Selected Topics From CS: Assignment 3

Anirudh Srinivasan (2015A7PS0382H)
Bharat Raghunathan (2015AAPS0263H)
Nitish Ravishankar (2015A7PS0152H)

April 22, 2018

Contents

1 Task 1: ANN (Artificial Neural Network)

The aim of this task was to develop an Artificial Neural Network to classify handwritten digits given the images in 28×28 matrix of pixels. The formula used in a 3-layer Artificial Neural Network are as follows: For the first layer

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{1}$$

and each of these a_j is then transformed by a differentiable non-linear *activation* function

$$z_j = h(a_j) \tag{2}$$

Following (2), these values are again linearly combined to give the equation for the second layer

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{3}$$

where $w_{k0}^{(2)}$ and $w_{j0}^{(1)}$ are the bias parameters Finally, the outputs are given by

$$y_k = \sigma(a_k) \tag{4}$$

where

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{5}$$

is the sigmoid function.

1.1 Main subtasks presented by the problem

1. **Preprocessing:-** The pixels either have to be binarized (i.e. $\{0, 1\}$ values) or they have to be between the range [0, 1] since the Neural Network weights are randomly initialized in the range [0, 1]

2. **Hyperparameters:-** We had chosen the *Adam* optimizer. However, since an accuracy of well over 95% was obtained with just 30 epochs, the number of epochs was limited to 30.

2 Task 2: CNN (Convolutional Neural Network)

The aim of this task was to develop a Convolutional Neural Network to classify handwritten digits given the images in 28×28 matrices of pixels. The equations followed are pretty much the same as above, with the only differences being:-

i) **Local Receptive Fields:** The feature invariances are captured locally in subregions of the image, rather than globally in the whole image.

ii) **Weight Sharing:** All the weights mapping from one set of $n \times n$ pixels to one region in the feature map, share the SAME weights.The value of n is specified in the *kernel_size* parameter

iii) **Subsampling:** The pixels are subsampled typically using a *pooling* layer which simply takes the maximum/minimum/average of every $k \times k$ window of pixels. The pixel window k is specified as a parameter.

2.1 Main subtasks presented by the problem

1. **Preprocessing:-** The pixels either have to be binarized (i.e. $\{0, 1\}$ values) or they have to be between the range [0, 1] since the Neural Network weights are randomly initialized in the range [0, 1]

2. **Hyperparameters:-** We had chosen the *Adam* optimizer, a 2×2 window for MaxPooling and a 5×5 kernel for Convolutional Layer. However, since an accuracy of well over 95% was obtained with just 50 epochs, the number of epochs was limited to 50.

3 Results

The results for ANNs and CNNs and the model summary are displayed for 1, 2 and 3 layers, respectively

```
Epoch 30/30
48000/48000 [==============================] - 2s 34us/step - loss: 0.1300 - acc: 0.9631 - val_loss: 0.1765 - val_acc: 0.9488
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 20)                15700
_________________________________________________________________
dense_2 (Dense)              (None, 10)                210
=================================================================
Total params: 15,910
Trainable params: 15,910
Non-trainable params: 0
_________________________________________________________________
None
10000/10000 [==============================] - 0s 46us/step
Test Loss:  0.17368382148742675
Test Accuracy:  0.9512
```

(a) 1 Layer ANN

```
Epoch 30/30
48000/48000 [==============================] - 2s 37us/step - loss: 0.0497 - acc: 0.9871 - val_loss: 0.1246 - val_acc: 0.9635
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 40)                31400
_________________________________________________________________
dense_2 (Dense)              (None, 20)                820
_________________________________________________________________
dense_3 (Dense)              (None, 10)                210
=================================================================
Total params: 32,430
Trainable params: 32,430
Non-trainable params: 0
_________________________________________________________________
None
10000/10000 [==============================] - 1s 51us/step
Test Loss:  0.11362889020163566
Test Accuracy:  0.9669
```

(b) 2 Layers ANN

```
Epoch 30/30
48000/48000 [==============================] - 2s 42us/step - loss: 0.0332 - acc: 0.9932 - val_loss: 0.1520 - val_acc: 0.9621
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 48)                37680
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1568
_________________________________________________________________
dense_3 (Dense)              (None, 16)                528
_________________________________________________________________
dense_4 (Dense)              (None, 10)                170
=================================================================
Total params: 39,946
Trainable params: 39,946
Non-trainable params: 0
_________________________________________________________________
None
10000/10000 [==============================] - 0s 49us/step
Test Loss:  0.14058110155854375
Test Accuracy:  0.9649
```

