****

**Department of Electrical Engineering**

**Fall 2018**

**EE 258 -PROJECT-I REPORT**

**Team members:**

**Sidharth Sharma**

**Onkar Deshpande**

**Under the guidance of,**

**Prof. Birsen Sirkeci**

Overview of the report:

In this project we developed 3 NN models to classify the CIFAR 10 dataset. We iteratively increased the performance of the models by changing the parameters. The first model uses RGB images directly. The second model converts the images to grey scale and then uses this gray scale data to train the model. Model 1 and 2 were fully connected feed forward NN. Finally, the model-3 uses convolutional NN on the RGB images. This project documents the iterations we made on each model to increase its accuracy.

**MODEL-1.0:**

**METHODOLOGY:**

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x3 layer

**2nd layer:** 128 units of Relu

**3rd layer:** 64 units of Relu

**4th layer:** 10 units of softmax

**Cross validation:** validation split = 33% while the batch of 10,000 images (batch 5) is used as test.

**DATA:**

The maximum value of each pixel is 255 but we normalize it between 0 and 1.

The shape of **train dataset** is (50000,32,32,3), where 50,000 is number of images while each image consists of 32x32 pixels of RGB channels.

The shape of **train labels** is (50,000,) consisting of labels for train data.

The shape of **test datasets** is (10,000,32,32,3) where 10,000 is the number of images while each image consists of 32x32 pixels of RGB channels.

We do not expose this dataset to model and finally used this dataset to predict the accuracy of the model.

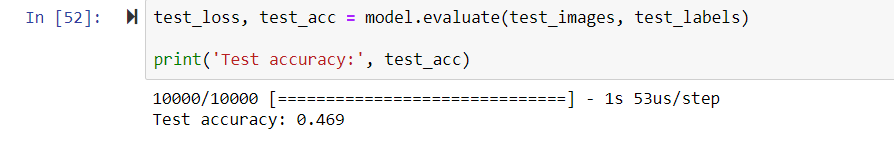
Labels: The images are classified into 10 labels as: ['airplane', 'automobile', 'bird', 'cat', 'deer',

'dog', 'frog', 'horse', 'ship', 'truck']

**SIMULATIONS:**

Using Stochastic gradient descent optimizer for 20 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

**TEST ACCURACY**= 46.9%



A close up of a map

Description generated with very high confidence

Model: 1 Accuracy on training and validation data

A close up of a map

Description generated with high confidence

Model: 1 Loss on training and validation data

**LET US INCREASE THE EPOCHS AND SEE WHAT HAPPENS**

Reasoning: I was trying to find if the same model can learn more features from the data set by just increasing the epochs. It was highly unlikely as we can see after 2.5 epochs the loss and the accuracy almost stay constant.

A close up of a map

Description generated with very high confidence

A close up of a map

Description generated with high confidence

**CONCLUSION: We can see that the training error is almost constant after 5th epoch. The test error is also decreasing very slowly. Let’s increase its complexity by adding a few layers and see what happens.**

**MODEL 1.1:**

Everything is same as model 1 except another layer has been added at the end and now the classifier is a:

**METHODOLOGY:**

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x3 layer

**2nd layer:** 128 units of Relu

**3rd layer:** 64 units of Relu

**4th layer:** 32 units of Relu

**5th layer:** 10 units of softmax

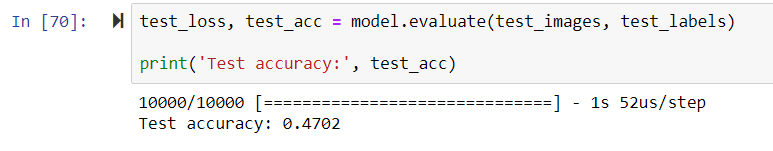
**Cross validation:** same as model 1.0

**DATA:**  Same as Model-1

**SIMULATIONS:**

Using Stochastic gradient descent optimizer for 30 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

**TEST ACCURACY**= 47.02%



**We see a little improvement. Although its negligible.**

A close up of a map

Description generated with very high confidence

Model: 1.1 Accuracy on training and validation data

A close up of a map

Description generated with very high confidence

Model: 1.1 Loss on training and validation data

Now let us experiment with the activation function for the same network:

**MODEL 1.2:**

Everything is same as model 1.1 except all the activation functions have been changed to tanh.

