Training Deep Neural Networks to Rationalize with LIME Feedback

Nikhil Kadapala[*]
Department of Computer Science
University of New Hampshire
Nikhil.Kadapala@unh.edu

Devin Borchard[†]
Department of Computer Science
University of New Hampshire
Devin.Borchard@unh.edu

Abstract

Most Neural Networks in use today are black boxes that are hard to interpret due to their complex nature and high dimensionality. Interpretability of these Machine Learning Models helps human decision-makers understand how these models rationalize their behavior and verify the predictions ultimately building trust and improving safety, especially in high-stakes applications like medicine to diagnose patients. Through this study, we show how the black-box nature of these Neural Nets obscures us from the underlying spurious correlations they sometimes tend to make between the features that might not align with human reasoning and still manage to deliver predictions with high accuracy. This can be a problem when sensitive variables are involved and misclassifications can have catastrophic consequences. To that end, we evaluate a classifier designed using a standard Neural Network architecture with LIME [1] to show the discrepancy between the model's perceived high-importance features and human annotations in the training data. We then propose a custom Neural Net model incorporating a feedback layer from LIME Explanations and an attention layer that learns from the received feedback to minimize the loss between LIME Explanations and human annotations at several randomly chosen time steps in the training loop.

1 Task

1.1 What task are you solving?

Definition: Given input W, predict a class $\hat{Y}$ with the highest possible accuracy and an EA (Explanation- Annotation) Alignment score by training the model to learn to attend to features or parts of the input that align better with human rationale in making the predictions.

Inputs: The input training data fed to the model will have a text example, the corresponding classification label for the entry, and a list of human-given annotations in the text that explains why the text was classified so. The human-given annotations and examples will be given as pre-trained GloVe embeddings. However, the annotations are pre-processed to extract similar emotion words and phrases using clustering methods.

[*]There are four neural networks: two standard and two custom, with one of each implemented by both authors.

[†]The code and documentation are available at `https://github.com/Nikhil-Kadapala/NeuralNets` and `https://gitlab.cs.unh.edu/dmb1035/nn-final-project`

Outputs: The outputs are the class predictions and the EA alignment scores along with the annotations highlighting the top 5% features to inform the user of both classification confidence and underlying reasoning.

Reimplementation: This study does not propose any novel methods but intends to reproduce the ERASER [2] benchmark evaluation method by teaching the Neural Networks to Rationalize on their own during the training instead of training a classifier on extracted rationales [4]. The data set we're using with interpretability annotations can be found on the website of [2]. The interpretability evaluation method we will be implementing is LIME which was introduced in the 2016 paper Ribeiro et al. [1]

2 Motivation

2.1 How would this make the world a better place?

Domain: Interpretability AWS defines interpretability as: "The degree to which a human can understand the cause of a decision. The higher the interpretability of an ML model, the easier it is to comprehend the model's predictions. Interpretability facilitates Understanding, Debugging, and auditing ML model predictions, Bias detection to ensure fair decision-making, Robustness checks to ensure that small changes in the input do not lead to large changes in the output, Methods that provide recourse for those who have been adversely affected by model predictions"

Importance: Interpretability plays a vital role in making important decisions that must be transparent, fair, and accountable in high-stakes applications such as healthcare, finance, and law. Interpretability can help understand a model's behavior, detect biases, debug issues, and build trust in the model by revealing how it understands data and behaves in different scenarios.

Helpful: Models like linear regression or decision trees are considered more interpretable because they have clear relationships between input features and predictions. In contrast, neural networks are typically less transparent due to their multi-layered, complex black-box nature. Adding an interpretability functionality to neural networks can help gain an understanding of how the model behaves and makes decisions.

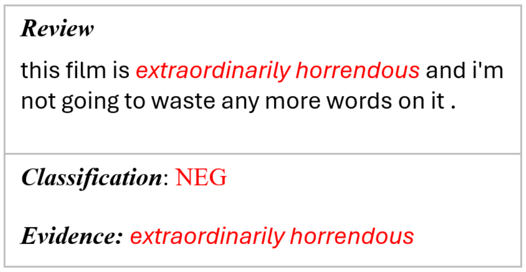
3 Dataset

Movie Reviews: We use the movie reviews data set from the ERASER benchmark [2]. This dataset contains reviews written by humans for movies. For each review, there is a classification label indicating if the review is positive or negative along with annotations provided by human evaluators highlighting the sentences or parts of the review that influenced the classification as seen in Figure 1.

Train/Test/Validation split: The dataset comes pre-split with 1600 training and 200 test and validation examples.

