Software project 1 (Part 3)

Buciu Ștefan

Learning hypothesis

We need a model that learns to recognize the digit from a given image. Let’s denote an image as I, then a function that would map the given image to the digit it represents would look as follows:

Our learner needs to find an hypothesis h such that h(i) = f(i) for the given images from I.

# CNN Solution

ML Architecture

### Layers

### Conv2D

### The Conv2D layer creates a convolution kernel on the given input.

* MaxPooling2D

This layer is usually used to down-sample the input. It uses a sliding window approach and takes the maximum value from the given area within the full input.

* Flatten

This layer is used to flatten the 2D input array to an 1D array. In our case, it is going to transform a 28x28 image to an array of 784 pixels.

* Dense

This layer is a fully connected. It maps each input with the given number of neurons.

Optimizer

Adam (Adaptive Moment Estimation)

Where classical SGDs maintain a single learning rate for all weight updates and the learning rate does not change during training, ADAM maintains a learning rate for each network weight (parameter) and that gets adapted separately as learning unfolds.

Loss Function

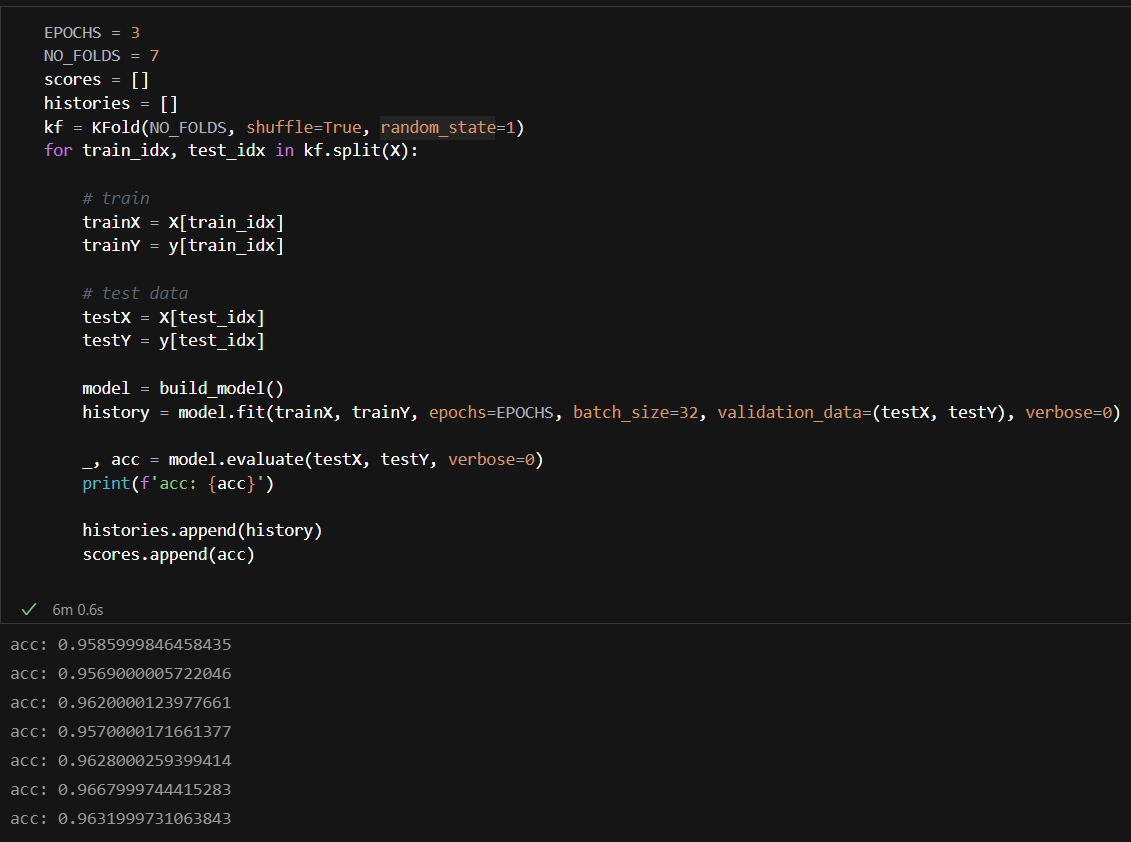
Sparse Categorical Cross-Entropy

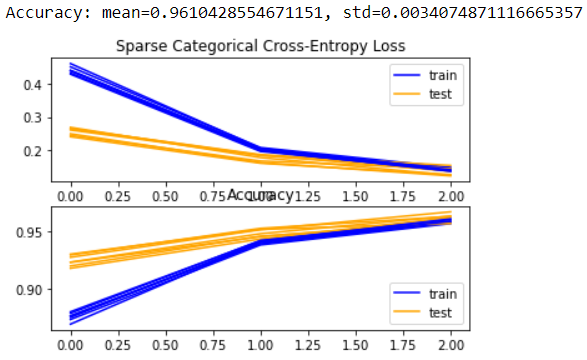
There is no need for one hot encoding. The output layer must be configured with n nodes (one for each class). Softmax activation can be used in the output layer.