**METHODOLOGY:**

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x3 layer

**2nd layer:** 128 units of tanh

**3rd layer:** 64 units of tanh

**4th layer:** 32 units of tanh

**5th layer:** 10 units of softmax

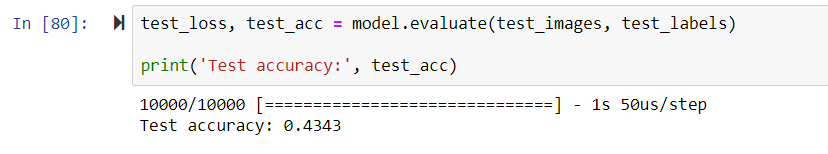
**Cross validation:** same as model 1.0

**DATA:**  Same as Model-1

**SIMULATIONS:**

Using Stochastic gradient descent optimizer for 30 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

**TEST ACCURACY**= 43.43%



**We get a worse accuracy**

A screenshot of a map

Description generated with very high confidence

Model: 1.2 Accuracy on training and validation data

A close up of a map

Description generated with high confidence

Model: 1.2 Loss on training and validation data

**Now let us try 1st layer with tanh and 3rd and 4th layer with Relu**

**MODEL 1.3:**

Everything is same as model 1.1 except all the activation functions.

**METHODOLOGY:**

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x3 layer

**2nd layer:** 128 units of tanh

**3rd layer:** 64 units of Relu

**4th layer:** 32 units of Relu

**5th layer:** 10 units of softmax

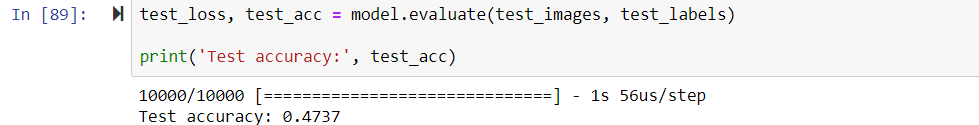
**Cross validation:** same as model 1.0

**DATA:**  Same as Model-1

**SIMULATIONS:**

Using Stochastic gradient descent optimizer for 30 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

**TEST ACCURACY**= 47.37%



**This performed slightly better than the all Relu model. The graphs are also more consistent and less oscillating.**

Reasoning: might be because Relu cuts off the negative values and our model finds the negative firing neurons a little useful in the first layers, but I can be wrong.

**A close up of a map

Description generated with very high confidence**

Model: 1.3 Accuracy on training and validation data

**A close up of a map

Description generated with very high confidence**

Model: 1.3 Loss on training and validation data

**Let us make another test of all Relu VS initial layers of tanh+Relu we increase another layer in our model. First let’s try the all Relu model.**

**MODEL 1.4:**

**METHODOLOGY:**

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x3 layer

**2nd layer:** 256 units of Relu

**3rd layer:** 128 units of Relu

**4th layer:** 64 units of Relu

**5th layer:** 32 units of Relu

**6th layer:** 10 units of softmax

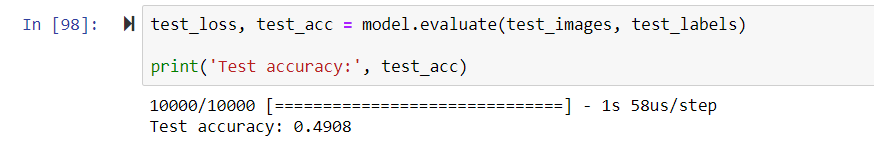
**Cross validation:** same as model 1.0

**DATA:**  Same as Model-1

**SIMULATIONS:**

Using Stochastic gradient descent optimizer for 30 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

**TEST ACCURACY**= 49.08%



A close up of a map

Description generated with very high confidence

Model: 1.4 Accuracy on training and validation data

A close up of a map

Description generated with very high confidence

Model: 1.4 Loss on training and validation data

**MODEL 1.5:**

Everything is same as model 1.4 except all the activation functions.

