Exploring Model Fusion Techniques to Combat Misinformation

Pranav Bhoopala
University of Michigan
pranavb@umich.edu

Daniel Chechelnitsky
University of Michigan
dchechel@umich.edu

Haotian Qiao
University of Michigan
qhaotian@umich.edu

1 Introduction

In today's digital age, individuals across the world have access to information right at their fingertips. However, with an increased access to information comes an increased amount of misinformation, particularly concerning the credibility of news/media articles. Individuals can share articles on social media easily without any thought into the verification of whether the article is spreading misinformation or not. With rapid rise of social media and internet connectivity across the world, fake news can be spread at an alarming rate.

While there are many fake news classification services (websites, twitter accounts, etc.), it is impossible to classify every single news article as real/fake. Thus, we propose using machine learning models to aid in the fight to stop misinformation across the internet. Particularly, we look to use a technique from natural language processing called multi-class classification for textual information. These machine learning models will take in the news article headlines and output whether or not it believes that the given headline is real/fake. We believe this is an appropriate approach as we are looking to identify which class a news article falls into (real or fake).

However, many classifiers exist online to solve this same problem. Thus, we looked into a subdomain of machine learning called model fusion. Model fusion is a machine learning technique where multiple learners are combined where each of these learners use different methods to obtain better fitting performance and smaller errors than a single model (Zhen et al., 2023). We believe that model fusion can aid us in creating a highly accurate, unified model that brings the strengths of various models to light. We look to analyze the most important information of each model and understand how fusioning it with different model architectures can bring us closer to solving this problem of real/fake news.

2 Related Work

Classification problems have become increasingly common in the field of NLP in the past few years. Therefore, there are various related research papers discussing classifying whether news is fake/real.

- In 2023, Kozik et al. published a paper which discusses various approaches to classify misinformation of news articles using different ML models (Kozik et al., 2023). Some of the models used include CNNs, transformer based models, and classical classifier models. This paper reports that transformer based models performed the best on this binary classification task, with the best models outputting an accuracy of 75%.

- In 2019, Zhou et al. trained a classifier model named Fakebox which uses McIntires' fake-real-news-dataset to classify misinformative news articles (Zhou et al., 2019). This paper was able to obtain an accuracy of 62.40% on this binary classification task and highlights other approaches to view in the future.

Additionally, model fusion problems have been on the rise in the past few years. We would like to understand how we can perform model fusion on this classification task to yield better results.

- A report from Zhen et al. in 2023 discusses approaches for model fusion for machine learning in a visibility prediction scenario (Zhen et al., 2023). While this doesn't necessarily relate to the classification task we look to explore, they explain the benefits of model fusion within a machine learning task and various approaches that could be taken to create ensemble models.

- Bi et al. explores a multimodal late fusion model for E-Commerce classification (Bi et al., 2020). They train two models, a text classifier (BERT) and an image classifier (ResNet) and then approach combining the two at the feature level and at the decision level. This type of fusion is more in line with what our approach would be for this project, and we look to approach our task similarly as the one in this paper.

While we have found various papers discussing the actual classification task of news articles, we found none explicitly addressing model fusion on this particular classification task. Thus, we decided to pursue this route for our final project to understand why this task may not have been done in the past and if successful results can be yielded.

3 Approaches

We decided to split up the task into two distinct portions: first, each of us building a classifier model and then second, building a unified model created via fusion of our three created classifiers.

3.1 Selecting an appropriate dataset

As part of our misinformation classification task, selecting a high-quality dataset is important to the success of our trained models.

After some research, we came across a website called Poltifact which analyzes news articles, social media posts, and more and classifies them as fake or real news. The Politifact team classifies a news article to be part of one of six classes: true, mostly-true, half-true, barely-true, false, and pants-fire.

As such, we have decided to use a dataset from Kaggle titled "Fake-Real News" which draws from Politifact themselves (Yadav, 2020). Our chosen dataset has 10,000 news article headlines with their appropriate classification from the Politifact team. The raw data contained in the dataset consists of the news article headline, the source, the date of creation, and of course the classification label. Figure 1 highlights the initial distribution of the dataset.

Based off of the distribution in Figure 1, we can see that most of the labels are somewhat equally distributed across the dataset, with the exception of label "false" occuring slightly more frequently than the rest (and consequently less of label "true"). Thus, we felt that this initial cleaned dataset would be sufficient for our tasks such that certain labels will not dominate the others when training.

