ELL793 Computer Vision
NN on MNIST & CNN on CIFAR-10

Abhinava Sikdar
2017MT10724

Yashank Singh
2017MT10756

November 24, 2020

1 Introduction

Neural Networks and related Deep Learning techniques are state of the art methods used today for a wide variety of applications in fields such as Computer Vision, NLP, Finance, Diagnostics, etc. In this assignment, consisting of two parts, we use their power to classify the MNIST handwritten digit dataset and the CIFAR-10 dataset.

2 NN on MNIST and Regularization Methods

The MNIST handrwitten digit dataset consist of 28×28 grayscale images. It is made up of about 60,000 images for the purpose of training and about 10,000 for the purpose of validation. As stated in the problem statement, we use a simple feedforward neural network of 2 hidden layers with 500 neurons each. All the models, regularised or otherwise are trained for 250 epochs. The training dataset was normalized. Here are some feature about our models:

- Softmax layer applied after the second hidden layer & activation for yielding a valid probability distribution over all the ten classes.
- Rectified Linear Unit (ReLU) used as an activation function throughout.
- Cross Entropy loss is used
- Batch size is taken to be 512 (w/ shuffling), learning rate $= 5 \times 10^{-5}$
- Adam optimizer is used

2.1 Normalization

The training data was normalised to have a zero mean, unit variance distribution. The value of mean and standard deviation of the non-normalized training set are

$$\mu = 0.1307 \text{ and } \sigma = 0.3081$$

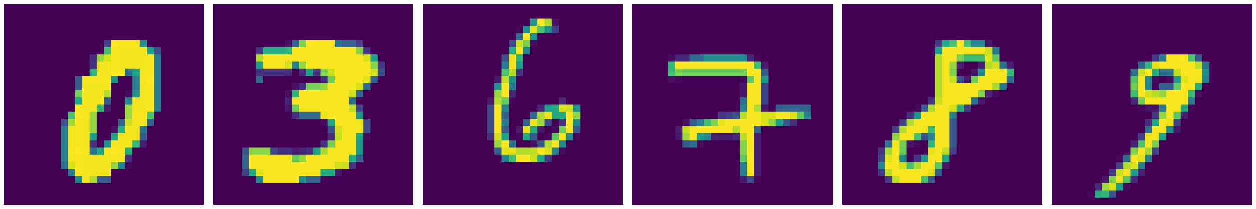

Figure 1: The MNIST Handwritten Digit Dataset

2.2 Loss Function

The Cross Entropy loss is a derived from KL-Divergence, which is an information theoretic quantity used to measure the similarity between two probability distribution $p(x)$ and $q(x)$. Let, $p(x)$ denote the probability distribution which our model outputs for some image $\mathcal{I}$ and $p^*(x)$ denote the optimal probability distribution over the possible classes χ. Then the KL divergence is given by

$$\begin{aligned} KL(P^*||P) &= -\sum_{x \in \chi} p^*(x) \log \frac{p(x)}{p^*(x)} \\ &= -\sum_{x \in \chi} p^*(x) \log p(x) + \sum_{x \in \chi} p^*(x) \log p^*(x) \end{aligned}$$

The optimal distribution $p^*(x)$ is fixed. Hence, for the purpose of optimization, we ignore the second term. Further, while taking $p^*(x)$ to be the one hot encoding of the class to which image $\mathcal{I}$ belongs isn't optimal [1], it works reasonably well in almost all practical settings. This finally gives us the Cross Entropy Loss for the image $\mathcal{I}$ belonging to class $x_{\mathcal{I}}$ as

$$\mathcal{L}(\mathcal{I}) = -\log p(x_{\mathcal{I}})$$

2.3 The Unregularized Model

Here, we simply train the model described in Section 2 with shuffled, normalized data and without any regularization for 250 epochs. Our Neural Network with two hidden layers and 500 neurons each has a very high model complexity especially with respect MNIST which is relatively an easier dataset to classify. Hence, it is quite predictable that our model will overfit the data. Hence, our results as seen in Figure 2. are in line with out intuition. Herein, we achieved a training accuracy of **100%** in contrast to a validation accuracy of only **98.22%**. On closer inspection, we can see that while the training accuracy increases continuously, the curve for the validation error is rather bouncy.

