CORD-19 MeSH Label Classification
Open-Ended Machine Learning
Sabrina Peng, Morgan VandenBerg

Problem Overview

The global pandemic caused by COVID-19 has marked a significant event in history and altered the way people think, act, and go about daily life. Those of us studying computer science can contribute towards academic efforts aimed at fighting the pandemic. In this program, we use natural language processing and machine learning techniques on the CORD-19 (COVID-19 Open Research Dataset), which is a collection of medical research articles consisting of an abstract and full text.

The specific problem to be solved in this program is the classification of paper topics by medical subject heading (MeSH) labels, which describe what broad topic a certain paper covers. Papers can have one or more MeSH labels, and our goal is to assign MeSH labels to new articles using supervised learning and training on a labelled corpus of existing research articles. This research can be particularly useful in finding articles that may be relevant to current and rapidly evolving scientific research on COVID-19, thus allowing scientists to extract and apply valuable information found to the current situation.

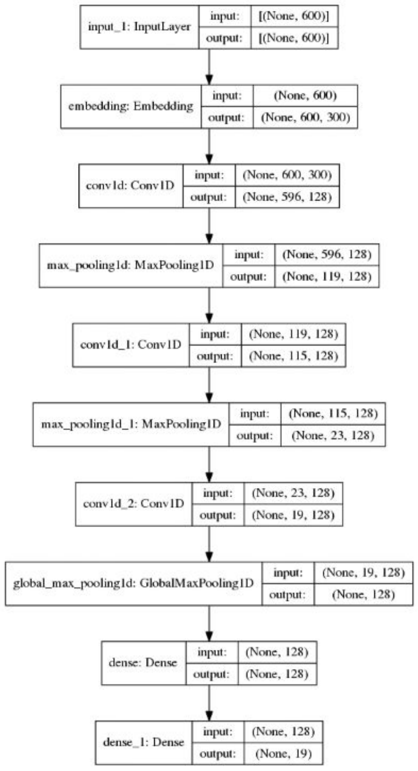
Classification Methods

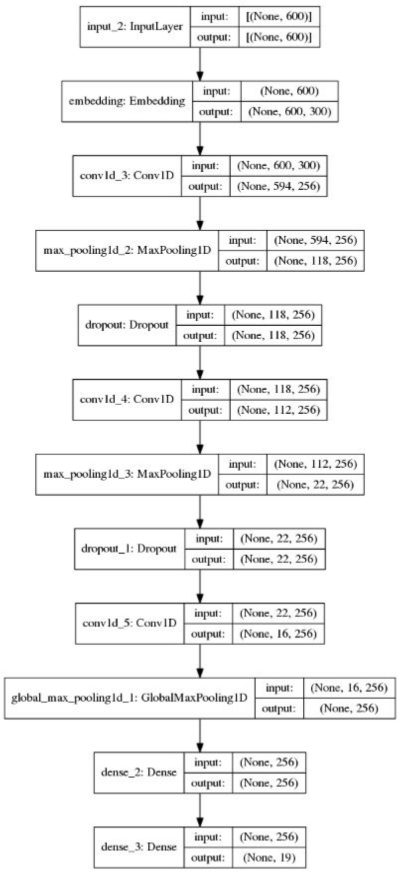
The approach used in this program was exploring the use of recurrent neural networks and convolutional neural networks in executing a range of classification methods. Recurrent neural networks are artificial neural networks that are designed to recognize patterns in sequences of data, taking time and sequence into account and storing information about past inputs in order to add a temporal dimension to learning and results. Specifically, LSTM (long short-term memory) and GRU (gated-recurrent-unit) RNNs were the types of RNN chosen, as they have been proven to be well-suited for text classification and overcome the vanishing and exploding gradient problems experienced by traditional RNNs. Convolutional neural networks are artificial neural networks that are typically used to classify two-dimensional image data, but can also be used with one-dimensional data such as text documents. These neural networks include convolutional layers, which multiply input data by a two-dimensional array of weights (called a filter or a kernel) in order to distill features from the data. CNNs were chosen as an approach because the convolutions mimic RNN behavior and the neural networks can be run in parallel, resulting in much faster training than for RNNs. The machine learning and deep learning libraries used to carry out classification using these deep neural networks include *scikit-learn*, *tensorflow*, and *keras*.

We trained five models in this experiment: a small CNN, a medium CNN, a large CNN, a LSTM RNN, and a GRU RNN. Note that the “size” of the convolutional networks were assigned based on the dimension of the kernel and convolutions used in their architecture.

These architectures were chosen in part based on models proposed in the Keras documentation for IMDB sentiment classification. Their hyperparameters were picked largely arbitrarily in order to explore performance differences across a range of architectures; in the future, tuning could be performed to obtain a more accurate model.

All five architectures are visualized below, and can be viewed in more detail in the ModelComparisons notebook or the PA03 notebook.


input_1: InputLayer | input: [(None, 600)] | output: [(None, 600)]
embedding: Embedding | input: (None, 600) | output: (None, 600, 300)
conv1d: Conv1D | input: (None, 600, 300) | output: (None, 596, 128)
max_pooling1d: MaxPooling1D | input: (None, 596, 128) | output: (None, 119, 128)
conv1d_1: Conv1D | input: (None, 119, 128) | output: (None, 115, 128)
max_pooling1d_1: MaxPooling1D | input: (None, 115, 128) | output: (None, 23, 128)
conv1d_2: Conv1D | input: (None, 23, 128) | output: (None, 19, 128)
global_max_pooling1d: GlobalMaxPooling1D | input: (None, 19, 128) | output: (None, 128)
dense: Dense | input: (None, 128) | output: (None, 128)
dense_1: Dense | input: (None, 128) | output: (None, 19)


input_2: InputLayer | input: [(None, 600)] | output: [(None, 600)]
embedding: Embedding | input: (None, 600) | output: (None, 600, 300)
conv1d_3: Conv1D | input: (None, 600, 300) | output: (None, 594, 256)
max_pooling1d_2: MaxPooling1D | input: (None, 594, 256) | output: (None, 118, 256)
dropout: Dropout | input: (None, 118, 256) | output: (None, 118, 256)
conv1d_4: Conv1D | input: (None, 118, 256) | output: (None, 112, 256)
max_pooling1d_3: MaxPooling1D | input: (None, 112, 256) | output: (None, 22, 256)
dropout_1: Dropout | input: (None, 22, 256) | output: (None, 22, 256)
conv1d_5: Conv1D | input: (None, 22, 256) | output: (None, 16, 256)
global_max_pooling1d_1: GlobalMaxPooling1D | input: (None, 16, 256) | output: (None, 256)
dense_2: Dense | input: (None, 256) | output: (None, 256)
dense_3: Dense | input: (None, 256) | output: (None, 19)