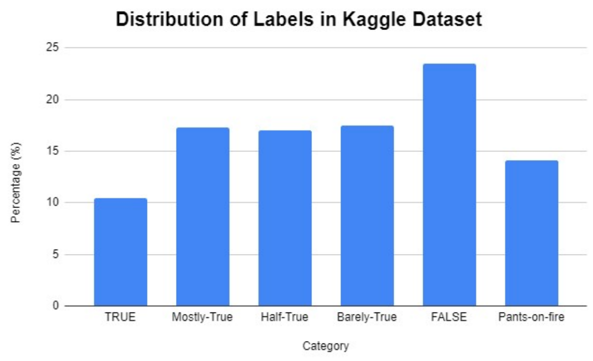


Figure 1: "Fake-Real News" Kaggle Politifact Dataset Distribution

3.2 Training and developing classifier models

As we had three of us working on this task, we decided it would be best to view how different models that each of us trained performed individually and then together when performing fusion. Text classification has become increasingly prominent in NLP, thus, we trained classifiers that we believed would perform best and/or give us interesting results in our analysis of model fusion.

3.2.1 Recurrent Neural Network Model

The first model that we chose to create was a recurrent neural network (RNN) model. RNN's are a type of neural network that are not feed-forward, unlike traditional neural networks; thus, they are able to maintain information as a "memory" of the previous inputs using hidden states (Amidi and Amidi, b).

Due to this structure, they are a great candidate for tasks where we deal with sequences, such as text processing and sentiment analysis for this particular scenario within this project.

The RNN model starts with an embedding layer to aid in converting the textual information into a vector representation. Additionally, there are LSTM layers to ensure that our RNN model can capture the dependencies and contextual information within the news article headlines more effectively. Finally, the model outputs from a dense layer which takes in 100 features as input, and outputs 6 features before ultimately performing the classification on that particular news headline. The full architecture for the trained RNN is shown in Figure 2.

3.2.2 Convolutional Neural Network Model

The second model that we chose to create was a convolutional neural network model (CNN).

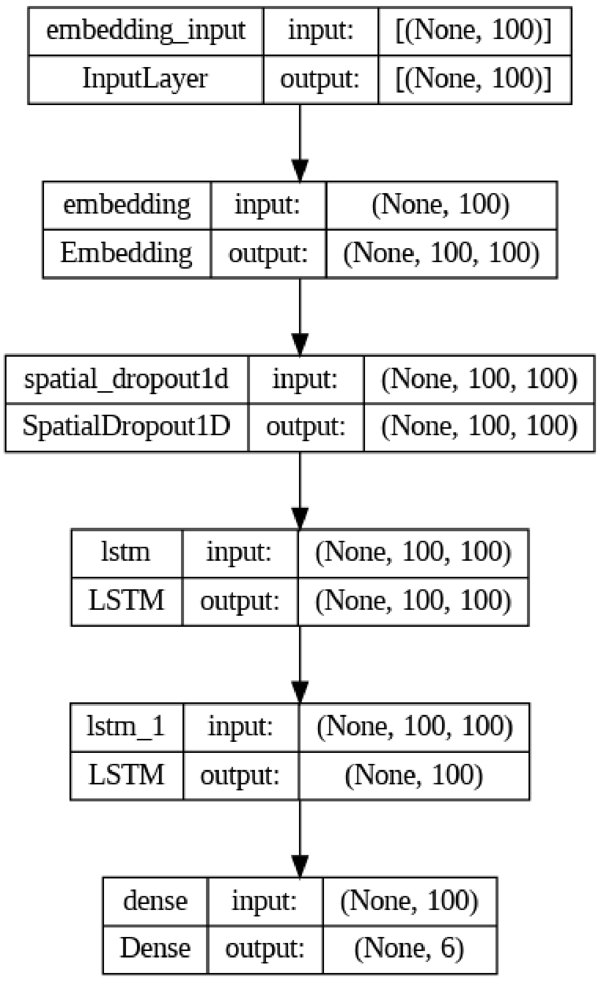

embedding_input | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
embedding | input: | (None, 100)
Embedding | output: | (None, 100, 100)
spatial_dropout1d | input: | (None, 100, 100)
SpatialDropout1D | output: | (None, 100, 100)
lstm | input: | (None, 100, 100)
LSTM | output: | (None, 100, 100)
lstm_1 | input: | (None, 100, 100)
LSTM | output: | (None, 100)
dense | input: | (None, 100)
Dense | output: | (None, 6)

Figure 2: RNN Model Architecture for Classification

CNN's are a type of feed-forward neural network that are generally composed of a convolution and pooling layer which can be fine-tuned for the task at hand (Amidi and Amidi, a). CNN's first generally gained popularity in the computer vision space, where it works extremely well on recognizing an object based on a defined category. However, recently, CNN's are becoming increasingly popular in terms of NLP textual classification tasks, which is why we were interested in seeing how a CNN would perform on our real/fake news classification task (Kim, 2014).

