SENTIMENT ANALYSIS USING NEURAL NETWORKS

Part I: Research Question

A. Describe the purpose of this data analysis by doing the following:

1. Summarize **one** research question that you will answer using neural network models and NLP techniques. Be sure the research question is relevant to a real-world organizational situation and sentiment analysis captured in your chosen dataset.

Is it possible to predict a customer's positive or negative review of a product based on past data using neural networks, specifically natural language processing (NLP)?

2. Define the objectives or goals of the data analysis. Be sure the objectives or goals are reasonable within the scope of the research question and are represented in the available data.

The objective of this analysis is to create a model that interprets and classifies the sentiment of a new product review to determine if it is positive, or negative.

3. Identify a type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences on the selected data set.

The deep learning neural networks used are Keras and TensorFlow. Keras is a closed-source neural network library, and TensorFlow is an open-source library. Both serve a variety of functions in machine learning. **TensorFlow provides both high-level and low-level APIs** while Keras provides only high-level APIs[1]. Since the goal is to predict if our customers have a positive or negative review of our products, we will be using a natural language processing model.

Part II: Data Preparation

B. Summarize the data cleaning process by doing the following:

1. Perform exploratory data analysis on the chosen dataset, and include an explanation of each of the following elements:
   - presence of unusual characters (e.g., emojis, non-English characters, etc.)

Upon analysis, there are three different emojis present throughout the data that do not impact the overall analysis.

```
In [45]: #Checking for popular emojis

         #:)
         for i in all_data_df.review:
             substring = ":)"
             if substring in i:
                 print("Happy Emoji Present")

         #:(
         for i in all_data_df.review:
             substring = ":("
             if substring in i:
                 print("Sad Emoji Present")

         #:-
         for i in all_data_df.review:
             substring = ":-"
             if substring in i:
                 print("Disappointing Emoji Present")

Happy Emoji Present
Happy Emoji Present
Happy Emoji Present
Happy Emoji Present
Sad Emoji Present
Disappointing Emoji Present
Disappointing Emoji Present
```

Figure 1: Presence of Popular Emojis

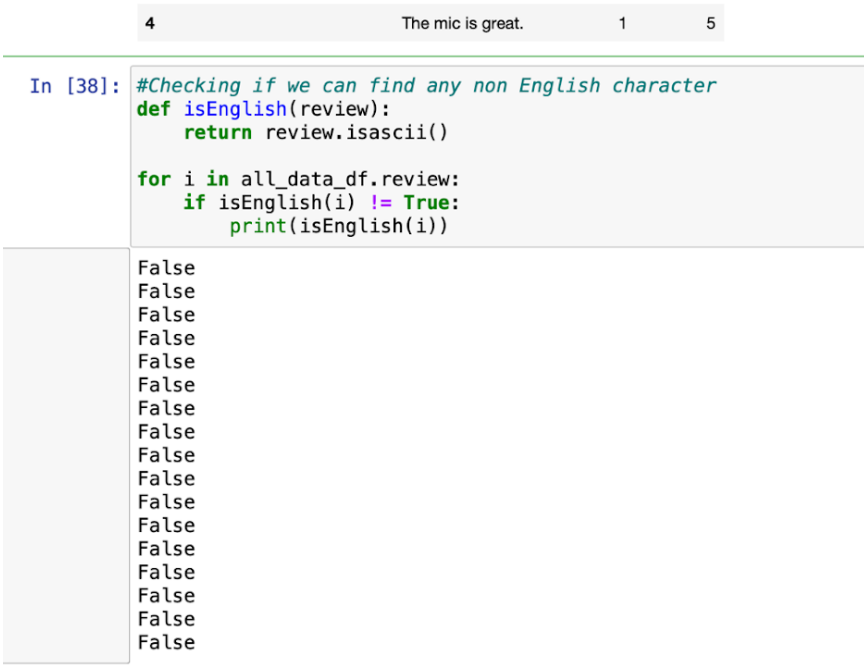

Figure 2: Non English Character Test

- vocabulary size

The set of unique words used in the text corpus is referred to as the **vocabulary**. When processing raw text for NLP, everything is done around the vocabulary[6].

```
In [41]: #Calculating how many unique words we have
unique_words = set(all_data_df['review'].str.replace('[^a-zA-Z]', '').str.lower().str.split(' ').sum())

#Printing the length of unique_words
print('There are ' + str(len(unique_words)) + ' unique words')
# print(unique_words)
```

```
There are 2717 unique words
```

