cs577 Assignment 2

Sahil Riyaz Sheikh(A20518693)

Department of Computer Science
Illinois Institute of Technology

October 06, 2022

Abstract

In this assignment I'll select a three-class classification dataset from UCI repository. We will be making a 3-layer neural network which will consist of 2hidden layers and 1 output layer. I have to use the specified loss and activation function for hidden layer and output layer. I'll create various version of the model by fine tuning the hyperparameters and then compute the graph for training loss and validation loss, training accuracy and validation accuracy. In the end we will test our best model on the report the performance.

1. Problem Statement:

I had shortlisted two datasets to from UCI repository. Seeds Classification and Iris dataset. The Seed class dataset consist of 7 attributes and 210 instances with no missing values and the target consisting of 3 classes whereas the Iris dataset, which was published in 1988 consisting of only 4 attributes and 150 instances. The goal for both of these datasets is to classify the samples to their respective classes.

2. Proposed Solution:

I will be creating a simple model consisting of two hidden layer and one output layer as defined in the question. The output layer will have Softmax as activation function and the hidden layers will consist of sigmoid activation. I will use Categorical Cross Entropy loss for loss function. As I will be using categorical cross entropy will need to one hot encode the target part of the dataset. The target consists of 3 classes. After the one hot encoding they would be mapped as

-- class1 0,0,1
-- class2 0,1,0
-- class3 1,0,0

This model is inspired from the code I saw in lecture 9/15 on loss.

3. Implementation Detail:

The first major issue I faced was that implementing the seeds dataset is quite difficult as it is in ".txt" format which makes it tedious job to convert it into csv format. I tried to do this but do it by using pd.readcsv ("link") function but it proved to be useless as in the dataset the columns are not separated by "," but rather made use of space to separate them. To work on the dataset, I had to convert it into the csv format and then manually adjusting it. I decided to utilized the popular Iris dataset, which has few papers written on it as well as it's easier to import in python as we could directly load it from "from sklearn.datasets import load_iris" rather than importing it through pandas. I tried to load through the link ". I made a function "def load_dataset ():" which does the following:

1. Load Iris dataset from sklearn.datasets
2. Assigning the data, target, names and feature names to a specific variable to be used in the code further
3. I one hot encoded the targets from dataset by importing OneHotEncoder from sklearn.preprocessing. I did faced a issue here when I was importing the dataset directly from UCI website. When I imported the data through sklearn.datasets and assigned the variable labelY to them. I had to perform fit transform on the labelY to able to hot encode it and then I converted it to an array to simplify the operations which would be performed on it later. After hot encoding I was left with

Iris Setosa	0,0,1
Iris Versicolour	0,1,0
Iris Virginica	1,0,0

4. Now I normalized the data by calculating the mean of data by using .mean () function and subtracted it with the data so the mean of data is 0 and then calculated standard deviation of data and divided it by data so the standard deviation of the dataset is 1.
5. I split this normalized data into training set, testing set, training targets and testing targets. This was done by utilizing the function "train_test_split" with parameters test size =0.1 and random_state=3.
6. Then I split this training data into training and validation using the same function as specified above.
7. Finally this load_dataset() function will plot the graph of dataset.

The next was to initialize the hyperparameters such as epochs, batch size, loss function by importing it using tf.keras.losses.CategoricalCrossentropy(from_logits=True), optimizer from tf.keras.optimizers.Adam(learning_rate=0.09) with a suitable learning rate 0.09. I did not want the learning to be very small as then the model might learn at a very slow rate and I avoided having a very large learning rate as it cause the model to overshoot.

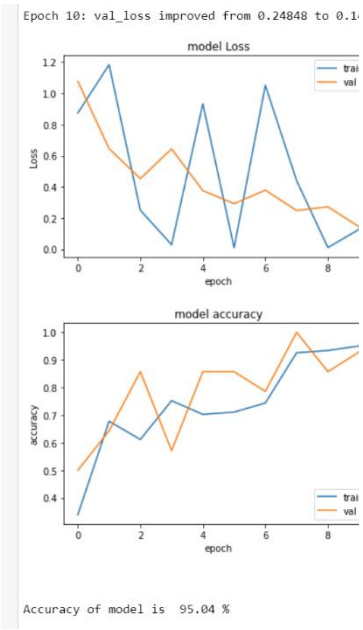
I prepared the training set and validation set by using the tf.data.Dataset.from_tensor_slices((train, train_lab)). Now implemented the training loop by keeping in the structure of professor's code from the lecture. I made use of from keras.callbacks import CallbackList t create a ModelCheckpoint() to save the weights of the model. I made use of for loop to fit the model and train it. The ModelCheckpoint stores the weights in a h5 file which will be created by the function and saved in the folder where the source code is saved.

