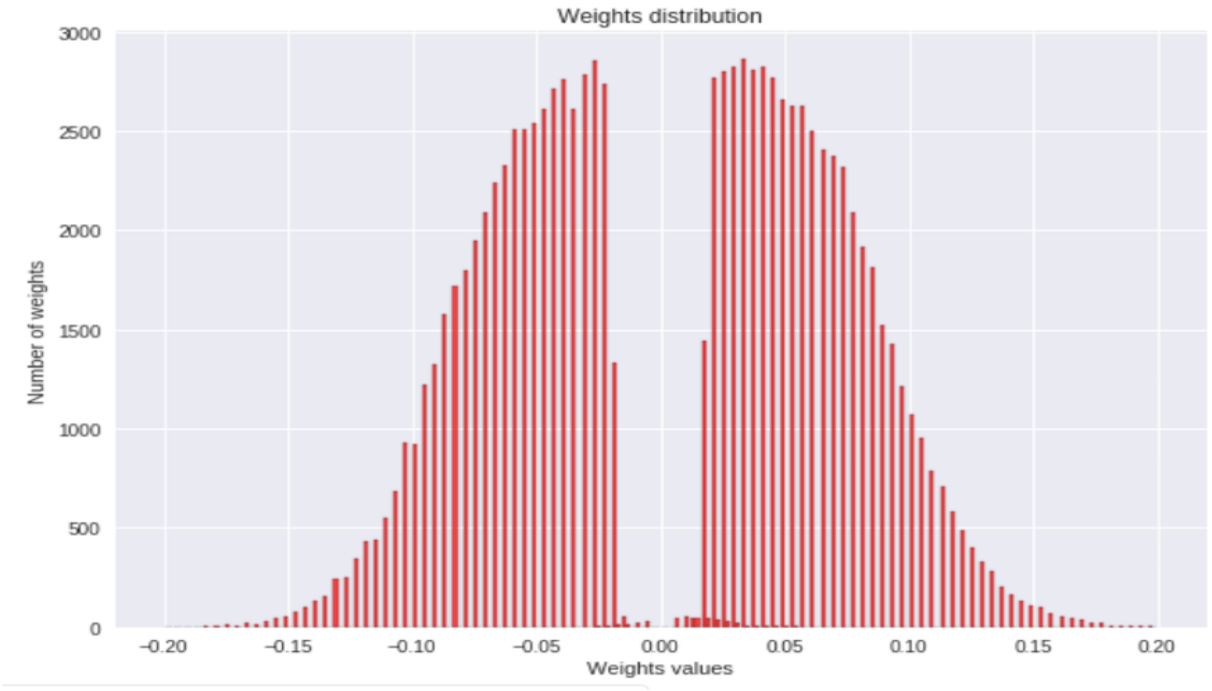
In CNN model the weights and biases are adjusted to minimize the loss through backpropagation, Once the network is fully trained all the important neurons which are contributing to make a decision are assigned with the highest weights and the neurons which are associated with 0 weights or relatively less weights are considered to be inactive. In order to reduce the size of the network we tried to remove the neurons associated with near zero weights. To achieve this, we have sorted the neurons of the last dense layer according to their absolute value and then we have removed around 20% of the neurons which has near zero weights.

Weight Distribution before Pruning:

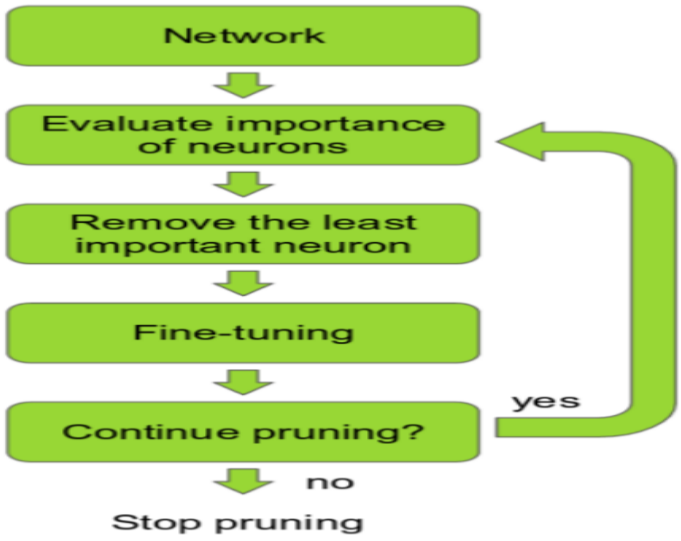


Once the neurons are removed from the network it means that the new model has that many less neurons to train, that improves the training time and it uses less computational power to run the deep neural network.

Weight Distribution after Pruning:



Low weights do not means removing the negative values because, in neural network training negative weights are equally important as the positive weights. We have taken the absolute value of the weights and then removed the 20% of neurons from the network. By doing this we are removing around 30,000 parameters from the neural network which makes the network lighter. This process can be repeated multiple times and this method can also be applied to other layers of neural network. The scope of this research is limited to pruning of only the last dense layer of the neural network.



Change in model architecture after pruning:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
dropout_7 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 4096)              0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               524416
_________________________________________________________________
dense_10 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_11 (Dense)             (None, 410)               105370
_________________________________________________________________
dropout_9 (Dropout)          (None, 410)               0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                4110
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 686,312
Trainable params: 686,312
Non-trainable params: 0
_________________________________________________________________
```

Results:

	INITIAL MODEL	PRUNED MODEL
NETWORK SIZE	5616Kb	5448Kb
PARAMETERS	713,546	686,312
ACCURACY	71.47%	70%

Summary:

Network size is reduced by **200kb**
Accuracy reduced by **1.5%**
New model has **30k less parameters**

Conclusion:

We have tried both the approaches to remove the neurons from the model and based on the results we can say that the removing neurons by low variance was slightly better effect than the removing the low weighted neurons. Since, we have tried this method only for one iteration and that to on the last layer only, so we do not have a solid conclusion which can support our thesis. In future we will try to apply this methods on other layers as well and try to prune the model iteratively.

Acknowledgment:

We would like to show our sincere gratitude to professor Nik Bear Brown for guiding us during this project.

References:

https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py
https://github.com/ex4sperans/pruning_with_tensorflow
https://www.tensorflow.org/api_docs/python
https://gist.github.com/hollance/d15c0cc6004a5479c00ac26bce61ac8d
https://keras.io/getting-started/faq/