Leila Suasnabar
3109-9816-56
lsuasnab@usc.edu

EE 569: Homework #4

Problem 1 : CNN Training and Its Application to the CIFAR-10 Dataset

1.1 Motivation

There are many machine learning algorithms used to classify data. Nevertheless, it wasn't until recently that convolutional neural networks (CNN) were employed. This type of artificial neural network learns appropriate features automatically. Unlike previous algorithms, the best results can be obtained by using a multilayer neural network such as CNN. The advantage of this, is that the model can connect all the neurons from previous layers and optimize the classifying result accordingly.

These type of neural network is important, since it allows the computer to recognize images/scenes/etc based on its features. Nowadays, image recognition is employed in gaming, augmented reality, medical imaging, image search, security and many other areas,

1.2 Approach

1.2.1 CNN Architecture and Training

The CNN architecture can be decomposed as: [1]

a. The fully Connected Layer:

In this layer, every neuron is connected to any neurons from previous layers. We can think of these layers similar as convolutional layer except these are not only locally connected. This type of layer allows us to learn (possible) non-linear functions in the feature space obtained from the convolutional layers.

b. The convolution layer:

It's main purpose is to extract features by convolving the input image with nxn filters. The amount of filters indicates the quantity of resulting feature maps. Different types of filters detect different features within an image. For instance, these filters can perform edge detection, curve detection, smoothening, sharpening, etc.

c. The max pooling layer:

It reduces the dimensionality of the input data. For a stride of 2, the data is minimized to half its original size. This is done by passing a sliding box of kxk, and storing the maximum value from this box.

d. The activation Function:

The main purpose of this function is to make the neural network non-linear. In other words, its main function is to obtain a non-linear decision boundary. A linear decision boundary may not work for all data types, hence it's better to use a generalized model to classify the data.

e. The softmax function:

The main purpose of this function is to transform the data into a new range [0,1] that add up to one. We can think of this as putting the data into a pdf.

These functions of these components are to:

a. Connect previous layers while keeping the weights
b. Obtain features from the image
c. Downsample the input image while keeping its depth
d. Assure that the data is classified with a non-linear decision boundary
e. In a sense obtain a one_hot label in a pdf form, to obtain which class value is the highest for that particular image

The main difference between CNN and multi-layer perceptron (MLP) is that CNN uses convolution and pooling. The feature maps that the convolutional layer outputs can detect local patterns, while pooling reduces the dimension of the input which is then passed onto a multi -layer perceptron (fully connected layers). Additionally the initialization process is different between these two methods. While the CNN provides more initialization parameter freedom, MLP are typically initialized using random numbers.

Unlike traditional computer vision methods, CNN uses multiple layers to find new feature spaces. This allows, us to obtain more information that can be used to classify the data. We are even able to sharpen our feature spaces by performing multiple convolution layers consecutively. While non CNN models, use hand crafted feature spaces which ultimately are not as efficient as CNN's feature spaces.

Back propagation allows us to train a network regardless of the quantity of layers within the network. Furthermore, it is able to optimize the weights while minimizing the error/cost/loss. This procedure is done by computing gradient values that are obtained from the implicit error values (obtained from the loss function).

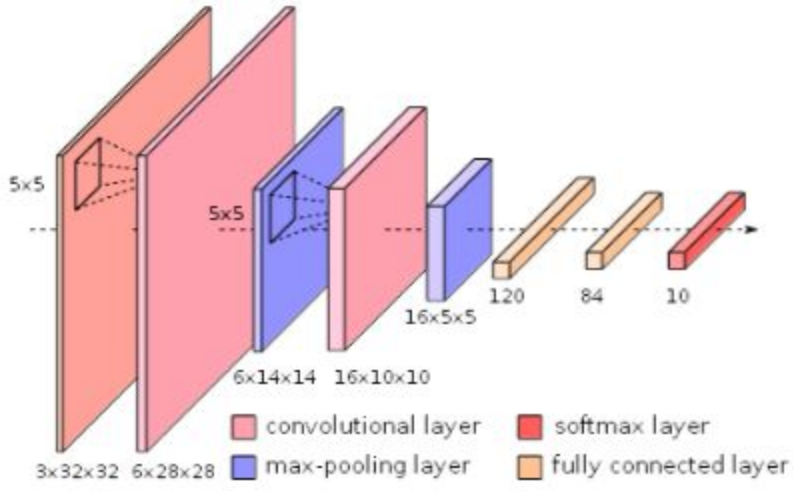
Lastly, the loss function measure the compatibility between the prediction labels and ground truth labels for all images[1]. For this problem, we'll use the cross-entropy loss function:

$$H(y,\hat{y}) = \sum_i y_i \log \frac{1}{\hat{y_i}} = -\sum_i y_i \log \hat{y_i}$$