Figure 3: Vocabulary Size

- proposed word embedding length

Word embedding is the action of capturing language in numbers, in a format that is meaningful for a computer. Each word is represented by a direction in a multidimensional space where the dimensionality is determined by the number of words we have included in the vocabulary[7]. The word embedding length is the dimensionality which is **100**.

```
In [61]: #Import TensorFlow
import tensorflow as tf

#Build deep learning model with TensorFlow
#1st layer - embedding layer where parameters are defined
#2nd layer - makes the vector flatter
#3rd layer - hidden layer
#4th layer - softmax activation function

model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length = max_length),
    tf.keras.layers.GlobalAveragePooling1D(),
    tf.keras.layers.Dense(6, activation = 'relu'),
    tf.keras.layers.Dense(1, activation = 'softmax')
])

# Compile model and set loss function
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

# Print model summary table
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 16)           32000

 global_average_pooling1d (G  (None, 16)               0
 lobalAveragePooling1D)

 dense (Dense)               (None, 6)                 102

 dense_1 (Dense)             (None, 1)                 7

=================================================================
Total params: 32,109
Trainable params: 32,109
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4: Word Embedding Length: 100

- statistical justification for the chosen maximum sequence length

The maximum sequence length was chosen to be 100. The basic statistics of review is as follows:

```
In [26]: #Lets analyze 'Review'
         review_ = all_data_df.review.str.len()
         print(review_)
         print(review_.describe())
```