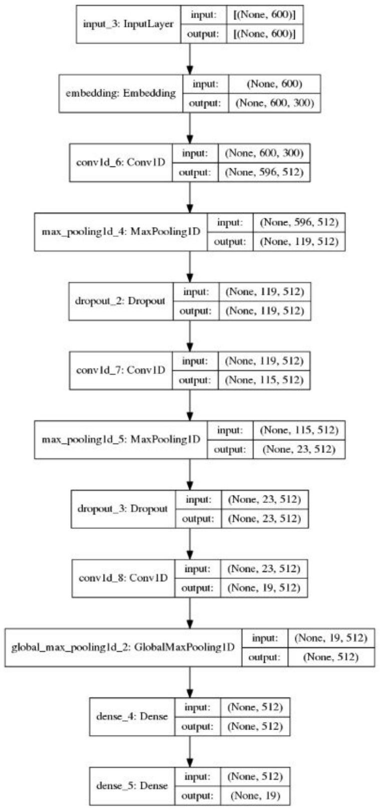

input_3: InputLayer | input: [(None, 600)] | output: [(None, 600)]
embedding: Embedding | input: (None, 600) | output: (None, 600, 300)
conv1d_6: Conv1D | input: (None, 600, 300) | output: (None, 596, 512)
max_pooling1d_4: MaxPooling1D | input: (None, 596, 512) | output: (None, 119, 512)
dropout_2: Dropout | input: (None, 119, 512) | output: (None, 119, 512)
conv1d_7: Conv1D | input: (None, 119, 512) | output: (None, 115, 512)
max_pooling1d_5: MaxPooling1D | input: (None, 115, 512) | output: (None, 23, 512)
dropout_3: Dropout | input: (None, 23, 512) | output: (None, 23, 512)
conv1d_8: Conv1D | input: (None, 23, 512) | output: (None, 19, 512)
global_max_pooling1d_2: GlobalMaxPooling1D | input: (None, 19, 512) | output: (None, 512)
dense_4: Dense | input: (None, 512) | output: (None, 512)
dense_5: Dense | input: (None, 512) | output: (None, 19)

Figure 1.1: *Small CNN* ***Figure 1.2:*** *Medium CNN* ***Figure 1.3:*** *Large CNN*

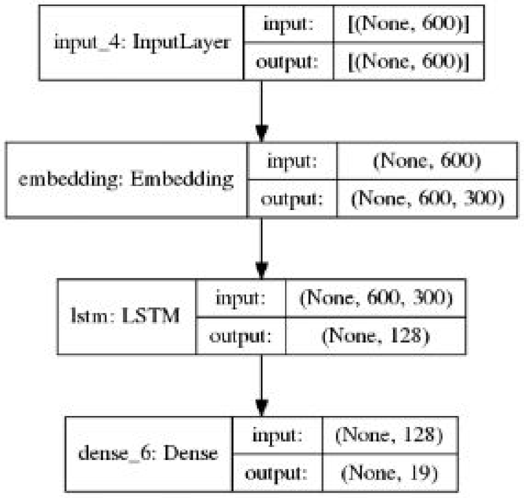


Figure 1.4: *LSTM RNN*

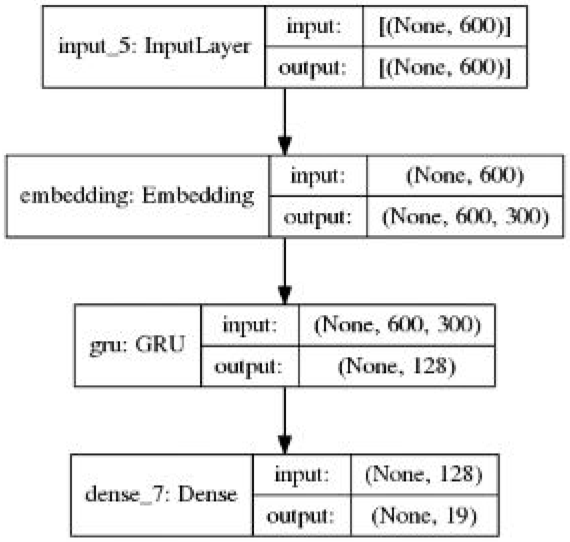


Figure 1.5: *GRU RNN*

Proposed Wrinkle

For our wrinkle, we propose a form of text splicing inspired partially by image recognition augmentation techniques.

In image recognition tasks, train samples are often modified (with minor rotations, hue / color adjustments, or luminance adjustments) to generate "new" training data that mirrors real-world use scenarios. It is known that this can significantly increase generalized model performance, particularly in cases where only a small training set is available.

We propose that the given text samples of the same class can be spliced together to form new articles in a similar fashion. While it should be noted that this combines train samples (and is therefore fundamentally different from image augmentation), we believe that it may improve model performance. In doing this, we are training the model with new, unique texts that are still known to be of a certain class, but contain content that is different from other samples.

We will attempt to splice the texts together in two ways, proposed as follows:

- **Full-Splice:** form new documents using ten random sentences from unique articles of each given tag.
- **Half-Splice:** form new documents by splicing together two halves of articles that have the same tag.

We will include the augmented / wrinkle data only in the training set, as it should not be considered to be fully-valid, real-world data.

Data Labeling Note

Some of the corpus articles have multiple MeSH tags, as defined in the metadata CSV file. We chose to perform all experimentation presented here with only the primary tag (derived from the file name). This was in response to the preliminary results which we presented on April 28th, in which a "many-to-many", "many-to-one", and a "duplicated one-to-one" methodology was tested.

