Assignment 1 Report

Harsh Vora
CWID : A20445400
CS577 – S20

February 17, 2020

Task 1:

Problem Statement:

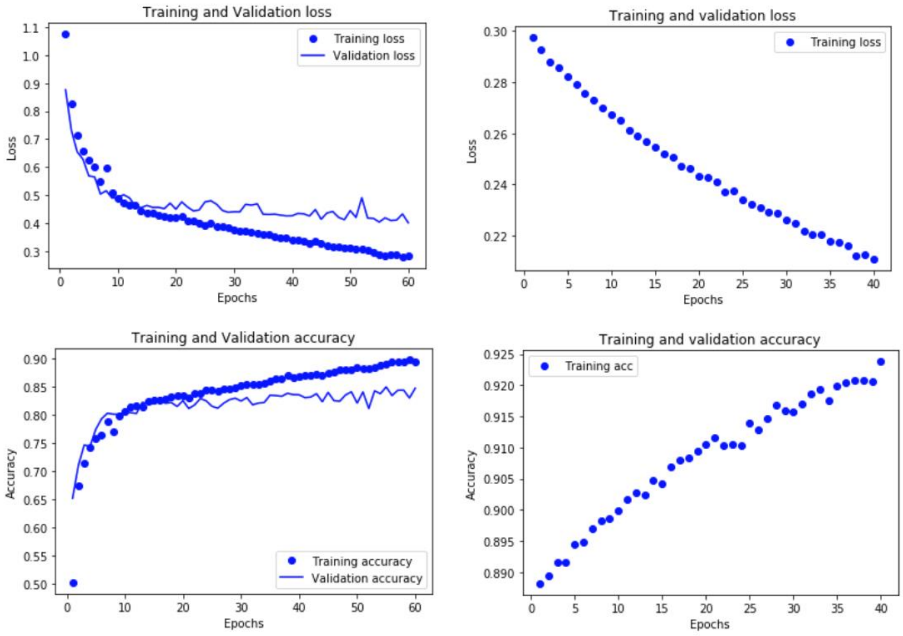
- Load the CIFAR dataset and select a subset of three classes.
- Split the training set into training and validation subset.
- Vectorize the images and encode the know class labels using categorical encoding.
- Design a fully connected neural network to perform multi-class classification of this data
- Justify your network design decisions (number of hidden layers, number of units per layer, loss function, and evaluation metric)
- Build and compile the network designed.
- Plot training and validation loss as a function of epochs, plot training and validation accuracy.
- Tune model hyper – parameters to improve performance.
- Retrain the final mode and test performance on the test collection.
- Report the performance obtained.
- Save and load the weights of the trained network

Implementation Details:

- Import all packages that are necessary such as keras, matplotlib and numpy.
- Load the CIFAR10 data and split the data into training and testing subset.
- Select only three classes of the training data. This is achieved by using following numpy function.
  train_images[np.where(train_labels[:,0] < 3)[0],:]
- Similar equations are made for train_labels, test_images and test_labels.
- Verify the size of new training and testing dataset that have only 3 class out of 10, by using following method:
  train_images_subset.shape, train_labels_subset.shape, test_images_subset.shape, train_labels_subset.shape
- We convert the training_labels_subset and testing_labels_subset to canonical encoding by using to_categorical method
  ku.to_categorical(train_labels_subset,3)
  ku.to_categorical(test_labels_subset,3)
- To vectorize we convert the training_images_subset and testing_images_subset to float and divide each pixel by 255.
- We split the training data into partial training and validation data.
- Build a sequential model with input shape (32,32,3) i.e. the dimension of images in the CIFAR10 dataset.

- Then we add Conv2D layer with (3,3) padding across the edges along with activation function relu.
- Then we add a Flatten() layer which will reduce the size of the image by half.
- Then we add a Dense layer with output size 3 (3 classes out of 10) and an activation function softmax.
- Then we compile the model using the optimizer SGD() (Stochastic Gradient Descend)and Loss Function as categorical_crossentropy and metrics as accuracy.
- We train the model using fit() function for partial training data with a batch size of 512 and 60 epochs and store the training details in history variable.
- The model is validated using the validation data.
- Then we plot training and validation loss with respect to epochs by using matplotlib.
- Similarly we plot the training and validation accuracy by using matplotlib.
- Retrain the model using learning rate of 0.001 as the model accuracy was dropping after every 8 epochs. Also I have trained the final model for 40 epochs.
- Saved the model as a hdf5 file
- Reload the model and test the model accuracy on test data. I obtain 85.03 % accuracy on the testing data.
- I loaded an image of a dog from test images and the model makes the right prediction

Results and Discussion:



Task 2 :

Problem Statement

- Load the spam email data from UCI repository "spambase" (https://archive.ics.uci.edu/ml/datasets/spambase)
- Prepare the data you load as tensor suitable for neural network.
- Normalize the features as needed.
- Repeat the steps for Task 1.

