# Determining Advertisement vs News in Television Broadcast

UMSL CS5300 FS2021

Brian Lindemann

December 2021

"All models are wrong; some are useful."
- George Box

**Abstract**

It is important for various agents to be able to determine whether the current programming on a television broadcast is an advertisement or not. The data set used herein was created in 2014 based on five television news stations broadcasts, and is classified as Advertisement or News. In 2014, a 85% accuracy was obtained using the full data set and a Support Vector Machine (SVM) classifier. This project looks at a subset of the parameters and creates a model with a neural network implemented in Tensorflow/Keras. This report shows that the performance of the neural network on the subset of parameters performs as well as the SVM did with the full data set. Future exploration is suggested as well.

## 1 Introduction

The author has had a career involved in aerospace and communications; most recently deeply involved in the broadcast communications arena. As such, a data-set that involved broadcast data was of great interest. This specific data set is concerned with commercials on television news broadcasts. The broadcast company that the author was involved with sold a rudimentary product to allow broadcasters to 'prove' airing of advertisements (commercials). While the viewing audience sees commercial breaks as nuisances, the broadcaster sees them as their revenue stream.

Since a broadcaster only gets paid when advertisements are running, it is important for broadcast organizations to be able to know at any point in time whether the programming airing currently is an advertisement or regular programming. This information can be used for reporting, recording, or for operational purposes. In the past there has been human interface required for tracking, or reliance on other programming automation interfaces to make the determination. With an Artificial Intelligence element the reporting can be passively ascertained and reported.

The project was broken into various phases as the final model was generated. This report is prepared in chronological order.

For this project, a data set of more than 100,000 samples that was created in 2014 is utilized. In the 2014 paper [1] that was the genesis of this data set, the team reported a generic precision of 83.33%, recall of 89.73% for an F-score of 86.4% before post processing. The post processing the original team performed consisted of further analysis of the data collected including the time duration of the programming and the blocking of the programming. This post processing is not utilized in this current work.

Compared to the original work, this project made use of no post-processing, and only 18 of the original 4,125 features.

All work presented in this paper was been done by the author (Brian Lindemann) individually. The Tensorflow processing was done utilizing Jupyter Notebook running on a Alienware Aurora computer with an Nvidia GPU graphics card installed. This actually proved to be a factor of two times faster than running the same notebook on Google's free version of Colab.

[1] "Commercial Block Detection in Broadcast News Videos", https://dl.acm.org/doi/proceedings/10.1145/2683483

## 2 Data Collection and Initial Processing

### 2.1 Data Set Selection

The data-set being used is available from the UC Irvine Machine Learning Repository [2], and is entitled "TV News Channel Commercial Detection Dataset".

The specific time that this data set was created is not specified, however it was utilized in the publication of the aforementioned paper in 2014, and was uploaded to the UCI repository in 2015.

### 2.2 Data Set Information

The data in the downloadable file is broken into five separate files, one for each of five international news television stations: BBC, CNN, CNNIBN, NDTV and TIMESNOW. During certain phases of the project, a subset of the data was utilized, for instance for initial processing and understanding of the data, only the data from TIMESNOW was used. For the final project, the complete sample set from all five was used.

The original data set used also included a so-called "word list". Though it was referred to as a word list, it is actually a set of Mel Frequency Cepstral Coefficients (MFCC) further describing the received signal. These were not used in this classification project. (See Future Work at the end of this report.)

These fields are stripped out for the purposes of this project, leaving only numerical attribute and binary classification columns in the data.

A separate parser was written in Python to convert the data from the format that was provided in the data set into the Comma Separated Value (csv) format that is used for the rest of the project.

In the original data files, the format of a typical line that looks like (there is no line break in the data line):

```
1  1:29 2:3.821209 3:1.567568 4:13.547628 5:7.242389 6:0.019883
7:0.012195 8:0.067241 9:0.049107 10:3406.866211 11:1363.990601
12:6796.552246 13:2719.626709 14:1021.359192 15:940.742371
16:102.607803 17:60.178288 18:0.538928 19:0.038365 20:0.020429
21:0.016987 22:0.016315 23:0.015634 24:0.015342 25:0.015516
26:0.014319 27:0.013871 28:0.012144 29:0.012063 30:0.011293
31:0.010508 32:0.009797 33:0.008709 34:0.007828 35:0.007598
36:0.006682 37:0.005798 38:0.005481 39:0.005144 40:0.004751
41:0.004803 42:0.004613 43:0.003977 44:0.004036 45:0.003770
46:0.003425 47:0.003060 48:0.002849 49:0.002399 50:0.001995
51:0.001910 52:0.001493 53:0.001131 54:0.000998 55:0.000800
56:0.000574 57:0.000565 58:0.528130 59:0.125689 60:0.043599
61:0.034053 62:0.019172 63:0.013895 64:0.012414 65:0.011797
66:0.010213 67:0.007679 68:0.005947 69:0.005660 70:0.005088
71:0.004640 72:0.004240 73:0.003481 74:0.003246 75:0.003195
76:0.003175 77:0.003153 78:0.003315 79:0.004922 80:0.004800
81:0.000528 82:0.000033 83:0.000004 84:0.000000 90:0.003973
91:0.007881 92:0.009475 93:0.165708 94:0.082617 95:0.075356
96:0.329906 97:0.121332 98:0.081927 99:0.074856 100:0.010632
101:0.020546 102:0.263981 103:0.354996 104:0.384623 105:0.015420
106:0.018685 107:0.023046 108:0.173355 109:0.094493 110:0.095674
111:0.236807 112:0.098715 113:0.144719 114:0.231744 115:0.044777
116:0.108719 117:0.137887 118:0.192764 119:0.187372 296:0.208333
448:0.020833 491:0.010417 572:0.052083 573:0.145833 580:0.479167
762:0.104167 816:0.020833 4124:0.813823396832 4125:0.397715407997
```