Generally, in a computer vision/image recognition task, CNN's are generally comprised of a 2D convolutional layer as images are represented in a 2D grid of pixels (PNG, JPEG, etc. formats). However, for our text classification task, a 1D convolutional layer should be sufficient as the actual textual information is sequential by nature, which works well with a 1D layer. Figure 3 highlights a sample CNN in which we can see how an input can be passed via a 1D convolutional layer in a CNN to output a final classification. In general, we believe this type of structure can work with good accuracy on textual classification tasks. We have decided to structure our model using a similar approach.

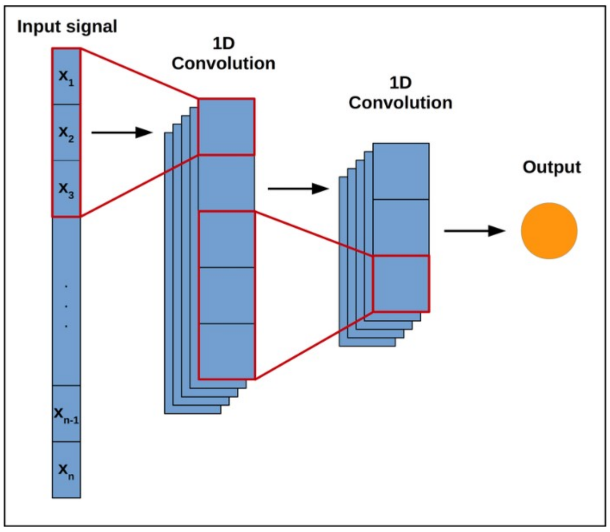


Figure 3: Simple 1D convolutional neural network (CNN) architecture with two convolutional layers (Shenfield and Howarth, 2020)

Our CNN starts with an embedding layer as before with the RNN to convert the textual information passed into it to a vector representation. Then, a 1D convolutional layer is used to process the actual text sequences before being passed onto a 1D pooling layer for downsampling. Finally, the model finishes off with two dense layers having the last one output 6 features as before, and then ultimately performing the classification appropriately. The architecture described for our CNN is depicted in Figure 4.

3.2.3 Transformer Based Models

Lastly, we chose to implement a transformer based model, BERT, as our third and final model. BERT stands for Bidirectional Encoder Representations from Transformers; essentially, BERT models are pre-trained on a large corpus of text, and then are fine-tuned for their specific tasks. BERT uses a transformer to computer vector-space representations of the actual natural language. Thus, it is a suitable model that can be used for text classification.

We start by loading a pre-trained BERT model called 'bert-base-uncased'. Then, we added some hidden layers to the output of the model as part of our hyperparameter-tuning. The BERT model

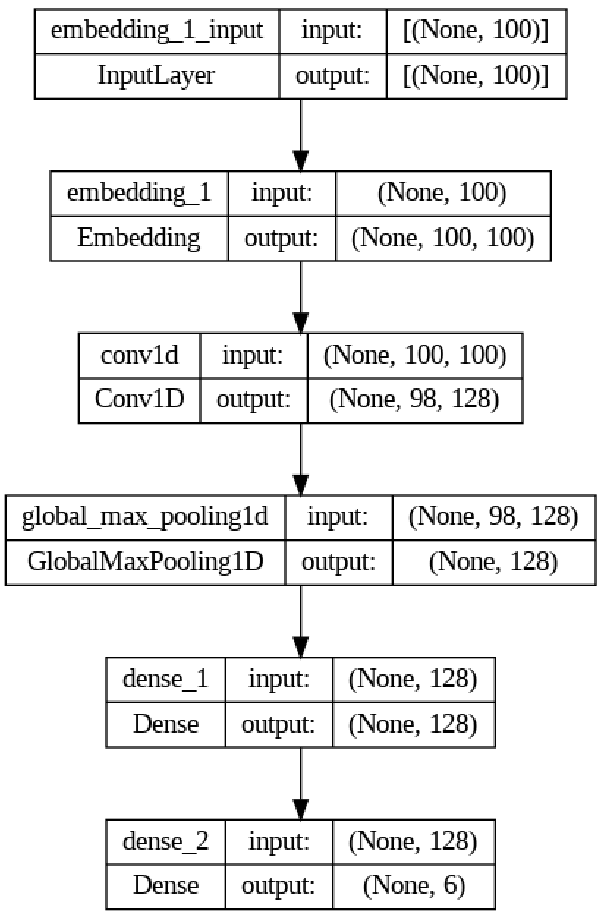

embedding_1_input | input: | [(None, 100)]
InputLayer | output: | [(None, 100)]
embedding_1 | input: | (None, 100)
Embedding | output: | (None, 100, 100)
conv1d | input: | (None, 100, 100)
Conv1D | output: | (None, 98, 128)
global_max_pooling1d | input: | (None, 98, 128)
GlobalMaxPooling1D | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 128)
dense_2 | input: | (None, 128)
Dense | output: | (None, 6)

