Gurmukhi Digit Classification

Revant Teotia

July 2020

Observations by analyzing the dataset

1. Dataset contains 1000 training and 178 validation images
2. Images belong to 10 classes (1 for each digit)
3. Each image is 32x32 with 3 color channels
4. All 3 channels have the same pixel values thus making the image effectively 1 channel image
5. Pixels are either Black(0) or White(255) : making black digits on white background

Example image of a 5

Link to notebook used for analysing the dataset

Data preprocessing

1. Converted images from RGB (3-channel) to grayscale (1 channel)
2. Rescaled pixels values from 0-255 to 0-1
3. Batched the data into batches of 32 images

Training Neural Network to classify images

notebook used for training : here

Loss Function used for Optimization : **Cross Entropy loss** : $-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$
where:
M - number of classes
log - the natural log
y - binary indicator (0 or 1) if class label c is the correct classification for observation o
p - predicted probability observation o is of class c

Optimizer used: **Adam (Adaptive Moment Estimation)**
with learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-07$

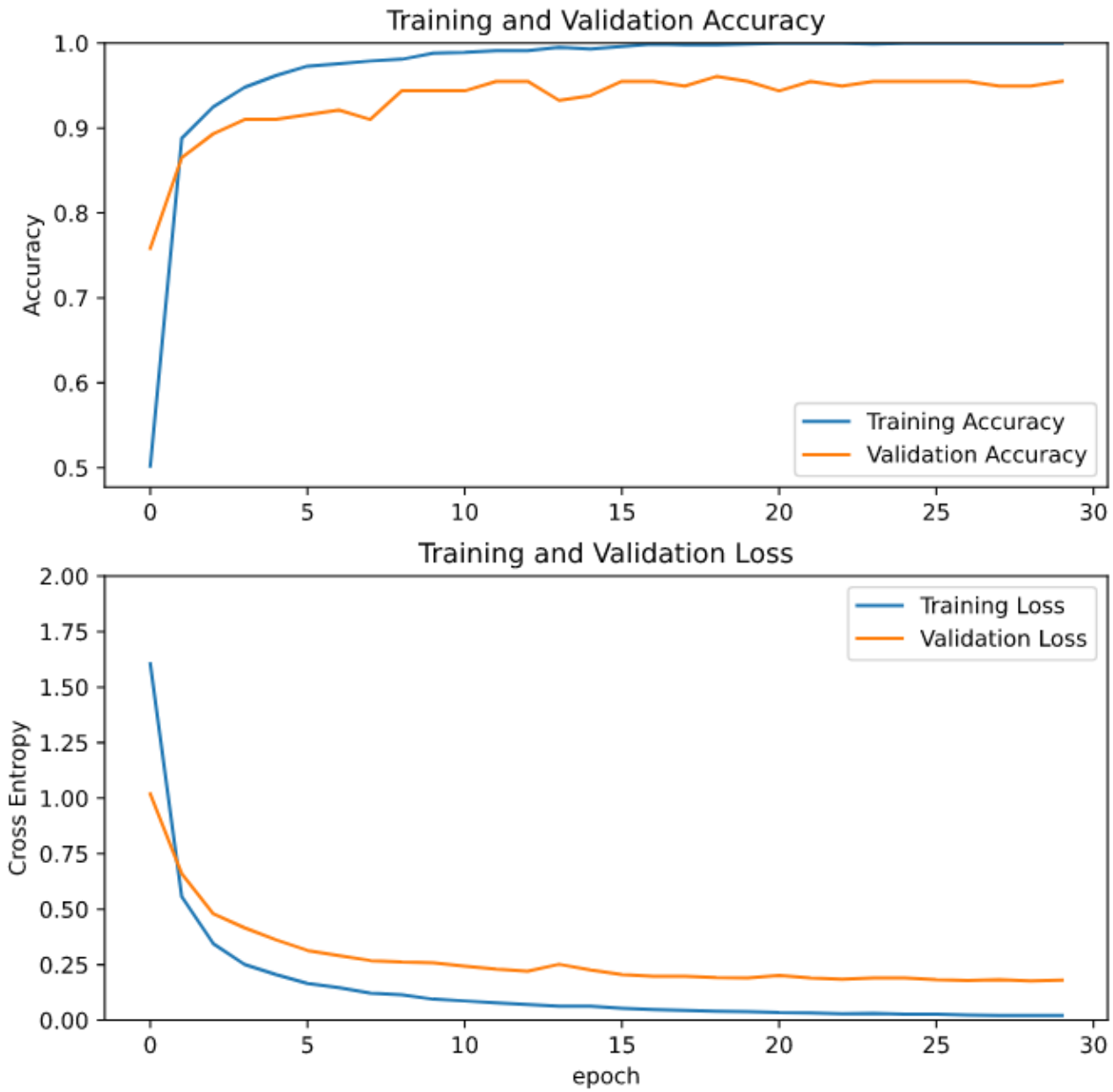
Training 1-layer NN with no hidden layer as baseline