This algorithm uses one_hot_encoded data (yi - hat) and one valued (yi) labels. Additionally, this loss function doesn't let the weights change into very small values since this would stall the training model [7]

1.2.2 Application of the CNN to CIFAR-10 Dataset

For this part, we base our model on a network architecture derived from LeNet:



The following parameters are changed and compared to find the best solution:

```
# Parameters
learning_rate
training_epochs
batch_size
```

For the preprocessing step, the data is subtracted by its global mean per channel, and subsequently the data is passed through a global contrast normalization using euclidean distance. This normalization type computes the mean and standard deviation across features per image. Next, ZCA whitening is performed on each batch. This will remove any correlation between one pixel and another[2]. This process can be done with the following implementation[3]:

```
#ZCA Whitening
sigma = np.cov(dataMatrix,rowvar=True)
U,S,V = np.linalg.svd(sigma)
#To prevent division by 0
epsilon = 1e-5
dataProcessed = np.dot(U,np.dot(np.diag(1.0/np.sqrt(S+epsilon)),U.T))
dataProcessed = np.dot(dataProcessed,dataMatrix)
```

It is important to normalize the data before performing any whitening. After whitening the data is decorrelated thus making it simpler to model it. Another important step is to choose a random network initializer. It is important to initialize the neural network with the correct

weights. If we use very small weights, then the model can become non-linear and hence lose any advantages from using multiple layers [4].

For this problem, we chose an xavier initializer for the weights and biases. This type of initializer keeps the variance the same with each passing layer. Ultimately, this initializer preserves the backpropagated signal well [5]. Lastly, for this problem, the model was optimized with an Adam Optimizer. This is a robust optimizer for large datasets, for this optimizer can do bias correction [6].

1.2.3 K-means with CNN

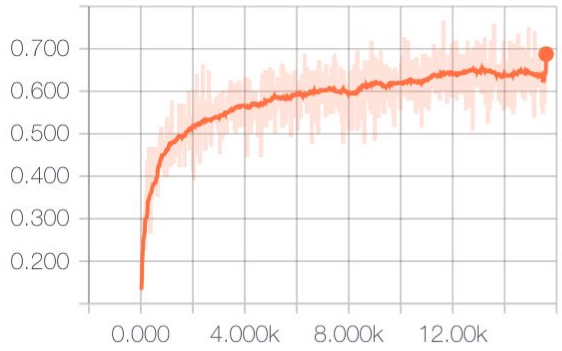
For this part, we compare the performance of using a random initializer with a k-means initializer. The k-means initialization works by getting inputs from a feature space, and after normalization we can find a hyper-sphere of these values. From this, we can use k-means to find the weight of the layers based on the centroids from the normalized and whitened image batches. We repeat this process for all the convolutional and fully connected layer. And lastly, we compare the new results with the labels to find the accuracy on the test and train dataset. Overall, this neural network method is trained until convergence in performance [10] Unfortunately, I was unable to fully solve this problem, hence why the accuracies for the test and train data are not displayed.

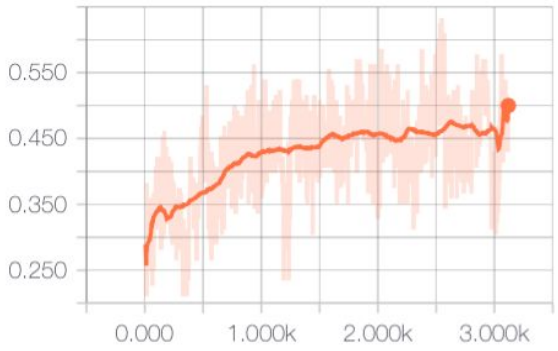
1.3 Results

1.3.1 CNN Architecture and Training

Explanation in 1.2.1.