**METHODOLOGY:**

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x3 layer

**2nd layer:** 256 units of tanh

**3rd layer:** 128 units of tanh

**4th layer:** 64 units of Relu

**5th layer:** 32 units of Relu

**6th layer:** 10 units of softmax

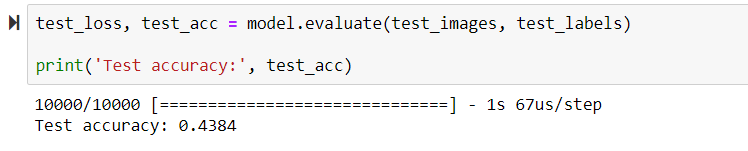
**Cross validation:** same as model 1.0

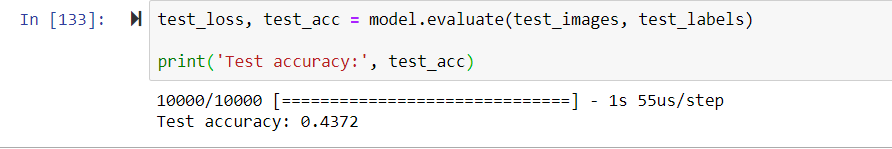
**DATA:**  Same as Model-1

**SIMULATIONS:**

Using Stochastic gradient descent optimizer for 30 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

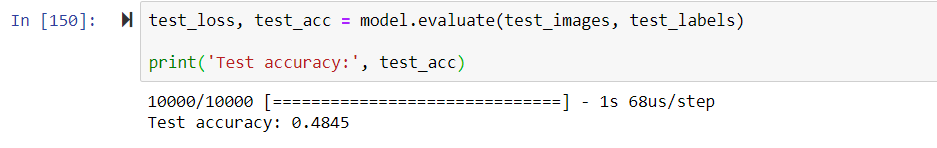
**TEST ACCURACY**= 43.84% with initial 2 tanh layer, 43.72% with just 1st tanh layer.

**Ok that wasn’t good. Let’s try it with just 1 layer of tanh first.**



**Still almost the same. So, we will stick with the all Relu network.**

**TEST ACCURACY**= 48.45% with all Relu



A close up of a map

Description generated with very high confidence

Model: 1.5 Accuracy on training and validation data with all Relu

A close up of a map

Description generated with very high confidence

Model: 1.5 Loss on training and validation data with all Relu.

**Now we will play with the optimizer. Let us change the optimizer to Adam optimizer and see if we get any improvement. The model is the same as the previous (all relu with 6-layer architecture) just the optimizer is changed.**

**MODEL 1.6:**

Everything is same as model 1.5 with all Relu activation function.

**METHODOLOGY:**

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x3 layer

**2nd layer:** 256 units of Relu

**3rd layer:** 128 units of Relu

**4th layer:** 64 units of Relu

**5th layer:** 32 units of Relu

**6th layer:** 10 units of softmax

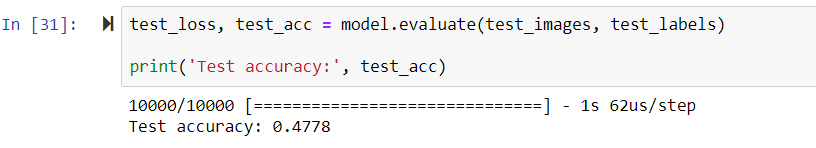
**Cross validation:** same as model 1.0

**DATA:**  Same as Model-1

**SIMULATIONS:**

Using Stochastic gradient Adam optimizer for 30 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

