

[CSCE460402 - Pract Machine Deep Learning](https://blackboard.aucegypt.edu/webapps/blackboard/execute/launcher?type=Course&id=_78911_1&url=)

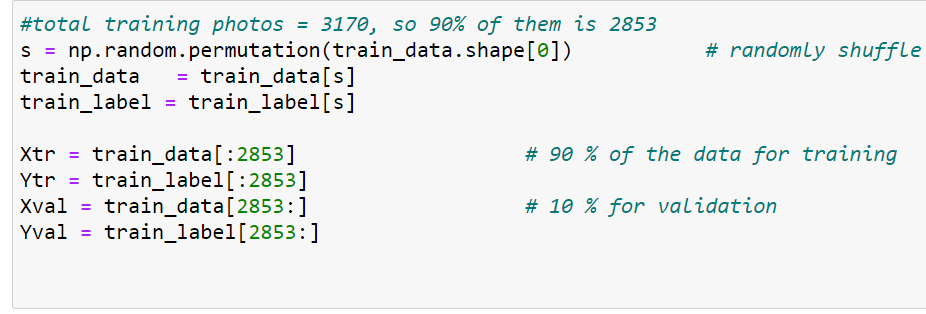
**Report on assignment #3**

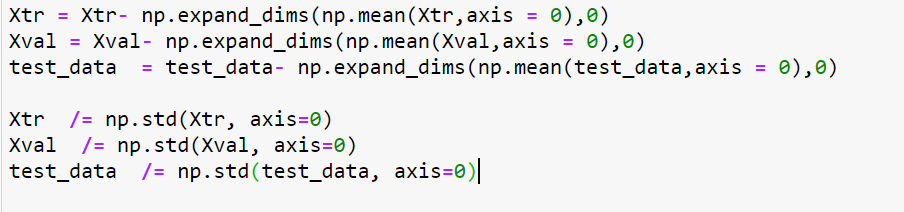
**Name:** Donia Ghazy

**ID:** 900172124

**1. Data Pre-processing:**

* The data is divided into training set (90%) and validation set (10%).
* The image’s mean is taken in order to zero center the outputs.
* The image’s standard variation is taken in order to normalize the values.
* After instantiating the conv module, data augmentation has been performed to increase the variety of the dataset. Through rotating both horizontally and vertically and flipping it into vertical axis.





**2. Finding a suitable architecture and fine-tuning the hyper-parameters:**

* **For the architecture**: I decided to start my design with a simple arch which was: Conv layer-> active fun -> Conv Layer -> active func-> max pooling-> flattened->FC. The design was so fast in execution, however, it gave a very low accuracy compared even to my previous implementation for NN. That’s why I have decided to add more layers so, the final architecture implemented is:

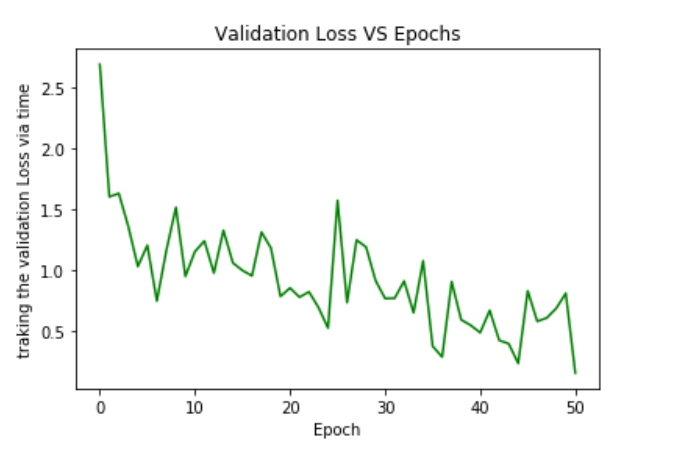
Conv layer-> active fun -> Conv Layer -> active func-> max pooling-> Conv layer-> active fun -> Conv Layer -> active func-> max pooling-> flattened->FC.

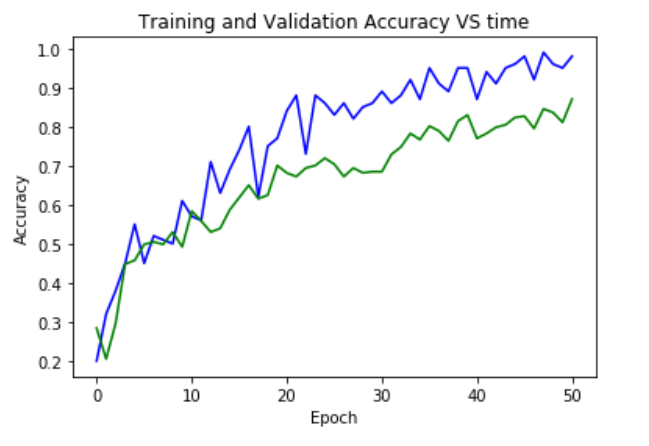
* **For the FC**, I have tried to use only two layers. Although it kept the architecture very simple but what I figured out is the non-linearity is not enough for overcoming the overfitting that’s why I added a third one. I did not even try to add more layers because I calculated the total memory space and it sucked.
* **For the filter sizes and filter numbers**: I tried different values for the number of filters (32,64,128) and the filter size (3x3, 5x5) by training on a small portion of the dataset (only 50 photo) and I settled on the values which yielded the best testing accuracy
* **For the Weights and Biases:** Gradient descent was used to minimize the loss as much as possible by trying to update the bias and weights by the end of each epoch.
* **For the learning Rate:** I tried to get the minimum and maximum range of values by try and error on small portion of the data and I found that 1e-4 yields the best results for my design.

**3. When to stop the training**:

I have tried at first small number of epochs but they only result in low accuracies, I tried to go higher by iterating over 50 epochs. Starting, roughly, from epoch 47, I should have stopped the training since the loss was saturated and the model starts to over-fit as the training accuracy reaches 95+% but I ran it extra three epochs in order to make sure it is actually overfitting.

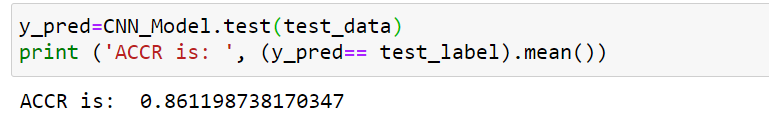




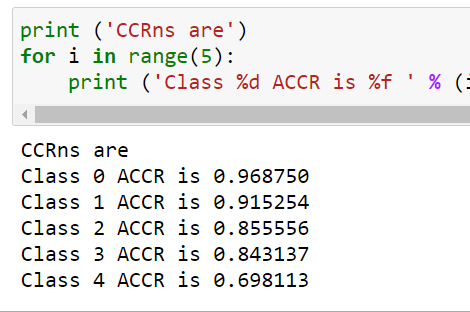


**4. CCR and ACCR:**

ACCR for this model is 86.11% which is higher than the NN (53.4%) and KNN (43.85%)



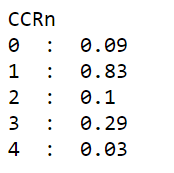
CCR for each class in this model is:



CCR for the NN was:



CCR for the KNN was:



My CNN works exceptionally in the first two classes but its performance starts to decrease gradually after that.