The first number (a 1 in the line above) is the classification value. The data set has 1 for "is a Commercial" or "Advertisement" and $-1$ for "not a commercial" or "News". The rest of the line of data is formatted with attribute number followed by a colon followed by the attribute for each of the remaining attributes.

There are 4,125 attributes in the data set, but not all are given on every line. This is because the data set utilizes bins for certain attributes. These binned attributes are not considered in this project and are referenced below as "UNUSED".

The attributes saved for further processing are:

[2] https://archive.ics.uci.edu/ml/machine-learning-databases/00326/TV_News_Channel_Commercial_Detection_Dataset.zip

```
1 - Shot Length: SL*
2 - Motion Distribution Mean: MD-M
3 - Motion Distribution Varaince: MD-V
4 - Frame Difference Distribution Mean: FDD-M
5 - Frame Difference Distribution Variance: FDD-V
6 - Short Time Energy Mean: STE-M
7 - Short Time Energy Variance: STE-V
8 - Zero Crossing Rate Mean: ZCR-M
9 - Zero Crossing Rate Variance: ZCR-V
10 - Spectral Centroid Mean: SC-M
11 - Spectral Centroid Variance: SC-V
12 - Spectral Roll Off Mean: SR-M
13 - Spectral Roll Off Variance: SR-V
14 - Spectral Flux Mean: SF-M
15 - Spectral Flux Variance: SF-V
16 - Fundamental Frequency - Mean: FF-M
17 - Fundamental Frequency - Variance: FF-V
18 to 58: Motion Distribution on the screen. UNUSED
59 to 91: Frame Difference Distribution UNUSED
92 to 122: Text Area Distribution UNUSED
123 to 4123: Bag of Words MFCC coefficients UNUSED
4124 - Edge Change Ratio Mean: ECR-M
4125 - Edge Change Ratio Variance: ECR-V
```

* Though Shot Length was saved, it was not be used in this classification process. It has a wild difference in lengths and it appears to do with how the data was gathered more than the actual data gathered. This parameter may have been originally used in the secondary post processing done by the original team.

After processing, the data attributes are as shown in Table 1.

One interesting thing about the data is that it includes means and standard deviations of measurements as attributes. So, for instance the line corresponding to MD-M is the mean of motion distribution; and the column for "mean" is the average of all mean motion distributions for the data set. It should be noted that the mean for "Commercial" is 0.64 - which corresponds to 64% of the samples in the TIMESNOW data set being advertisements (1).

| | Maximum | Minimum | Mean | Median |
|---:|---:|---:|---:|---:|
| SL : | 17729.00 | 25.00 | 104.93 | 48.00 |
| MD-M : | 15.75 | 0.01 | 2.61 | 2.03 |
| MD-V : | 37.36 | 0.00 | 1.62 | 1.37 |
| FDD-M : | 67.29 | 0.22 | 12.48 | 10.03 |
| FDD-V : | 63.40 | 0.07 | 8.72 | 7.04 |
| STE-M : | 0.04 | 0.00 | 0.02 | 0.02 |
| STE-V : | 0.02 | 0.00 | 0.01 | 0.01 |
| ZCR-M : | 0.39 | 0.01 | 0.11 | 0.10 |
| ZCR-V : | 0.21 | 0.01 | 0.06 | 0.06 |
| SC-M : | 4005.92 | 957.71 | 3594.32 | 3618.70 |
| SC-V : | 1669.64 | 67.91 | 1056.09 | 1098.64 |
| SR-M : | 7821.28 | 5861.00 | 7054.40 | 7094.83 |
| SR-V : | 2931.56 | 206.83 | 2051.33 | 2131.48 |
| SF-M : | 8821.18 | 0.00 | 1152.39 | 1129.08 |
| SF-V : | 7423.56 | 0.00 | 1429.77 | 1384.21 |
| FF-M : | 186.24 | 55.39 | 118.30 | 118.99 |
| FF-V : | 83.01 | 15.93 | 50.18 | 49.79 |
| ECR-M : | 1.00 | 0.00 | 0.50 | 0.50 |
| ECR-V : | 1.00 | 0.00 | 0.50 | 0.50 |
| Commercial : | 1.00 | 0.00 | 0.64 | 1.00 |