Note that these methods were defined as follows:

- **One-To-One (used for all further tests):** use primary MeSH tag to predict a primary MeSH tag
- **Many-To-Many:** use all training MeSH tags to predict one or multiple MeSH tags on the test data
    - **Any Match metric:** a "correct" classification has at least one of the predicted tags
    - **All Match metric:** a "correct" classification requires exactly all of the predicted tags
- **Many-To-One:** use all training MeSH tags to predict one primary MeSH tag on the test data
- **Duplicated One-To-One:** create single-tag copies of documents for every associated MeSH tag, then train using a one-to-one approach.

We found that, using the same architectures presented here, the small CNN yielded the following results:

	One to One	**Many to One** (standard)	**Many to One** (duplicated 1:1)	**Many to Many** (any match)	**Many to Many** (all match)
Accuracy	0.82	0.04	0.65	0.72	0.60

Given the vast discrepancy between accuracy values observed here, we chose to only use the one-to-one approach for the rest of the project.

Preprocessing

Each text was considered as the combination of the abstracts and full texts from each document.

Past research has suggested that neural network models learn best from data that is (largely) in its raw form, with minimal preprocessing. For the sake of completeness, this report will explore results on an unprocessed dataset, as well as one with English NLTK stopwords removed and lemmatization on all terms via the WordNetLemmatizer.

Using deep learning techniques such as RNNs and CNNs requires that data be transformed such that all text sequences have the same length. In order to accomplish this, the limit for truncation and padding was determined by plotting the distribution of word counts in all documents. Documents were truncated to approximately 600 words, which allowed the vast majority of the text in the corpus to be retained. Only approximately 0.1% to 0.65% of the documents (depending on preprocessing) were truncated, leading us to believe that the 600-word truncation boundary is an effective splitting bound. All text sequences were padded with 0 values or truncated to 600 words to ensure the data could be used in our chosen deep learning algorithms.

Google News 300-dimension pre-trained word vectors were used in order to form embeddings and represent terms in the texts of the corpus.

A 80/20 stratified train/test split was used on each corpus in order to separate training and validation data for use with our deep learning methods.

Our chosen evaluation metric for the effectiveness of our classifier was the accuracy of assigned labels on the validation data set. Though the distribution of documents across all MeSH labels was not uniform, the non-uniformities only affect two out of the nineteen available MeSH labels, and the stratified split will be able to negate the small effect that this non-uniformity might have on our results. Therefore, though the F1 score tends to be a better metric when imbalance classes exist, accuracy can still be used as a valid evaluation metric for the purposes of this program.

Results

Analysis Methodology

To compare our network models, we will use a McNemar test. This allows us to statistically evaluate if there is a significant difference in the proportion of errors made by the classifier. In addition, this does not require us to retrain the model *n* times for a sample size and set of metrics, saving significant training time and computational power.

We will define a helper function to calculate the McNemar test statistic and then perform hypothesis testing with a chi-squared statistic and one degree of freedom. The hypotheses and test statistic are given as follows:

$$\begin{aligned} H_0 &: p_b = p_c \\ H_1 &: p_b \neq p_c \end{aligned}$$

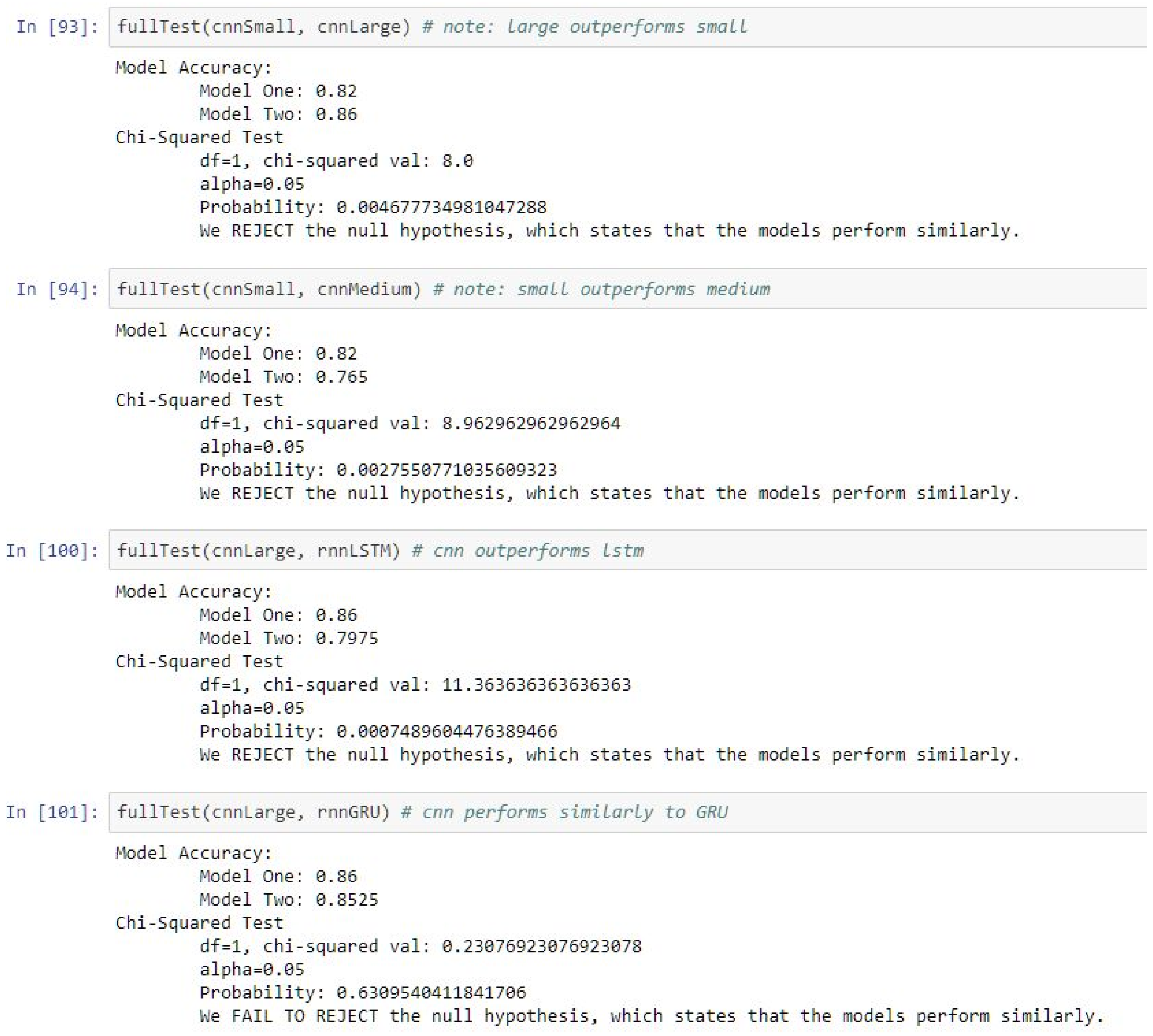
$$\chi^2 = \frac{(b-c)^2}{b+c}$$

where a, b, c, and d are counts of correct and incorrect predictions.[1]

[1] https://en.wikipedia.org/wiki/McNemar%27s_test

Standard Model Comparisons

We will first compare the standard models, which have no additional preprocessing or “wrinkle” (augmented) data.

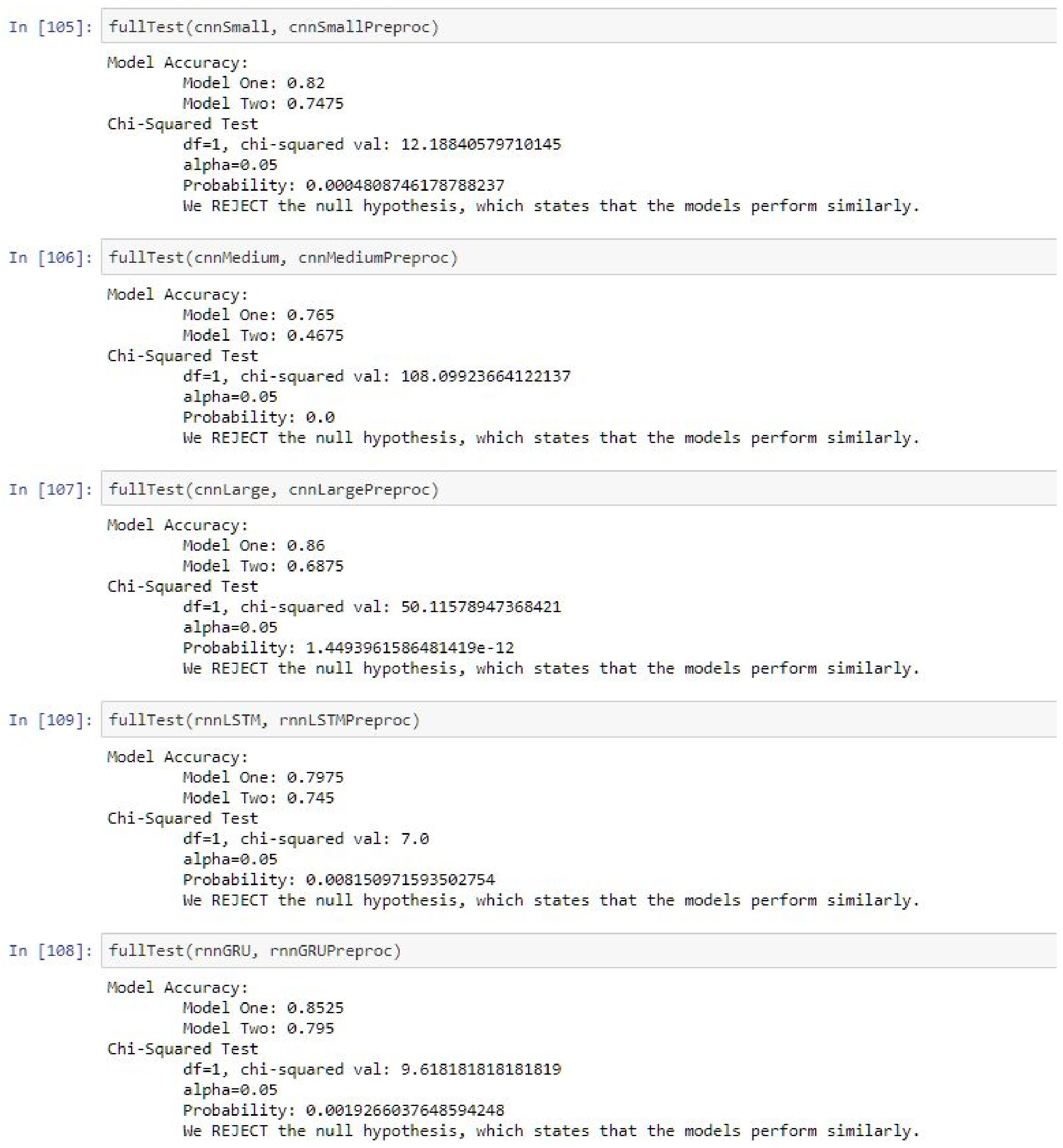


```
In [93]: fullTest(cnnSmall, cnnLarge) # note: large outperforms small

         Model Accuracy:
                 Model One: 0.82
                 Model Two: 0.86
         Chi-Squared Test
                 df=1, chi-squared val: 8.0
                 alpha=0.05
                 Probability: 0.004677734981047288
                 We REJECT the null hypothesis, which states that the models perform similarly.

In [94]: fullTest(cnnSmall, cnnMedium) # note: small outperforms medium

         Model Accuracy:
                 Model One: 0.82
                 Model Two: 0.765
         Chi-Squared Test
                 df=1, chi-squared val: 8.962962962962964
                 alpha=0.05
                 Probability: 0.0027550771035609323
                 We REJECT the null hypothesis, which states that the models perform similarly.

In [100]: fullTest(cnnLarge, rnnLSTM) # cnn outperforms lstm

         Model Accuracy:
                 Model One: 0.86
                 Model Two: 0.7975
         Chi-Squared Test
                 df=1, chi-squared val: 11.363636363636363
                 alpha=0.05
                 Probability: 0.0007489604476389466
                 We REJECT the null hypothesis, which states that the models perform similarly.

In [101]: fullTest(cnnLarge, rnnGRU) # cnn performs similarly to GRU

         Model Accuracy:
                 Model One: 0.86
                 Model Two: 0.8525
         Chi-Squared Test
                 df=1, chi-squared val: 0.23076923076923078
                 alpha=0.05
                 Probability: 0.6309540411841706
                 We FAIL TO REJECT the null hypothesis, which states that the models perform similarly.
```

We find that the large CNN outperforms all models by a statistically significant margin at an alpha cutoff of 0.05, with the exception of the GRU RNN.

