Assignment 1:MLP for Image Classification: Connectionist AI

Traditional MLP:

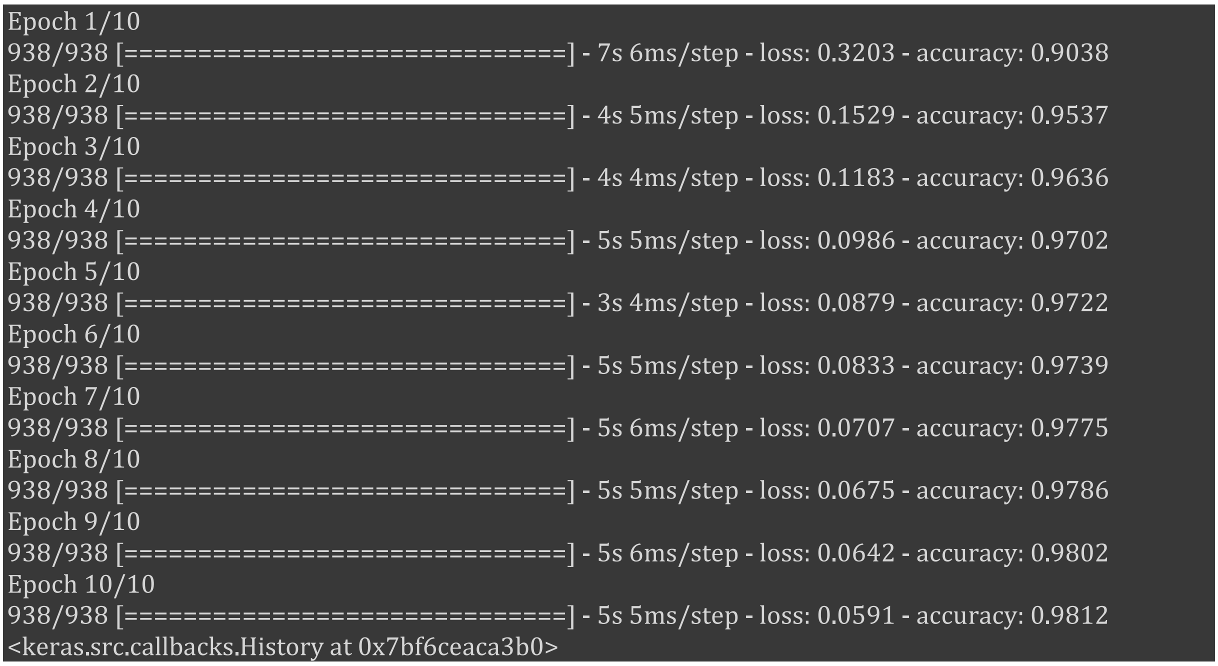
Dataset out put is one-hot encoded since having a tensor as an output would be easier since we are also giving a tensor as an input.

It’s just a simple three-layer Perceptron which uses categorical crossentropy as a loss function and Relu as an activation function for the hidden layers as a general trend. And the output layer uses SoftMax since it resolves the problem of vanishing gradients in a multi-class classification problem as common knowledge also it outputs a probability as an output and in a normalized way too.

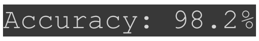
Model summary:

```
Model: "sequential_190"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_358 (Dense)           (None, 256)               200960    
                                                                 
 activation_3 (Activation)   (None, 256)               0         
                                                                 
 dropout_175 (Dropout)       (None, 256)               0         
                                                                 
 dense_359 (Dense)           (None, 256)               65792     
                                                                 
 activation_4 (Activation)   (None, 256)               0         
                                                                 
 dropout_176 (Dropout)       (None, 256)               0         
                                                                 
 dense_360 (Dense)           (None, 10)                2570      
                                                                 
 activation_5 (Activation)   (None, 10)                0         
                                                                 
=================================================================
Total params: 269322 (1.03 MB)
Trainable params: 269322 (1.03 MB)
Non-trainable params: 0 (0.00 Byte)
```