Trained a NN with only one input layer and one output layer. No hidden layers used. Model looked like this

Image (32x32)
↓
Input Layer (1024 units)
↓
Output Layer (10 units with softmax activation)

Results obtained : for best weights saved during training using early stopping strategy:
Validation loss: 0.2166 and **Validation accuracy: 96.07%**

Progress of loss and accuracy during training is shown below :



Observation noted for 1-layer NN

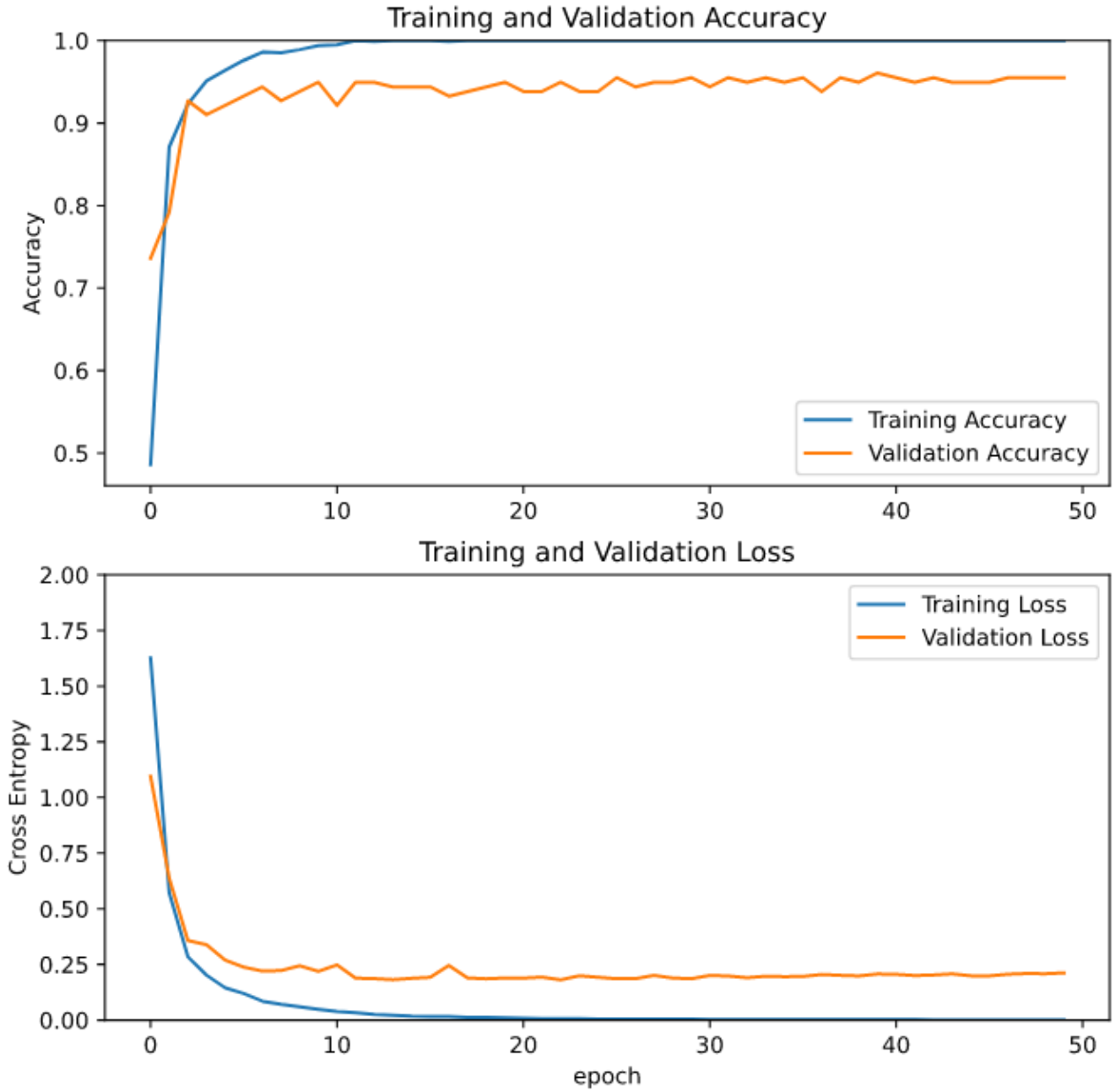
1. Starts overfitting just after training for 2-4 epochs
2. Training accuracy reaches 100% accuracy in 12-15 epochs while validation accuracy hovers around 94% during later epochs : indicating clear overfitting
3. Training loss keeps on decreasing while training for larger number of epochs while validation loss reaches its minimum in 15-20 epochs of training : again showing clear overfitting