2.4 Model with L^2 Parameter Regularization

Known also as ridge regression or Tikhonov regularisation, L^2 regularisation adds a wight penalty to the loss function as

$$\mathcal{L}_{L^2} = \mathcal{L} + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

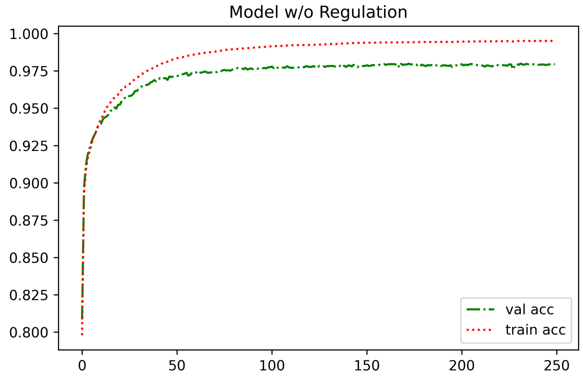

Figure 2: Validation & Training Accuracy vs Epochs for the Unregularized Model

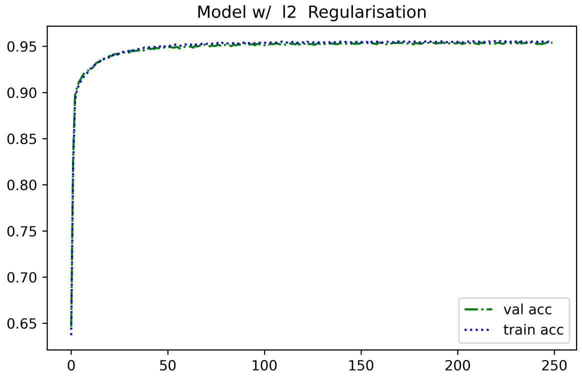

Figure 3: Validation & Training Accuracy vs Epochs for the L^2 Regularized Model

where $\mathcal{L}_{L^2}$ is the L_2 regularized loss, $\mathcal{L}$ is the unregularized loss, $\mathbf{w}$ the vector of all the parameters of the network and λ is the regularisation hyperparameter. The norm penalty ensure that only the directions along which the parameters contribute significantly in reducing the loss are relatively preserved while the other parameters are decayed throughout the training. This prevents overfitting and ensures that the training and validation accuracy do not start diverging after certain number of epochs as seen in Figure 3. Here the **regularization parameter**, λ was set to $\mathbf{7 \times 10^{-3}}$. Herein, we achieved a training accuracy of **95.47%** and a validation accuracy of **95.33%** which are very comparable.

2.5 Model with Dropout Layer Regularization

Ensemble methods are a well known class of algorithms used for regularization especially in decision trees. However extending the same to Neural Networks is computationally very heavy and infeasible. Dropout [2] is a clever way to get around this. A neuron in a given hidden layer to which dropout is applied, say with a parameter p, is stochastically dropped while training i.e. it neither participates in the forward pass nor does the weights attached to it get updated during the backward pass. However, during validation, all such neurons are present for the forward pass and all the weights connected to such a dropout neuron are multiplied by its dropout probability p. This promotes the neurons to generalize since instead of learning extremely fine features of the training data causing overfitting, they are engaged in adapting to the mistakes of previous layers, making the network more robust. We use dropout values of **0.75** in the first hidden layer and **0.65** in the second one. As seen in Figure 4., while the model is still a bit susceptible to overfitting due to its large capacity, it generalises reasonably well. Herein, we achieved a training accuracy of **98.40%** and a validation accuracy of **97.15%**.