Figure 4: CNN Model Architecture for Classification

starts with the tokenized input created from the tokenizer from 'bert-base-uncased,' and then we have a dropout layer followed by a dense layer that outputs 6 features. All of these hidden layers are applied on the actual output of the BERT model that we have used with our dataset. The full architecture for our pre-trained BERT model can be found in figure 5.

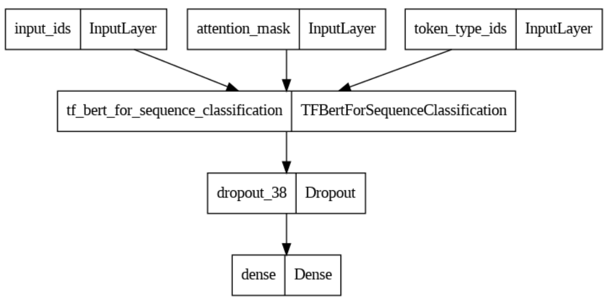

input_ids | InputLayer
attention_mask | InputLayer
token_type_ids | InputLayer
tf_bert_for_sequence_classification | TFBertForSequenceClassification
dropout_38 | Dropout
dense | Dense

Figure 5: BERT Model Architecture for Classification

3.3 Unified Fusion Model Approaches

As part of our project, we looked to create a unified model which could hopefully take the best strengths of each of our individually created models (as described above) and then give us one 'best' guess on this classification task. However, with fusion, there are various avenues we can take to create our unified model:

1. Early Fusion: In early fusion, the raw data modalities are integrated together into a unified representation *before* the learning/feature extraction process of the models. As an example, early fusion could involve concatenation of the inputs to each of the models along a new dimension (Boulahia et al., 2021). While this may sound simple in theory, there are few papers/works that have adopted early fusion strategies as part of their multimodal model fusion work.

2. Late Fusion/Hidden Representation Level Fusion: Late fusion is generally a merging strategy that will occur outside the monomodal classification models. A late fusion approach will combine the decisions of each classifier to produce new decisions that may be more precise (Boulahia et al., 2021). A common approach of late fusion is to combine the hidden states of each of the monomodal classification models (Hidden Representations) and then use that as input to a newly created, unified model for classification. This way, we can optimize for the best strengths of each of the models to be highlighted in the unified model.

We have decided to move forward with late fusion for our unified model as this seemed to make the most sense for our situation and current understanding of ML/NLP techniques. Additionally, we found that the Keras Functional API provided us with the tools to do so appropriately and easily (Chollet).

3.3.1 Late Fusion Model Architecture

As our classifier models were created using Keras with Tensorflow, the Keras Functional API provided us with a structure to easily create a unified model using late fusion. In our unified model, we have decided to use a late fusion approach where we combine the hidden states of each of the classifier models to act as input for our unified model. This allows us to pass on what each of the three individual models learned and are best at working with. That way, each of their strengths can be highlighted and hopefully be showcased in our fusion model. The hidden layers for each of our classifiers can be viewed in Figures 2, 4, and 5 for the RNN, CNN, and BERT models, respectively.

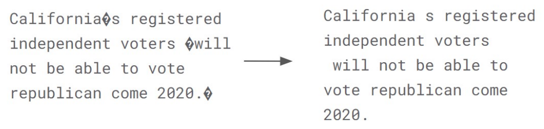
After concatenating each of the hidden layers to produce the unified model, we can also apply additional layers within this unified model before outputting the final classification result. We decided to have three dense layers, where the first one takes in 256 features as input, the second layer taking in 128 features, and then the last one taking in 6 features. The full architecture of our fusion model can be viewed in Figure 7.

4 Evaluation

4.1 Dataset Setup and Cleaning

For each of our classifier models, we trained them using a randomized 80%/20% train-test split. Each of the train and test portions are the same across each of the models to ensure consistency. Additionally, we ensured that each input to the model had a consistent length, so we padded the inputs to have a maximum length of 100.