Training:



```
Epoch 1/10
938/938 [==============================] - 7s 6ms/step - loss: 0.3203 - accuracy: 0.9038
Epoch 2/10
938/938 [==============================] - 4s 5ms/step - loss: 0.1529 - accuracy: 0.9537
Epoch 3/10
938/938 [==============================] - 4s 4ms/step - loss: 0.1183 - accuracy: 0.9636
Epoch 4/10
938/938 [==============================] - 5s 5ms/step - loss: 0.0986 - accuracy: 0.9702
Epoch 5/10
938/938 [==============================] - 3s 4ms/step - loss: 0.0879 - accuracy: 0.9722
Epoch 6/10
938/938 [==============================] - 5s 5ms/step - loss: 0.0833 - accuracy: 0.9739
Epoch 7/10
938/938 [==============================] - 5s 6ms/step - loss: 0.0707 - accuracy: 0.9775
Epoch 8/10
938/938 [==============================] - 5s 5ms/step - loss: 0.0675 - accuracy: 0.9786
Epoch 9/10
938/938 [==============================] - 5s 6ms/step - loss: 0.0642 - accuracy: 0.9802
Epoch 10/10
938/938 [==============================] - 5s 5ms/step - loss: 0.0591 - accuracy: 0.9812
<keras.src.callbacks.History at 0x7bf6ceaca3b0>
```



```
Accuracy: 98.2%
```

MLP Mixer:

The MLP mixer uses patches to improve the model’s ability to recognize local spatial patterns by dividing the input into several patches and analysing them individually.

Before the data enters the MLP layers we reshape them to tuples like ‘(batch_size, num_patches, patch_dim)’ to be taken by the next layer.

The MLP layers consist of two Sequential layers with a ‘Gelu’ activation function and then apply layer normalization for more stable learning.

Moreover, we also have a residual connections to promote a healthy flow of information between the layers and to capture hierarchical features.

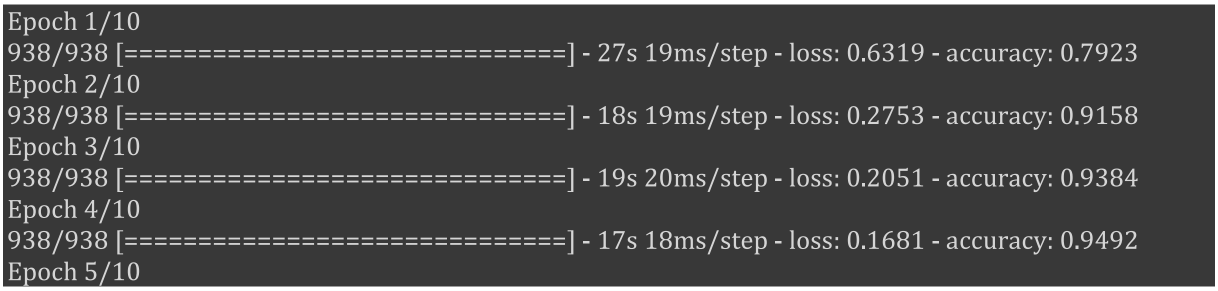
Towards the end we use global average pooling to reduce spatial dimensionality and make the model more robust to translations of images.

Similar to Traditional MLP the output layer is ‘Softmax’ activated and sparse_categorical_crossentropy as a loss function in model training.