Training NN with one or more hidden layers

Models looked like this

Image (32x32)
↓
Input Layer (1024 units)
↓
One or more Hidden Layer (5 to 128 units with relu activation)
↓
Output Layer (10 units with softmax activation)

Progress of loss and accuracy during training of NN with hidden layers :



Observation noted for NN with one or more hidden layers

1. Starts overfitting just after training for 2 epochs of training
2. Training accuracy reaches 100% accuracy in 10-12 epochs of training while validation accuracy hovers around 94% during later epochs : indicating **more overfitting than the baseline model**
3. Training loss almost vanishes after 15-20 epochs of training while validation loss reaches its minimum in 5-7 epochs of training and ddoes not decrease further : again showing large overfitting
4. **NONE of the NNs with hidden layers could outperform the baseline model with no hidden layer**

Using Image augmentation by adding random distortions in images

Added random **rotation and shear distortions in training images**.

1. Mimicking distortions present hand written digits.
2. Effectively increasing the size of training data.

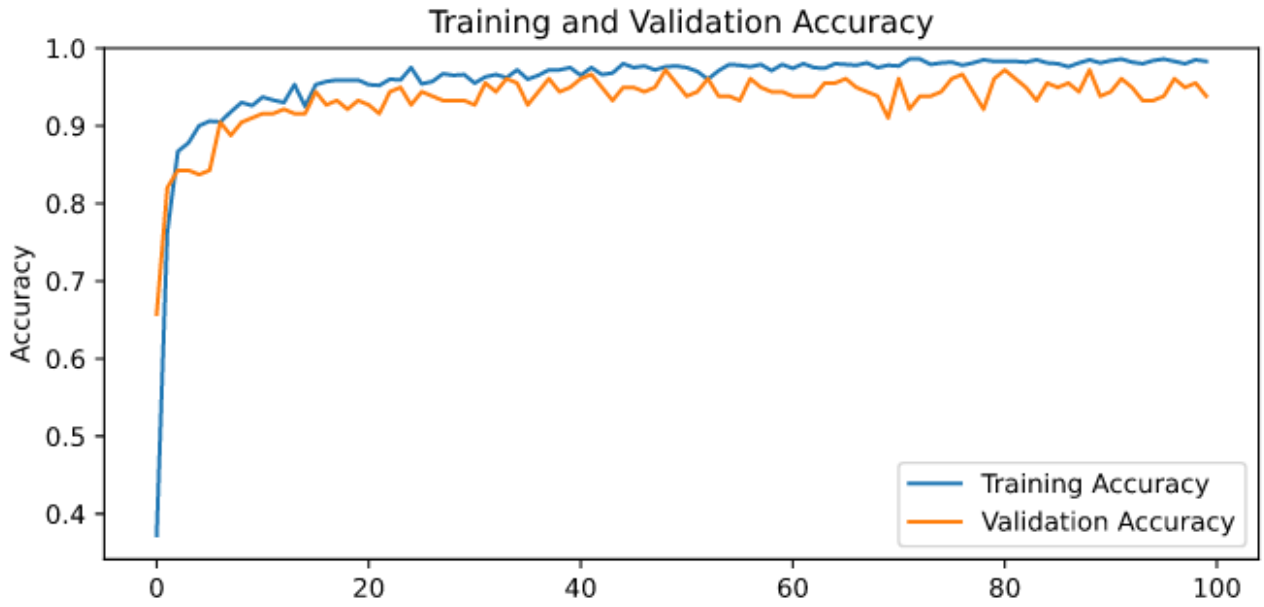
Range of random shear and rotation distortions applied = 0°-15°