We then performed data pre-processing to ensure that our dataset used was cleaned accordingly so that our model was not learning on unnecessary features of the news headline. Special characters, certain punctuation, and more were all removed to ensure that each input to the models are consistent and will not be influenced by outside factors. One important thing to note, however, is that stop words were *not* removed from the dataset to help preserve context within the news headline itself.



```
California�s registered
independent voters �will
not be able to vote
republican come 2020.�
```

$\longrightarrow$

```
California s registered
independent voters
 will not be able to
vote republican come
2020.
```

Figure 6: Example of text pre-processing

After all the pre-processing has been done, we have our same six different labels as explained earlier, with around 10,000 total news article headlines and their associated labels. We can therefore deduce that a random-guessing model will have a 16.67% accuracy on deciding which label to assign a news headline to. Ideally, each of our classifiers will improve on this accuracy and then our fusion model will have the best accuracy of all the models created.

4.2 Multi-Class Classification Analysis

In this portion we discuss the performance and results of the three classifier models we trained (RNN, CNN, BERT). We evaluated all the models on the 20% test set split as mentioned previously. Thus, all models are tested on the same, consistent test set. Additionally, each of the models use categorical cross-entropy as the loss functions to remain consistent within the changing models.

4.2.1 RNN Model Performance

We fine tuned our RNN with the architecture as previously explained, and found that an Adam optimizer worked best in terms of accuracy on the test set. There are various hyperparameters we tested the RNN on, and Table 1 highlights these different configurations and the corresponding accuracy.

Epochs	Batch Size	LR	Accuracy
10	64	10^{-5}	23.64%
10	128	10^{-5}	23.64%
10	64	10^{-6}	23.64%
10	8	10^{-4}	23.64%

Table 1: RNN Results on Various Hyperparameters

It was interesting to observe how regardless of the actual hyperparameter setup that we chose, the actual performance of the RNN did not necessarily change. There can be various reasons that may cause this output, such as data imbalance issues or overfitting on the actual data itself due to this dataset being on the smaller side of 10,000 entries.

4.2.2 CNN Model Performance

As before, we fine tuned our CNN with the architecture as previously explained in section 3 of the paper. We used an Adam optimizer on the test set and measured accuracy as our evaluation metric. Table 2 highlights the various configurations and the corresponding accuracy obtained from the CNN models.

Epochs	Batch Size	LR	Accuracy
10	64	10^{-3}	26.59%
10	128	10^{-3}	23.53%
10	64	10^{-2}	24.15%
10	128	10^{-2}	25.46%