**TEST ACCURACY**= 47.78%



**The accuracy is almost the same just a little less than the SGD.**

A close up of a map

Description generated with very high confidence

Model: 1.6 Accuracy on training and validation data

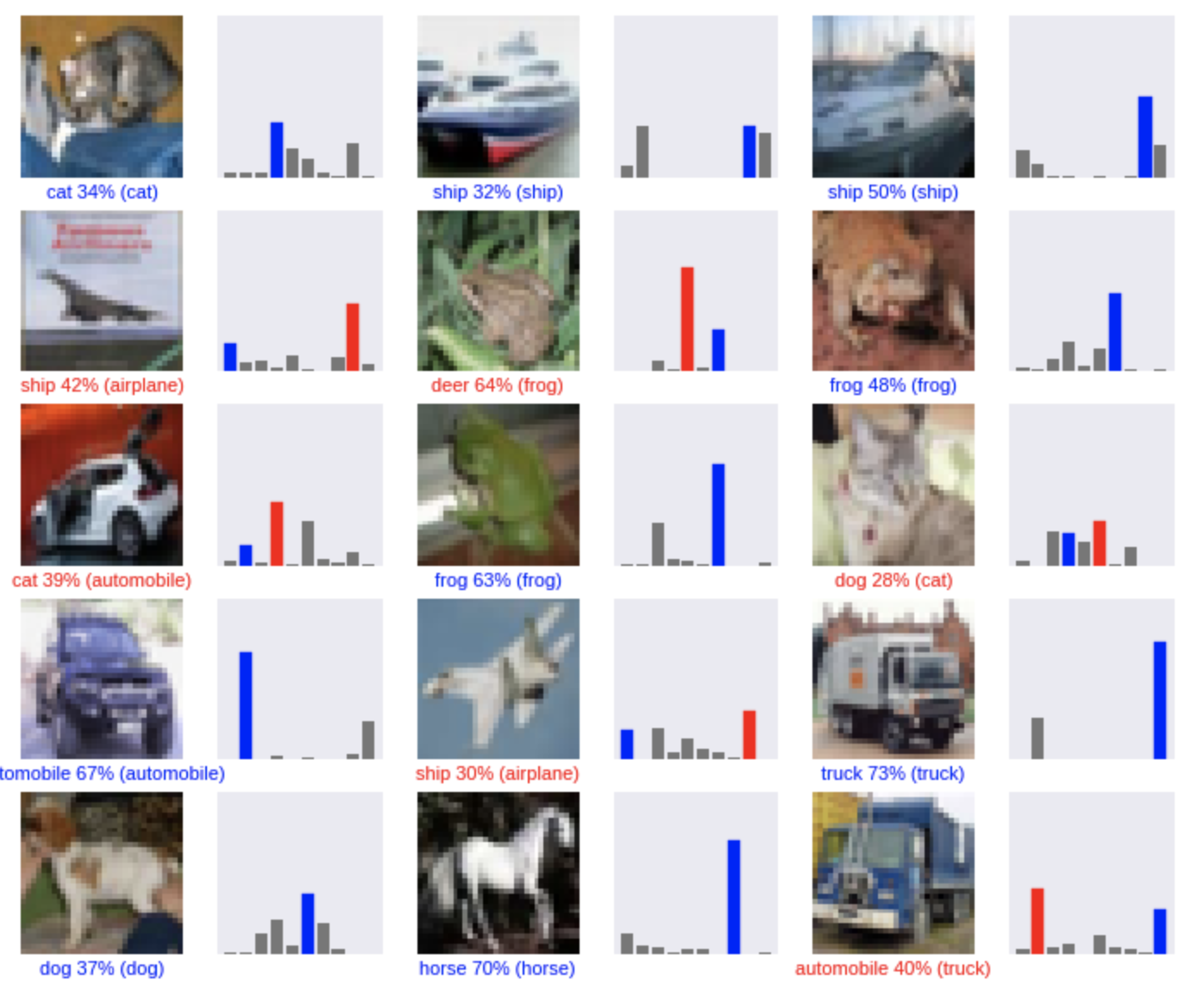
A close up of a map

Description generated with very high confidence

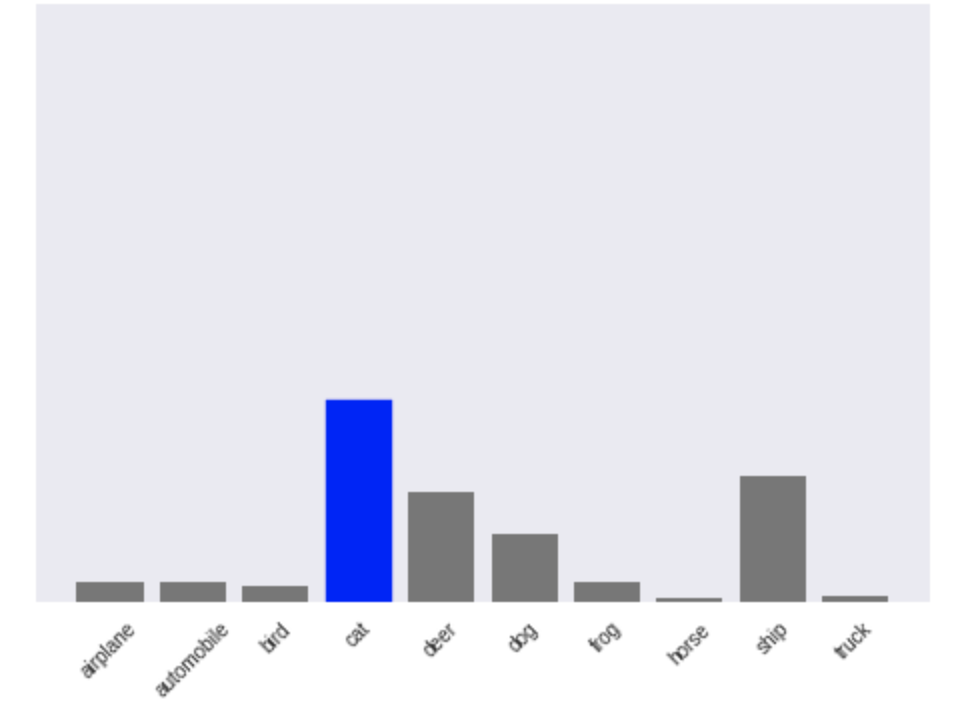
Model: 1.6 Loss on training and validation data

**From the result above I have decided to stick with RELU as the activation function and SGD as the optimizer for all future models**

**Observations:**



As seen in the figure, the red plots represent the wrong predictions and blue are the correct predictions. In our sample dataset, taken from test data, the model predicts 6 wrong predictions out of 15.



We can also view the statistics of each image. The probability of the image falling into each class given by the softmax function.

Using the mlflow library we can also save the model and experiment with the already trained model by splicing it with other trained models.

import mlflow.keras

mlflow.keras.save\_model(model, 'model1')