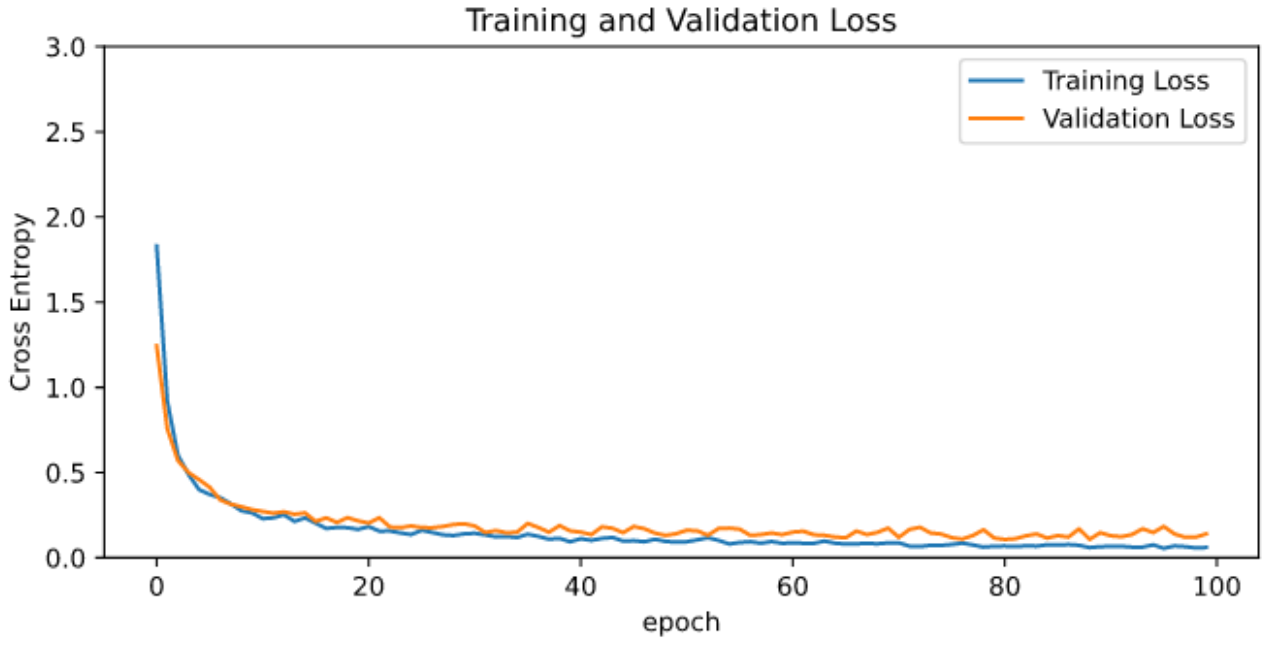
Training 1-layer NN with no hidden layer with augmented data

Model looked like this

Image (32x32)
↓
Adding random distortions to image (only for training set)
↓
Input Layer (1024 units)
↓
Output Layer (10 units with softmax activation)

Progress of loss and accuracy during training is shown below :





Observations noted for 1-layer NN

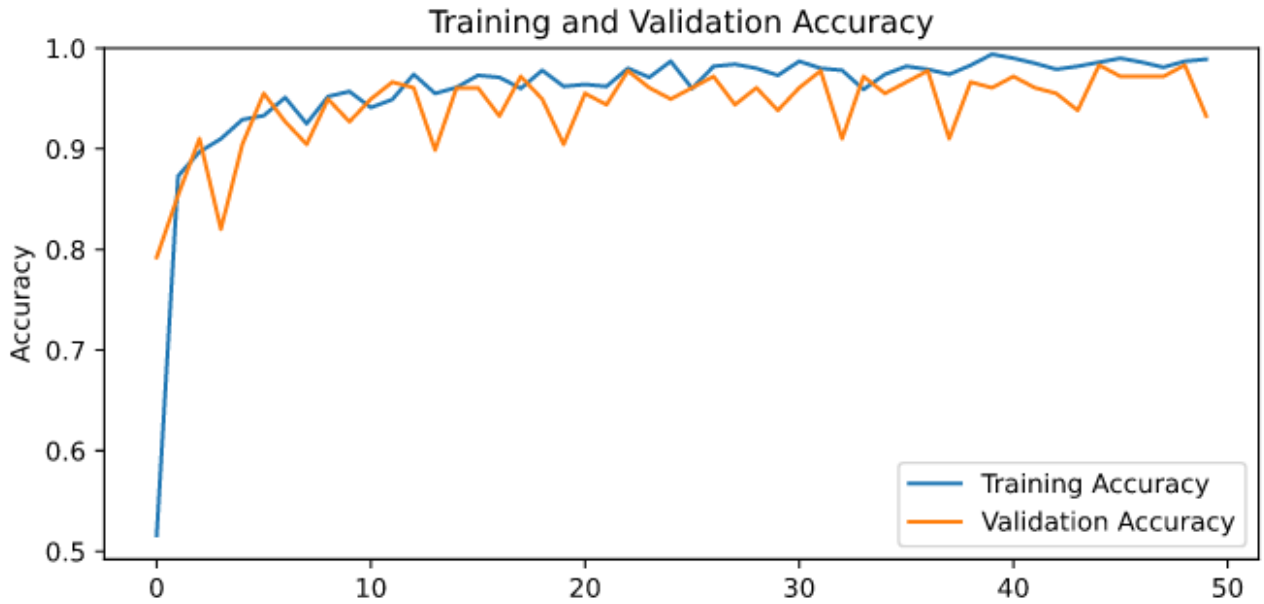
1. Significantly less overfitting than the one without image augmentation
2. Improved loss and accuracy (97.19%) for validation set than the one without image augmentation.
3. OVERALL : **adding random distortions improved the baseline 1-layer model**.