Table 2: CNN Results on Various Hyperparameters

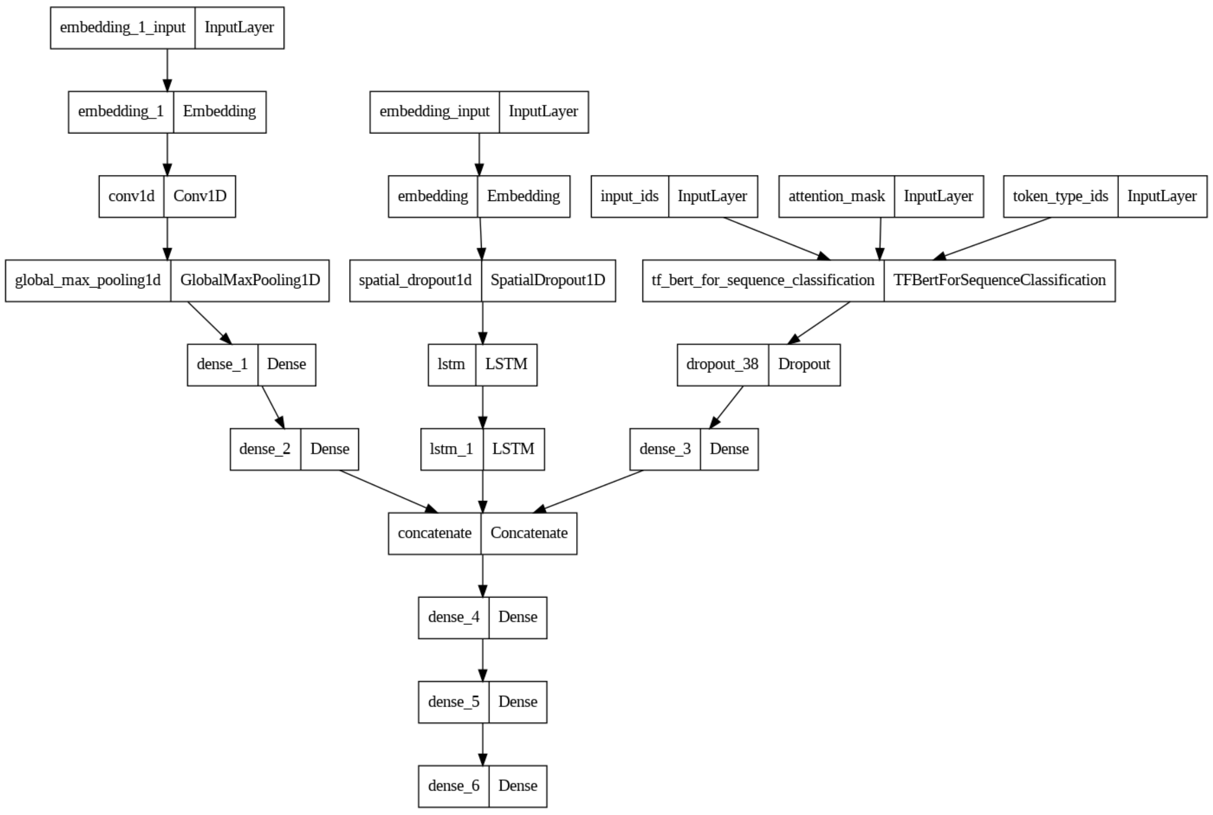

embedding_1_input | InputLayer
embedding_1 | Embedding
conv1d | Conv1D
global_max_pooling1d | GlobalMaxPooling1D
dense_1 | Dense
dense_2 | Dense
embedding_input | InputLayer
embedding | Embedding
spatial_dropout1d | SpatialDropout1D
lstm | LSTM
lstm_1 | LSTM
input_ids | InputLayer
attention_mask | InputLayer
token_type_ids | InputLayer
tf_bert_for_sequence_classification | TFBertForSequenceClassification
dropout_38 | Dropout
dense_3 | Dense
concatenate | Concatenate
dense_4 | Dense
dense_5 | Dense
dense_6 | Dense

Figure 7: Fusion Model Full Architecture

4.2.3 BERT Model Performance

We used a pre-trained BERT model based off of 'bert-base-cased' in which we tokenized the inputs of the news article headlines before feeding it into the BERT model. The architecture of the BERT model is as explained before, and we used an Adam optimizer on the test set and measure accuracy as our evaluation metric. We tested our BERT model on various hyperparameters shown below in Table 3. It is a pre-trained model so comparing to models above, it was trained/finetuned for only 3 epochs.

Epochs	Batch Size	LR	Accuracy
3	64	10^{-3}	18.21%
3	64	10^{-4}	28.46%
3	64	10^{-5}	23.64%

Table 3: BERT Results on Various Hyperparameters

4.2.4 Full Fusion Model

As before, we have constructed our fusion model to contain the architecture explained in section 3. In order to be as consistent as possible, we used an Adam optimizer on the test set and measure accuracy as our evaluation metric. The following table highlights a few hyperparameters we tested on our fusion model and the obtained accuracy.

Epochs	Batch Size	LR	Accuracy
2	64	10^{-3}	18.21%
2	64	10^{-5}	21.87%
2	32	10^{-3}	25.26%

Table 4: RNN, CNN, Bert Fusion Results on Various Hyperparameters

4.2.5 Fusion on RNN and CNN

Additionally, we were interested in performing fusion on only two models to see how the obtained performance differs from that of infusing that of three models. In the interest of time, we decided to only fuse RNN and CNN as the two models as these two took the least amount of time to train and test, in addition to having complementary strengths that can highlight the best of both the neural networks. Just like before, we kept the architectures of the underlying models the same while having two dense layers before outputting in this second fusion model. Table 5 highlights the output of the fusion with RNN and CNN, with differing learning rates.

Epochs	Batch Size	LR	Accuracy
5	64	10^{-3}	23.85%
5	64	10^{-4}	23.94%
5	64	10^{-5}	25.41%

Table 5: RNN and CNN Fusion Results on Various Hyperparameters

5 Discussion

For each of our models created, we evaluated them on our 20% test split as described earlier. Our evaluation metric was accuracy, where accuracy is measured as the proportion of classifications that were classified correctly by the model performing the testing. Since we have 6 different labels, this makes our 'baseline' for the accuracy 16.67%.

5.1 Notes on our Models

5.1.1 RNN Model

When training the RNN, we noticed that it would generally converge to the same training accuracy after 1-2 epochs on our best performing model. At first, we believed that this could be due to the fact that our learning rate was either too high/low. However, after training with various learning rates, we found that while the training accuracy within epochs varied slightly, the accuracy on the test set was the same regardless of the hyperparameters we chose.