**MODEL 2.0:**

After analyzing the images on all RGB channels, now I decided to convert the images to greyscale and then feed it in the network. **As we can see that the model incorrectly classifies a frog as a deer and an aeroplane as a ship. This might be because of the background is simlar. These kinds of noise in the dataset can be eliminated by using grayscale images.**

To see the performance of grey scale images we keep the architecture as same as possible

**METHODOLOGY: (**same as model 1.6)

**Classifier:** Fully connected MLP

**1st layer:** Flattening the I/P to form a 32x32x1 layer (as only gray channel)

**2nd layer:** 256 units of Relu

**3rd layer:** 128 units of Relu

**4th layer:** 64 units of Relu

**5th layer:** 32 units of Relu

**6th layer:** 10 units of softmax

**Cross validation:** same as model 1.0

**DATA:**

We convert the RGB images into grayscale using Luma encoding. The shape of **train dataset** is (50000,32,32,1), where 50,000 is number of images while each image consists of 32x32 pixels of single gray channels.

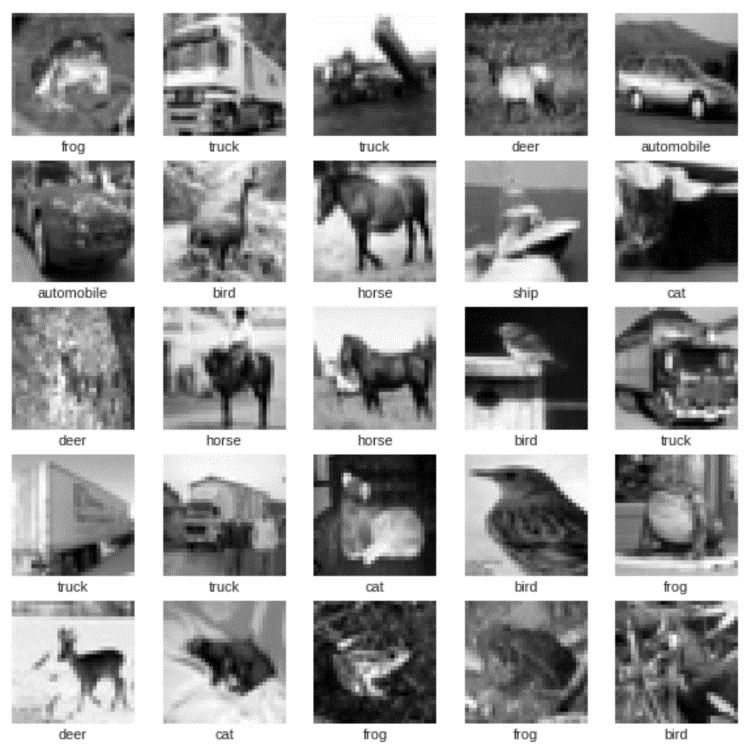
The shape of **train labels** is (50,000,) consisting of labels for train data.