```
0       82
1       27
2       22
3       79
4       17
        ..
2743    63
2744    92
2745    32
2746    20
2747    75
Name: review, Length: 2748, dtype: int64
count    2748.000000
mean       71.528384
std       201.987266
min         7.000000
25%        32.000000
50%        55.000000
75%        87.000000
max      7944.000000
Name: review, dtype: float64
```

Figure 5: Review Info

We can see that the mean length of review is 71 with a standard deviation of 201. I chose 100 since it is a little bit greater than the average.

2. Describe the goals of the tokenization process, including any code generated and packages that are used to normalize text during the tokenization process.

Package in Python: Scikit-learn, TensorFlow and Keras

Language in its original form cannot be accurately processed by a machine, so we need to process the language to make it easier for the machine to understand. The first part of making sense of the data is through a process called *tokenization*, or splitting strings into smaller parts called *tokens*.

3. Explain the padding process used to standardize the length of sequences, including the following in your explanation:
   - if the padding occurs before or after the text sequence - post
   - a screenshot of a single padded sequence – Figure 5 below

The padding is necessary since we have sentences with different lengths[5]. We need to have inputs with the same size.

We are doing post padding in sentences with length less than 100 words.

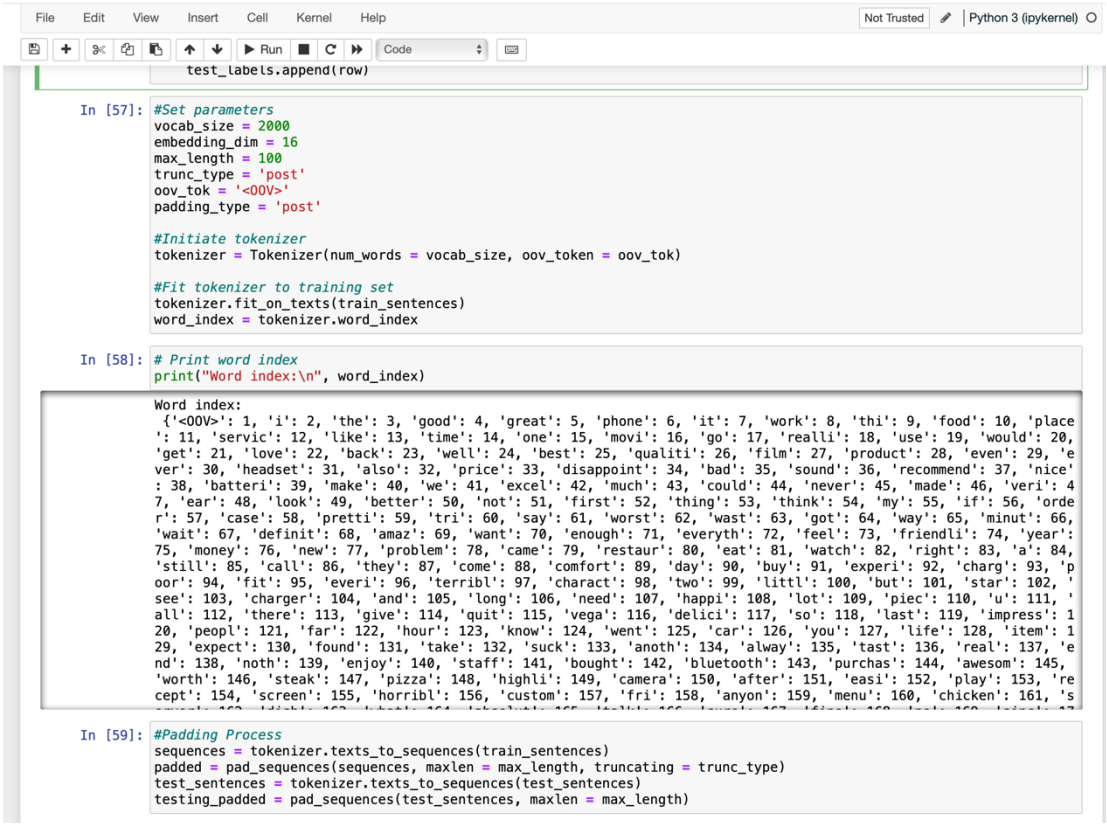

Figure 6: Padding Parameters and Sequence

4. Identify how many categories of sentiment will be used and an activation function for the final dense layer of the network.

There are only two categories of sentiments: 0 for negative reviews and 1 for positive reviews. The activation function for the final dense layer of the network is softmax[4].

```
In [105]: #Import TensorFlow
          import tensorflow as tf

          #Build deep learning model with TensorFlow
          #1st layer - embedding layer where parameters are defined
          #2nd layer - makes the vector flatter
          #3rd layer - hidden layer
          #4th layer - softmax activation function

          model = tf.keras.Sequential([
              tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length = max_length),
              tf.keras.layers.GlobalAveragePooling1D(),
              tf.keras.layers.Dense(6, activation = 'relu'),
              tf.keras.layers.Dense(1, activation = 'softmax')
          ])

          # Compile model and set loss function
          model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

          # Print model summary table
          print(model.summary())

          Model: "sequential_4"
          _________________________________________________________________
           Layer (type)                 Output Shape              Param #
          =================================================================
           embedding_4 (Embedding)      (None, 100, 16)           43200

           global_average_pooling1d_4   (None, 16)                0
           (GlobalAveragePooling1D)

           dense_8 (Dense)              (None, 6)                 102

           dense_9 (Dense)              (None, 1)                 7

          =================================================================
          Total params: 43,309
          Trainable params: 43,309
          Non-trainable params: 0
          _________________________________________________________________
          None