Table 1: Features Numerical Analysis (TIMESNOW)

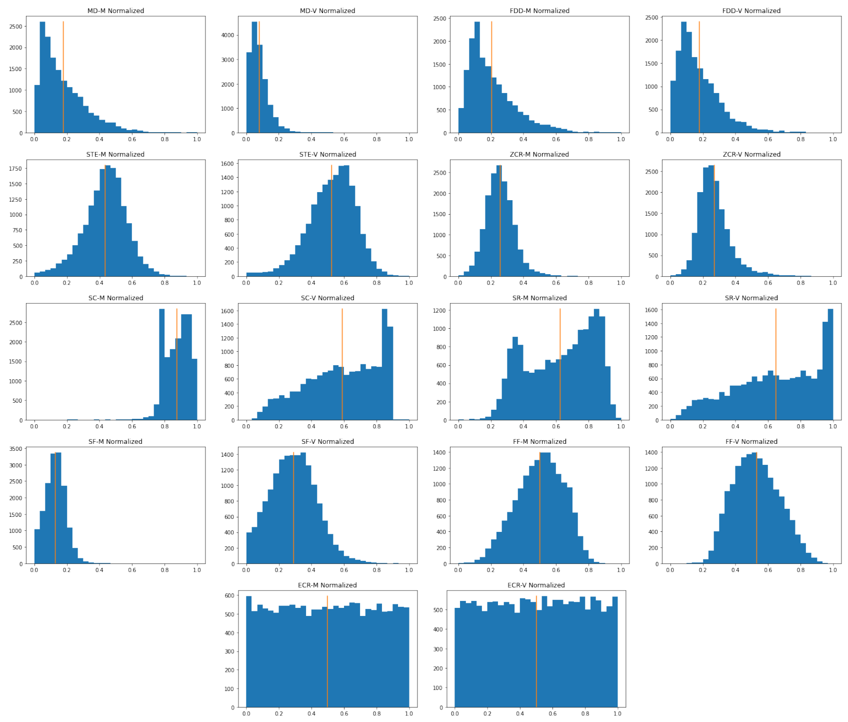

Figure 1: Feature Distribution Graphs

## 2.3 Data Set Normalization

Based on the initial look at the data, it was decided that normalization should be applied. Some of the data ranges from zero to a relatively small number (e.g. MD, FDD, ZCR), but other range from large numbers to larger numbers (e.g. SC, SR, SF, FF). A process of min-max normalization should was used to put all data into the 0 - to - 1 range.

$$x'_i = \frac{x_i - min}{max - min}$$

## 2.4 Data Set Visualization

The data set after normalization is shown in the graphs in figure 1.

## 2.5 Initial fitting

A basic model was generated to do a logistic fitting of the data. This model was run against 8000 lines of data selected from the TIMESNOW data set, where the first 4000 Advertisements and first 4000 News segments were selected from the set. This gave a 50% initial likelihood for the data and eliminates the skew in the original data caused by the 64% Advertisement rate that was representative in the file.

After 400 epochs, the training loss was 0.4597 and the training accuracy was 0.8019.

From a visualization standpoint, the two graphs in figure 2 show what the comparison of the predictions to the actual classifications using the training data for predictions. The x-axis is the value output from the logistic function with the threshold being 0.5 between Advertisement and News (Commercial vs NonComm respectively in the figure.) The orange represents the miss-classified data samples. About 80% of the combined bar chart is blue; 20% orange.

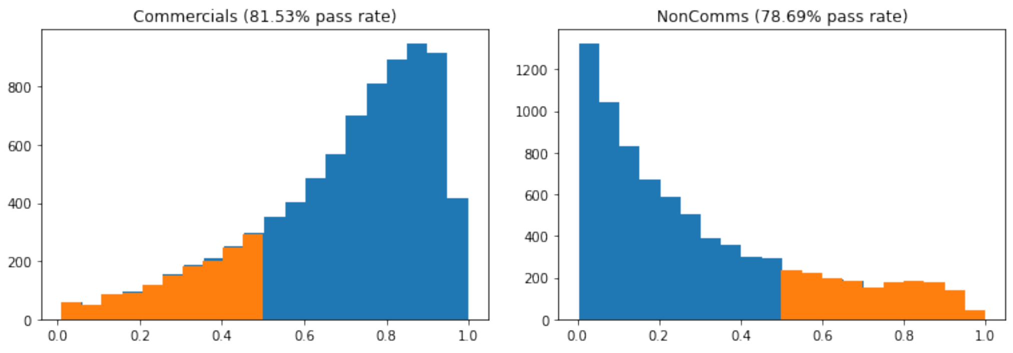

Figure 2: Initial Fitting

## 3 Neural Network Sizing

Experimentation was performed to determine a good size for the neural network to perform classification.

This was done by making a model that was large enough to completely overfit the data, and then trimming back until overfitting no longer occurred.