Training NN with one or more hidden layers with data augmentation

Models looked like this

Image (32x32)
↓
Adding random distortions to image(only for training set)
↓
Input Layer (1024 units)
↓
One or more Hidden Layer (5 to 2048 units with relu activation)
↓
Output Layer (10 units with softmax activation)

Progress of loss and accuracy during training of NN with hidden layers :



Observation noted for NN with one or more hidden layers with image distortions

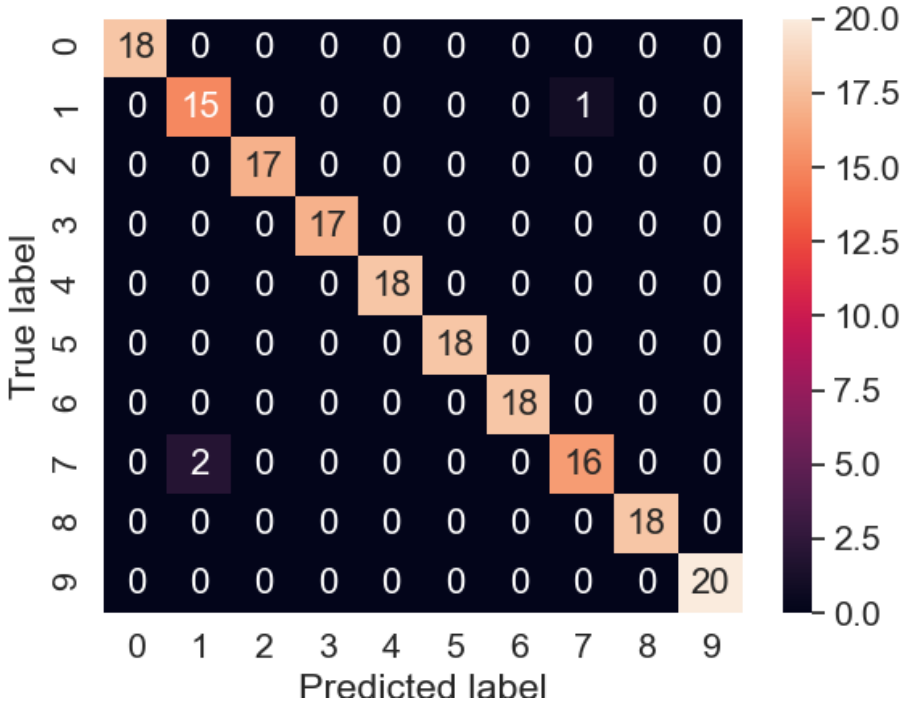
1. Very less (negligible) overfitting
2. Perform very well with $> 98\%$ accuracy on validation data.
3. Only 2-3 images out of 178 images in validation set are misclassified

Best model and its performance on validation data

With 2 hidden layers having 1024 units each with relu activation. Looked like this :

Image (32x32)
↓
Adding random distortions to image(only for training set)
↓
Input Layer (1024 units)
↓
Hidden Layer : 1024 units with relu activation
↓
Hidden Layer : 1024 units with relu activation
↓
Output Layer (10 units with softmax activation)

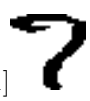
Result :Accuracy on validation data = 98.31%, Only 3 images misclassified



Confusion Matrix

Observation from confusion matrix

Model finds it hard to differentiate between 1 and 7. Maybe because they look similar

[Example image of a 1]  [Example image of a 7] 

1 and 7 look similar

Code used :

(also containing more details of experiments done) :

- For analysing the dataset
- For classification w/o image distortions
- For classification with image distortions
- To test and plot confusion matrix