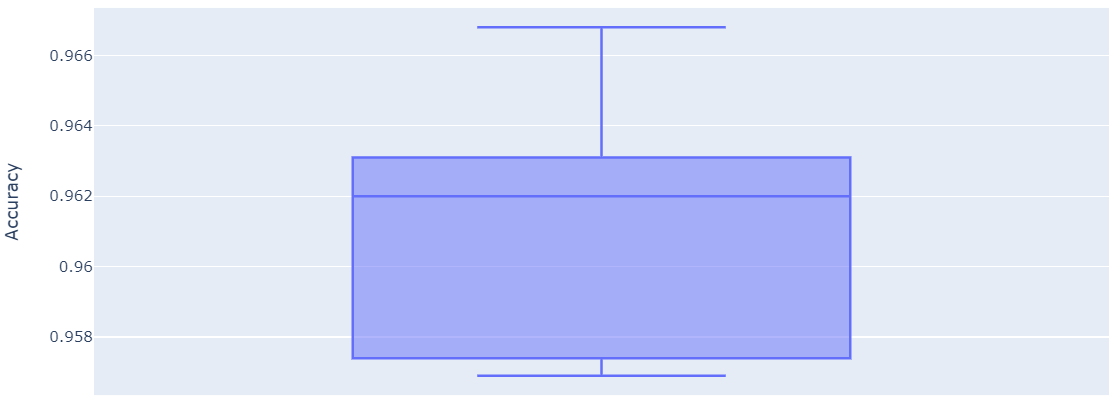


Results

Used K-Fold Cross Validation where K is 7. These are the accuracy scores for each iteration:







# SGD Solution

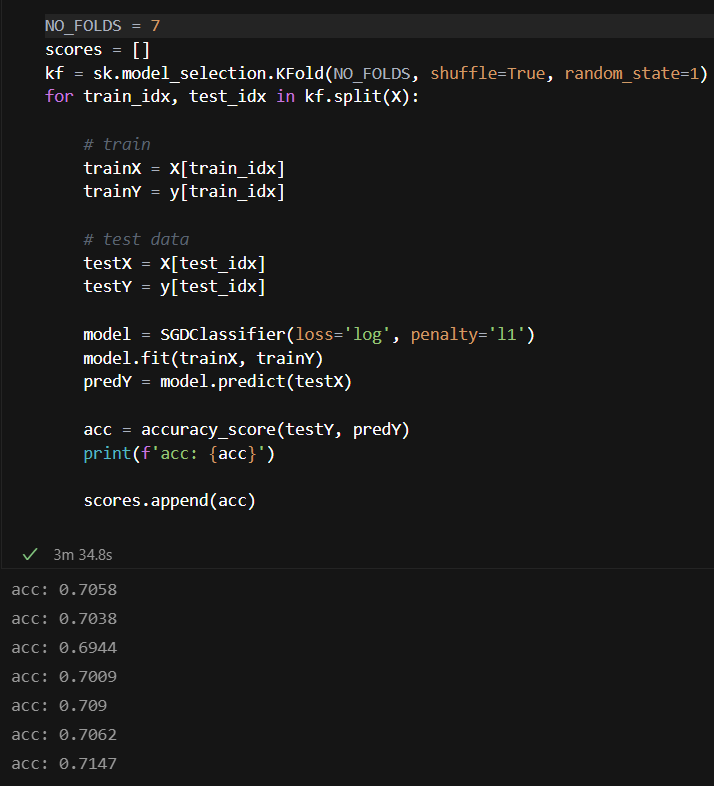
ML Architecture

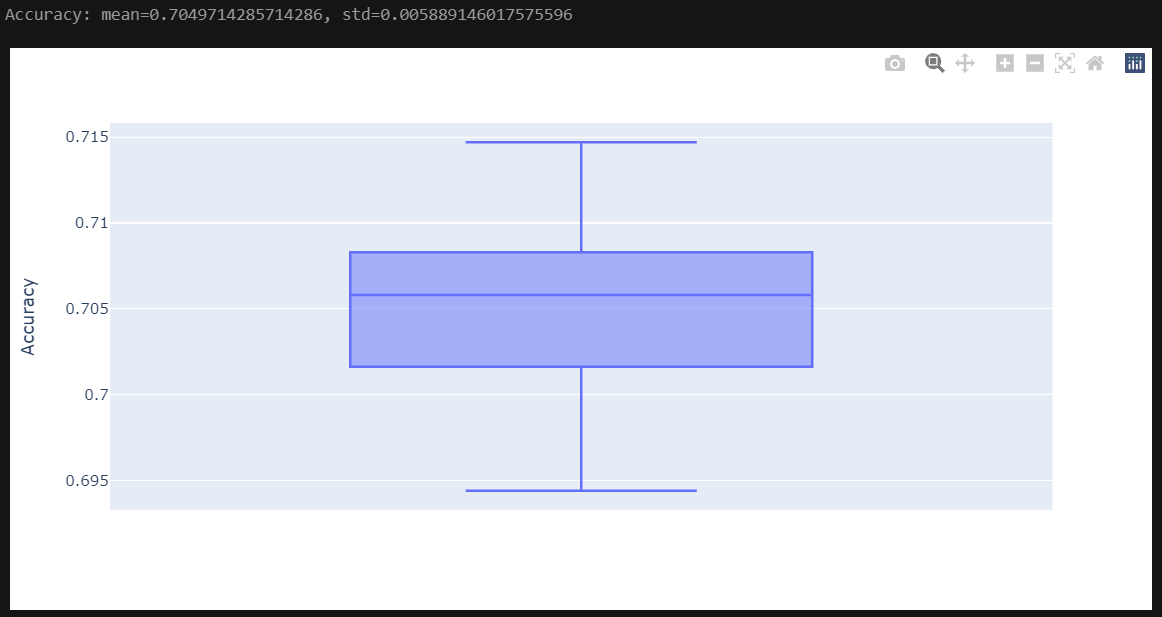
Used SGD Classifier with the following settings:

Loss function: log

Penalty: l1

Results





Comparative analysis of the results

We can see that the CNN model performs way better than the classical SGD Classifier. The data was prepared in the same style. Probably with some tweaks to the parameters of the SGD Classifier we can improve the results, but probably not around 20%.