After further analysis, it is possible that the range hyperparameters we chose to experiment with was not 'wide' enough, and that we should have tested with more different hyperparameters like different optimizers, dropout rate, learning rate, etc. Moreover, this could also be due to the RNN structure itself in relation to the dataset we are training the RNN with. It is plausible that the dataset may not have been cleaned properly and/or is too small to use with an RNN. Thus, the same features would be learned with the RNN regardless of the hyperparamaters that are used.

5.1.2 CNN Model

While training our CNN model on the various hyperparameters, we noticed that it would generally overfit on the test data, easily obtaining a 100% accuracy after various epochs. There may be various reasons that this could occur with the CNN. For one, the dataset could be too small and therefore the model could be too complex for the dataset. Since there are only 10,000 entries, the CNN could easily pick up on features and thus overfit on the train data with >5 epochs used during training. In the future, we can explore applying regularization techniques to aid in ensuring that overfitting does not occur in the training phase. We can look into techniques such as applying a dropout to understand whether this could impact the train accuracies obtained with the CNN.

5.1.3 BERT Model

The BERT model performed the best out of the three, which lines up with what we were expecting, as BERT models in general perform exceptionally well on textual analysis and classification tasks. However, after various hyperparameter testing, we were not expecting the BERT to obtain a <30% accuracy on our dataset.

After trying various hyperparameter configurations on the BERT model, the accuracies were not up to our expectations. Thus, we believe that the dataset could be creating the lower than expected accuracies. For instance, its possible that BERT models perform better on very large scale datasets so that they can capture more contextual information seen within the dataset (in our case news headlines). Thus, with our dataset size being on the smaller side, the BERT model may not generalize well to unseen examples.

5.1.4 Fusion Model

With the fusion model, we were expecting results that would match or exceed the greatest accuracy we have seen from each of the individual classifier models. However, we were only able to get up to an accuracy of up to 25% on our various hyperparameters that we tested.

Based off of our original interpretation of the fusion model, we wanted the 'strengths' of each model to be used and would thus give us the highest accuracy on this classification task. However, the lower accuracy from the RNN and CNN seem to be pulling the performance of the unified model to be lower than expected.

There are various reasons this may occur. One of them, for instance, could be the fact that this task may not necessarily benefit from combining/concatenating multiple modalities to achieve a higher accuracy. We may be able to deduce this based on the lower accuracy we obtained, in addition to the fact that other papers did not approach this classification task using model fusion.

Additionally, there may be mismatches when it comes to the actual hidden layers and architectures concatenated between the three models. Each of the three models process and learn the information in different ways. RNN feeds back its results learned from the neural network, while CNN uses a feed-forward network with convolutional and pooling layers. BERT in contrast, uses various embeddings to create textualization representations of the phrases it is working with. Thus, it is completely possible that combining these various architectures might result in feature mismatches and loss of information.

5.2 Limitations

5.2.1 Dataset

After training each of our models, it became increasingly apparent that the dataset could have been improved by including a lot more news headline samples, in addition to better pre-processing methods. It could be entirely possible that our models could have performed better with more information to train on compared to the limited 8,000 entries that we used to train our models with. This would also have helped alleviate the overfitting we saw for some models like the CNN.

Finding a higher quality dataset is paramount for this particular task, as classifying misinformation can vary from dataset to dataset. In the future, we look to potentially combine various datasets from different sources to train our models on. This way, we are ensuring that we have larger amounts of data from different sources so that our model will not necessarily be overfit.

5.3 Future Work

Our results show that the BERT model was able to has the highest accuracy, even over the fusion model, ultimately demonstrating that our late fusion approach was sub-optimal. There are many places in the model training and processing pipeline that fusion can take place, and more work can still be done to see at which of these points would be the best for the overall infusion model accuracy. Techniques like early fusion, for example, could still prove to be better performing.

With regards to fusion techniques as a whole, it would also be interesting to see if there are ways to incorporate other kinds of methods outside of the Keras library: specifically those that use a different pooling structure. As other libraries become more optimal, it will be interesting if any of those can be used in a more optimal way.

Looking beyond the fusion model: our other two individual models, the RNN and the CNN models, could also further have been optimized with experiments by further pre-processing the data for RNN and by adding a dropout layer for the CNN respectively, for examples.

Finally, instead of using our dataset of 10,000 entries, we could apply this to a larger scale dataset of real-fake news and see if the accuracies are proportional even on those that are trained on the datasets of larger size and more variety.