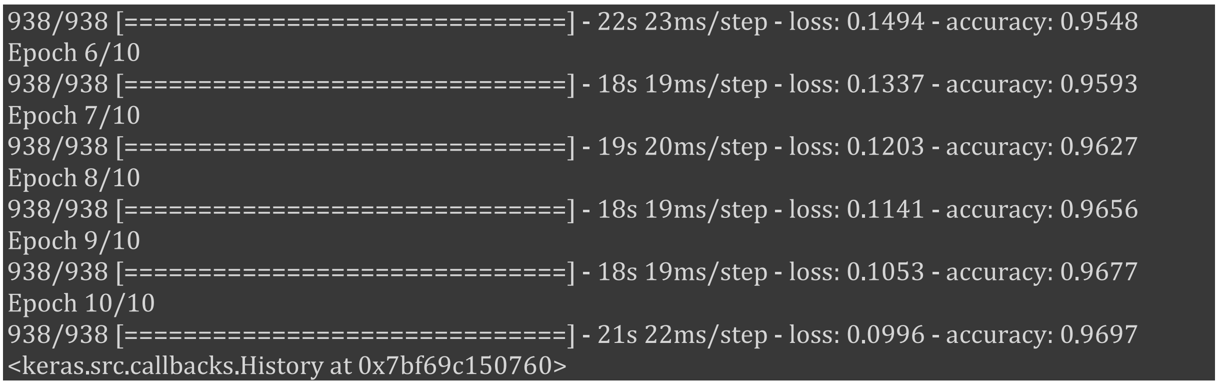
Model summary:

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_17 (InputLayer)       [(None, 28, 28, 1)]       0         
                                                                 
 patches_16 (Patches)        (None, 49, 16)            0         
                                                                 
 dense_320 (Dense)           (None, 49, 64)            1088      
                                                                 
 sequential_171 (Sequential  (None, 49, 64)            45652     
 )                                                               
                                                                 
 global_average_pooling1d_2  (None, 64)                0         
  (GlobalAveragePooling1D)                                       
                                                                 
 dropout_156 (Dropout)       (None, 64)                0         
                                                                 
 dense_321 (Dense)           (None, 10)                650       
                                                                 
=================================================================
Total params: 47390 (185.12 KB)
Trainable params: 47390 (185.12 KB)
Non-trainable params: 0 (0.00 Byte)
```

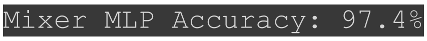
Training:



```
Epoch 1/10
938/938 [==============================] - 27s 19ms/step - loss: 0.6319 - accuracy: 0.7923
Epoch 2/10
938/938 [==============================] - 18s 19ms/step - loss: 0.2753 - accuracy: 0.9158
Epoch 3/10
938/938 [==============================] - 19s 20ms/step - loss: 0.2051 - accuracy: 0.9384
Epoch 4/10
938/938 [==============================] - 17s 18ms/step - loss: 0.1681 - accuracy: 0.9492
Epoch 5/10
```



```
938/938 [==============================] - 22s 23ms/step - loss: 0.1494 - accuracy: 0.9548
Epoch 6/10
938/938 [==============================] - 18s 19ms/step - loss: 0.1337 - accuracy: 0.9593
Epoch 7/10
938/938 [==============================] - 19s 20ms/step - loss: 0.1203 - accuracy: 0.9627
Epoch 8/10
938/938 [==============================] - 18s 19ms/step - loss: 0.1141 - accuracy: 0.9656
Epoch 9/10
938/938 [==============================] - 18s 19ms/step - loss: 0.1053 - accuracy: 0.9677
Epoch 10/10
938/938 [==============================] - 21s 22ms/step - loss: 0.0996 - accuracy: 0.9697
<keras.src.callbacks.History at 0x7bf69c150760>
```



```
Mixer MLP Accuracy: 97.4%
```

Overall: The Traditional model achieves test accuracy of 98.2% in 10 epochs while the Mixer-model attains test accuracy of 97.4% in 10 epochs. Making Traditional model more suitable for the MNIST dataset.

References

- https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/
- https://www.v7labs.com/blog/neural-networks-activation-functions
- https://www.baeldung.com/cs/learning-rate-batch-size#:~:text=Batch%20size%20defines%20the%20number,training%20set%20in%20one%20epoch.
- https://arxiv.org/abs/2105.03404v2
- https://sh-tsang.medium.com/review-resmlp-feedforward-networks-for-image-classification-with-data-efficient-training-4eeb1eb5efa6
- https://stats.stackexchange.com/questions/162988/why-sigmoid-function-instead-of-anything-else/318209#318209