Suitability: The data comes with the classification labels required for the initial classification task along with human notations that mark the strings that led to the classification result. These annotations make the dataset perfect for measuring interpretability and alignment between the model's reasoning and human rationale.

Data Pre-processing: We will first convert the human annotations provided in the dataset to their corresponding GloVe embeddings and extract a feature set or rationales for each example. We will first build a dictionary of vocabulary with GloVe vectors of the words from all three datasets i.e. train, validation, and test. Then we will convert each example to its corresponding GloVe vector embedding to be used for training. We will load these dictionaries to local files to save processing time and faster access.


Review

this film is *extraordinarily horrendous* and i'm not going to waste any more words on it .

Classification: NEG

Evidence: *extraordinarily horrendous*

Figure 1: An example of a review with negative classification. The rationale for the predictions is shown under evidence

4 Standard Neural Network Approach

The text inputs will be tokenized, and converted to corresponding pre-trained word embeddings with imported word embedding libraries like GloVe.

4.1 Standard CNN implemented by Nikhil

We first approach our rational model development goal with a vanilla CNN for a typical sentiment analysis task[3] using the dataset described in the previous section. We hope that if we can achieve our goal with reasonable accuracy and stay aligned with human rationales, we can then advocate for future work to build on top of the idea, refine it, and apply it on a large scale for high-stakes applications like medicine, content moderation, etc. This is similar to the approach behind the work done in [3].

Architecture and Degrees of Freedom: The model will start without an embedding layer since we use pre-trained GloVe embeddings. We have three convolution layers with 64, 128, and 256 filters each with a kernel size of 3, 5, and 7 respectively. Each of the convolution layers is followed by ReLU activation, Batch Normalization, and dropout to ensure non-linearity, training stability, and appropriate fitting. A max-pooling layer will follow the convolution layers. Then we have a fully connected layer that gives us the raw logits for classification. These raw logits are then passed to the loss function i.e. BCEWithLogitsLoss which handles the sigmoid activation internallly and is more numerically stable than BCELoss.

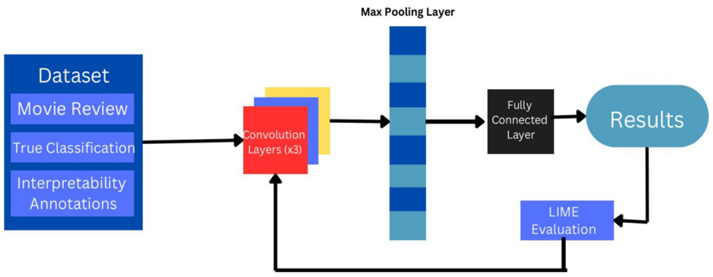


Figure 2: CNN with LIME Feedback

Training: We will normally train the CNN using Adam optimizer with a weight decay and learning rate scheduler to handle early plateaus in learning. We will apply LIME evaluation to generate explanations at a randomly chosen time step. We will then calculate a metric i.e., Explanation-Rationale Score to measure the alignment between LIME Explanations and Human Annotations. As long as the alignment score is less than the set acceptable threshold(0.5), we will retrain the model by adjusting the weights to assign more importance to the features that are missing from the LIME-generated explanations.

[3]The CNN models implemented in this paper are a variant of the model implemented in [5] the difference being sequential conv1d layers as opposed to parallel layers.

For simplicity's sake, we will not enforce the comprehensiveness of rationales but try to at least align the model's reasoning with the top 5% or n (n = number of rationales) features. A set of rationales that are missing from the LIME generated Explanations are identified by comparing them with the human annotations provided for the example in the dataset. A weight mask or filter matrix is created with zeros except for the indices of the words identified from the missing rationales set to a scaling factor. The model is then trained on that specific example by multiplying the final output of the convolution layers with the weight mask before passing through a pooling layer. The Loss calculated on this prediction is used for a single backward pass by the optimizer to compute gradients and adjust the weights. With this approach, we hope that the model eventually learns to find associations between features that align with the human rationales by at least 50%.

To look at the pseudocode for the forward function of the model described above and the training loop, refer to Appendix A.1 and A.2. Refer to the GitHub repository for the full implementation of the model.

4.2 Standard Transformer by Devin

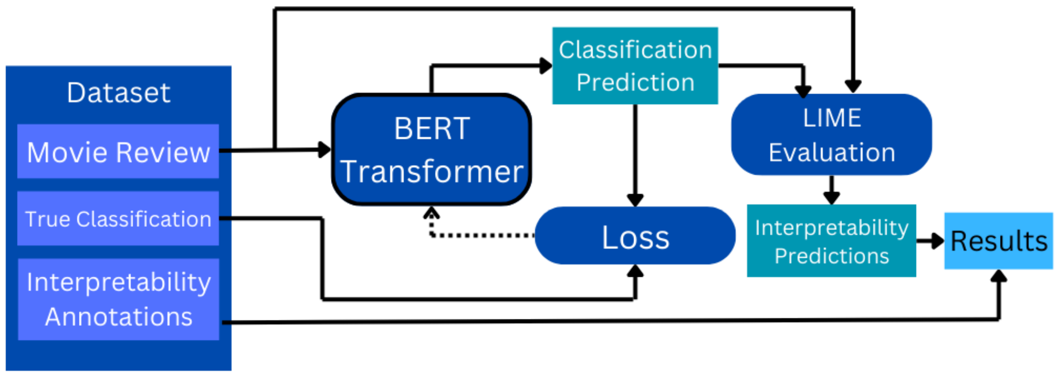
Transformers are the current state-of-the-art architecture used for NLP-related problems. They use the attention mechanism to solve the context issue present in CNNs.

Data Pre-processing: The preprocessing will be the same as the standard CNN outlined above.

Training: The transformer will be performing a classification and so will be using a log loss function for training.

Architecture: The transformer model will start as a general-purpose NLP BERT transformer from huggingface. The model will be fine-tuned with the training data and adjusted to give a single classification output. The LIME interpretability method will be used to measure the alignment of the model features used for making predictions, with the given human-made annotation in the dataset.

Code: The following is a diagram of the implementation of the standard transformer.



Dataset
Movie Review
True Classification
Interpretability Annotations
BERT Transformer
Classification Prediction
LIME Evaluation
Loss
Interpretability Predictions
Results

Figure 3: standard Transformer

5 Custom Neural Network Approach

5.1 Custom CNN with Dynamic Attention

The traditional CNN we discussed in the previous section can effectively capture the sentiments at the lower level by processing the phrases or continuous related words i.e. n-grams but will ultimately fail to keep track of the overall emotion of the entire text since the reviews are very long. We know that transformer-based models excel at capturing long-range dependencies in text using the Attention mechanism. To take advantage of the attention mechanism, we will integrate a dynamic attention layer in the standard CNN model that gets activated during training based on the alignment scores we calculate following the LIME Evaluation. We will then retrain the model this time including the attention layer outputs. The hope is that over time, the model learns to focus on the features extracted from the annotations and generalizes well on the test data.

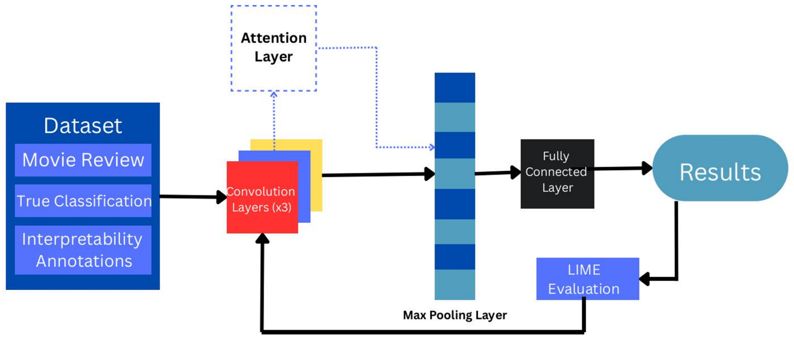


Figure 4: CNN with Attention

Training: The training follows the same flow as the standard model except for an attention layer following the convolution output that is multiplied by the feature weights mask. This attention layer performs self-attention on the convolution output using Multi-head attention with heads equal to the number of output channels of the final convolution layer.

5.2 Custom Interpretable Transformer

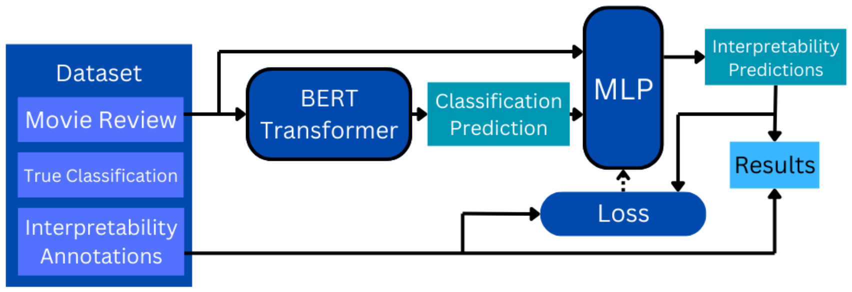

Dataset
Movie Review
True Classification
Interpretability Annotations
BERT Transformer
Classification Prediction
MLP
Interpretability Predictions
Results
Loss

Figure 5: Custom Transformer

One of the issues with any NLP model in the domain of interpretability is that the interpretability method used, LIME in this case, will often up weight and highlight features that humans would associate with the correct classification. Ideally, the interpretability methods output would match the human annotations. To try to achieve this well be using a similar transformer model as the standard approach above, but well be adding another layer that is trained on the loss between the human annotations and predicted annotations. The alignment score of this model is measured by the final MLP output against the human given annotations in the dataset

6 Results

6.1 Table with results from evaluation

	AP	MR	CA
Standard CNN	0.69±0.23	0.78±0.49	0.82
Attention CNN	0.73±0.22	0.83±0.49	0.82
Standard Transformer	0.37±0.29	0.88±0.10	0.86
Custom Transformer	0.40±0.27	0.94±0.06	0.86

Table 6.1 shows the scores of alignment between the LIME generated Explanations and human-annotated rationales, misalignment between explanations and rationales i.e. the LIME explanations

that are not part of the human rationales annotated for that example, and the classification accuracies of all four models.

The Alignment score (AS) is the fraction of rationales that have at least one match with any feature in the explanations.AS represents how well a set of "explanations" align with a set of "rationales".

Misalignment Rate (MR) quantifies how well the provided explanations align with the rationales. A higher value indicates more mismatched features and a lower value indicates that the model's reasoning is consistent with human understanding.

AS and MR act as proxies for the explanations' quality and trustworthiness of the model respectively. CA is the classification accuracy of the model. More details on how AS and MR are calculated are provided in Appendix A.3.

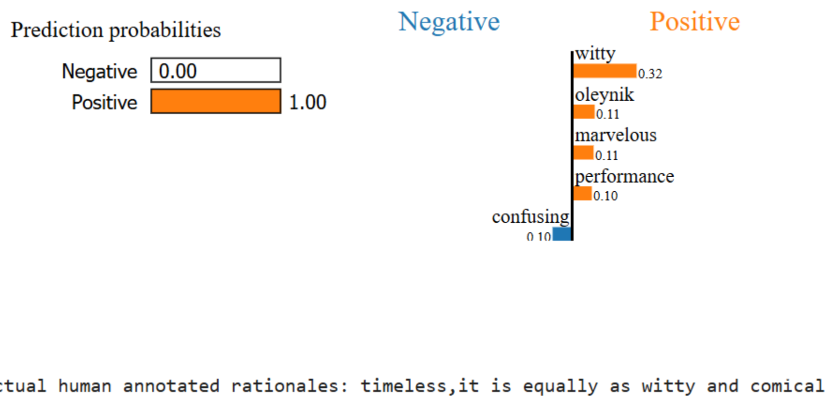


Figure 6: LIME Explanation of Standard CNN Classification

We can see in Figure 6 from the LIME Explanations that a standard CNN classifier can predict a class or sentiment of the review with absolute certainty for reasons that do not necessarily align with the provided human annotations. In the above example, the model assigns high feature importance to the word '*witty*' which is a part of the annotation but also assigns high importance to other words that are not part of the human rationale even though meaningful and correct in this context such as '*marvelous*.

The goal here is to teach or train the model to make predictions based on a feature set that closely aligns with human rationales.

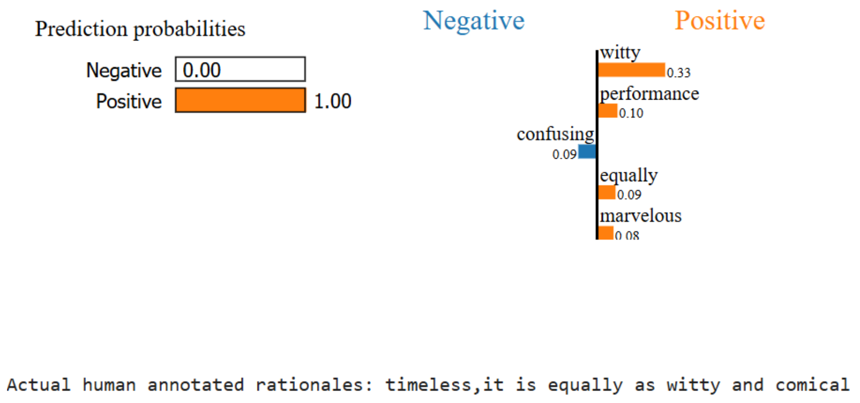
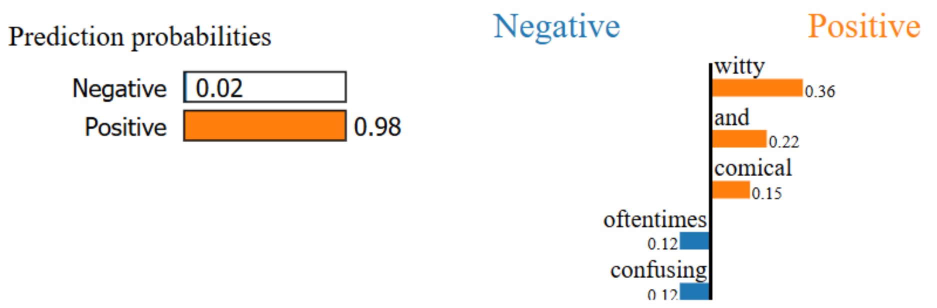


Figure 7: LIME Explanation after Feature Weight Scaling

Figure 7 shows the LIME explanations generated for the same example after retraining the model with scaled feature weights mask applied over the convolution output. After the retraining, we now see different words appear in the explanations where the newly added word '*equally*' is a part of the rationales. This shows an improvement in the model's performance. Since we've only generated the explanations based on the top 5 features, we don't see other features whose importance or attribution scores have also increased compared to the base classifier.



```
Actual human annotated rationales: timeless,it is equally as witty and comical
```

Figure 8: LIME Explanation after re-training with Attention

Figure 8 shows the explanations generated using the custom CNN classifier re-trained with an additional dynamic attention layer that processes the feature weight scaled convolution output at random epochs during training. Here we can see a slightly improved result with three consecutive words from a rationale in the explanations i.e. '*witty and comical*' that reinforces the fact that the attention mechanism indeed is better at capturing the dependencies between words. Though not a big improvement on the base results, the model's reasoning after using an additional attention layer better aligns relatively with human understanding, forming associations with coherent and meaningful n-grams. Notice that the prediction probability has slightly reduced and this indicates that there must be a trade-off between accuracy and interpretability of the model.

7 Conclusions

Despite some improvements observed in the model's reasoning through LIME Explanations, the behavior was found to be inconsistent and was highly dependent on the length of the input review and human-annotated rationale set. The model still made associations with meaningless and incoherent features besides those that do not align with the human rationale though not necessarily in the top 5 range. This is not an acceptable performance and even the improvements observed are not statistically significant to conclude that the proposed approach can effectively teach the black-box deep neural networks to rationalize during training and generalize well on unseen data.

In conclusion, some model approaches are easier to interpret than others. Although transformer-based models had higher classification accuracy, CNN models were more interpretable. CNN's AP scores were around 50% better than the transformers on average, while also having a marginally smaller standard deviation.

Although the MRs were comparable between models, the CNNs once again slightly outperformed the transformers. However, neither model's MR were what I would consider acceptable. For this method to be practical, the predictions should not be misaligned this frequently. This high misalignment rate suggests that the model's high alignments are a result of over-guessing.

8 Future Work

One of the serious limitations of the approach that we have followed here lies in the way we have handled our rationale set in the re-training phase. A better approach would have been to retrain the model with scaled weights for features that are more meaningful, emotionally charged, and less frequent across examples and avoid most commonly found words like pronouns, prepositions, auxiliary verbs, and conjunctions.

We believe that this would considerably improve the model's reasoning over the base results. Simply upscaling every word from the rationale set that is not present in the LIME explanations could have overamplified the importance of irrelevant and frequent words resulting in a lower alignment with the human rationale set.

Future work can be done by adopting a more clean and streamlined approach using a pre-trained language model or the custom CNN with dynamic Attention by focusing only on a set of important words extracted using a POS tagger and clustering methods.

Hopefully, this can achieve a better feature alignment with human reasoning by inherently learning to reason like a human and generalize well with an equally balanced accuracy, unlike the ERASER benchmark that explicitly trains on the extracted rationales. If it does not, then it is obvious why the straightforward approach adopted by the ERASER benchmark works better by training only on the rationales that are sufficient and comprehensive enough to make an accurate classification.

References

[1] Marco Tulio Ribeiro and Sameer Singh and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv:1602.04938*, 2016.

[2] DeYoung, Jay, et al. ERASER: A Benchmark to Evaluate Rationalized NLP Models. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[3] Zaidan et al. Using "Annotator Rationales" to Improve Machine Learning for Text Categorization. *The Conference of the North American Chapter of the Association for Computational Linguistics* 2007.

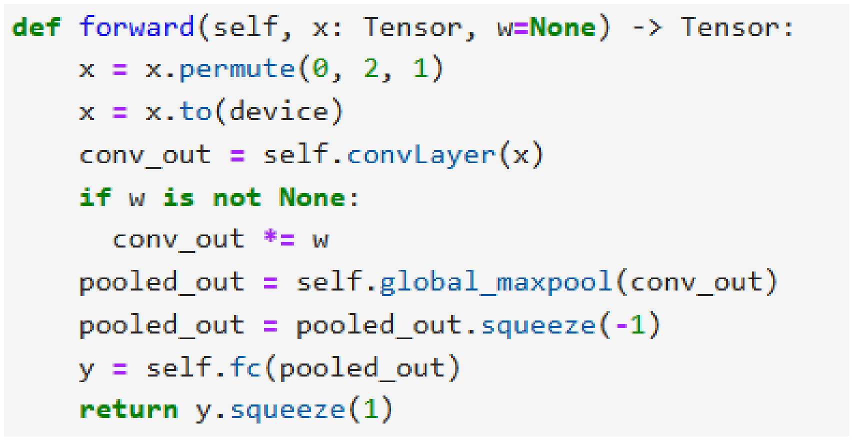
[4] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 107–117* 2016.

[5] Sentiment Analysis: Using Convolutional Neural Networks *d2l.ai-chapter16.3*

[6] lime, Marco Tulio Ribeiro and Sameer Singh and Carlos Guestrin github.com/marcotcr/lime

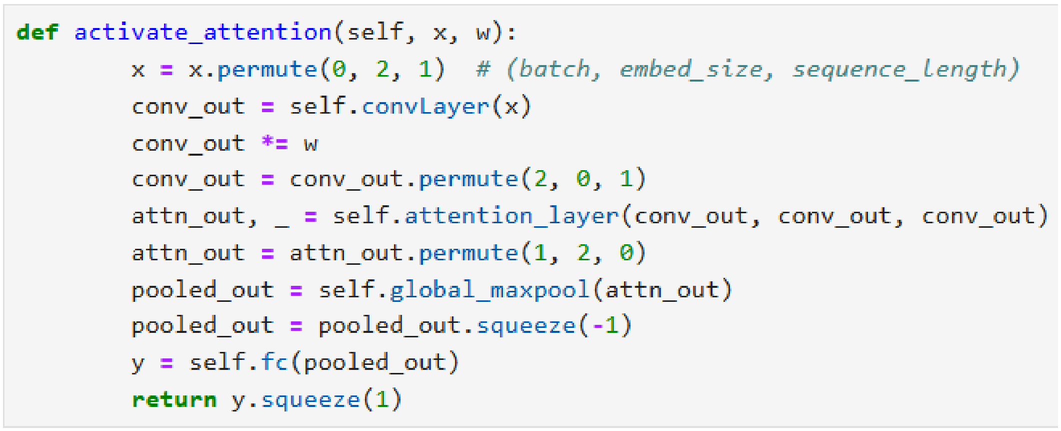
A Appendix

A.1 Pseudocodes



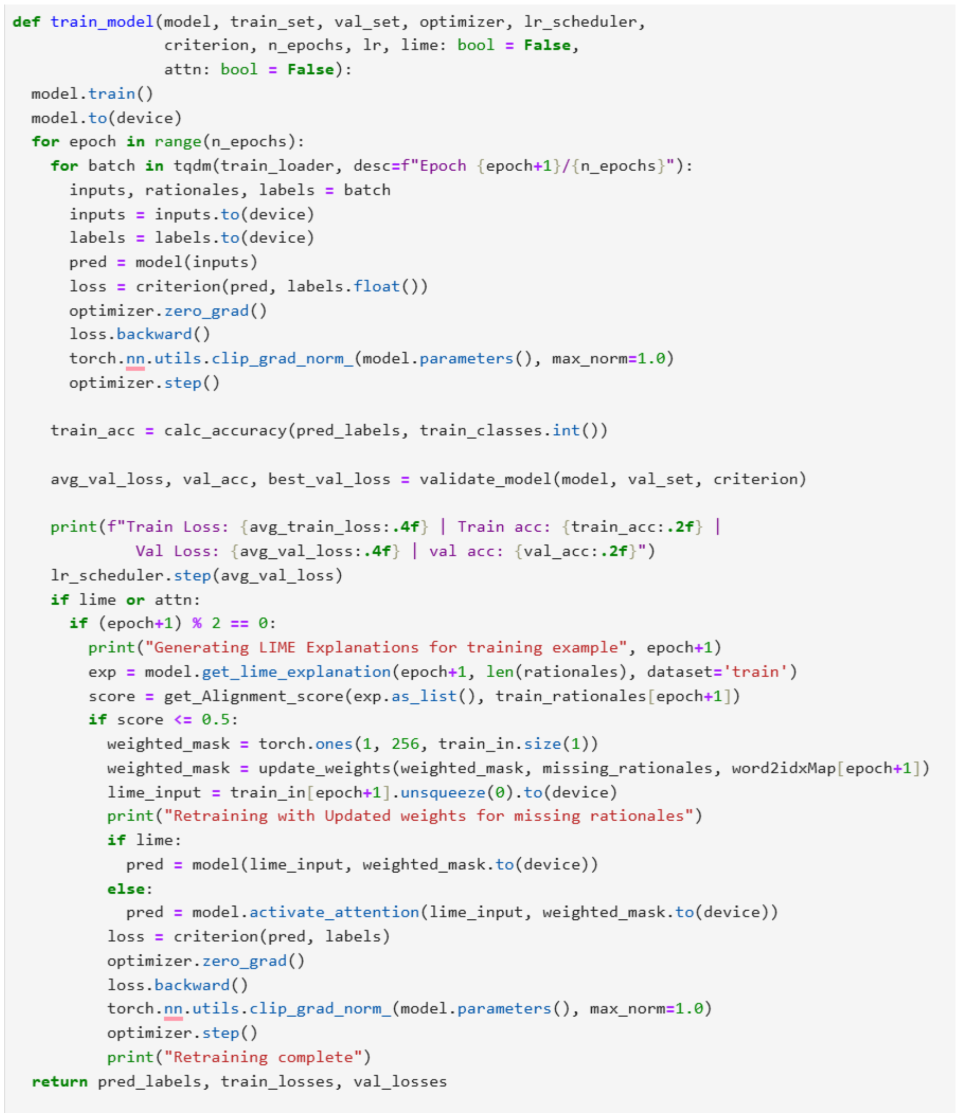
```
def forward(self, x: Tensor, w=None) -> Tensor:
    x = x.permute(0, 2, 1)
    x = x.to(device)
    conv_out = self.convLayer(x)
    if w is not None:
      conv_out *= w
    pooled_out = self.global_maxpool(conv_out)
    pooled_out = pooled_out.squeeze(-1)
    y = self.fc(pooled_out)
    return y.squeeze(1)
```

Figure 9: Standard CNN Model



```python
def activate_attention(self, x, w):
        x = x.permute(0, 2, 1)  # (batch, embed_size, sequence_length)
        conv_out = self.convLayer(x)
        conv_out *= w
        conv_out = conv_out.permute(2, 0, 1)
        attn_out, _ = self.attention_layer(conv_out, conv_out, conv_out)
        attn_out = attn_out.permute(1, 2, 0)
        pooled_out = self.global_maxpool(attn_out)
        pooled_out = pooled_out.squeeze(-1)
        y = self.fc(pooled_out)
        return y.squeeze(1)
```

Figure 10: Custom CNN Model



```python
def train_model(model, train_set, val_set, optimizer, lr_scheduler,
                criterion, n_epochs, lr, lime: bool = False,
                attn: bool = False):
  model.train()
  model.to(device)
  for epoch in range(n_epochs):
    for batch in tqdm(train_loader, desc=f"Epoch {epoch+1}/{n_epochs}"):
      inputs, rationales, labels = batch
      inputs = inputs.to(device)
      labels = labels.to(device)
      pred = model(inputs)
      loss = criterion(pred, labels.float())
      optimizer.zero_grad()
      loss.backward()
      torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
      optimizer.step()

    train_acc = calc_accuracy(pred_labels, train_classes.int())

    avg_val_loss, val_acc, best_val_loss = validate_model(model, val_set, criterion)

    print(f"Train Loss: {avg_train_loss:.4f} | Train acc: {train_acc:.2f} |
             Val Loss: {avg_val_loss:.4f} | val acc: {val_acc:.2f}")
    lr_scheduler.step(avg_val_loss)
    if lime or attn:
      if (epoch+1) % 2 == 0:
        print("Generating LIME Explanations for training example", epoch+1)
        exp = model.get_lime_explanation(epoch+1, len(rationales), dataset='train')
        score = get_Alignment_score(exp.as_list(), train_rationales[epoch+1])
        if score <= 0.5:
          weighted_mask = torch.ones(1, 256, train_in.size(1))
          weighted_mask = update_weights(weighted_mask, missing_rationales, word2idxMap[epoch+1])
          lime_input = train_in[epoch+1].unsqueeze(0).to(device)
          print("Retraining with Updated weights for missing rationales")
          if lime:
            pred = model(lime_input, weighted_mask.to(device))
          else:
            pred = model.activate_attention(lime_input, weighted_mask.to(device))
          loss = criterion(pred, labels)
          optimizer.zero_grad()
          loss.backward()
          torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
          optimizer.step()
          print("Retraining complete")
  return pred_labels, train_losses, val_losses
```

Figure 11: Training loop

A.2 Classification Metrics

As seen from the confusion matrices in both cases of the standard and custom models, there is no significant difference or improvement in the classification accuracy on the test dataset.

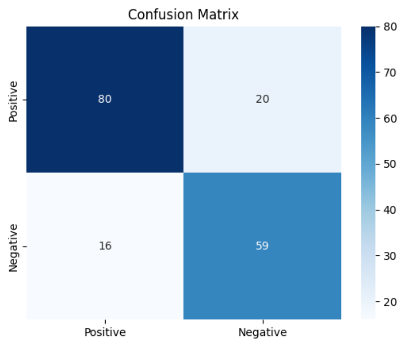


Figure 12: CNN with feature prioritization

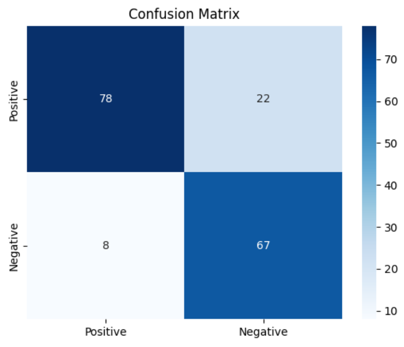


Figure 13: CNN with feature prioritization and Attention

Figure 14: Classification metrics for both CNN models

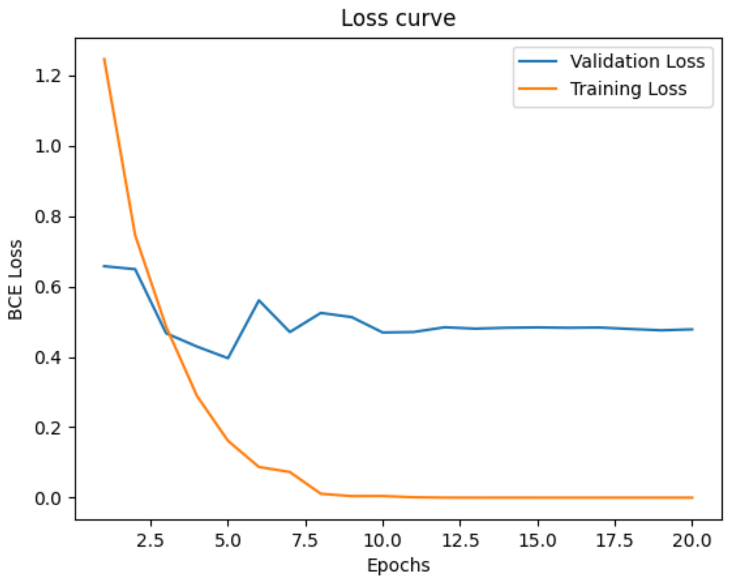


Figure 15: Training and validation loss during training

Though the training loss improved closer to zero, the training accuracy never improved beyond 52% which we believe is due to the lack of diversity of training data and the small size of the dataset(1600 examples). But for the learned parameters, the model was able to generalize well on relatively much smaller validation and test datasets with just 200 examples each. As mentioned in the earlier section, we tried implementing the CNN model with parallel conv1d layers similar to [5] but the model quickly overfit on training data with 92% accuracy in as low as 5 epochs and never generalized above 51% accuracy on the held-out data.

A.3 AS, MR, and CA

AP - Annotation prediction (mean $\pm$ standard deviation) It is unrealistic to expect any algorithm or model to align perfectly with the human annotations. For this reason, we came up with our own metric to measure the prediction accuracy of the annotations. For the sake of this project, we decided that if any portion of the annotation was present in the explanations then the annotation was considered to be successfully found. In the example of Figure 1, the annotation is *extraordinarily horrendous*. If either *extraordinarily* or *horrendous* were found in the explanation then the AP for that example would be a 1.0. The total AP for a given example was calculated by

$$(FA)/(TA)$$

where FA = number of found annotations, and TR = total number of rationales in the example

MR - Misalignment Rate (mean $\pm$ standard deviation)

Misprediction Rate is a measurement of how inaccurate the explanations are. This is calculated as

$$(WE)/(TE)$$

where WE = number of wrong explanations and TE = number of total explanations. If all predicted explanations were aligned with the annotations then MR would calculate to 0.0. Alternatively, if none of the explanations were aligned with annotations the MR would result in 1.0

CA - Classification Accuracy The classification accuracy is the performance of the base model when doing the sentiment analysis classification task on the dataset. This measurement is not related to the annotation alignment scoring.