HOMEWORK -2 CONVOLUTIONAL NEURAL NETWORK

**Name:** Titash Mandal, Ajay Thorve

**Net Id:** tm2761, aat414

Given the cifar-10 Dataset which contains 45,000 training examples and 10 output labels, we implemented a Convolutional Neural Network for image recognition. The process description is as follows:

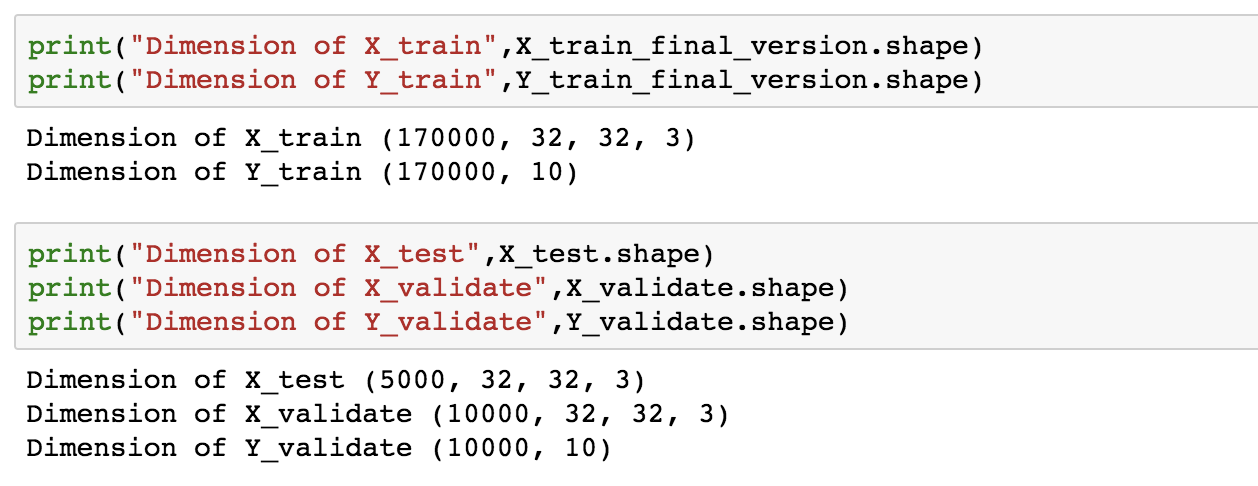
**1. Duration of timing:** The training dataset was trained for a continuous 5 hours for 100 epochs after applying data augmentation. This process was repeated several times by saving the checkpoints of the previous run iterations.

**ARCHITECTURE**

**2.** **Image Augmentation:**

Data augmentation can help reduce overfitting on models, especially when the training data is less, by increasing the size of the training data using information only in our training data. Doing so helped to make the model more robust in identifying features in the training dataset. To implement this, we used tensor flow’s inbuilt functions to create a set images which are flipped from left to right that is translated along the X axis, flipped from top to bottom that is translated along the Y axis and added some values to the RGB channels of the image to change their contrast. This resulted in creating a dataset of X\_train whose dimensions are 180000, 32,32,3.

**The size of the training dataset is as follows:**





Initial CNN tensor flow architecture used was as follows.

**CONVOLUTION**

**RELU**

**MAXPOOL**

**NORMALIZATION**

**CONVOLUTION**

**RELU**

**NORMALIZATION**

**MAXPOOL**

**FLATTEN**

**FULLYCONNECTED**

**FULLYCONNECTED**

**SOFTMAX**

**A.** One of the main disadvantages of Convolution with a filter is that the image size reduces after each convolution step. To prevent this from happening we used **Same Padding in the all the steps of tensor flow architecture to keep the image size constant.**

**A.1.** In the first Convolution step, we used strides= 1,1 along the height and the width of the image and applied **8 filters** each of dimension **4\*4\*3**.

**A.2.** After applying ReLU we implemented MAXPOOL with a **window size of 8\*8** and applied local response normalization to the output.