In [106]: #Numpy array to confirm when to stop
```

Figure 7: Softmax Activation Function

5. Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split.

1-) Load data sets (Amazon, Yelp and IMDB) using Pandas
2-) Combine all datasets into one
3-) Assign sentiment values: positive reviews have a sentiment of 1, and negative, 0
4-) Review all datasets
5-) Review Basic information about our dataset using info, shape, describe, types and looking for missing data
6-) Remove URLs – they do not add meaningful value
7-) Remove unnecessary characters (e.g “good!” is the same as “good”)
8-) Convert all captioned letters to lowercase to make the input text as generic as possible
9-) Remove all words that do not add information (the, and, in, etc)
10-) Remove rare words
11-) Correct spelling
12-) Perform Stemming and Lemmatization steps
13-) Split dataset into *training* (80%) and *testing* (20%)
14-) Tokenization
15-) Prepare dataset for extraction

```
In [112]: #Training and Testing Datasets
          from sklearn.model_selection import train_test_split

          #Defining predictor and outcome
          X = all_data_df['clean_review']
          y = all_data_df['sentiments']
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 42)

          #Print
          print('How many samples in Training:', X_train.shape[0])
          print('How many sample in Test:', X_test.shape[0])

          How many samples in Training: 2198
          How many sample in Test: 550
```

Figure 8: Training And Testing Split

6. Provide a copy of the prepared dataset.

The prepared dataset will be uploaded with this document as ***prepared_sentiment.xls***

Part III: Network Architecture

C. Describe the type of network used by doing the following:

1. Provide the output of the model summary of the function from TensorFlow.

The output is shown in Figure 9 below:

```
In [105]: #Import TensorFlow
          import tensorflow as tf

          #Build deep learning model with TensorFlow
          #1st layer - embedding layer where parameters are defined
          #2nd layer - makes the vector flatter
          #3rd layer - hidden layer
          #4th layer - softmax activation function

          model = tf.keras.Sequential([
              tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length = max_length),
              tf.keras.layers.GlobalAveragePooling1D(),
              tf.keras.layers.Dense(6, activation = 'relu'),
              tf.keras.layers.Dense(1, activation = 'softmax')
          ])

          # Compile model and set loss function
          model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

          # Print model summary table
          print(model.summary())
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 100, 16)           43200

 global_average_pooling1d_4   (None, 16)               0
 (GlobalAveragePooling1D)

 dense_8 (Dense)             (None, 6)                 102

 dense_9 (Dense)             (None, 1)                 7

=================================================================
Total params: 43,309
Trainable params: 43,309
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 9: Model Output

2. Discuss the number of layers, the type of layers, and total number of parameters.

As we can see in Figure 9 above, four layers were used.

1-) First layer (embedding layer): vocabulary size is 2700, 16 embedding dimensions, max length of a sentence is 100. Total of 43,309 parameters.

2-) Second Layer (GlobalAveragePooling1D): the objective is to flatten the vector so it is unidimensional. Total number of parameters is zero.

3-) Third Layer: (Dense and Hidden layer): best practice number neuron of six. Total of one hundred and two (102) parameters.

4-) Fourth Layer (Softmax Activation Function): It is a dense layer like the previous one, with seven (7) parameters.

3. Justify the choice of hyperparameters, including the following elements:

- **activation functions:** The ReLU function is a non-linear activation function that has gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. The main

advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time[10].

- **number of nodes per layer:** we have a small dataset. Utilizing six nodes on the third layer and one on the fourth is enough to obtain meaningful results.

- **loss function:** Binary Cross-Entropy Loss is the default loss function to use for binary classification problems. It is intended for use with binary classification where the target values are in the set {0, 1} [11].

- **optimizer:** Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iteratively, based on training data[9].

- **stopping criteria:** EarlyStopping - Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset[8].

- **evaluation metric:** we use the accuracy metric to compare how similar the measured and known values are

Part IV: Model Evaluation

D. Evaluate the model training process and its relevant outcomes by doing the following:

1. Discuss the impact of using stopping criteria instead of defining the number of epochs, including a screenshot showing the final training epoch.