Preprocessed Model Comparisons

Now, we compare each standard model to the identical version which was trained on the data preprocessed with stop word removal and lemmatization.



```
In [105]: fullTest(cnnSmall, cnnSmallPreproc)

          Model Accuracy:
                  Model One: 0.82
                  Model Two: 0.7475
          Chi-Squared Test
                  df=1, chi-squared val: 12.18840579710145
                  alpha=0.05
                  Probability: 0.0004808746178788237
                  We REJECT the null hypothesis, which states that the models perform similarly.

In [106]: fullTest(cnnMedium, cnnMediumPreproc)

          Model Accuracy:
                  Model One: 0.765
                  Model Two: 0.4675
          Chi-Squared Test
                  df=1, chi-squared val: 108.09923664122137
                  alpha=0.05
                  Probability: 0.0
                  We REJECT the null hypothesis, which states that the models perform similarly.

In [107]: fullTest(cnnLarge, cnnLargePreproc)

          Model Accuracy:
                  Model One: 0.86
                  Model Two: 0.6875
          Chi-Squared Test
                  df=1, chi-squared val: 50.11578947368421
                  alpha=0.05
                  Probability: 1.4493961586481419e-12
                  We REJECT the null hypothesis, which states that the models perform similarly.

In [109]: fullTest(rnnLSTM, rnnLSTMPreproc)

          Model Accuracy:
                  Model One: 0.7975
                  Model Two: 0.745
          Chi-Squared Test
                  df=1, chi-squared val: 7.0
                  alpha=0.05
                  Probability: 0.008150971593502754
                  We REJECT the null hypothesis, which states that the models perform similarly.

In [108]: fullTest(rnnGRU, rnnGRUPreproc)

          Model Accuracy:
                  Model One: 0.8525
                  Model Two: 0.795
          Chi-Squared Test
                  df=1, chi-squared val: 9.618181818181819
                  alpha=0.05
                  Probability: 0.0019266037648594248
                  We REJECT the null hypothesis, which states that the models perform similarly.
```

We note that, in every case, the standard model outperforms the preprocessed version by a statistically significant margin. This is likely because the lemmatization affects the word embeddings that serve as model inputs, changing the semantics of a sentence and negatively impacting classifier performance.

Wrinkle Model Comparisons

In our implementation of the proposed wrinkle, we append 50 full-splice and 250 half-splice documents to the training dataset (note that the test set is not augmented in any way). We then statistically compare the resulting model with its original form (trained on the standard dataset without preprocessing).