1.3.2 Application of the CNN to CIFAR-10 Dataset





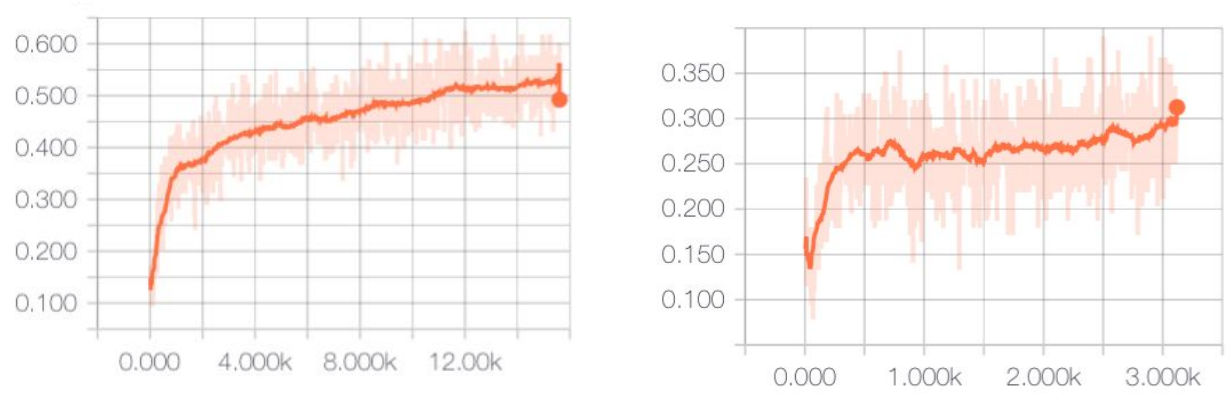


Fig 1. Batch_size = 128, epochs = 40. Trying different learning rates with . Left: Train accuracy. Right: Test accuracy. Top: Learning_rate = 0.001. Bottom: Learning_rate = 0.0001