(c) 3 Layers ANN

Figure 1: Accuracy with Number of Layers

```
Epoch 50/50
48000/48000 [==============================] - 3s 55us/step - loss: 0.0123 - acc: 0.9956 - val_loss: 0.0655 - val_acc: 0.9852
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 64)        1664
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                92170
=================================================================
Total params: 93,834
Trainable params: 93,834
Non-trainable params: 0
_________________________________________________________________
None
10000/10000 [==============================] - 1s 64us/step
Test Loss:  0.06203978628458117
Test Accuracy:  0.9861
```

(a) 1 Layer CNN

```
Epoch 50/50
48000/48000 [==============================] - 3s 69us/step - loss: 5.2636e-04 - acc: 1.0000 - val_loss: 0.0364 - val_acc: 0.9907
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 64)        1664
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          102464
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 114,378
Trainable params: 114,378
Non-trainable params: 0
_________________________________________________________________
None
10000/10000 [==============================] - 1s 72us/step
Test Loss:  0.031815983698792294
Test Accuracy:  0.9908
```

(b) 2 Layers CNN

```
Epoch 50/50
48000/48000 [==============================] - 4s 76us/step - loss: 3.7431e-04 - acc: 1.0000 - val_loss: 0.0387 - val_acc: 0.9910
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 64)        1664
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 64)          102464
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 64)          36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 141,706
Trainable params: 141,706
Non-trainable params: 0
_________________________________________________________________
None
10000/10000 [==============================] - 1s 76us/step
Test Loss:  0.03067033268496707
Test Accuracy:  0.9909
```

(c) 3 Layers CNN

Figure 2: Accuracy with Number of Layers

3.1 Understanding of Results

1. Accuracy of the Artificial Neural Network is above 95%, so most of the invariance has already been captured.
2. More number of hidden layers take a lesser no of epochs to converge and also give higher Accuracy, but after a point, they may start *overfitting*

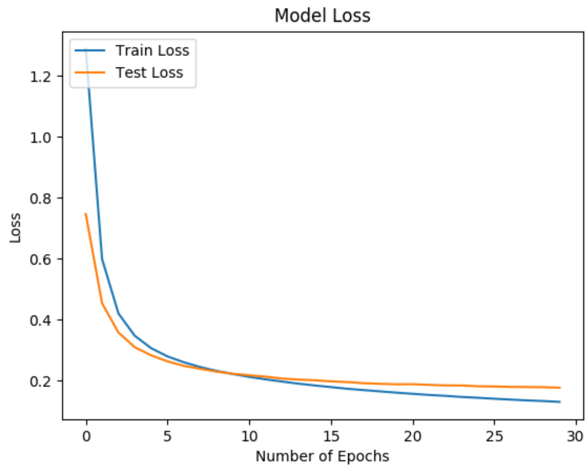
Number of Layers of ANN	Accuracy
1	95.12%
2	96.69%
3	96.49%

Table 1: Accuracy for different number of ANN Layers

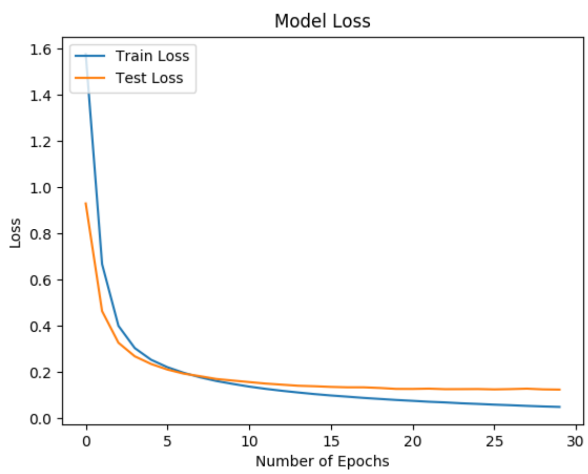
Number of Layers of CNN	Accuracy
1	98.61%
2	99.08%
3	99.09%