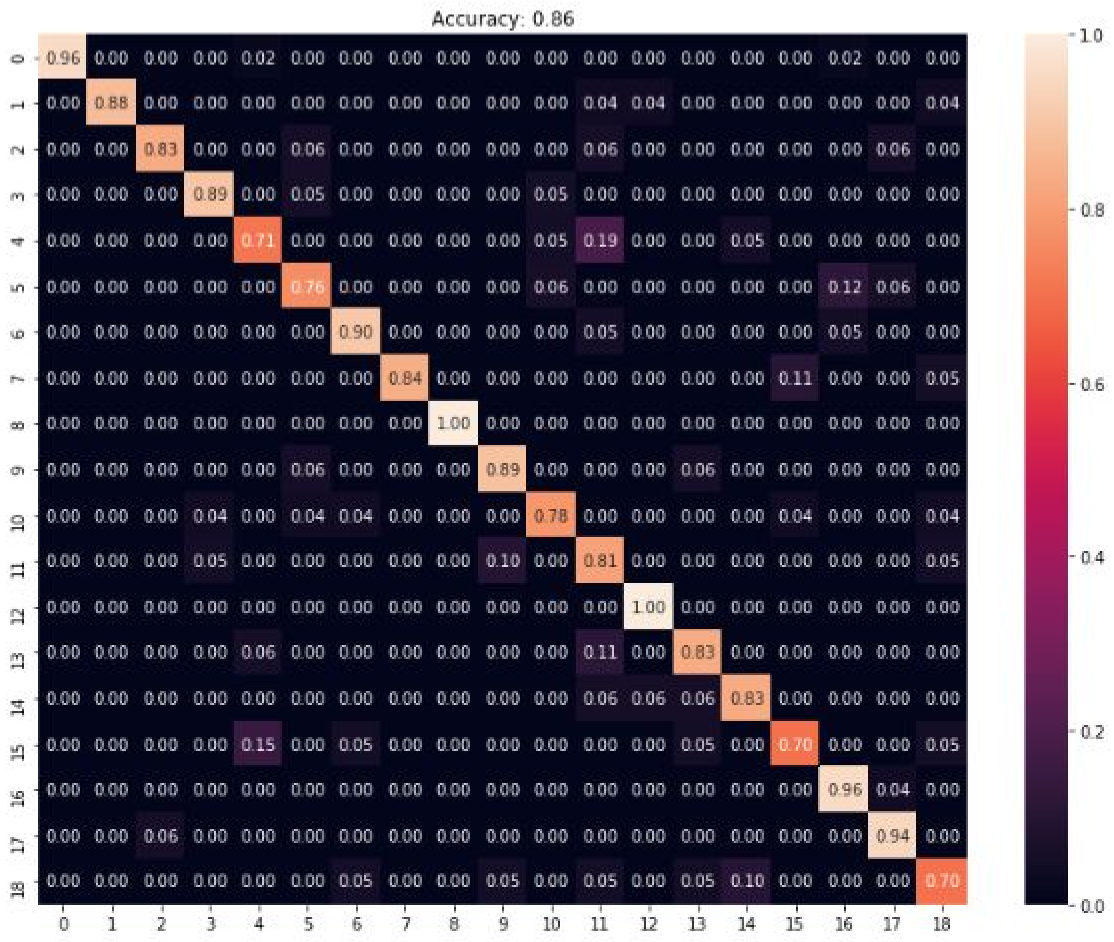
```
In [95]: fullTest(cnnSmall, cnnSmallWrinkle) # wrinkle doesn't have significant impact until alpha > 0.052
         print()
         fullTest(cnnSmall, cnnSmallWrinkle, alpha=0.1)

         Model Accuracy:
                 Model One: 0.82
                 Model Two: 0.85
         Chi-Squared Test
                 df=1, chi-squared val: 3.789473684210526
                 alpha=0.05
                 Probability: 0.05157586362087696
                 We FAIL TO REJECT the null hypothesis, which states that the models perform similarly.

         Model Accuracy:
                 Model One: 0.82
                 Model Two: 0.85
         Chi-Squared Test
                 df=1, chi-squared val: 3.789473684210526
                 alpha=0.1
                 Probability: 0.05157586362087696
                 We REJECT the null hypothesis, which states that the models perform similarly.

In [96]: fullTest(cnnMedium, cnnMediumWrinkle) # wrinkle has no statistical impact here

         Model Accuracy:
                 Model One: 0.765
                 Model Two: 0.7575
         Chi-Squared Test
                 df=1, chi-squared val: 0.36
                 alpha=0.05
                 Probability: 0.5485062355001471
                 We FAIL TO REJECT the null hypothesis, which states that the models perform similarly.

In [97]: fullTest(cnnLarge, cnnLargeWrinkle) # wrinkle has no statistical impact here

         Model Accuracy:
                 Model One: 0.86
                 Model Two: 0.8525
         Chi-Squared Test
                 df=1, chi-squared val: 0.2727272727272727
                 alpha=0.05
                 Probability: 0.6015081344405899
                 We FAIL TO REJECT the null hypothesis, which states that the models perform similarly.

In [102]: fullTest(rnnLSTM, rnnLSTMWrinkle) # wrinkle has statistically significant positive impact here

         Model Accuracy:
                 Model One: 0.7975
                 Model Two: 0.8325
         Chi-Squared Test
                 df=1, chi-squared val: 6.533333333333333
                 alpha=0.05
                 Probability: 0.010587137334056917
                 We REJECT the null hypothesis, which states that the models perform similarly.

In [103]: fullTest(rnnGRU, rnnGRUWrinkle) # wrinkle has no statistical impact here

         Model Accuracy:
                 Model One: 0.8525
                 Model Two: 0.8575
         Chi-Squared Test
                 df=1, chi-squared val: 0.18181818181818182
                 alpha=0.05
                 Probability: 0.6698153575994166
                 We FAIL TO REJECT the null hypothesis, which states that the models perform similarly.
```

We find that the wrinkle seems to have a statistically significant impact on the small convolutional neural network (when the alpha cutoff is slightly higher than 0.05), as well as the LSTM network (at the original alpha cutoff of 0.05). We moreover infer that the wrinkle may be particularly useful for increasing validation accuracy in models trained with a small corpus, or models that tend to overfit.

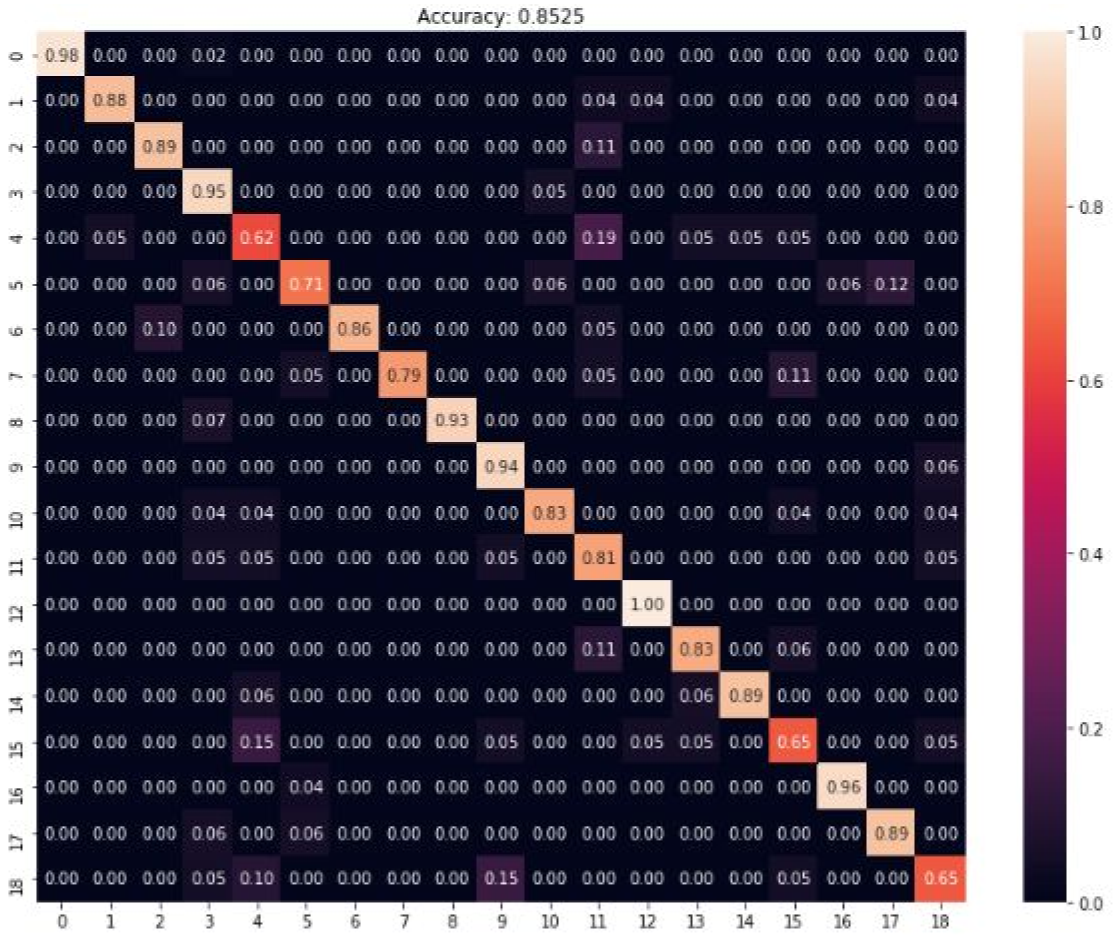
Performance Visualizations

We will visualize confusion matrices for the two highest performing models (which are not statistically different), the large CNN and the GRU RNN.