The shape of **test datasets** is (10,000,32,32,1) where 10,000 is the number of images while each image consists of 32x32 pixels of single gray channels.

We do not expose this dataset to model and finally used this dataset to predict the accuracy of the model.

Labels: The images are classified into 10 labels as: ['airplane', 'automobile', 'bird', 'cat', 'deer',

'dog', 'frog', 'horse', 'ship', 'truck']

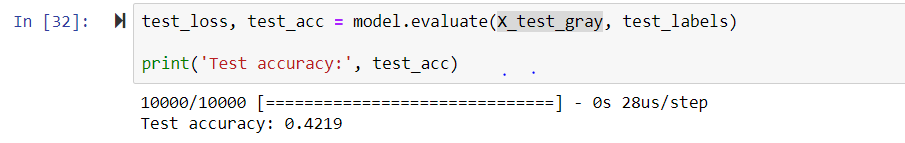


**SIMULATIONS:**

Using Stochastic gradient descent optimizer for 30 epochs, decay=1e-6, momentum=0.9, loss = sparse categorical cross entropy

**TEST ACCURACY**= 42.19%

**We see that the accuracy is lower when we used the RGB image.**



A close up of a map

Description generated with very high confidence

Model: 2.0 Accuracy on training and validation data. Gray scale dataset

A close up of a map

Description generated with very high confidence

Model: 2.0 Loss on training and validation data. Gray scale dataset

**Finally we decided to stick with rgb image with our final convnet model.**

**MODEL 3.0:**

**METHODOLOGY:**

**Classifier:** Convolutional Neural Network

**1st layer:** convolutional layer; 32 channels of 5x5 feature map; relu activation function; Stride=1. (individual matrix in the channel is of size 32x32x3)

**2nd layer:** max pooling layer consisting of 32 channels of 2x2 max pooling matrices. Stride is 2 (now each individual matrix in the channel is of size 16x16x3)

**3rd layer:** convolutional layer; 64 channels of 5x5 feature map; relu activation function; Stride=1. (individual matrix in the channel is of size 16x16x3)

**4th layer:** max pooling layer consisting of 32 channels of 2x2 max pooling matrices. Stride is 2 (now each individual matrix in the channel is of size 8x8x3)

**5th layer:** Flattening the IPs. Fully connected layer of size (8x8x3x64)

**6th layer:** Fully connected layer with 1000 units and Relu activation.

**7th layer:** 10 units of softmax

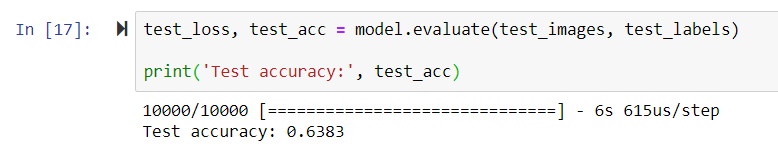
**Cross validation:** same as model 1.0

**DATA:**  same as model 1. As we realized that working on the 3 channel RGB gives better performance than using the gray scale.

**SIMULATIONS:**

Using Stochastic gradient Adam optimizer for 30 epochs, loss = sparse categorical cross entropy

**TEST ACCURACY**= 63.83%



A close up of a map

Description generated with very high confidence

Model: 3.0 Accuracy on training and validation data. CNN.

A close up of a map

Description generated with very high confidence

Model: 3.0 Loss on training and validation data. CNN.

A picture containing photo, wall

Description generated with high confidence

We can see that the simulation results of the CNN are far better than the MLP only 3/15 images classification are wrong.

**Future improvements on the present models:**1) Use data augmentation to increase the dataset

2) Try to splice separately trained models.

3) Adding drop out layers in the present CNN