4. Results:

I implemented two models with similar structure. The main difference between then is that in first layer of model 1 I initialized 3 units for inputs. This helps to simplify the data as there are 4 attributes and in the model 2 I kept the number of inputs equal to the number of attributes i.e. 4. Here are the graphs for model 1

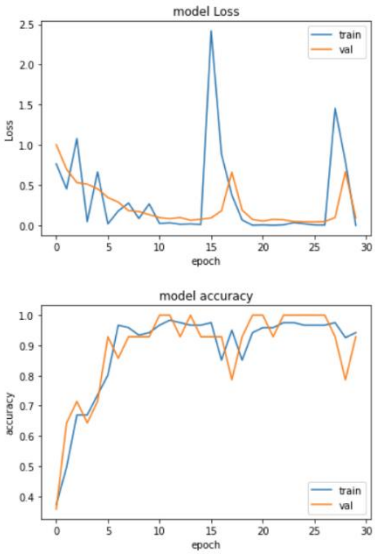
Epoch = 10

Batch size= 20



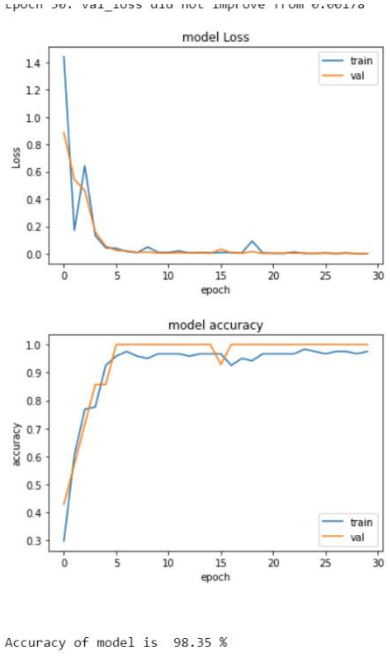
Epoch = 30

batch size=20



Epoch = 30

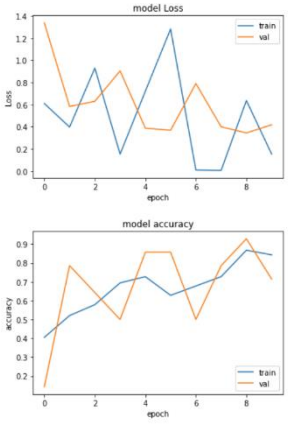
Batch size= 10



Here are the graphs for model 2:

Epoch = 10

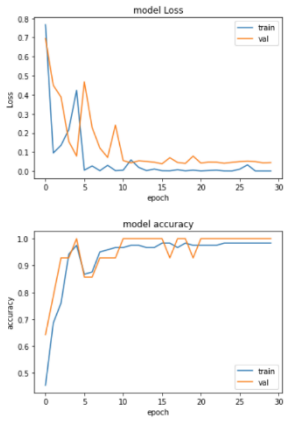
Batch size= 20



Accuracy of model is 86.78 %

Epoch = 30

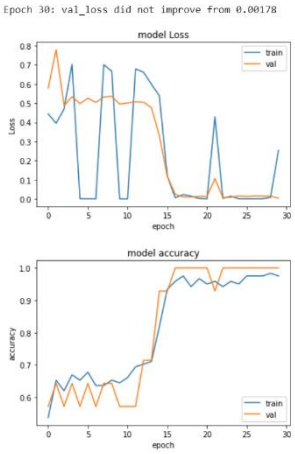
batch size=20



Accuracy of model is 98.35 %

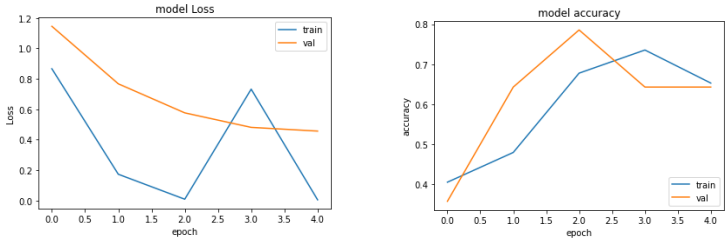
Epoch = 30

Batch size= 10



Accuracy of model is 98.35 %

After comparing their performances, I decided the model2 was slightly better when hyper parameters were "epoch =30 and batch size = 20". This model overfits after 5th epoch so I tried to run it for 5 epochs and this was the results obtained.



The drawback of our best model is the validation loss which is approximately in the range of 0.40-0.45. Our main goal is to reduce this validation loss without overfitting the model. The validation accuracy dips a little in the beginning but increases as the number of epoch increases. This could be due the less numbers of instances due to which we have less data to train on.

On right is our performance when model is evaluated on test set. We obtained a accuracy of 80.00 percentage and a loss of 32.16 percent

```
Epoch 5: val_loss improved from 0.48063 to 0.45634, saving model to best_weights.h5


Accuracy of model is  73.55 %
1/1 [==============================] - 0s 220ms/step - loss: 0.3217 - accuracy: 0.8000
Test accuracy of model is  [0.3216777443885803, 0.800000011920929]
```

5. References:

Lecture notes of Lesson 08 on loss.
https://stackoverflow.com/questions/57622988/unable-to-transform-the-categorical-variable-showing-categories-auto-error
https://stackoverflow.com/questions/57622988/unable-to-transform-the-categorical-variable-showing-categories-auto-error
https://stackoverflow.com/questions/57622988/unable-to-transform-the-categorical-variable-showing-categories-auto-error