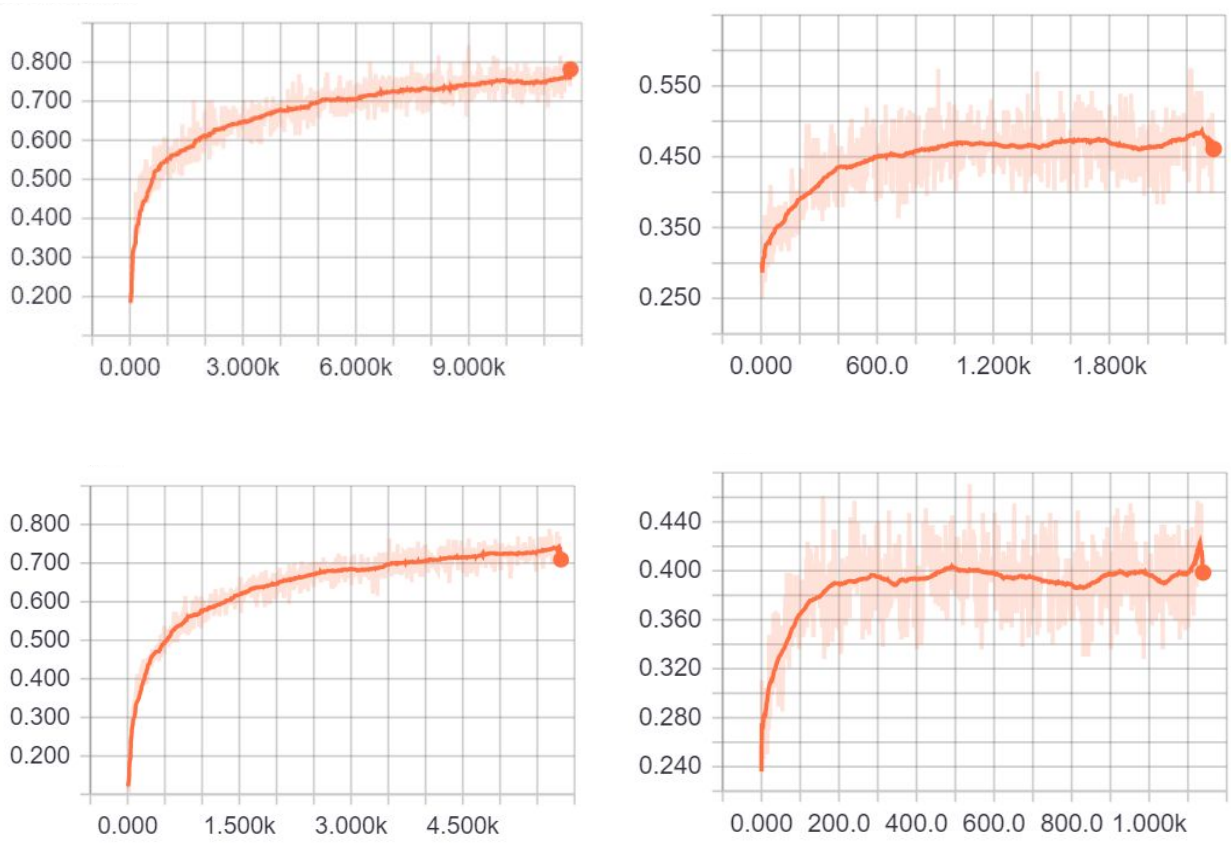


Fig 2. Trying different batches size. Left: Train data. Right: Test data. Top: batch_size = 256. Bottom: batch_size = 512

