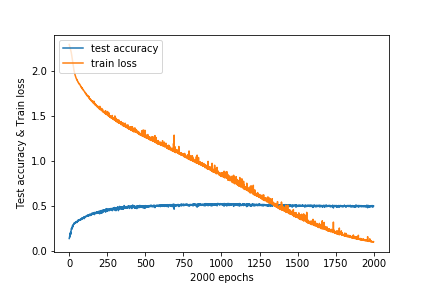
* 1. **Part A**
  2. **1. Train the network by using mini-batch gradient descent learning. Set batch size =128, and learning rate 𝛼=0.001. Images should be scaled.**
  3. **a. Plot the training cost and the test accuracy against learning epochs.**



*Figure 1a*

Test accuracy reached the maximum of around 0.5 at epoch 500. Afterwards, the loss is decreasing but accuracy stays the same, suggesting overfitting. Subsequent training will be done with 750 epochs.

* 1. **b. For any two test patterns, plot the feature maps at both convolution layers (𝐶1 and 𝐶2) and pooling layers (𝑆1 and 𝑆2) along with the test patterns.**

|  |  |  |
| --- | --- | --- |
|  | Test Pattern 1 | Test Pattern 2 |
| Raw |  |  |
| C1 |  |  |
| S1 |  |  |
| C2 |  |  |
| S2 |  |  |

**2. Using a grid search, find the optimal numbers of feature maps for part (1) at the convolution layers. Use the test accuracy to determine the optimal number of feature maps.**

Searching these grids:

C1 = {40,50,60}

C2 = {50,60,70}

|  |  |  |  |
| --- | --- | --- | --- |
| Test Accuracy | C1:40 | C1:50 | C1:60 |
| C2:50 | 0.517 | 0.5165 | 0.5185 |
| C2:60 | 0.515 | 0.516 | 0.516 |
| C2:70 | 0.5155 | 0.5075 | 0.5265 |

From the table above, optimal number of feature maps {C1, C2} is {60, 70}.

**3. Using the optimal number of filters found in part (2), train the network by:**

**a. Adding the momentum term with momentum 𝛾=0.1.**

**b. Using RMSProp algorithm for learning**

**c. Using Adam optimizer for learning**

**d. Adding dropout to the layers**

**Plot the training costs and test accuracies against epochs for each case.**

|  |  |
| --- | --- |
| Momentum | RMSProp |
| Adam Optimizer | Dropouts |

**4. Compare the accuracies of all the models from parts (1) - (3) and discuss their performances.**

GradientDescent – GradientDescent has the slowest training rate, reach 0.5 accuracy after 500 epochs. There are also many small spikes during training.

Momentum – In this case, Momentum has a similar performance to Gradient Descent. This might be because the gradient during training is too gentle for Momentum to pick up any speed. Alternatively, the constant spikes has cancelled out any built up momentum.

RMSProp – RMSProp has the fastest training rate, finishing training before 100 epochs. When slowing down, there are huge spikes while its stabilising. However, there is a slight loss in accuracy compared to GradientDescent.

Adam – Adam has the second fastest training rate, finishing training at around 120 epochs. When slowing down, it has similar spikes to RMSProp. After flatlining for a while, there is a sudden spike in loss at 400 epochs. Afterwards, training loss decreases and seems to converge at 0.7. There is an apparent loss of 0.1 accuracy as well. It could be that epoch 400 has a particularly unlucky mini-batch, causing the loss to spike.

Dropouts- After introducing Dropouts to Gradient Descent, there are less spikes during training but training takes a longer time.

For all the models, the test accuracy does not increase past 0.5 despite the decrease in loss. If we compare models where loss = 0, accuracy stays around 0.4-0.5. This suggests that the model has a max performance of 50% accuracy when used outside of training. To fix this, we might have to introduce more robust datasets or change the model.

In this case, RMSProp has the best performance out of all the models. Despite not showing better accuracy, it finished its training in 100 epochs, which is 5 times faster than some models.