EE258 Project 1 Report
Chandrahas Soman
012428377

Methodology

Our approach was to start with simple things and keep adding complexities step by step to improve the performance. If we observe any degradation in performance then we took a step back and tried new things. Whenever we added a functionality (complexity or a new parameter) we kept it for all the models after that.

We kept two parameters constant for measuring the performance of different classifiers on the same scale. We restricted number of epochs to 30 and we will see below that model overfits as we increase number epochs so this was a good idea. We used softmax activation function at the output layer because we needed a ten class classifier. For every model we flattened our input image to one dimension i.e from 32* 32*3 to 3072.

We developed a simple model without normalizing the data and observed that it is giving really ugly results. Therefore, we have used normalize data for training rest all the models as normalized data gave us way better results.

The following are classifier descriptions. I am highlighting the complexity brought up in new models. Results of these models are compared and discussed in simulations section.

Model 0 (Baseline model)

We started with a simple 4 layer neural network. (3072-256-256-10) with tanh and ReLU activation functions in the hidden layers and softmax activation function for the output layer. We used stochastic gradient descent for weight updation. We observed accuracy and categorical entropy loss.

Model 1

This is a **3 layer** neural network (3072-512-10) with ReLU activation in the hidden layer and softmax activation function in the output layer. We used stochastic gradient descent for weight updation. We observed accuracy, categorical entropy loss and also the **f1 score**. We defined a function to calculate f1 score.

Model 2

This is a **5 layer** neural network (3072-384-384-48-10) with ReLU activation function in **all** hidden layers and softmax activation function in the output layer. We used stochastic gradient descent for weight updation. We observed accuracy, categorical entropy loss and f1 score.

Model 2 with validation split

This model is same as Model 2 but fitted with **validation split** equal to 0.2 and 0.33.
(end of code part 1)

Model 2 with 10 fold cross validation

This model is same as Model 2 but fitted with **10 fold cross validation**.

Model 3

This is a **7 layer** neural network (3072-512-256-128-64-32-10) with ReLU activation function in all hidden layers and softmax activation function in the output layer. We used **adam optimizer** for weight updation. We observed accuracy, categorical entropy loss and f1 score.

Model 4

This is a 7 layer neural network (**3072-512-384-256-128-64-10**) with ReLU activation function in all hidden layers and softmax activation function in the output layer. We used **stochastic gradient descent with learning rate of 0.01, decay of 10^-6 and momentum constant of 0.9** for weight updation. We observed accuracy, categorical entropy loss and f1 score.

Model 5

This is a **6 layer** neural network (3072-512-384-128-64-10) with ReLU activation function in all hidden layers and softmax activation function in the output layer. We used stochastic gradient descent with **learning rate of 0.1, decay of 10^-6 and momentum constant of 0.7** for weight updation. We observed accuracy, categorical entropy loss and f1 score.

Model 6

This is a 6 layer neural network (**3072-512-384-160-64-10**) with ReLU activation function in all hidden layers and softmax activation function in the output layer. We used **stochastic gradient descent with learning rate of 0.01, decay of 10^-6 and momentum constant of 0.9** for weight updation. We observed accuracy, categorical entropy loss and f1 score.

Model 6 with validation split

This model is same as Model 6 but fitted with **validation split** equal to 0.2 for observing the plots.

Model 7

This is a 6 layer neural network (3072-512-384-160-64-10) with ReLU activation function in all hidden layers and softmax activation function in the output layer. We used stochastic gradient descent with **learning rate of 0.001, default decay** and momentum constant of 0.9 for weight updation. We observed accuracy, categorical entropy loss and f1 score.

Model 8 with validation split

This is a **5 layer** neural network (3072-3072-1024-3072-10) with **alternate ReLU and linear activation function** in hidden layer and softmax activation function in the output layer. This model is fitted with validation split equal to 0.2 for observing the plots.

Dataset Description

The CIFAR-10 dataset (Canadian Institute For Advanced Research) is a collection of images that are commonly used to train machine learning and computer vision algorithms. It is one of the most widely used datasets for machine learning research. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. The first 50000 images are used for training and last 10000 images are used for testing.

Interpretation of Results

Each simulation is performed with number of epoch = 30

Model 0

This is a baseline model.

Training accuracy without normalization = 10%
Training accuracy with normalization = 50.58%

(from now on every model is trained on normalized data)

Model 1

The performance of this model is worse than baseline because it underfits the data as it is only 2 layer neural network.

Training accuracy = 40.82%
F1 score = 0.2567

Model 2

This gives us better performance than baseline model as we have increased number of neurons. It better fits the data.

Training accuracy = 54.10%
F1 score = 0.7026

Model 2 with validation split

When we use only ⅔ data for training we aren't using less amount of data as compared to when we use ⅘ of data for training. That is why we are getting good results when we are using validation split of 0.2.
This is not in general but observed in this dataset.

validation split = 0.2 and number of epoch = 3

Training accuracy = 56.03%
Validation accuracy = 54.14%

Training F1 score = 0.7397
Validation F1 score = 0.7253

validation split = 0.33 and number of epoch = 3

Training accuracy = 57.96%
Validation accuracy = 53.38%
Training F1 score = 0.7805
Validation F1 score = 0.7406

Model 2 with 10 fold validation

Here we can see that validation accuracies are almost 10% less than training accuracies.
We can say that we are overfitting the data.