Table 2: Accuracy for different number of CNN Layers

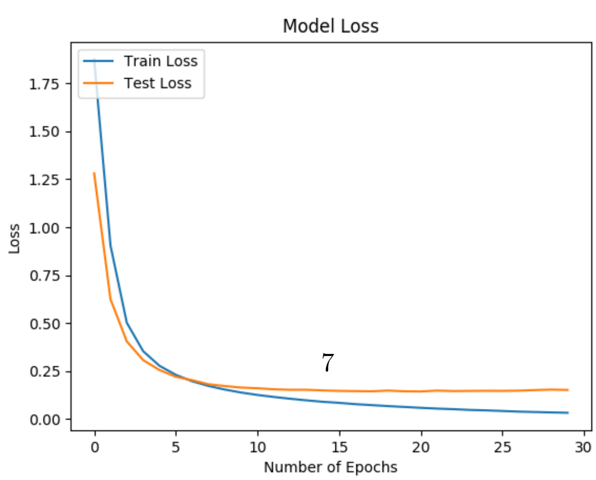
The training and test losses for ANN and CNN for 1, 2 and 3 Layers respectively are also plotted



(a) 1 Layer ANN

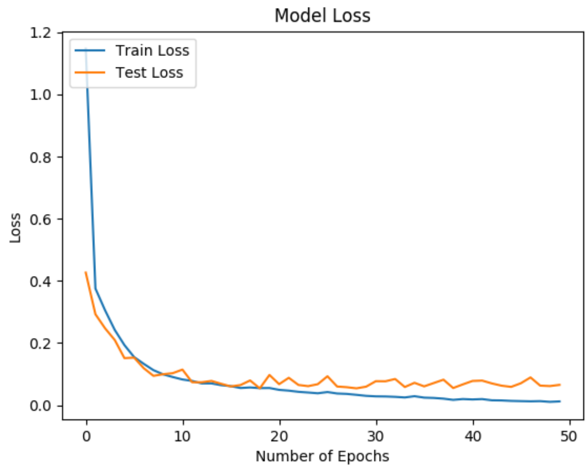


(b) 2 Layers ANN

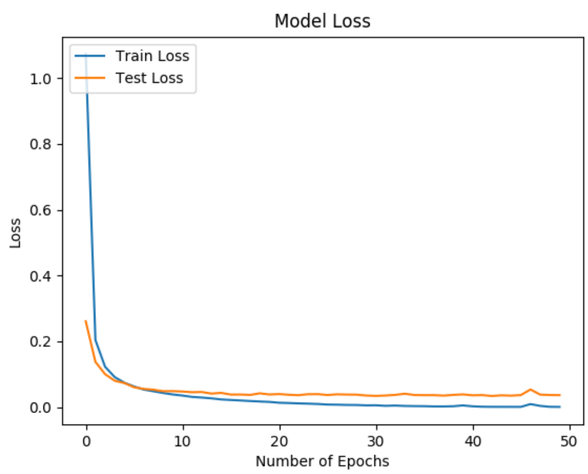


(c) 3 Layers ANN

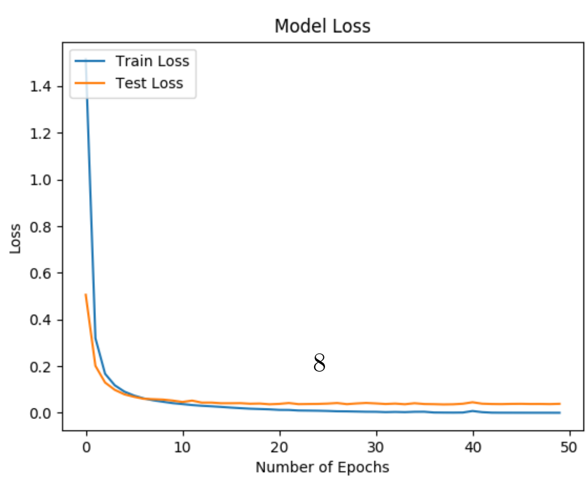
Figure 3: Train and Test Losses with Number of ANN Layers



(a) 1 Layers CNN



(b) 2 Layers CNN



(c) 3 Layers CNN

Figure 4: Train and Test Losses with Number of CNN Layers

A Notation used for Equations

$$w_{jk}^{(l)} = \text{Weight from } k^{th} \text{ neuron in } (l-1)^{th} \text{ layer to } j^{th} \text{ neuron in } l^{th} \text{ layer}$$

$$a_j = \text{Value of } j^{th} \text{ neuron before activation}$$

$$z_j = \text{Value of } j^{th} \text{ neuron after activation}$$

$$exp(a) = \text{Exponential Function}$$