Within this report, shorthand is used to show the neural network topography. All networks utilize "Dense", Rectified Linear Unit activation for the first nodes, and have a single sigmoid activation node for the final output to classify the data between "Advertisement" and "News". The shorthand for these is x:y:z:1 - where the x, y and z (and others as needed) represent the number of neurons at each layer. Given that they are "Dense", the first layer neurons that the raw data feeds into have 19 parameters each (18 X's and one bias) and subsequent neurons are constrained to have one parameter for each neuron in the prior layer plus a bias parameter. So, in the case of a network made up of 8 input neurons, 4 subsequent and a single sigmoid output, the shorthand used herein is 8:4:1, and the number of parameters in that network is $8 * (18 + 1) + 4 * (8 + 1) + 1 * (4 + 1) = 193$ parameters.

The model selected to move forward with was 125:125:2:1 based on the over-fitting criteria.

### 3.1 Initial Logistic Classification

The initial starting point in the data over-fitting process is a single neuron with a logistic (sigmoid) activation.

The neural network configuration is shown in Table 2.

| Layer (type) | Output Shape | Param Count |
|---|---|---|
| dense_13 (Dense) | (None, 1) | 19 |
| Total params: 19 | | |
| Trainable params: 19 | | |
| Non-trainable params: 0 | | |

Table 2: Single Logistic Element Model Structure

This single neuron gave a surprising graph. To get the divergence of data over time, the .fit() command was run using the validation_data parameter. The history was then plotted. In each of the plots in this report, the training accuracy is given in blue and the validation accuracy is given in red. At the start, the training and validation accuracy are both close to 64%, as would be expected from a random situation with this distribution of positives. See figure 3.

The validation accuracy tracked at a consistent 1 percentage point lower than the training accuracy. The 75% accuracy on the training data gave some hope that this overall experiment was going to succeed - that is, that the data set without the word lists would have enough information to be able to classify a given sample as "Advertisement" or "News". This hope was balanced against the fact that 75% is only 11 percentage points better than the nominal of 64% in the data set.

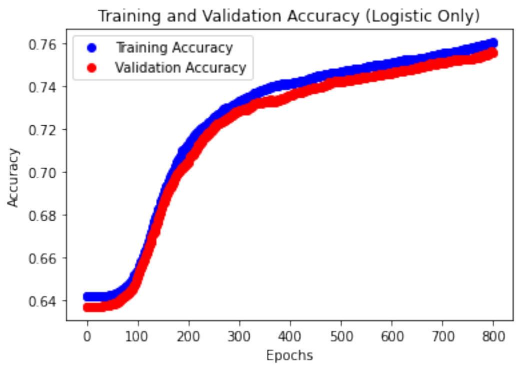

Figure 3: Logistic Only Performance

## 3.2 First Deeper Neural Layer

The next step in determining what the minimum set would be to give good over-fitting was to add a single layer with two neurons utilizing Rectified Linear Unit activation. In shorthand a configuration of 2:1. This was a minimal step just to get a feel for where the project needed to go.

For this, a length of 800 epochs was chosen, in keeping with the initial single Logistic element.

The model summary for this is shown in Table 3.

| Layer (type) | Output Shape | Param Count |
|---|---|---|
| dense_24 (Dense) | (None, 2) | 38 |
| dense_25 (Dense) | (None, 1) | 3 |
| Total params: 41 | | |
| Trainable params: 41 | | |
| Non-trainable params: 0 | | |

Table 3: Two Element Model Structure

The results were similar to the initial single Logistic implementation, in that there wasn't much difference between the accuracy of the training data and the accuracy of the validation data. However, in those 800 epochs, the accuracy achieved was slightly higher - around 80% compared 75% as shown in figure 4.

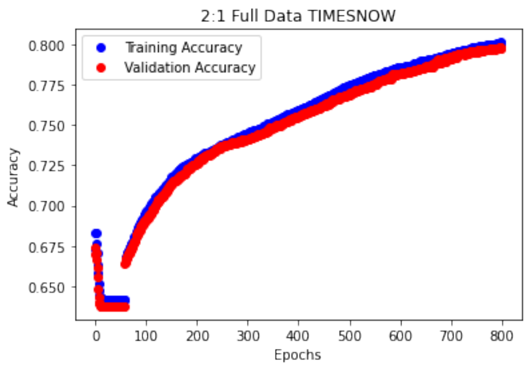

Figure 4: Two Layer Performance

An interesting artifact was noted in this sparse network: the initial training accuracy was at 67%, but subsequently dropped each epoch, bottoming out at 64.2% from epochs 13 through 59, at which point the accuracy began to climb.

## 3.3 Second Deeper Neural Layer

As a method to zero in on a good size, it was decided to "go big" and set a model that had a number of parameters that was close to the total number of parameters in the training data. The TIMESNOW training data has 18 x-values per sample, and 27,447 samples for a total of 494,586 total parameters. That seemed a bit high so something more reasonable, but still big was used. In shorthand: 250:250:250:250:250:1 - with a total of 256,001 parameters as shown in Table 4. This was Run for 400 epochs.

| Layer (type) | Output Shape | Param Count |
|---|---|---|
| dense_26 (Dense) | (None, 250) | 4750 |
| dense_27 (Dense) | (None, 250) | 62750 |
| dense_28 (Dense) | (None, 250) | 62750 |
| dense_29 (Dense) | (None, 250) | 62750 |
| dense_30 (Dense) | (None, 250) | 62750 |
| dense_31 (Dense) | (None, 1) | 251 |
| Total params: 256,001 | | |
| Trainable params: 256,001 | | |
| Non-trainable params: 0 | | |

Table 4: Many Large Elements Model Structure

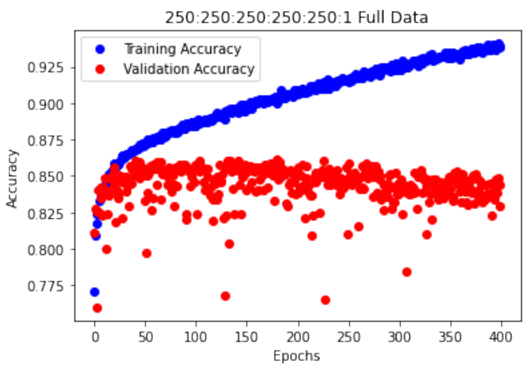

Figure 5: Many Large Elements Network Performance

This model got the training accuracy above 90% within 400 epochs with a batch size of 128. But, it is rather large and obviously prone to over-fitting. So with the two bounds in place, experimentation was performed on the number of layers and the number of neurons per layer.

## 3.4 Finding The Balance

With an upper bound of five levels of 250 neurons each followed by a sigmoid and a lower bound of one level of 2 neurons followed by a sigmoid, experimentation was performed to find the balance.

As shown in Figure 6 numerous different configurations were looked at.

Note: Not all sub-figures, are at the same scale.

All except one model showed promise - the 250:2:1 model is the exception.

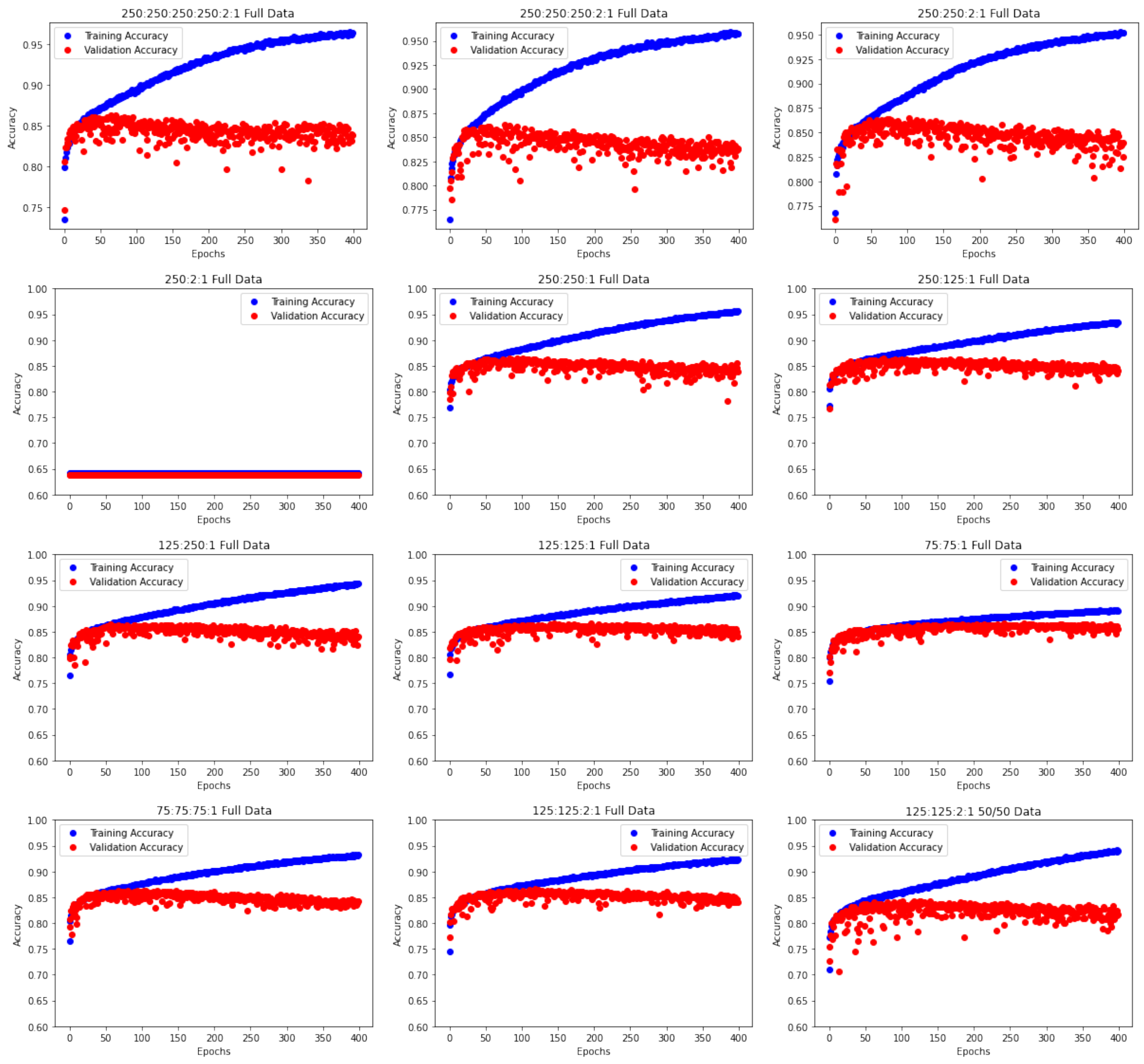

Figure 6: Various Configurations Performance

# 4 Detailed Analysis

Based on experimentation the model 125:125:2:1 appeared to over-fit the fastest. This model has a total of 18,380 parameters in 4 layers (one input, two hidden and one sigmoid output.) This model gave 85% accuracy for the validation data.

Experimentation was performed to determine whether there was a smaller model that gave similar validation accuracy. As shown in Section 4.1 experimentation was used to draw this down to 75:75:1 with a total of 7,201 parameters.

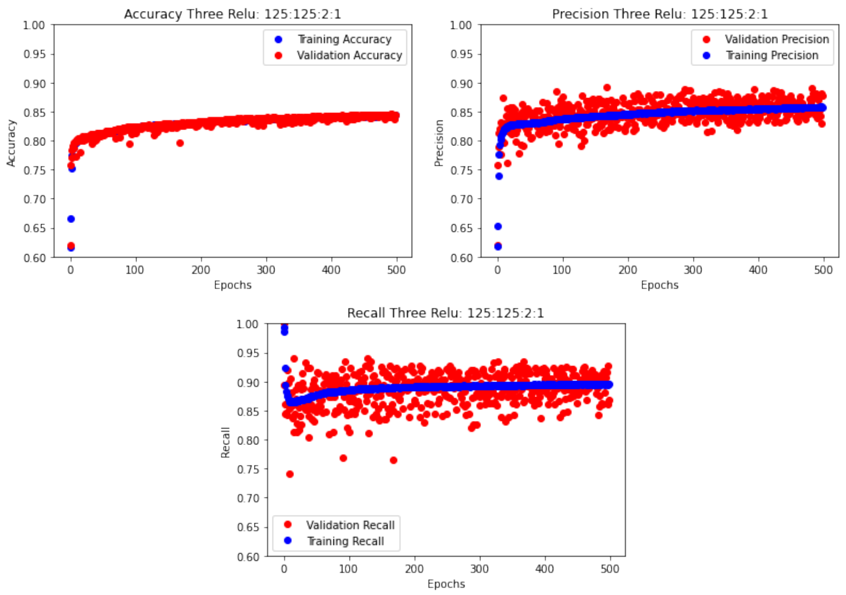

Figure 7: 125:125:2:1 Performance

## 4.1 Model Determination

Several criteria were used to get the final model for this phase - including now looking at precision and recall.

The measure of precision says of all the data samples that the model says are Advertisements, how many actually are Advertisements. At the extreme, a model that classifies one single Advertisement correctly and does not mis-classify any News samples would have a precision of 100% - that is every item it classified as an Advertisement was indeed an Advertisement. Though it might miss many Advertisements. ("How many samples classified as Advertisements are Advertisements?")

The measure of recall is a ratio of the number of items that were classified as Advertisements compared to the total number of Advertisements that were in the sample set. ("How many Advertisements were actually classified as Advertisements?") The performance of the 125:125:2:1 model is seen in Figure 7. By comparison, the accuracy, recall and precision is seen in Figure 8. Table 5 shows the accuracy, precision and recall after 500 epochs with the two different sized models. Based on the remarkable similarity, the 75:75:1 model was chosen.

## 4.2 Longer Training

The next step was to train the model longer and stop after a 'reasonable' time has passed rather than training for ever. To define 'reasonable', an early-stopping callback routine was implemented. With everything else left the same, the fit function was allowed to run up to 10,000 epochs. As shown in the listing, the stopping parameter was validation accuracy, with a patience value of 200. The best model was saved using the ModelCheckpoint function, again based on validation accuracy. So, the model with the best validation accuracy is kept.

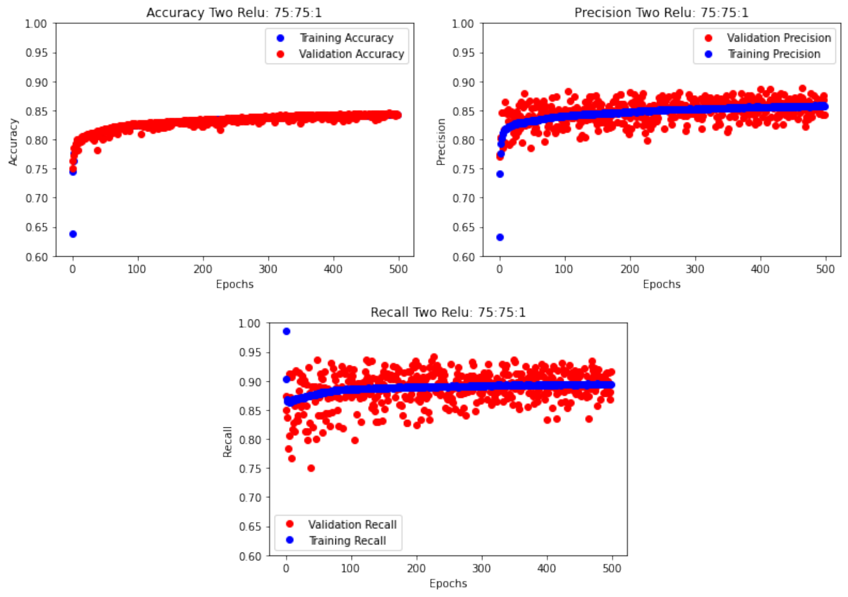

Figure 8: 75:75:1 Performance

| Size | Train Accuracy | Validation Accuracy |
|---|---|---|
| 125:125:2:1 | 84.40 | 84.37 |
| 75:75:1 | 84.31 | 84.28 |
| **Size** | **Train Precision** | **Validation Precision** |
| 125:125:2:1 | 85.78 | 87.75 |
| 75:75:1 | 85.91 | 85.60 |
| **Size** | **Train Recall** | **Validation Recall** |
| 125:125:2:1 | 89.62 | 86.88 |
| 75:75:1 | 89.26 | 89.68 |

Table 5: Model Size Parameters

Validation accuracy was used rather than validation loss because that is the end number that is being chased. Due to the large size of the data set the validation accuracy is a reasonably variable number. That is, if there were only 100 samples in the set, validation accuracy could be 0.89 or 0.90 but never 0.895. With more than 80,000 samples in the training set, the validation accuracy can be quite smooth.

The training stopped after an additional 679 epochs, with a final validation accuracy of 86.10% and a validation precision of 88.28% as shown in figure 9.

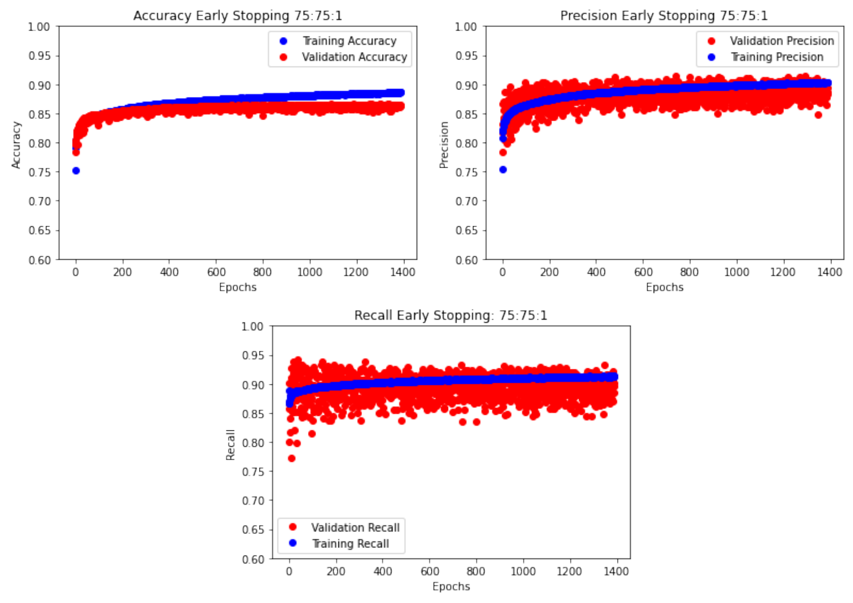

Figure 9: 75:75:1 With Early Stopping

## 5 Feature Reduction

With a model that provides a reasonable result, the model was looked at to determine if the number of features could be decreased.

To perform this, the model that was settled was used, but set to have a single feature input rather than 18 inputs. The model was then run a modest number of epochs; 50 was used on as a good point to show differentiation between features.

See section 2.2 for the definitions of the features, and the names used herein.

The top image in Figure 10 shows the model run with a single feature at a time. On that bar graph, the horizontal line shows the 63% level of samples that are Advertisements within the training set. (That is, if the model just said "Advertisement" it would be right 63% of the time.)

The two features that fall right at the line of having no impact on the model are "ECR-M" and "ECR-V". Refer to Figure 1 for the grpahs of the input information - these two features appear to be a uniform distribution. Indeed, the maximum, minimum, mean and median are 1.0, 0.0, 0.5 and 0.5 for both of them. The fact that they show basically no impact on the model suggest that they are indeed uniform random distributions with no correlation to whether the broadcast is News or Advertisement.