Average Training accuracy = 53.07%
Average Validation accuracy = 43.59%

Model 3

Here we have increased number of hidden layers and that's why we can see improvement in performance.

Training accuracy = 55.11%
Training F1 score = 0.7192

Test accuracy = 44.27
Test F1 score = 0.6996

Model 4

This model gives us best training accuracy so far. But training accuracy is not improved much. So we can say that the model is just memorizing the data.

Training accuracy = 64.33%
Training F1 score = 0.9567

Test accuracy = 47.78%
Test F1 score = 0.8786

Model 5

This model is simply off tuned. It is worse than worse models even though it has a lot of layers and a lot of neurons. This has happened because of bad choice of learning rate and bad choice of momentum constant.

Training accuracy = 49.54%
Training F1 score = 0.5939

Test accuracy = 4002
Test F1 score = 0.5168

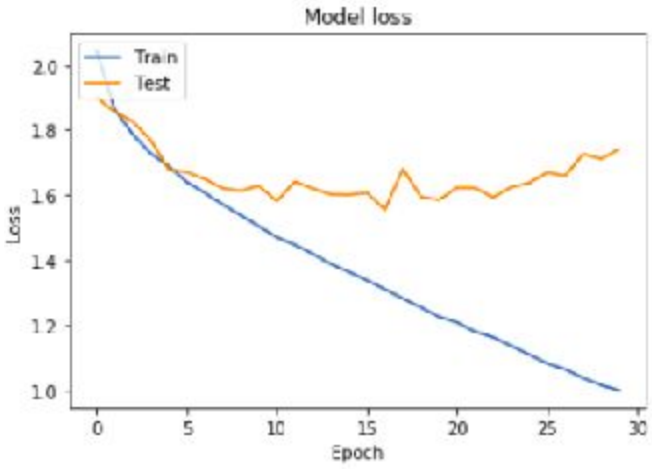
Model 6

This is a good model with so far best training accuracy with less number of hidden layers than model 4. However, we had paid price in terms of testing accuracy. Testing accuracy is almost 0.8% less than model 4.

This is a verification of computation power vs test accuracy trade off and computation time vs test accuracy trade off.

Training accuracy = 64.34%
Training F1 score = 0.9504

Test accuracy = 46.98%
Test F1 score = 0.9081



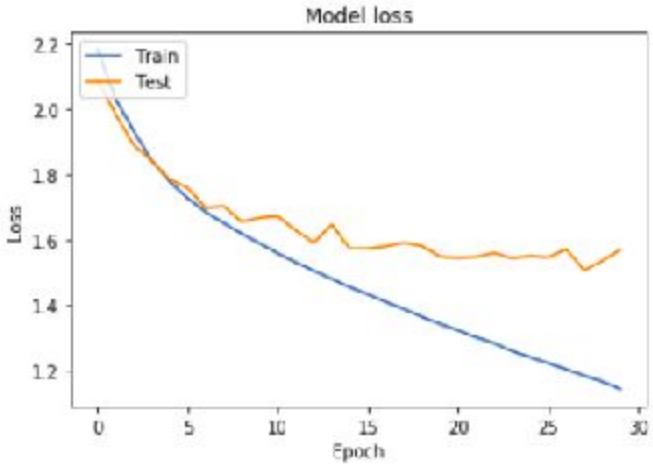
The above is the train-test error plot against number of epochs. We can see that as number of epochs increase we are overfitting the data. The U-shaped curve of test error is nicely depicted here.

Model 7

This model differs from model 6 only in terms of learning rate. This gives us the best test accuracy so far, almost 1.5% more than model 6. However, it gives 5% less training accuracy than model 6. We can from the plots that this model has not overfitted the data.

Even though, it's hard to compare this model with model 6 as we want both training and test accuracies as high as possible. But I will go with this model as the **best model** so far.

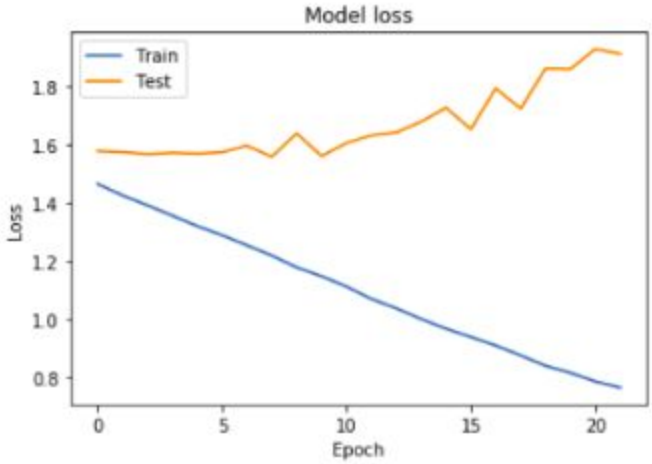
Training accuracy = 59.30%
Validation accuracy = 47.09%
Training F1 score = 0.7840
Validation F1 score = 0.7201
.



The above is the train-test error plot against number of epochs. We can see that we have not yet overfitted the data. There is a saturation of test error but no overfitting yet

Model 8

This model is perfect example overfitting. This model overfits the data because it has a lot of links. It just memorizes the data. It has the best training accuracy but testing accuracy is comparable to model 6 and 7.



Training accuracy = 72.70%
Validation accuracy = 48.18%
Training F1 score = 1.0979
Validation F1 score = 0.7201

Test accuracy = 47.02%
Test F1 score = 0.9813