EarlyStopping is used as a method to avoid the overfitting. The model tries to chase the loss function crazily on the training data, by tuning the parameters. Now, another set of data is kept as the test set and as it goes on training, a record of the loss function is kept on the test data, and when there is no more improvement on the test set, it stops, rather than going all the epochs. This strategy of stopping early based on the test set performance is called **Early Stopping**[12]. The "patience" parameter was set to four meaning if there is no improvement after four epochs, the model training will be stopped.

A problem with training neural networks is in the choice of the number of training epochs to use. Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. EarlyStopping is a method that allows you to specify an arbitrary large number of training epochs (in my case twenty) and stop training once the model performance stops improving on a hold out validation dataset[8].

```
In [70]: #Number of epochs
         num_epochs = 20

         #Fitting the model
         history = model.fit(padded,
                             training_labels_final,
                             epochs = num_epochs,
                             callbacks = EarlyStopping(monitor = 'val_loss', mode='min', patience=2, verbose=1),
                             validation_data = (testing_padded, test_labels_final))

Epoch 1/20
69/69 [==============================] - 0s 4ms/step - loss: 0.0976 - accuracy: 0.4950 - val_loss: 0.0965 - val_acc
uracy: 0.4950
Epoch 2/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0975 - accuracy: 0.4950 - val_loss: 0.0959 - val_acc
uracy: 0.4950
Epoch 3/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0961 - accuracy: 0.4950 - val_loss: 0.0943 - val_acc
uracy: 0.4950
Epoch 4/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0973 - accuracy: 0.4950 - val_loss: 0.0977 - val_acc
uracy: 0.4950
Epoch 5/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0925 - accuracy: 0.4950 - val_loss: 0.0914 - val_acc
uracy: 0.4950
Epoch 6/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0929 - accuracy: 0.4950 - val_loss: 0.0903 - val_acc
uracy: 0.4950
Epoch 7/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0918 - accuracy: 0.4950 - val_loss: 0.0894 - val_acc
uracy: 0.4950
Epoch 8/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0909 - accuracy: 0.4950 - val_loss: 0.0954 - val_acc
uracy: 0.4950
Epoch 9/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0896 - accuracy: 0.4950 - val_loss: 0.0876 - val_acc
uracy: 0.4950
Epoch 10/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0909 - accuracy: 0.4950 - val_loss: 0.0883 - val_acc
uracy: 0.4950
Epoch 11/20
69/69 [==============================] - 0s 3ms/step - loss: 0.0890 - accuracy: 0.4950 - val_loss: 0.0993 - val_acc
uracy: 0.4950
Epoch 11: early stopping
```

Figure 10: Final Training Epoch - 11

The model is trained to accuracy of 49.50% in the first epoch and it does not improve after that. Since the patience is set to 4, it reaches the 11th training epoch with no more minimization of the validation loss (parameter we are monitoring). Thus there is no need to proceed through all twenty epochs. The final training epoch is shown in the above picture.

2. Provide visualizations of the model's training process, including a line graph of the loss and chosen evaluation metric.

```
In [129]: #Visualize model and loss function

          plt.style.use('ggplot')

          acc = history.history['accuracy']
          val_acc = history.history['val_accuracy']
          loss = history.history['loss']
          val_loss = history.history['val_loss']
          epochs=range(len(acc))
          plt.plot(epochs, acc, 'r', 'Training Accuracy')
          plt.plot(epochs, val_acc, 'b', 'Validation Accuracy')
          plt.title('Training and Validation Accuracy')
          plt.figure()
          plt.plot(epochs, loss, 'r', 'Training Loss')
          plt.plot(epochs, val_loss, 'b', 'Validation Loss')
          plt.title('Training and Validation Loss')
          plt.figure()

Out[129]: <Figure size 432x288 with 0 Axes>
```

Figure 11: Training and Validation Accuracy/Loss

```
In [109]: #Model Eval
          loss, accuracy = model.evaluate(padded, training_labels_final, verbose = 0)
          print('Accuracy: %f' % (accuracy * 100))

          Accuracy: 49.499545
```

Figure 12: Model Accuracy

3. Assess the fitness of the model and any measures taken to address overfitting.

It appears that the model is overfitted. One sign is that the model's accuracy doesn't improve with the training epochs. EarlyStopping was used to avoid the problem of overfitting. As explained before, this method stops the model training when there is no more improvement of the parameter being monitored (validation loss), after four epochs (patience). Our model is trained up to the 11^{th} epoch.