Implementation Details:

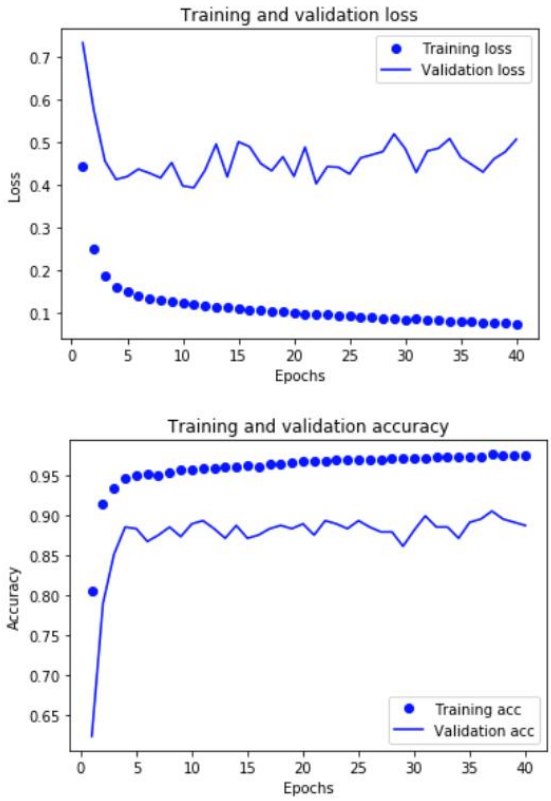
- Import all packages that are necessary such as keras, matplotlib and numpy.
- Create a function load_spam_data()
- Then we extract the values from the dataframe using dataframe.values
- Then we split the data into spam and non_spam datasets. So that training and testing subset contains equally proportional amount of spam and non_spam data.
- We use the following function to split the data:

  not_spam, spam = np.split(dataframe, np.where(np.diff(dataframe[:,57]))[0]+1)
- After splitting the data into equal amounts of spam and non_spam emails we again stack the data using the following function:

  train_data = np.vstack([train_ns_emails,train_s_emails])[:,:57]
- This type of stacking of data we do for all variables train_labels, test_data & test_labels
- Load the spambase data and split the data into training and testing subset.
- To Normalize the features, first we take the mean and standard deviation of training_emails.
- Then we subtract all the elements in the dataset with the mean and then we divide it with standard deviation.
- We split the training data into partial training and validation data
- Build a sequential model with input size (57,) i.e. the number of columns we have in the dataset.
- Then we add 1 dense layer of size 16 with an activation of relu and a hidden dense layer with activation of relu.
- Then at the end we add the output layer of size 1 (because it is a binary classification) and activation as sigmoid.
- Then we compile the model using the optimizer RMSprop() (Root Mean Square Propogation) and Loss Function as binary_crossentropy and metrics as binary_accuracy.
- We train the model using fit() function for partial training data with a batch size of 32 and 40 epochs and store the training details in history variable..
- The training details are stored in the history variable.
- Then we plot training and validation loss with respect to epochs by using matplotlib.
- Similarly we plot the training and validation accuracy by using matplotlib
- Retrain the model. In this case I didn't change any hyper parameters. But I train the model on the entire training data.
- Save the model as hdf5 file.

- Reload the model and test the model accuracy on test data. I obtain 86.52 % accuracy on the testing data.

Results and Discussion:



Task 3:

Problem Statement

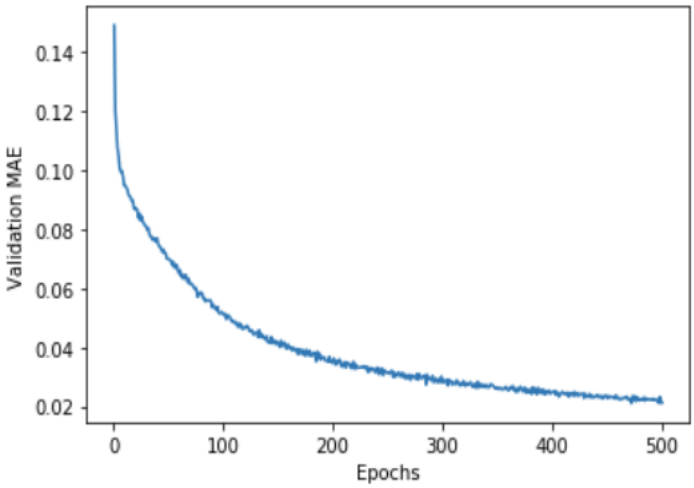
- Load the crime data from UCI repository “Communities and crime” (https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime).
- Prepare the data you load as a tensor suitable for a neural network.
- Normalize features as needed.
- Explain the steps you perform in preparing the data and justify them.
- Repeat the steps performed for task 1 expect that in this case perform the k-fold cross validation.

Implementation Details:

- Import all packages that are necessary such as keras, matplotlib and numpy.
- Create a function load_crime_data()
- Read the csv file using “pandas.read_csv”.and then replace “?” with “np.NAN”
- Then we extract the values from the dataframe using dataframe.values

- Split the data into training and testing subsets.
- We only consider columns from 6 to 127. Because columns from 1 – 5 are non predictive and also column 128 is a label. So we take column 128 as the class label.
- Since there are many np.nan values in the dataset so we take mean of the column where missing values are present and place it with the mean of that column.
- Build a sequential model with input size (57,) i.e. we have 122 trainable attributes in the training data.
- Then we add 1 dense layer of size 64 with an activation of relu and a hidden dense layer of size 64 with activation of relu.
- Then at the end we add the output layer of size 1 (because it is a binary classification) and activation as sigmoid.
- Then we compile the model using the optimizer RMSprop() (Root Mean Square Propogation) and Loss Function as mse(mean squared error) and metrics as mae(mean absolute error).
- Then we train the model for using k-fold cross validation. This is done typically because we have less data to work with.
- We split the training data into a subset of training and validation data. But for each fold we take a different set of training data as the training and validation data.
- We take scores of each of the fold and append it into an array.
- The mae(mean absolute error) is mean of all the mae of the K-fold.
- Then we Plot the results on the graph.
- The training details are stored in the history variable.
- Retrain the model using all training data for 20 epochs only.
- Save the model as hdf5 file.
- Reload the model and test the model accuracy on test data.

Results and Discussion:



References:

https://keras.io/