CNN-Large Heatmap



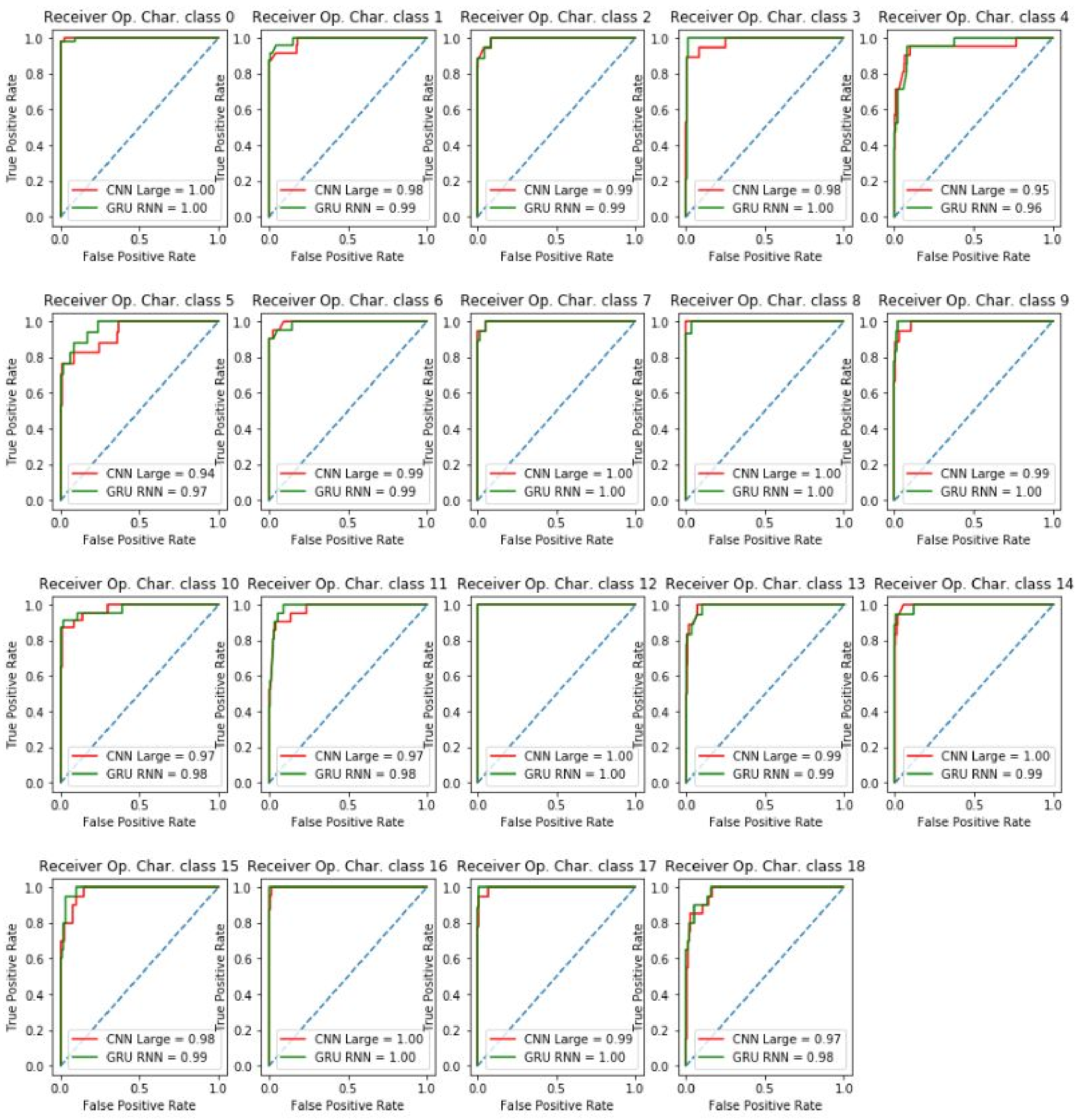
RNN-GRU Heatmap



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0.98	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.88	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.00	0.04
2	0.00	0.00	0.89	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.05	0.00	0.00	0.62	0.00	0.00	0.00	0.00	0.00	0.00	0.19	0.00	0.05	0.05	0.05	0.00	0.00	0.00
5	0.00	0.00	0.00	0.06	0.00	0.71	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.06	0.12	0.00
6	0.00	0.00	0.10	0.00	0.00	0.00	0.86	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.79	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.11	0.00	0.00	0.00
8	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.94	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06
10	0.00	0.00	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.00	0.83	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.04
11	0.00	0.00	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.81	0.00	0.00	0.00	0.00	0.00	0.00	0.05
12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.00	0.83	0.00	0.06	0.00	0.00	0.00
14	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.89	0.00	0.00	0.00	0.00
15	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.05	0.05	0.00	0.65	0.00	0.00	0.05
16	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.00	0.00
17	0.00	0.00	0.00	0.06	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.89	0.00
18	0.00	0.00	0.00	0.05	0.10	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.65