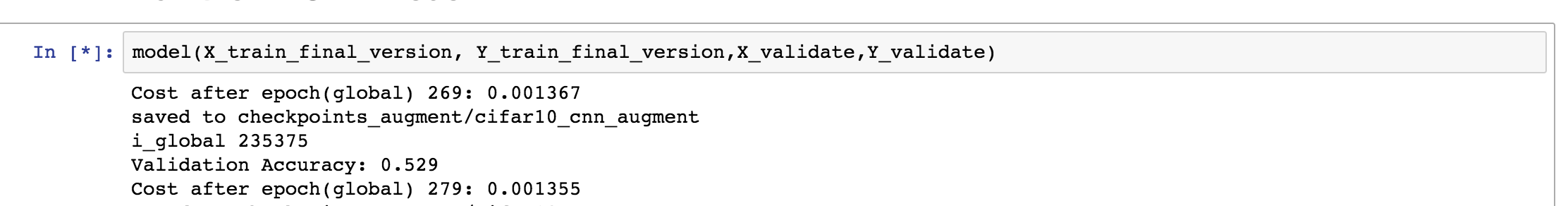
ReLU neurons have unbounded activations and we need LRN to normalize that. We want to detect high frequency features with a large response. If we normalize around the local neighborhood of the excited neuron, it becomes even more sensitive as compared to its neighbors.

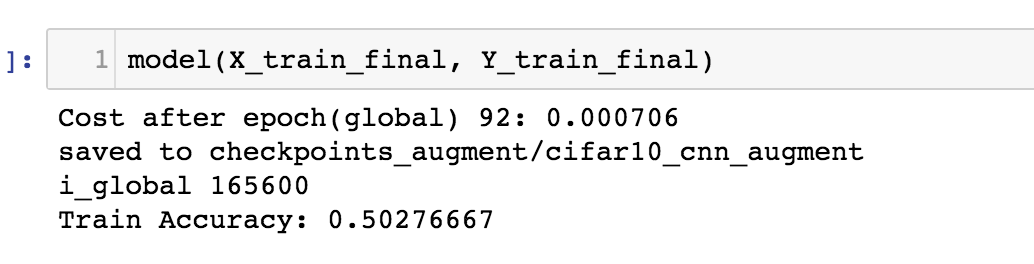
**A.3** For the second Convolution step we used **16 filters** to convolve the image with each having a dimension of **2\*2\*8** and applied ReLU to its output.

**A.4** To the output of ReLU, we applied local response normalization and again applied Maxpooling with a **window of size 4\*4 and stride 4.**

**A.5** The last steps were to apply flattening and implementing two fully connected layers of neurons 256 and 10 respectively. The Y predicted was determined by applying a softmax cross entropy function to the activation of the last layer.

**The accuracy received on the trained and validation dataset using this CNN Architecture is as follows.**





As the accuracy was not improving after training the dataset for a large number of epochs, we decided to change the CNN architecture and test the new model again.