Future iterations of this model could address the overfitting problem by using different activation functions and/or hyperparameters to achieve better accuracy.

4. Discuss the predictive accuracy of the trained network.

Accuracy is only 49%, which is far from ideal and means that the model will only be able to predict the sentiment of new reviews accurately less than half of the time.

Part V: Summary and Recommendations

E. Provide the code used to save the trained network within the neural network.

The code (***Task 2.HTML***) will be provided along with this report. Everything was developed using Jupyter Notebook.

F. Discuss the functionality of your neural network, including the impact of the network architecture.

Three data sources (Amazon, Yelp and IMDB) were combined to predict sentiments from unlabeled data. The goal was to make predictions based on positive and negative customer reviews. NLP NN was developed to use supervised learning in making those predictions. Based on the accuracy of 49%, it is not possible to make accurate predictions. This is likely a result of the size of the available dataset, and the chosen activation functions.

G. Recommend a course of action based on your results.

Since the model accuracy is only 49%, it is very hard to recommend a course of action. More data would need to be gathered in order to improve accuracy and better understand customer sentiment.

Part VI: Reporting

H. Create your neural network using an industry-relevant interactive development environment (e.g., an R Markdown document, a Jupyter Notebook, etc.). Include a PDF or HTML document of your executed notebook presentation.

A Jupyter Notebook in HTML format will be submitted with this report.

I. List the web sources used to acquire data or segments of third-party code to support the application.

[1-c] I used this page in StackOverflow to fix the issue I was having regards the download of the punk model. https://stackoverflow.com/questions/38916452/nltk-download-ssl-certificate-verify-failed

[2-c] (2021, April 18th) CHAUHAN, Amit. Understand NLP Model Building Approach with Python

J. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

[1] (2019, June 28th) CHOUDHURY, Ambika. TensorFlow vs Keras: Which one should you use? https://analyticsindiamag.com/tensorflow-vs-keras-which-one-should-you-choose/

[2] (2021, May 31st) UPADHYAY, Pratima Removing stop words with NLTK in Pyhton https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

[3] (2021, Dec 28th) NLTK Documentation: Sample Use for STEM https://www.nltk.org/howto/stem.html

[4] (2020, Oct 8th) VIRAHONDA, Sergio. An easy tutorial about Sentiment Analysis with Deep Learning and Keras

[5] (2020, Apr 2nd) CANER. Padding for NLP https://medium.com/@canerkilinc/padding-for-nlp-7dd8598c916a

[6] Natural Language Processing with Machine Learning https://www.educative.io/courses/natural-language-processing-ml/N0Wr9zwpEmv

[7] (2020, June 30th) HUSEBY, Kristin. Word embeddings, what are they really? https://towardsdatascience.com/word-embeddings-what-are-they-really-f106e1ff0874

[8] (2018, Dec 10th) BROWNLEE, Jason. Use Early Stopping to Halt the Training of Neural Networks At the Right Time https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

[9] (2017, July 3rd) BROWNLEE, Jason. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[10] (2020, Jan 30th) GUPTA, Dishashree. Fundamentals of Deep Learning – Activation Functions and When to Use Them? https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

[11] (2019, Jan 30th) BROWNLEE, Jason. How to Choose Loss Functions When Training Deep Learning Neural Networks https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[12] (2020, Aug 9th) GOSWAMI, Dr. Saptarsi. Introduction to Early Stopping: an effective tool to regularize neural nets https://towardsdatascience.com/early-stopping-a-cool-strategy-to-regularize-neural-networks-bfdeca6d722e