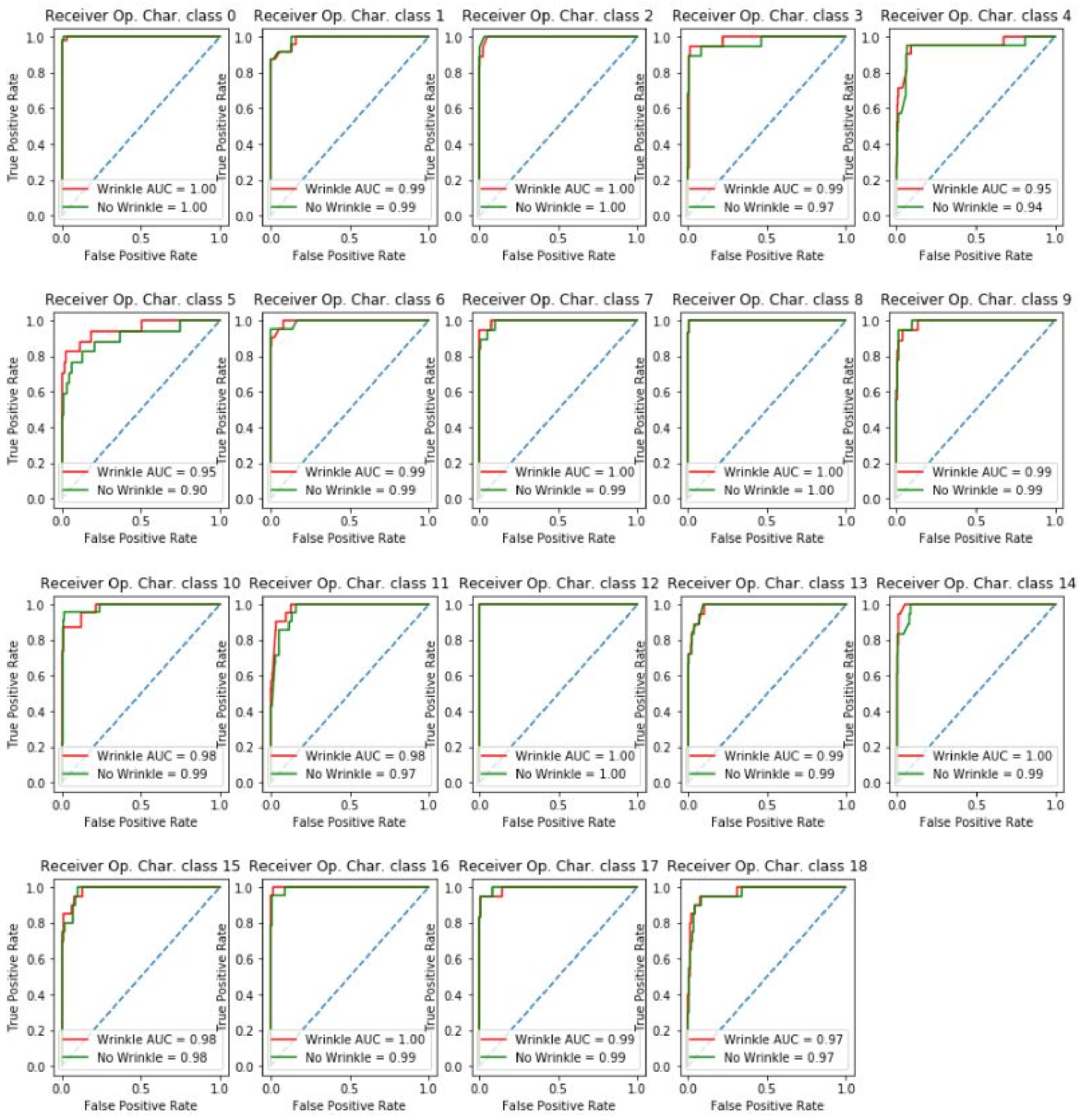
We note that many of the confusions are shared by the models: class four and eleven, in particular, are commonly misclassified by both models. However, other errors are not shared between the models (such as the class nine / class eighteen inaccuracy in the RNN). Therefore, it may be possible to gain (a small amount of) additional accuracy with an ensemble classifier of both models.

We can further visualize the comparison between the large CNN and the GRU RNN by plotting receiver operator characteristic plots for each class. This allows us to inspect the per-class performance of a model relative to the other.



The ROC curves show that, in most cases, the CNN and RNN perform similarly. However, there is a notable discrepancy on class five (where the RNN performs better, with 0.03 greater area-under-curve). This further suggests that an ensemble method of both models could improve general performance.

We will also plot a ROC curve to compare the small CNN and the small CNN trained on the augmented (wrinkle) data.



Here, we can see that the wrinkle has increased the network performance primarily on class five. In fact, the area-under-curve for that class surpasses that of the large CNN. This suggests that the wrinkle is effective in this use case, and shows promise for future investigation.

It should also be noted that we did not perform significant hyperparameter tuning with the wrinkle data. The number of items spliced in was chosen arbitrarily, and could be modified in the future to optimize performance.

Conclusions

In this program, we explored the use of recurrent neural networks (LSTM and GRU variants) and convolutional neural networks (of different sizes) in one-to-one MeSH tag classification of medical research papers based on the COVID-19 Open Research Dataset.

Five models were trained for this experiment - a small CNN, a medium CNN, a large CNN, a LSTM RNN, and a GRU RNN. These models were trained on a variety of different corpora, including the original corpus, a preprocessed corpus (stopwords removed and lemmatization), and an augmented corpus containing the original dataset, 50 full-splice, and 250 half-splice articles.

Our experiments with the five models trained on the original corpus showed that the large CNN significantly outperformed all other models except the GRU RNN. We also observed that the models trained on the original corpus significantly outperformed the corresponding models trained on the preprocessed corpus, which is likely due to the effect of lemmatization on the word embedding used as model inputs. Further experimentation with the augmented corpus that served as our wrinkle for this project showed that adding augmented articles had a statistically significant impact on the small CNN and LSTM RNN, which could indicate that adding full-splice and half-splice data can help increase validation accuracy in models trained with small corpora or models that tend to overfit.

As the large CNN and the GRU RNN provided the best performance on the classification task, performance visualizations using heatmaps and ROC curve plots allowed us to conclude that because some classification errors are not shared between the models, an ensemble classifier utilizing both of these models could achieve even greater accuracy. In addition, ROC curves plotted to compare the small standard CNN vs the small augmented CNN show that adding augmented data could be effective given certain use cases, particularly if hyperparameter tuning is conducted.