Overall though, since the data shows that the BERT model was able to have significantly higher accuracy than the fusion model, this makes a strong argument to simply just use BERT and other transformer models instead of using any fusion techniques. Thus, to put it generally, fusion approaches should probably not be pursued in this and other similar context classification tasks.

6 Challenges Faced

Throughout the development of our model and search for dataset, we definitely encountered some challenges that impacted the overall development and roadmap of our models.

6.1 Fusion

Actually performing the multimodal fusion task was quite ambiguous, as many of the resources available online were rather more explanation heavy and theoretical rather than actually depicting how model fusion would work practically. As this was a new topic to us in terms of machine learning, we spent a considerable amount of time just figuring out exactly what model fusion is and entails. After spending some time in office hours, we came to the conclusion that proceeding with the Keras functional API would make model fusion simpler, which helped us understand and execute the overall development process at a smoother rate than before.

6.2 Planning

At the start of our project, we did not plan too much regarding the actual development of the project aside from what was laid out in the project proposal. A lot of the main details were left unknown until after visiting office hours a few times or looking extensively online to get a better understanding of which direction we should proceed with our task. For instance, training BERT models on Google Colab took more time than expected, as the Colab instance would run out of memory fast or simply block GPU access after our allocated time had expired.

6.3 Choosing a dataset

Finding and choosing a dataset for this particular task proved to be a challenge as it is rather hard to find an accurate and properly distributed dataset for a fake/real news classification task. Additionally, we would have to understand what *actually* classifies a news article as misinformation or not, and what the line between real and fake is. Moreover, we were unable to find more recent, larger datasets for our problem which we felt could have improved the performance of our models compared to our smaller, 10,000 element dataset.

7 Conclusion

In summary, this project demonstrates three different individual approaches to classifying real-fake news articles, as well as a infusion model approach. We developed an RNN, CNN, BERT, and fusion model and ultimately classified each of these models in order to understand how model fusion can impact a classification task such as news misinformation. By comparing all four of these various methods of classification, we were able to see which models were more suited for performing this task while also giving providing insight into how classification can still be improved in the future.

We had to navigate through some challenges throughout the process to come to our end goal, but the results we obtained helped us gain a deeper understanding of the ever-changing world of NLP. It further gave us context into why methods like fusion model methods have not been widely used in more cutting-edge NLP research.

8 Division of Work

Our group consisted of three members: Pranav Bhoopala, Daniel Chechelnitsky, and Haotian Qiao. The work of our project was easily dividable into multiple parts so that we can all contribute equally to the project. Each of us trained a classifier model, where Pranav trained the RNN, Daniel trained the CNN, and Haotian trained the BERT model. The actual model fusion was done together as a group, where we all came together to figure out how to perform the model fusion via Keras and understand which architecture would work best for our textual classification problem. Lastly, we all worked together equally on finishing the overall report and the in-class presentation.

9 URL to Code Base

https://github.com/chechelnitskd/real-fake-model-infusion

References

Afshine Amidi and Shervine Amidi. a. Convolutional neural networks cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

Afshine Amidi and Shervine Amidi. b. Recurrent neural networks cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

Ye Bi, Shuo Wang, and Zhongrui Fan. 2020. A multimodal late fusion model for e-commerce product classification. *arXiv preprint arXiv:2008.06179*.

Said Yacine Boulahia, Abdenour Amamra, Mohamed Ridha Madi, and Said Daikh. 2021. Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. *Machine Vision and Applications*, 32(6):121.

François Chollet. The functional api. https://keras.io/guides/functional_api/#models-with-multiple-inputs-and-outputs.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Rafał Kozik, Aleksandra Pawlicka, Marek Pawlicki, Michał Choraś, Wojciech Mazurczyk, and Krzysztof Cabaj. 2023. A meta-analysis of state-of-the-art automated fake news detection methods. *IEEE Transactions on Computational Social Systems*.

Alex Shenfield and Martin Howarth. 2020. A novel deep learning model for the detection and identification of rolling element-bearing faults. *Sensors*, 20(18):5112.

Kajal Yadav. 2020. Fake-real news. https://www.kaggle.com/datasets/techykajal/fakereal-news.

Maochan Zhen, Mingjian Yi, Tao Luo, Feifei Wang, Kaixuan Yang, Xuebin Ma, Shengcheng Cui, and Xuebin Li. 2023. Application of a fusion model based on machine learning in visibility prediction. *Remote Sensing*, 15(5).

Zhixuan Zhou, Huankang Guan, Meghana Moorthy Bhat, and Justin Hsu. 2019. Fake news detection via nlp is vulnerable to adversarial attacks. *arXiv preprint arXiv:1901.09657*.