**FINAL CNN ARCHITECTURE USED IS AS FOLLOWS.**

**CONVOLUTION**

**ReLU**

**SOFTMAX**

**FULLY CONNECTED LAYER - 1**

REPEATED THREE TIMES

**DROPOUT**

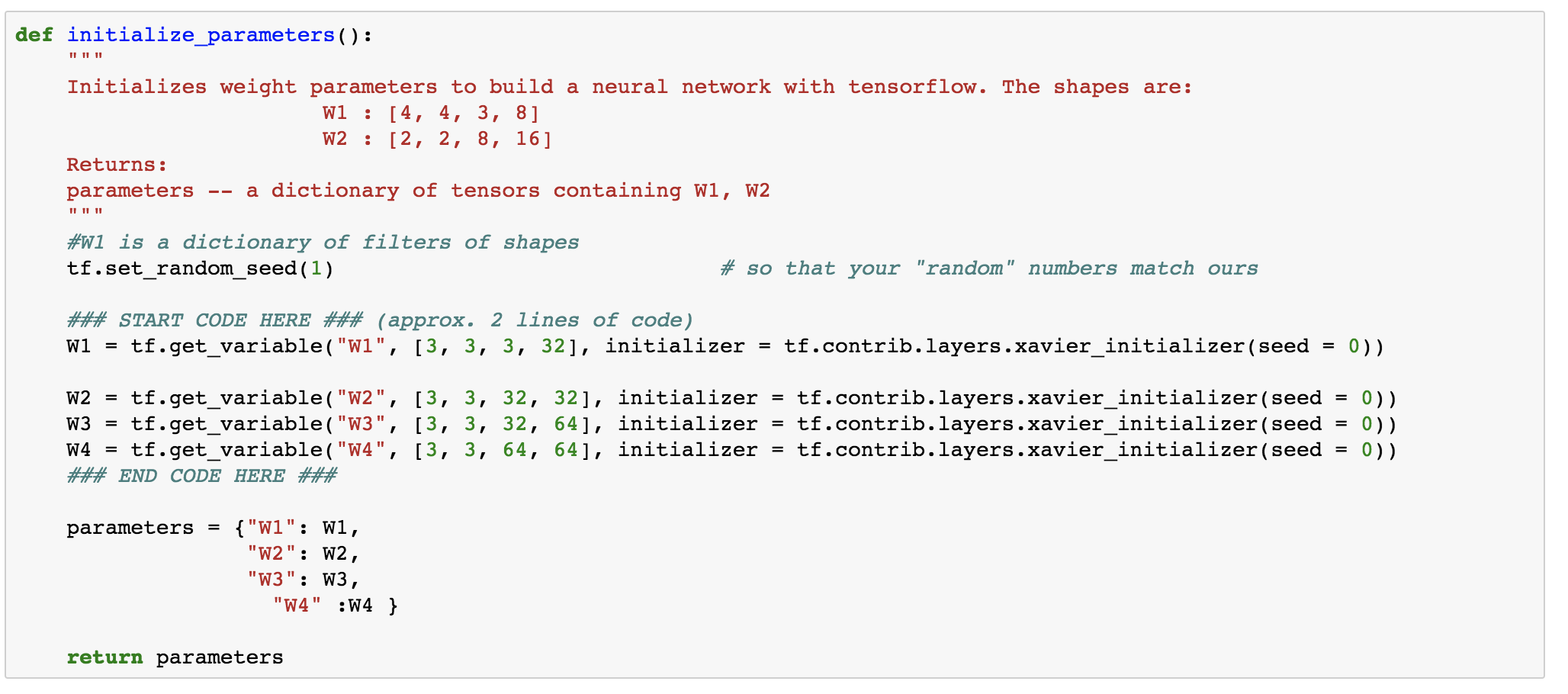
**MAXPOOLING**

**FLATTEN**

**RELU**

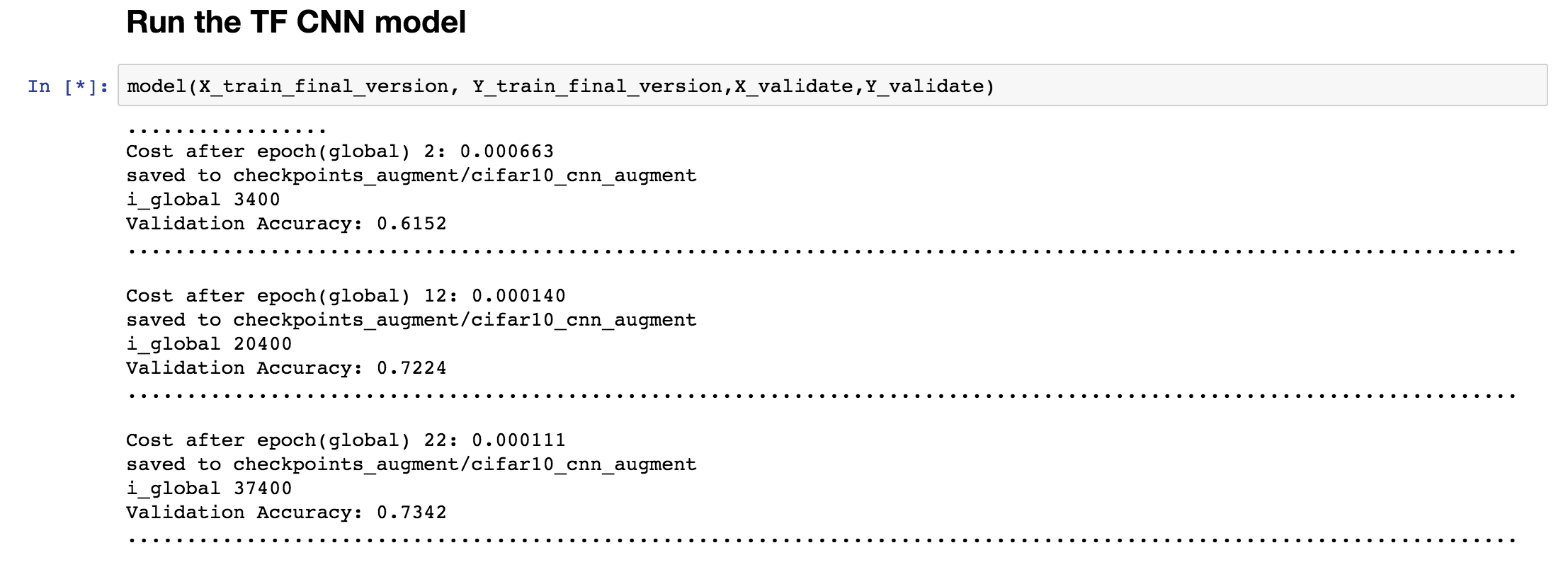
**CONVOLUTION**

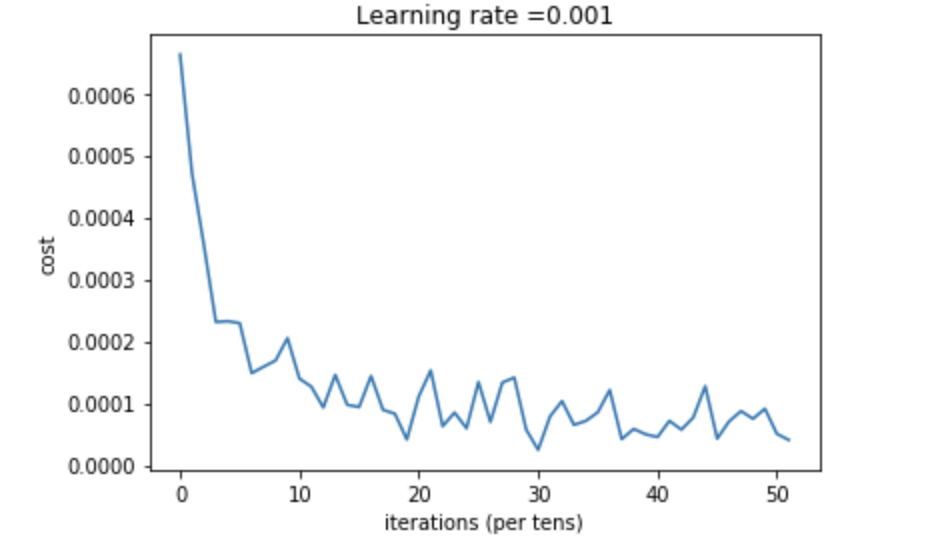
The filter or weight dimensions used for each convolution layer is as follows:



**Same padding with strides of 1** was used similar to the first architecture used.

Using this architecture helped improve our accuracy on the validation dataset. The improved values are shown by the screenshot below.





The tensor board graphs are:

