Making the Best Use of Review Summary for Sentiment Analysis

IE643 Course Project
DeepMind2.0

Swapnoneel Kayal 200100154
Yash Choudhary 200100173

Outline

Sentiment Analysis

Fundamental task in NLP, which predicts the subjectivity and polarity of the given text. It is widely used in E-commerce and movie reviews.

Problem Statement

- So far, people have devised several models that only take review information as input for classifying the sentiment.
- However, it has been empirically shown that meaningful summaries can help in sentiment classification as well and hence can be used as additional training signal.
- As a part of this project, we are interested in making the best use of both review and summary information to achieve an effective sentiment classification.

Work done before the mid-term project review

- Prior to the mid-term project review, most of our efforts went into thoroughly reading the paper [1] from which we drew our major inspiration, as well as two related papers. We also spent good amount of time to understand all aspects of the problem statement as well as the various methods used to tackle it.
- LSTMs and GRUs were previously unknown to us, and we understood not only these basic models but also advanced models such as Bi-LSTMs and Bi-GRUs. We theoretically understood the importance of the co-attention layer as well as global average pooling.
- We learned the basics of the PyTorch framework to make it easier for us to use it ahead. We also started our work by first procuring the usable dataset for Amazon reviews on Toys and Games which was available online and then used it to train the Bi-LSTM based review-centric attention model whose code was made available by Yang et al. on GitHub.

Work done before the mid-term project review

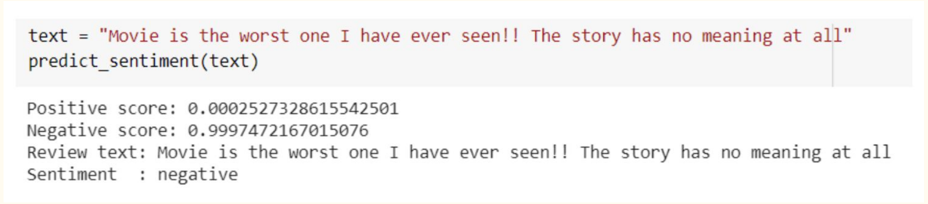
- We initially tried to train it locally on our PC setup which had Cuda 11.3 Environment. During the training phase, the data pre-processing part was completed successfully. However, we were able to train the model up to 2 epochs only post which the training did not show any progress since our PC setup was running out of GPU memory.
- We finally decided to leverage the GPU offered by Google Colab. There were a lot of ideas for modifications that could lead to even better sentiment classification that we had discussed but we spent time on understanding and selecting the ones that are worth trying. The main modification we planned to experiment with was to change the Bi-LSTM block of the model to a Bi-GRU.
- Some other modifications we felt were worth trying out were :
    - Train on a smaller subset of the training data and then consider increasing the subset.
    - Change the number of layers present in their proposed model from 2 to 3.
    - Replace PG-Net with something more state-of-the-art for summary generation
    - Add the summary generated by the model to the actual summary given by the user.

Major Comments given during the mid-term project review

- We can substitute the Bi-LSTM block with a Bi-GRU one and comment on the test accuracy as well as the test loss obtained after training.
- Number of hidden layers can be changed in order to understand it's impact on the final accuracy.
- We can replace PG-Net (summary generation model used in by the authors of the paper) with something more state-of-the-art. Since more meaningful summaries would lead to a better sentiment classification.
- Predictions on complicated reviews could be tried in order to understand how well the model performs in sarcastic sentiment analysis.

Team’s addressal to the comments given during the mid-term project review

- We initially distributed the work so that we two can work independently on it.
- Swapnoneel’s task was to:-
  - Substitute the Bi-LSTM block with the Bi-GRU one
  - Change the number of layers from 2 to 3
- Yash’s task was to :-
  - Analyze the model’s performance on sarcastic sentiments
  - Find summary generation models that can outperform the PG-Net and try to replace it and analyze the subsequent testing accuracy and loss that is obtained
- New ideas that came up:-
  - Transformer-based architectures like XL-Net can also be used for sentiment analysis.



```
text = "Movie is the worst one I have ever seen!! The story has no meaning at all"
predict_sentiment(text)
```

```
Positive score: 0.0002527328615542501
Negative score: 0.9997472167015076
Review text: Movie is the worst one I have ever seen!! The story has no meaning at all
Sentiment  : negative
```

Work done After the Mid-Term Project Review

- After the mid-term project review, we progressed with the construction of the Bi-GRU model that we had proposed in PyTorch and consolidated the dataset.
- We simultaneously successfully completed the training of the review-centric Bi-LSTM model. Hyper-parameters were also tuned appropriately and were kept at those values for which the model would not take a lot of time to train.
- Then we were also able to train the Bi-GRU model.
- We then tackled the problem of changing the number of layers within the Bi-LSTM model from 2 to 3. We also decided to experiment with the number of layers within the Bi-GRU as well and hence we trained a Bi-GRU model with 3 hidden layers.
- Finally as a part of our end-term project review, we also prepared a video wherein we go over our code along with another video wherein we demonstrate the different experiments that we have conducted.

Data set Details

- SNAP Amazon Review Dataset consists of around 34 million Amazon reviews in different domains, such as books, games, sports and movies.
- Each review mainly consists of a product ID, a piece of user information, a plain text review, a user-written summary and an overall sentiment rating which ranges from 1 to 5.
- For training purposes however, we have restricted ourselves to only 1 dataset i.e. Toys & Games. Within this domain, there are about 168,000 products with the average length of the reviews being around 99.9 and the average length of the corresponding summaries being around 4.4.
- For each dataset, the first 1000 samples are taken as the development set, the next 1000 samples as the test set, and the rest as the training set.
- Before using the dataset for training, few data preprocessing techniques were adopted and they are as follows:
    - Tokenizing the dataset
    - Processing word embedding files for the given dataset
    - Saving the tokenized dataset, word to index, and index to vector pickle files

Experimental Details

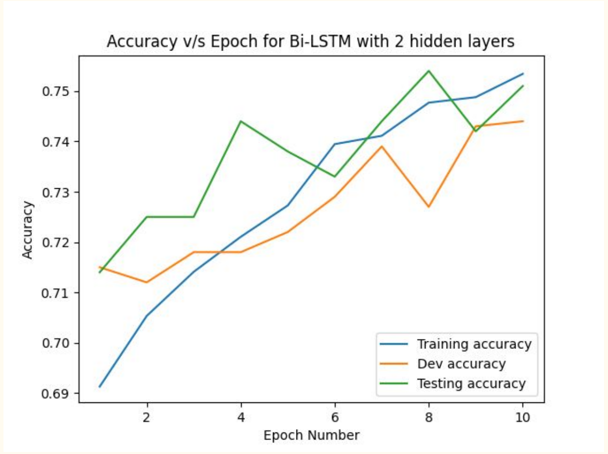
- Since we were using Google Colab, our training was heavily dependent on the GPU that was being allocated to us during that particular runtime.
- After having successfully trained the model as described in the paper using a Bi-LSTM, the first experiment that was done was changing the Bi-LSTM block of the model to a Bi-GRU (Bidirectional Gated Recurrent Unit) in PyTorch.
- In case of Bi-GRU, the number of gates are reduced hence the resulting model is far less complex. It is also approximately as good as a Bi-LSTM despite the reduction in complexity and is generally used when training needs to be done quickly and with decent accuracy, or if there are infrastructural constraints.
- We hypothesized that this could speed up training without compromising much on accuracy.

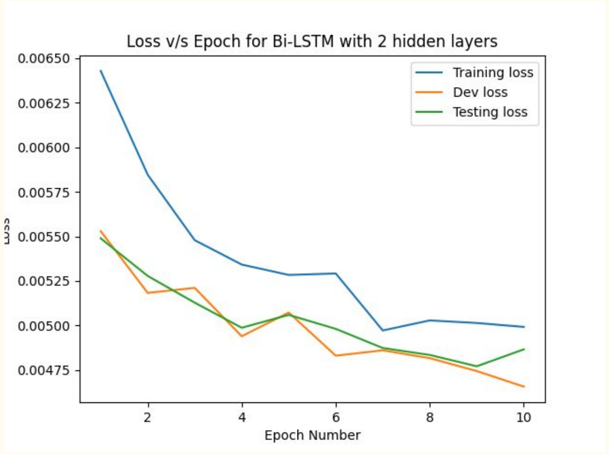
Experimental Details

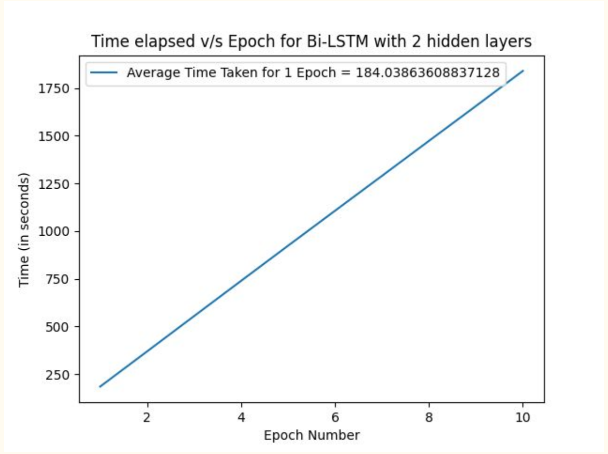
- When implementing the Bi-GRU model, all other model parameters and layers were kept the same as before.
- To be more precise, we also used GloVe : 300-dimensional embeddings as pre-trained word vectors.
- The Bi-GRU hidden size was set to 256.
- We used Adam to optimize the model, with a learning rate of 3e—4, decay rate of 1 and a decay interval of 200.
- The number of epochs for which the model was trained is 10 with a batch size of 128.
- The dropout rate was kept at 0.5 and number of attention heads was kept at 1.
- We also increased the number of layers within the Bi-LSTM as well as the Bi-GRU model from 2 to 3. We hypothesized that this would result in an increase in accuracy but at the cost of computational time.

Results

- After training the Bi-LSTM with 2 hidden layers, we were able to obtain a test accuracy close to 75.1, test loss of around 0.004834 and average test time for every epoch was around 184 seconds.
- Here are the corresponding plots obtained for accuracy, loss and test time :

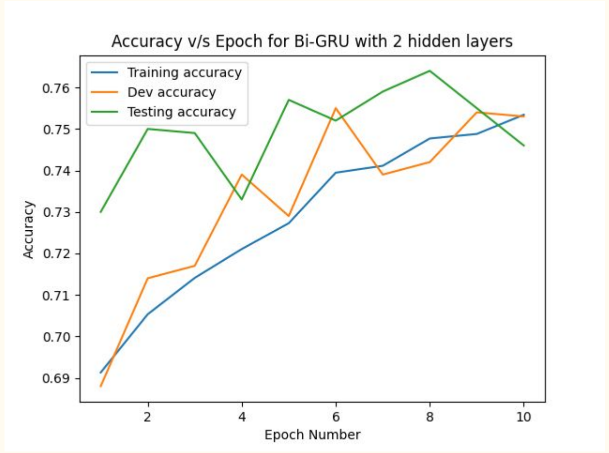


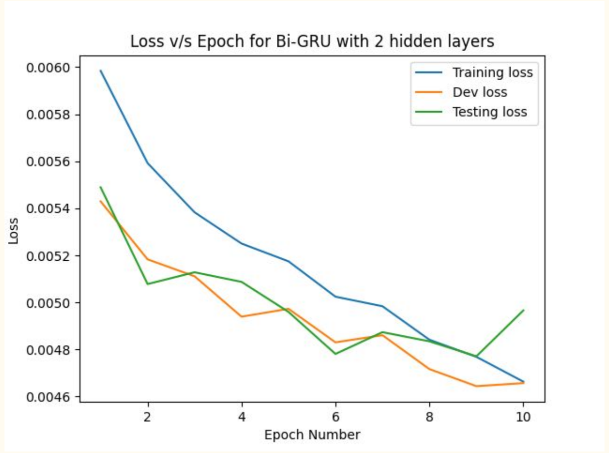


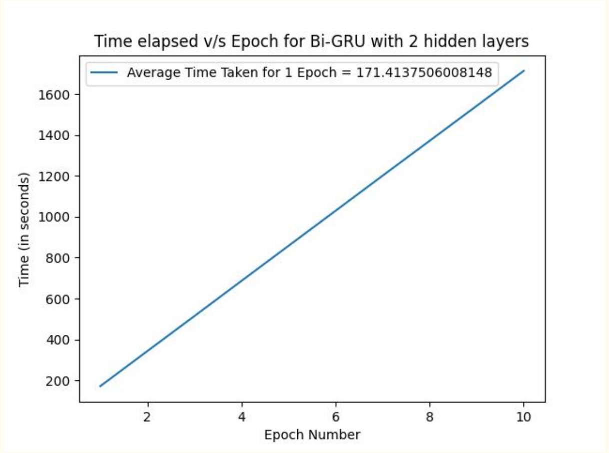


Results

- After training the Bi-GRU with 2 hidden layers, we were able to obtain a test accuracy close to 74.6, test loss of around 0.004966 and average test time for every epoch was around 171 seconds.
- Here are the corresponding plots obtained for accuracy, loss and test time :

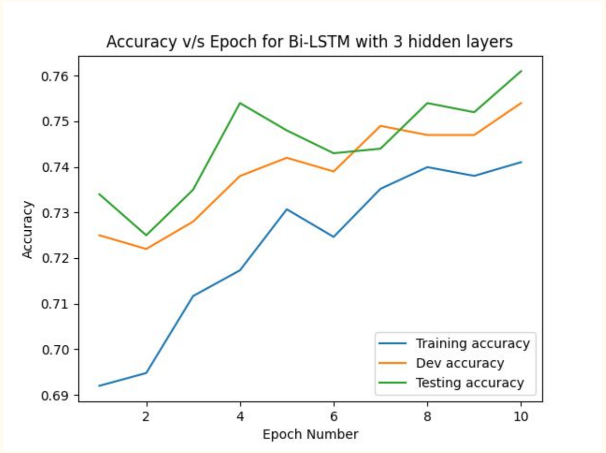


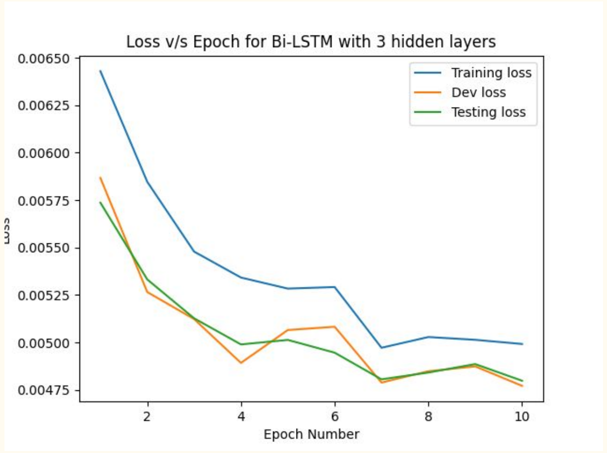


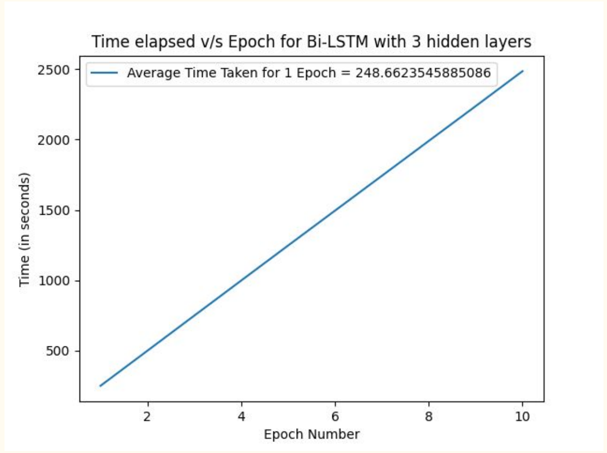


Results

- After training the Bi-LSTM with 3 hidden layers, we were able to obtain a test accuracy close to 76.4, test loss of around 0.00479 and average test time for every epoch was around 249 seconds.
- Here are the corresponding plots obtained for accuracy, loss and test time :

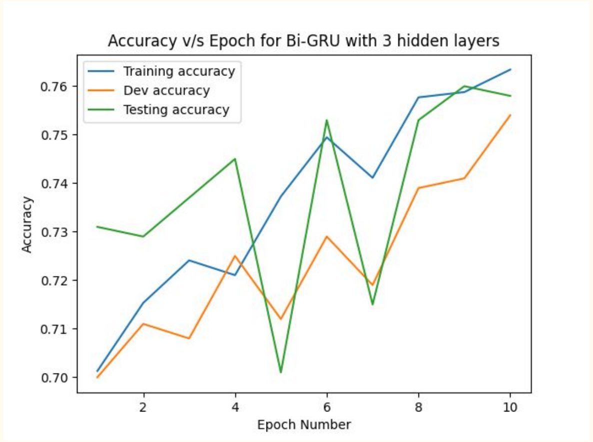


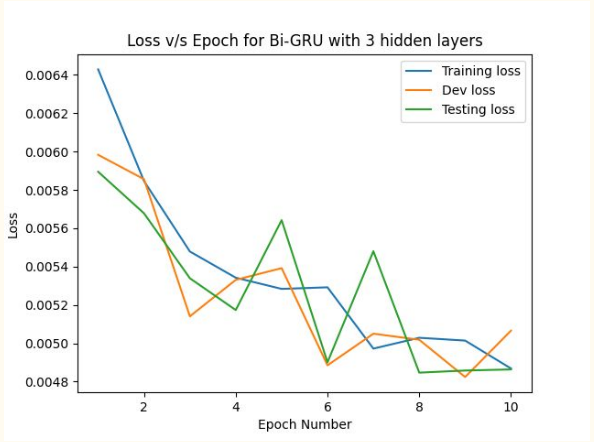


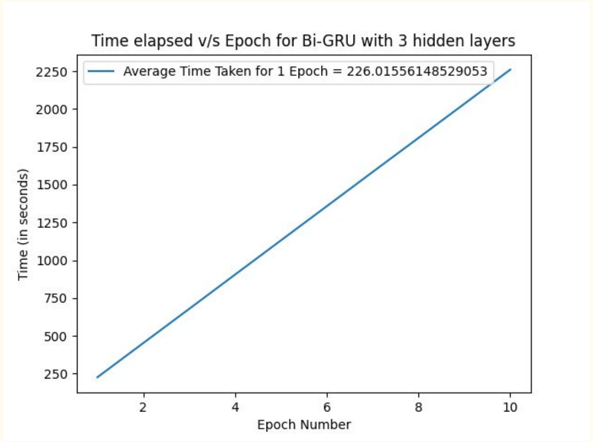


Results

- After training the Bi-GRU with 3 hidden layers, we were able to obtain a test accuracy close to 75.8, test loss of around 0.004824 and average test time for every epoch was around 226 seconds.
- Here are the corresponding plots obtained for accuracy, loss and test time :







Sarcasm Sentiment Analysis/Detection

- Sarcastic sentences are subjective, they carry sentiment and emotion-bearing information
- Convolutional neural network framework for learning sarcasm from a corpus(Using CNN helps integrate local and global aspects.)
- Pre-trained mood, emotion, and personality models may improve sarcasm detection.

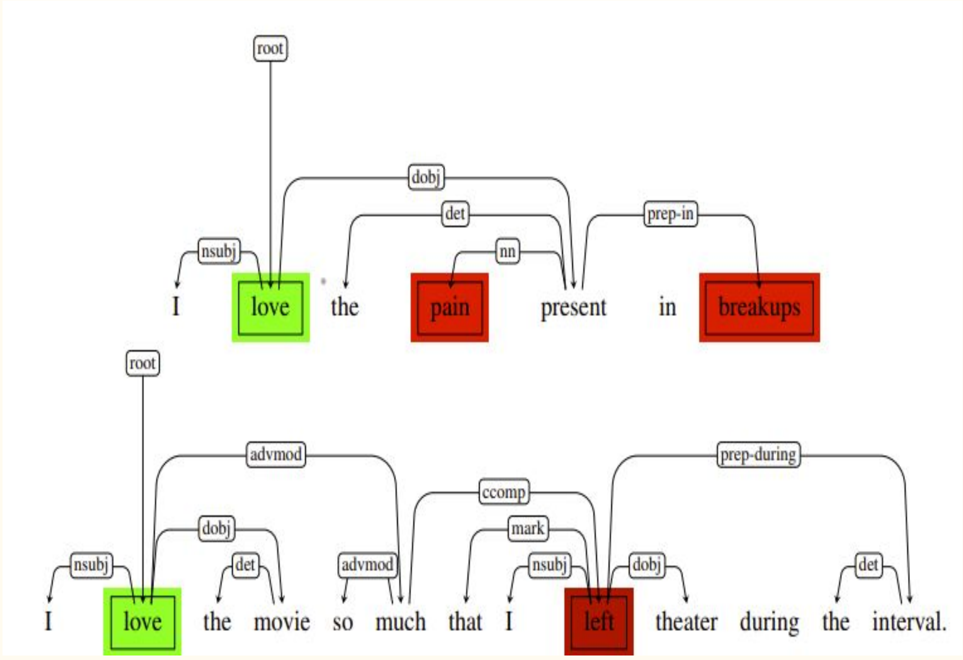
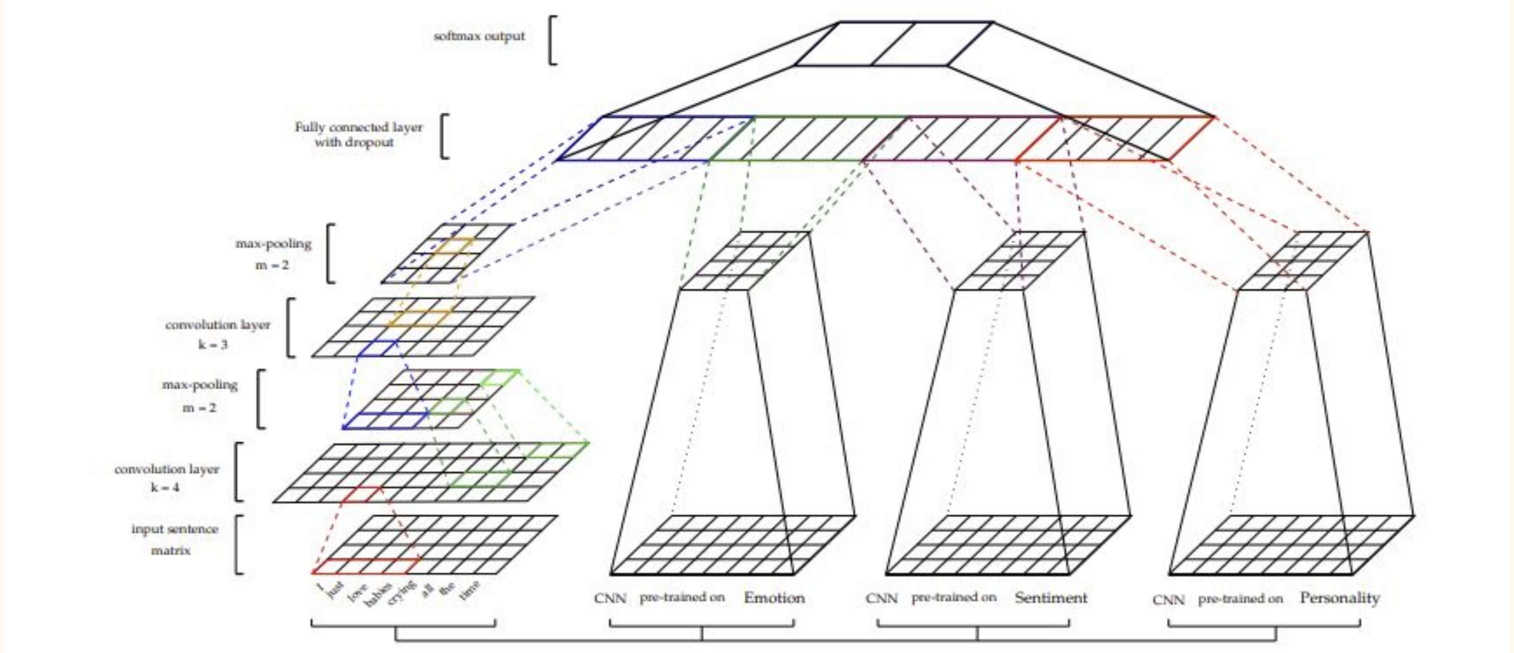


Figure 1: Sentiment shifting can be an indicator of sarcasm.

Framework Proposed

- We incorporate both sentiment and emotion clues, and the personality of the opinion holder in the framework.

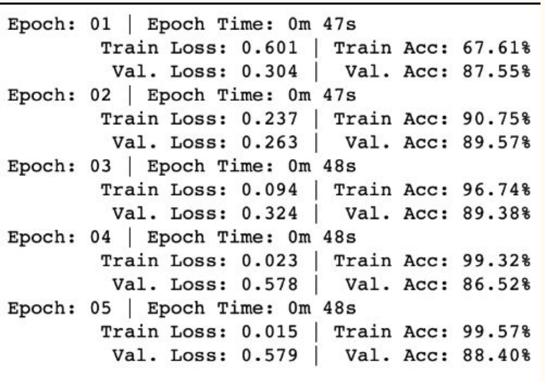


softmax output
Fully connected layer with dropout
max-pooling m = 2
convolution layer k = 3
max-pooling m = 2
convolution layer k = 4
input sentence matrix
I just love babies crying all the time
CNN pre-trained on Emotion
CNN pre-trained on Sentiment
CNN pre-trained on Personality

Figure 2: The proposed framework: deep CNNs are combined together to detect sarcastic tweets.

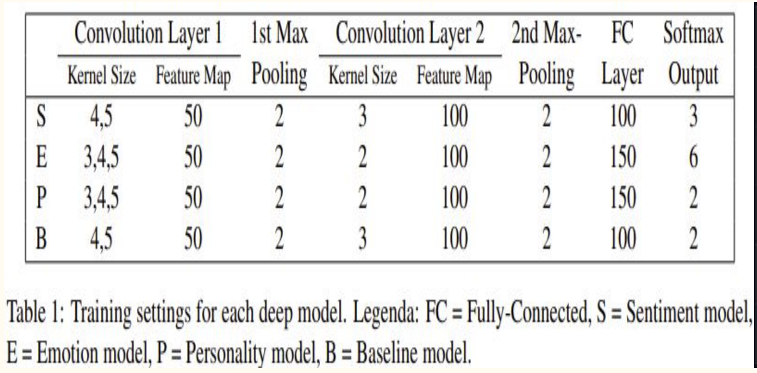
Dataset And Training Details

- We have obtained this dataset from "The Sarcasm Detector 2" . It contains 120,000 tweets, out of which 20,000 are sarcastic and 100,000 are non-sarcastic.
- Incorporate three model
  - Sentiment Feature Extraction Model : (positive, negative or neutral)
  - Emotion Feature Extraction Model : (Anger, Disgust, Surprise, Sadness, Joy and Fear)
  - Personality Feature Extraction Model : (Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism)
  - Baseline Model : CNN Architecture



```
Epoch: 01 | Epoch Time: 0m 47s
        Train Loss: 0.601 | Train Acc: 67.61%
         Val. Loss: 0.304 |  Val. Acc: 87.55%
Epoch: 02 | Epoch Time: 0m 47s
        Train Loss: 0.237 | Train Acc: 90.75%
         Val. Loss: 0.263 |  Val. Acc: 89.57%
Epoch: 03 | Epoch Time: 0m 48s
        Train Loss: 0.094 | Train Acc: 96.74%
         Val. Loss: 0.324 |  Val. Acc: 89.38%
Epoch: 04 | Epoch Time: 0m 48s
        Train Loss: 0.023 | Train Acc: 99.32%
         Val. Loss: 0.578 |  Val. Acc: 86.52%
Epoch: 05 | Epoch Time: 0m 48s
        Train Loss: 0.015 | Train Acc: 99.57%
         Val. Loss: 0.579 |  Val. Acc: 88.40%
```

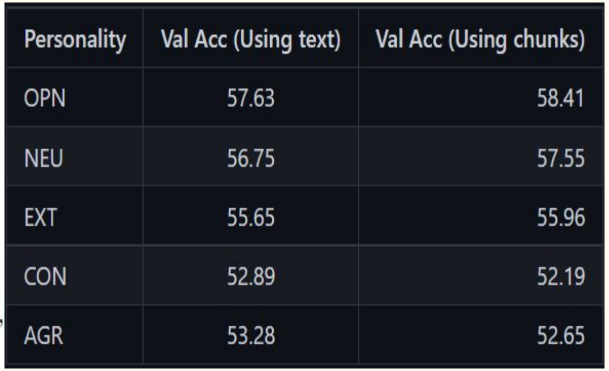
Fig 1 Sentiment Analysis



	Convolution Layer 1		1st Max	Convolution Layer 2		2nd Max-	FC	Softmax
	Kernel Size	Feature Map	Pooling	Kernel Size	Feature Map	Pooling	Layer	Output
S	4,5	50	2	3	100	2	100	3
E	3,4,5	50	2	2	100	2	150	6
P	3,4,5	50	2	2	100	2	150	2
B	4,5	50	2	3	100	2	100	2

Table 1: Training settings for each deep model. Legenda: FC = Fully-Connected, S = Sentiment model, E = Emotion model, P = Personality model, B = Baseline model.

Fig 2. Training Parameters

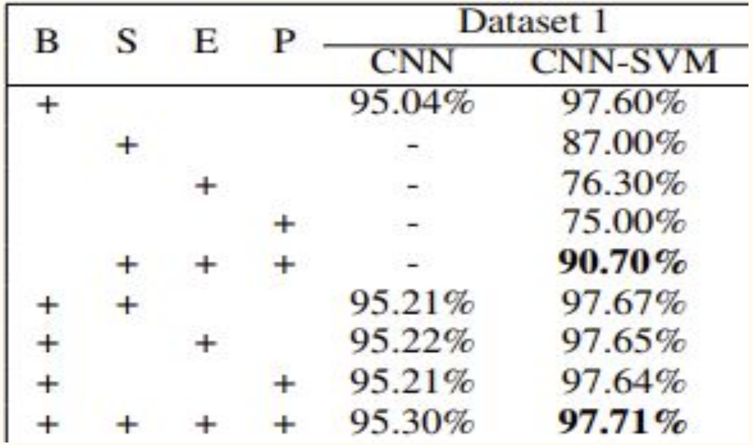


Personality	Val Acc (Using text)	Val Acc (Using chunks)
OPN	57.63	58.41
NEU	56.75	57.55
EXT	55.65	55.96
CON	52.89	52.19
AGR	53.28	52.65

Fig 3 Personality Analysis

Results

- In the conventional feature merging procedure, features are extracted from all deep CNN-based feature extraction models and then concatenated. SVM is then applied to the resultant feature vector.
- Use pre-trained model features as static CNN channels in another environment. These features are added to the CNN's hidden layer before the output softmax layer.



B	S	E	P	Dataset 1	
				CNN	CNN-SVM
+				95.04%	97.60%
	+			-	87.00%
		+		-	76.30%
			+	-	75.00%
	+	+	+	-	**90.70%**
+	+			95.21%	97.67%
+		+		95.22%	97.65%
+			+	95.21%	97.64%
+	+	+	+	95.30%	**97.71%**

Fig 2 : Conclusion : Baseline features outperform the pre-trained features for sarcasm detection. However, the combination of pre-trained features and baseline features beats both of themselves alone

Future Work

Things that can be explored to get more reliable and robust performances are:

- Current state of the art methods for various NLP problems, especially text classification, leverage attention based Transformer-based architectures like XL-Net. We can try a similar approach in an attempt to make a better use of review summary for effective sentiment classification.
- The summary generation model used in all of the models we trained was PG-Net. One can explore the possibility of replacing it with better and more state of the art summary generation model. This can definitely be a good research topic since meaningful summaries generally result in the correct sentiment getting classified.
- The accuracy of the model in correctly classifying sarcastic sentiments can also be researched upon.
- The sentiment distribution within the training set and its corresponding impact on the model's testing performance can be studied and explored.

Conclusions

- In this project, we researched and implemented solutions to the problem statement of better sentiment classification through the use of review summary.
- Yang et al., 2020 had proposed a review-centric attention model with Bi-LSTM at it's core. Upon training their model, for a total of 10 epochs on 1200 batches with every batch having a size of 128, on SNAP Amazon Review Dataset (Toys & Games) we were able to obtain a test accuracy close to 75.1%.
- The modification we proposed was substituting the Bi-LSTM block with Bi-GRU, which we hypothesized would reduce the parameters and the complexity of the network, therefore reducing training time without compromising on accuracy.
- Our expectations were met, as the Bi-GRU based model achieved almost the same accuracy on the test set while also taking around 8% less time to reach the same level of loss as the Bi-LSTM network had after 10 epochs.
- Therefore, Bi-GRUs offer a reliable method of achieving the same classification results with a less complex network architecture and training time.
- We also attempted training the network with different number of hidden layers. We observed that we were able to improve upon the accuracy by around 1.4% but this also resulted in 35% increase in computational time.

References

- https://sentic.net/sarcasm-detection-with-deep-convolutional-neural-networks.pdf
- https://snap.stanford.edu/data/web-Amazon.html
- https://arxiv.org/abs/1805.01089
- https://aclanthology.org/P16-1223/
- https://arxiv.org/abs/1611.01604