# Deep learning assignment

Introduction: in this assignment we will classify the images of cifar10, which contains 10 labels and 50000 train images and 10000 for test images.

We will build a model with 5 hidden layers and we will discuss the accuracy for this model and we will show the cost function. We will import the dataset from Keras dataset and then convert to binary vector that represents the label of the picture. Using the function to\_categorial , then we will normalize by dividing by 255( why 255? Because the values of RGB is 0-255) , the data and split to train and test data. In order to check the accuracy of our model after training. We declared in X placeholder having 3027 features , and that's because we have 32 width and high for every image multiply 3 because of the RGB for each image. We also defined a Y\_ place holder representing the predicted labels of images comparing it to y with argmax (which it’s a function that exits In numpy). This figure below describes the amount of every label exits in training data :

Classification : In this assignment we will compare two models and we will see the difference between them and how the hidden layers improve our model test accuracy and train accuracy and made a progress . The first model didn't has a hidden layers, it's just softmax activation function for our labels, and we will discuss the accuracy that was given by softmax , the activation of softmax generates a probabilitiesbetween 0-1 and each probability represents how relevant this image to this label.

Hence the softmax generates probabilities for classes (labels), the sum of the probabilities in 1. And we will see the accuracy of softmax. Same for cost function for each. We used Adam optimizer because it's made a good progress to model accuracy compared to gradient descent.

Here is a figure to our softmax code of Cifar10 classifications :

# 

# Figure 1

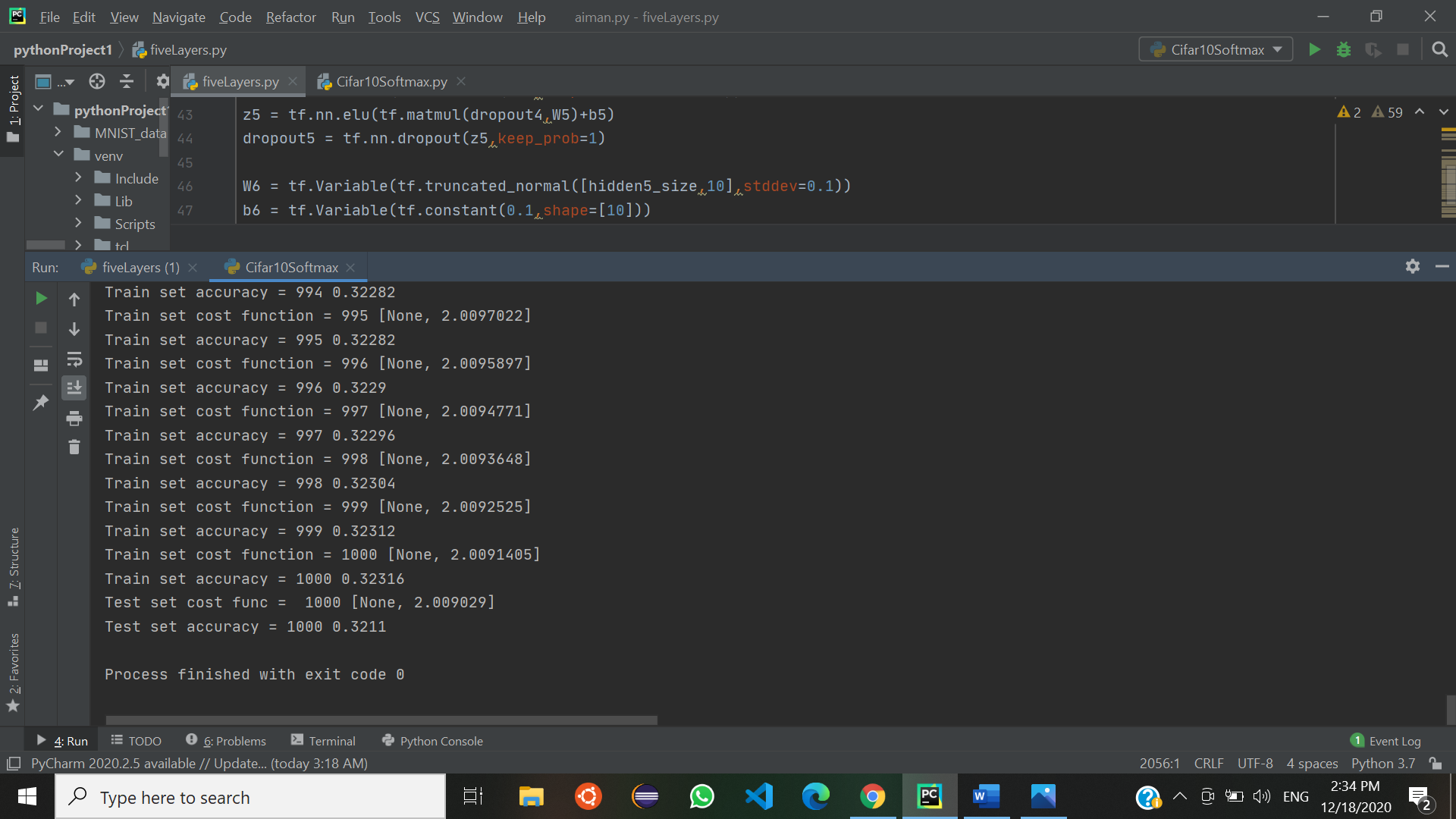
In the figure above we see that the cost function in Softmax is to high and reached 1.6821 and the accuracy of test is 0.413 which it's approximately is 41% accuracy for test. And for training we reached 0.45 which make sense why the accuracy of test is 0.4. what is the loss function? Our loss function is :

-tf.reduce\_sum(y\_ \* tf.log(y)

And the meaning is we are calculating the summation of our the log of the probability that is given by softmax and the value of y\_ which is usually is 1 or 0.

For example assume we have a 0.90 accuracy for a Deer or other label. The value of log(0.90) is to small and it's equal to -0.044.. and we will multiply this value to -1, hence this value is to small we will see that our cost function is small too , and that's what we are aiming to, because we want to find the perfect w and b for our cost function in intend to minimize it and get a good model with a good accuracy.

Using Adam optimizer made a progress in back and forward propagation searching for the weights and biases that minimize our cost function. And here is figure 2 with gradient descent for Softmax without hidden layers using gradient descent optimizer (the default nn optimizer):



# Figure 2

As we see the difference between these two figures is not just in accuracy, it's also depends on the cost function we see that the cost function with Adam is smaller so we assume that Adam optimizer helped as to get more effective weights and biases to maximize the accuracy and minimizing the cost function because he has done a good job in forward and back propagation.

Adding hidden layers to the model: How we improved our test accuracy?, The answer is we added 5 hidden layers for our model using Elu activation function in order to improve the model accuracy while training. We also used Adam Optimizer and softmax for the output layers. Here is a figure 3 to our accuracy for our model with 5 hidden layers:

# 

# Figure 3

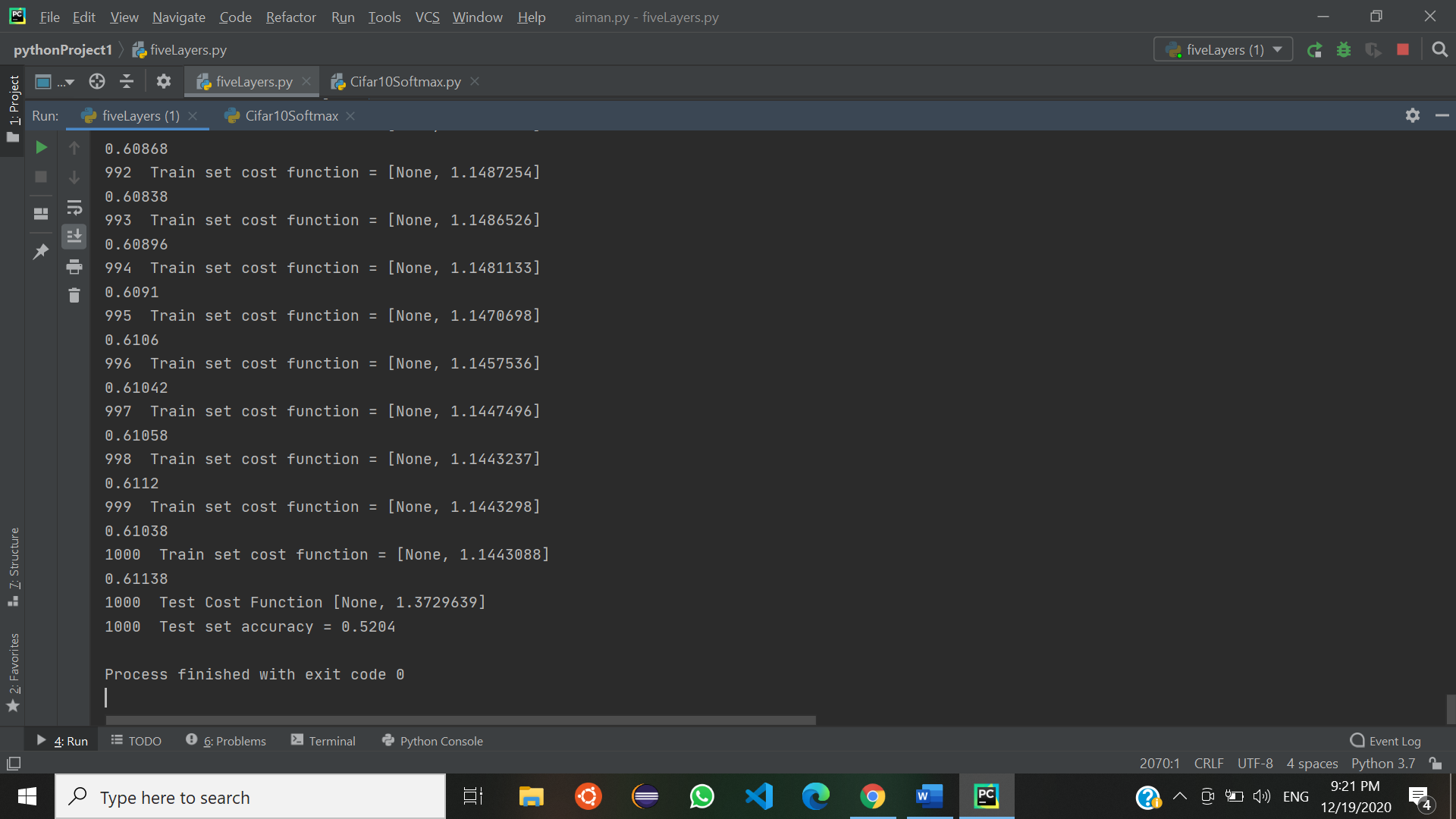
As we see here our model has made a considerable progress in the accuracy of the test adding 12% percent to our model. And we will also consider the cost function and we will compare between the first model without the hidden layers and this model which has 5 hidden layers. We will improving our model at the next time using CNN which is considered to be for images classification. And provides better performance at classification.

We will answer the question that why our model accuracy is better now, when we used to add layers for our model our model make a good progress because each layer will focus to identify the features for our labels , i.e the first layer will identify the ears and eyes , and the second the hair and the fur etc.

Now we will look at our model cost function in train set and test set, to judge if were is over fitting to our model, how we detect a overfitting in our model? By looking at the cost function in train data and test data, suppose that our cost function in train data is to small compared to our cost function in test data, this is called overfitting because of the high variance. Suppose we have high value to the both of our cost function , and that's is UF (Underfit) because of the high biases.

So generally we will observe the cost-function to determine our problem (if it's exit).

In this figure we are looking for the cost function in our model which considered to be build with 5 layers we did 1001 iterations and printed the accuracy of the model every 100 iterations:



# Figure 4

Note : in this figure we have done another try with the model , to print the cost function for test and train.

This time we got accuracy 0.52 for the test images and 0.61 approximately for the train images our cost function to test images is 1.372 and for train is 1.14 after 2 hours so close values.

As we see in the figure above there is not considerable difference between two models cost functions for train set and test set, the conclusion is clear our classification doesn't have a high variance problem which is called by the way Overfitting , but if we look at the cost functions ,clearly we can see that the cost function is not satisfying our goals , the problem is this model has a primitive methods and it's simple NN. So we will use Convolutional neural network which has a good methods too classify these images in future. Reaching our goal that we are looking too (a good classifier with an advanced methods).

The codes are attached to with this document in the zip file, you can run the model in order to check the accuracy and how these models are differ you better use Nvidia Gpu.

Summary: will we use a good neural network next time that's is too much more efficient to classify cifar10 images and this type of neural network is convolutional neural network using kernels to detect the edges as we learned and seen in the class using kernels and max-pooling methods to get a high accuracy and a lower cost function of course.