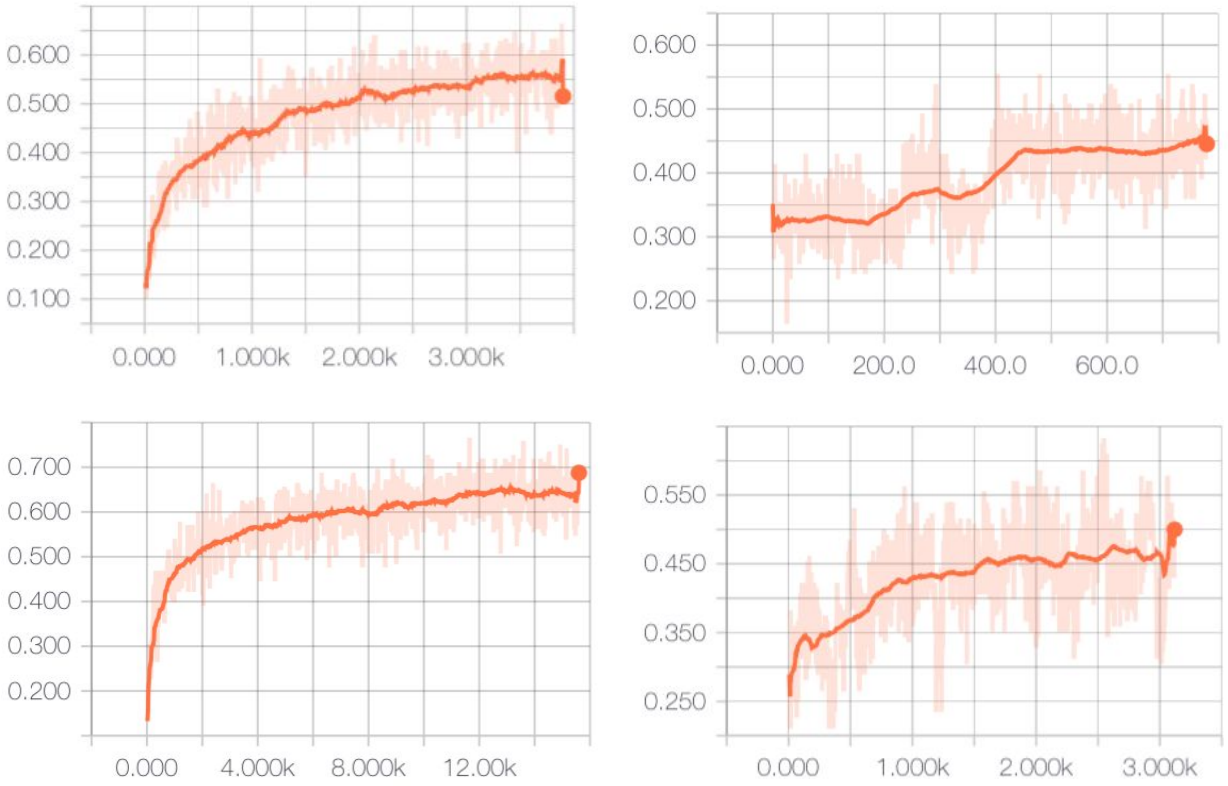


Fig 3. Batch_size = 128, learning_rate = 0.001. Trying different epochs number. Left: Train accuracy. Right: Test accuracy. Top: epochs = 10. Bottom: epochs = 40

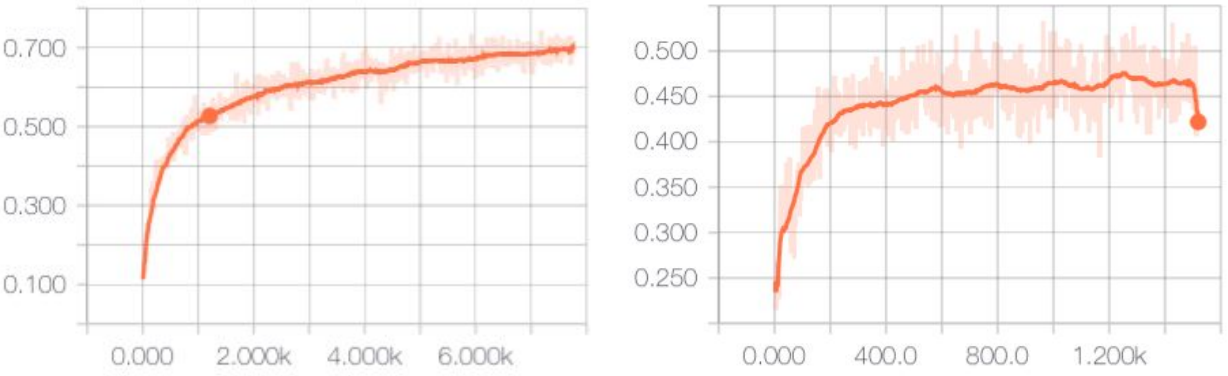


Fig 4. Best result. Learning rate = 0.001, batch_size = 256, epochs = 80

1.3.3 K-means with CNN

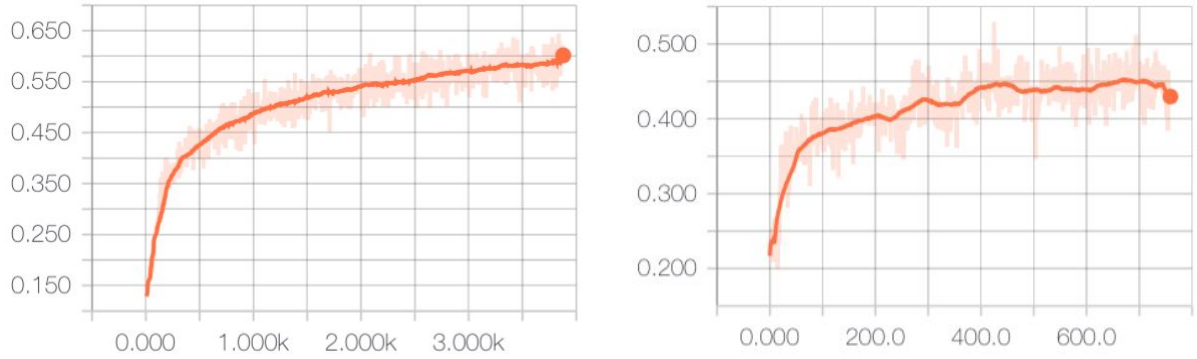


Fig 5. Batch size = 512, learning rate = 0.001.Left: Train accuracy. Right: Test accuracy. Top: Random Initialization.

1.4 Discussion

1.4.1 CNN Architecture and Training

Explanation in 1.2.1.

1.4.2 Application of the CNN to CIFAR-10 Dataset

From fig 1, we can see the effect of trying different learning rates. Particularly for this problem, the best result was obtained when using a learning rate of 0.001. The smaller the value of the learning rate, the more iterations we would need for the model to converge to the best values. Notice these two models were training under the same amount of epochs, therefore as observed in fig 1, the learning rate of 0.001 performed better than a learning rate of 0.0001

By trying different batch sizes, we can see how in fig 2 the best result is obtained when using 256 images per batch than using 512 images per batch. Also, as we increase the batch size, the computational time increases as well. So it is more efficient to avoid using large batch sizes. It's good to notice that using a larger batch size reduces the noise in the accuracy graphs.

In fig 3, we can see the effect of using different epochs. As expected, the more epoch we use the more the model is trained and hence produces better classification results.

With the best parameter from the previous experiments, we can see that the data has a train accuracy of ~70% and a test accuracy of ~46% (ignoring the last drop of value). This big difference in accuracy tells us that there model may have some overfitting problems. Nevertheless, the model is able to predict the correct label around half the time.

1.4.3 K-means with CNN

In the paper the model only works better than a random initialized network when the network is a simple model (MNIST). Based on this assumption, we can predict that for a deep multilayer neural network using the CIFAR-10, the data would work better for a random initializer model.

One of the advantages of using k means initialization (or any other unsupervised initialization), is that we don't need to rely on (potentially) expensive labeling. Fig 5, shows the accuracies using a random initializer.

Problem 2 : Capability and Limitation of Convolutional Neural Network

2.1 Motivation

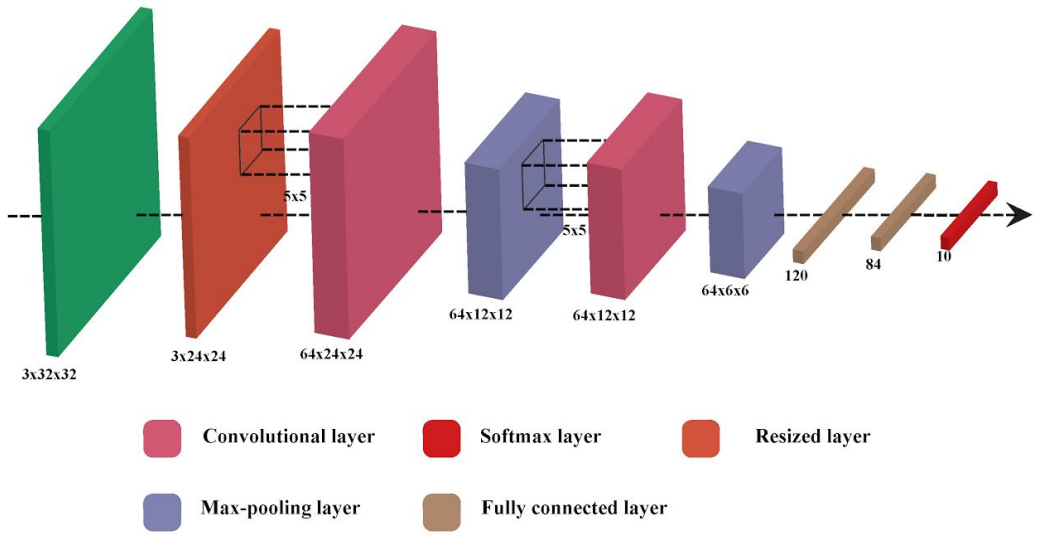
As we will see later in this problem, deep neural networks work more efficient than simple neural networks. Not only does the accuracy increase, but the overall effectivity of the model becomes more efficient. In fact, in the real world deep neural networks are widely more utilized than simple neural networks.

Additionally, we'll see how other state-of-the-art implementation work and understand the similarities and differences between this model and the modified and baseline model.

2.2 Approach

2.2.1 Improving Your Network for CIFAR-10 Dataset

Architecture for the modified convolutional neural network:



For this part, we have kept the baseline of the CNN from part 1a. The initial data setup was performed differently than in part 1a. For this part the 50000/10000 images were shuffled,

and set up in batches after preprocessing. Each image was randomly cropped to a 24x24 image, and subsequently randomly flipped, added a change in brightness, changed into a random contrast. Upon obtained the new modified image, this was normalized and then shuffled and put into batches.

Right after a convolutional layer (with padding), the features were passed by a dropout of probability 0.2. Dropout processing is a good way to decrease overfitting on large datasets. Lastly, right after the max-pooling layers (with padding), a local response normalization processing was done with the parameters from [8]. This step allows to normalize the ReLU neurons, and relatively boost the neurons that have a larger activation. Also, notice the increase in the convolutional layers, this allows us to obtain more features from which we can perform information.

Lastly, a very crucial change was the use of an exponential decay on the Gradient Descent optimizer employed in this problem. This exponential decay allows us to lower the learning rate as the training progresses.

The following are the overall model parameters used for this part:

```
# Exponential Learning Rate parameters
learning_rate = 0.1
decay_rate = 0.96
decay_steps = 300

# Set model parameters
iterations = 17000
batch_size = 128
prob = 0.2 #used for the dropout function
```

To maintain a low computational time, we decided to use only 128 image batches per training. Also notice the parameters of the exponential decay, this allows us to drop the learning rate value as the training takes place.

Note: Parts of this implementation were modified with tutorials from the Tensorflow website. More information found in the coding file comments.

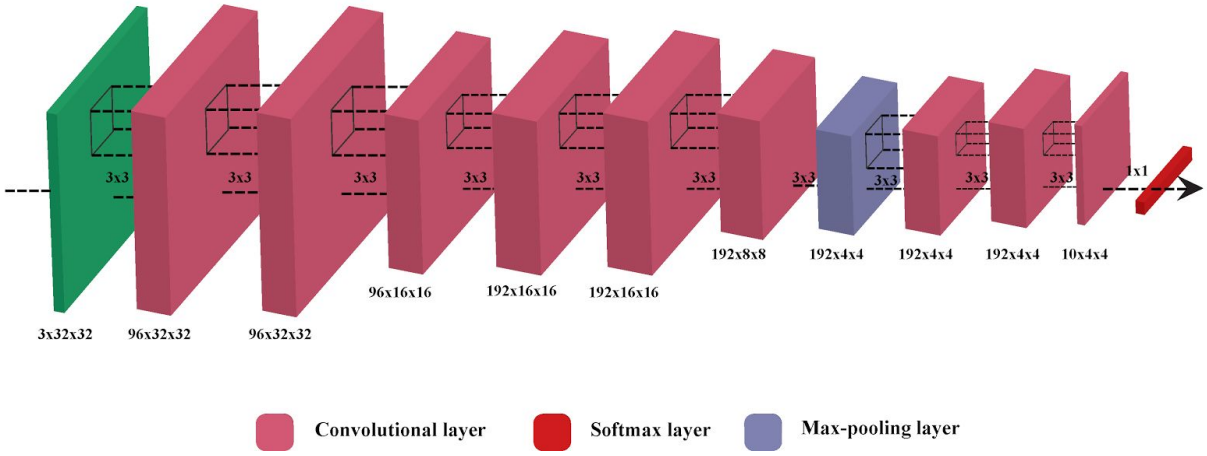
2.2.2 State-of-the-Art CIFAR-10 Implementation

In [9], we see that the authors implemented several models to find the best classification result on the CIFAR-10. From the paper we can see that some of the neural networks used are:

Model		
Strided-CNN-C	ConvPool-CNN-C	All-CNN-C
Input 32 × 32 RGB image		
3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU with stride $r = 2$	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU	3 × 3 conv. 96 ReLU 3 × 3 conv. 96 ReLU
	3 × 3 max-pooling stride 2	3 × 3 conv. 96 ReLU with stride $r = 2$
3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU with stride $r = 2$	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU	3 × 3 conv. 192 ReLU 3 × 3 conv. 192 ReLU
	3 × 3 max-pooling stride 2	3 × 3 conv. 192 ReLU with stride $r = 2$
3 × 3 max-pooling stride 2		
3 × 3 conv. 192 ReLU		
1 × 1 conv. 192 ReLU		
1 × 1 conv. 10 ReLU		
global averaging over 6 × 6 spatial dimensions		
10 or 100-way softmax		

Image from [9]

Architecture for the ALL-CNN-C neural network:



For this part of the problem, we'll focus on the ALL-CNN-C which as is later described on the paper, the model that obtained the lowest test classification error.

This neural network is implemented with the following key points:

a. No data augmentation
b. Vanilla stochastic gradient descent with momentum for the optimizer, which as stated in the paper, reaches a good performance without the need of activation functions, normalization or max-pooling[9]
c. This model is based on convolutional layers as the main way to obtain the features
d. Smaller convolutional kernel sizes is better
e. Use of strided convolutional instead of using max-pooling

As we can see from the neural network architecture, we start by setting up all the layers. The input images are first whitened and contrast normalized. This model uses adaptive learning rate which multiplies the original learning rate by 0.1 every predetermined amount of epochs [200,250,300]. Additionally, dropouts of 0.2 were used on the input image, and 0.5 after a pooling layer. The layers were regularized with a weight decay of 0.001.

In summary this model is primarily a stack of convolutional layers, with some convolution layers with stride 2 which serve as max-pooling layers but are more effective.

2.3 Results

2.3.1 Improving Your Network for CIFAR-10 Dataset

Fig 6. Accuracy Results

2.3.2 State-of-the-Art CIFAR-10 Implementation

Description is in part 2.2.2 and discussion in 2.4.2.

2.4 Discussion

2.4.1 Improving Your Network for CIFAR-10 Dataset

From fig 6, we can see that the accuracy on the training set is ~94.5 and ~80.5 on the test set. Notice that there's a difference of ~14% of accuracy, this indicates that the model does not have a large overfitting problem as in problem 1. We can see that the dropout and local response normalization processing improved the accuracy significantly. Also, this part was done on more iterations than the original CNN, which also contributed to its improvement. Furthermore, it is clear that using more filters and padding on the convolutional layers did indeed produce more features that overall benefitted our model results.

2.4.2 State-of-the-Art CIFAR-10 Implementation

In comparison, we can see that the deep multilayer neural network ALL-CNN-C performed significantly better compared to the previous results. The baseline CNN used a total of 2 convolutional layers with no padding +ReLU + max-pooling, 2 fully connected layers + ReLU, and a fully connected layer + softmax to obtain the class probabilities. The improved CNN used a total of 2 convolutional layers with padding and with more filters (64) + ReLU + dropout(0.5) + max-pooling + local response normalization, 2 fully connected layers + ReLU, and a fully connected layer with softmax. The ALL-CNN-C uses smaller kernel sizes for the convolutional layers with padding, instead of using a max-pool layer the model uses a strided convolutional layer to reduce the feature space dimensionality; furthermore, the algorithm uses a global averaging right before performing softmax on the final output.

Accuracy result:

- Baseline CNN: ~46%
- Improved CNN: ~80%
- ALL-CNN-C: ~91%

	PROS	CONS
Baseline CNN	● Fast computation time ● Can be computed on CPU	● Very low accuracy ● Doesn't use adaptive learning rate ● Doesn't fix overfitting with a dropout ● No data augmentation ● No padding on convolutional layers which reduces the

		feature space
Improved CNN	• Moderate computation time • Can be computed on CPU • Uses adaptive learning rate: exponential decay • Fixes overfitting with a dropout • Padding on the convolutional layers	• Adequate accuracy • Fixes overfitting only on convolutional layers • No data augmentation
ALL-CNN-C	• Slow computation time • Cost effective by not using max-pooling layers which don't always improve performance • Uses adaptive learning decay • Fixes overfitting with a dropout • Padding on the convolutional layers	• High accuracy • No data augmentation

References

[1] http://cs231n.github.io/neural-networks-2/

[2] Appendix A of *Learning Multiple Layers of Features from Tiny Images* by A. Krizhevsky.

[3] http://ufldl.stanford.edu/wiki/index.php/Implementing_PCA/Whitening

[4] https://prateekvjoshi.com/2016/03/29/understanding-xavier-initialization-in-deep-neural-networks/

[5] http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf

[6] https://arxiv.org/pdf/1412.6980.pdf

[7]https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/

[8]http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[9] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, *Striving for Simplicity: The All Convolutional Net*

[10]https://arxiv.org/pdf/1701.08481.pdf