It is interesting that the Short Time Energy Mean has little affect on the model while Short Time Energy Variance has a major impact.

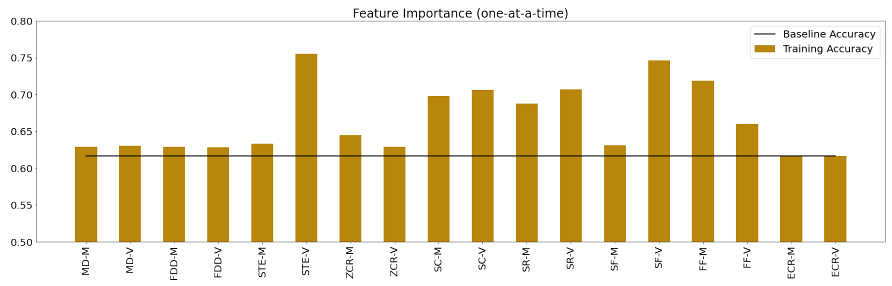

Figure 10: Feature Importance

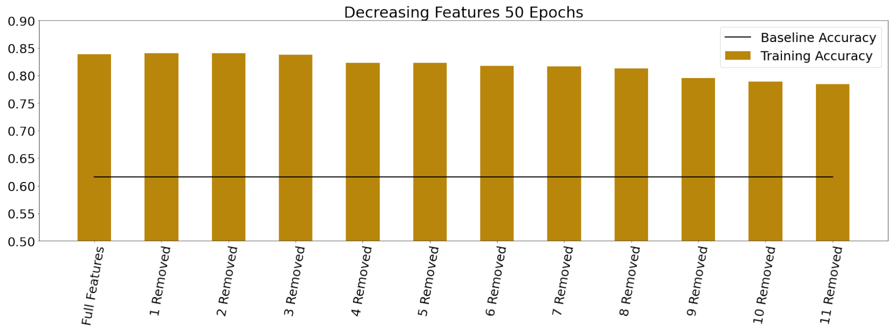

Figure 11: Effect of Removing Features

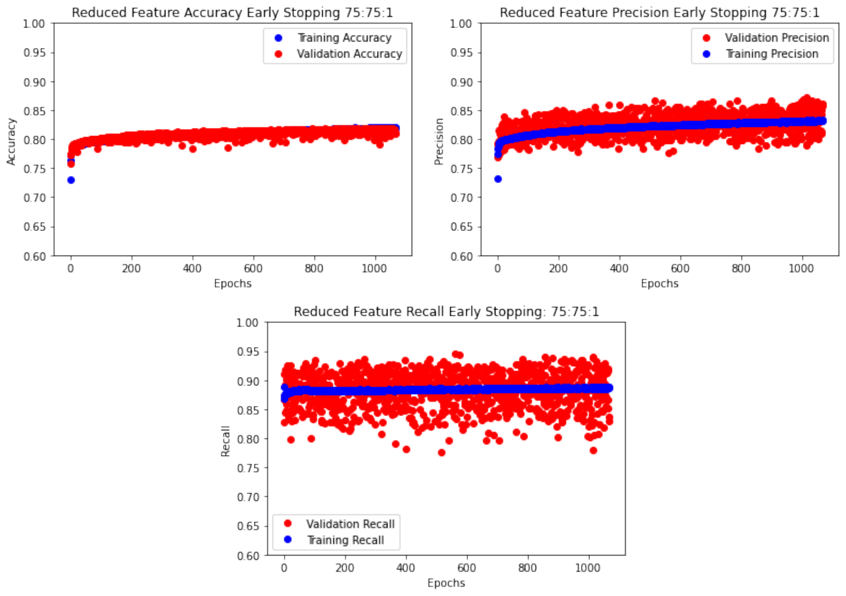

Figure 12: Reduced Features Full Run With Early Stopping

The features were removed in the following order, and the effect plotted as shown in Figure 11.
1) Feature ECR-M
2) Feature ECR-V
3) Feature FDD-V
4) Feature FDD-M
5) Feature ZCR-V
6) Feature MD-M
7) Feature MD-V
8) Feature SF-M
9) Feature STE-M
10) Feature ZCR-M - this is the one where it could get interesting
11) Feature FF-V - this might have a profound affect  After removing those the top 6 of those features - reducing the feature set to just 12 features - the full model was run with maximum of 10,000 epochs using early stopping as done during model selection.

As can be seen in Figure 12 compared to Figure 9, the accuracy has suffered only a small amount - going from 86.35% down to 84.89% - a change of 1.5 percentage points.

## 6 Conclusion

This project determined that given a data set from 2014 which contained over 4,000 features used to classify a particular broadcast as “News” or “Commercial”, only 12 features were necessary for a neural network implemented in Python utilizing TensorFlow/Keras to achieve results that are similar to those found using a Support Vector Machine construct.

For a future project, it would be interesting to look at the MFCC elements contained within the data set and determine whether these could make a stronger model than what is presented herein.