2.6 Model with Early Stopping

When using models with sufficiently large capacity, as seen in Section 2.3, it is common to see the training accuracy increase continuously over the epochs. However, the validation

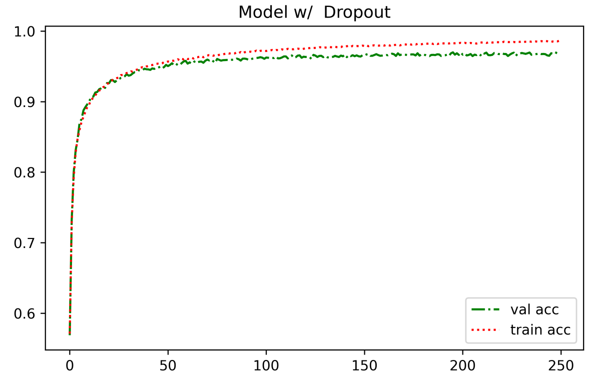

Figure 4: Validation & Training Accuracy vs Epochs for the Dropout Regularized Model

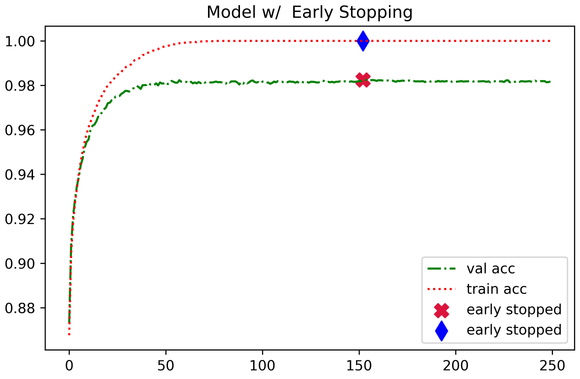

Figure 5: Validation & Training Accuracy vs Epochs for the Early Stopping Model

accuracy either starts to decrease or stagnates with fluctuations. This is classic overfitting. To circumvent this, one of the most simplest, efficient and hence widely used method to regularise is to simple stop the training as soon the validation accuracy is determined to be clearly decreasing. Practically this can be done by training the model for a large number of epochs while maintaining the maximum validation accuracy seen till that epoch along with the weights. As seen in Figure 5, the marks on the curve refer to the epoch at which highest validation accuracy was achieved. Herein, we achieved a training accuracy of **100.00%** and a validation accuracy of **98.24%**. This is clearly the method yielding the highest accuracy on the validation set.

3 CNN on CIFAR-10

The CIFAR-10 dataset is one of the most popular dataset in machine learning and computer vision. It consists of about 50,000 images for the purpose of training and about 10,000 for validation which are all of size 32×32. As suggested by the name, the images come from a total of 10 classes which are airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Unlike the MNSIT dataset, this dataset consists of RGB coloured images. Similar to the Section 2.1, the data was normalized to yield zero mean and unit variance across all the 3 color channels. The value of mean and standard deviation of the non-normalized training set are

$$\mu = [0.4914, 0.4822, 0.4465] \text{ and } \sigma = [0.2470, 0.2435, 0.2616]$$

3.1 Our Best Model

Well regularised models with large capacity i.e. more filters, convolutional blocks and deeper fully connected layers generally can yield a higher validation accuracy. However such models require multiple GPUs for training. Since we mostly use Google Colab for all the experiments, the best validation accuracy we could come up with in a simple CNN without any augmentation or advanced techniques is **80.06%**. In this CNN, we used an initial learning rate of 2×10^{-3} with Adam optimizer and a batch size of 512. Model architecture used was:

Figure 6: The CIFAR-10 Dataset

- Convolution Block 1
  - 32 Conv2D Filters, BN, ReLU, MaxPool2D
- Convolution Block 2
  - 64 Conv2D Filters, BN, ReLU, MaxPool2D
- Convolution Block 3
  - 128 Conv2D Filters, BN, ReLU, MaxPool2D
- Convolution Block 4
  - 256 Conv2D Filters, BN, ReLU, MaxPool2D
- Fully Connected 1
  - Dropout(0.3), 512 Neurons, ReLU
- Fully Connected 2
  - Dropout(0.3), 256 Neurons, ReLU

All the Conv2D filters were applied with a padding of 1 and had kernel size 3. All the MaxPool2D layers had kernel size 2 and stride 2. Additionally, to prevent overfitting, the weight decay parameter was set to 3×10^{-4} in the optimizer. The model was trained for 100 epochs.

References

[1] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.