Reproduction of Learned in Translation: Contextualized Word Vectors

Yasmine Khajjou, Diego Rejas, Malak Matar
TUW - Deep Learning for Natural Language Processing 2024W

Abstract
This report presents an implementation and reproduction of the paper "Learned in Translation: Contextualized Word Vectors" by McCann et al[5]. The paper introduces CoVe (Contextualized Word Vectors), which are generated using a deep bidirectional LSTM[3] (BiLSTM) encoder trained on machine translation tasks. These vectors aim to enhance the performance of NLP models by providing contextual word representations.

Our implementation focuses on recreating the attentional sequence-to-sequence architecture described in the paper, using GloVe embeddings for input representation and a BiLSTM encoder. We carefully designed a pipeline that includes robust preprocessing, integration of pretrained embeddings, and support for flexible encoder-decoder architectures. The end goal was to reach the results presented at the original paper.

Contents

1 Introduction

In natural language processing (NLP), contextual word representations have emerged as powerful tools for enhancing the understanding of textual data. Which led to the increased use of Transfer learning by reusing pretrained models to address various downstream tasks. This project focuses on the implementation and reproduction of the work presented in the paper *"Learned in Translation: Contextualized Word Vectors"* by McCann et al., which introduces Contextualized Word Vectors (CoVe). These vectors are generated using a deep bidirectional Long Short-Term Memory (BiLSTM) encoder trained on machine translation tasks.

Building on the success of transfer learning in computer vision, pretrained convolutional neural networks (CNNs) have become the standard for feature extraction and performance enhancement, McCann et al. extended this paradigm to NLP by leveraging machine translation (MT) as a source domain. Their approach integrates pretrained static word embeddings, such as GloVe, with dynamic, context-sensitive representations derived from a BiLSTM encoder trained on MT tasks. This integration shows significant performance improvements across a variety of NLP tasks, including sentiment analysis, natural language inference, and question answering.

To evaluate the effectiveness of CoVe embeddings, the authors employed the following methodology:

- Authors develop Biattentive Classification Network for classification tasks on datasets such as the Stanford Sentiment Treebank (SST) and the Stanford Natural Language Inference Corpus (SNLI),
- For question answering, the Dynamic Coattention Network (DCN)[10] was utilized, incorporating the outputs of MT-trained BiLSTMs (CoVe) alongside word vectors.

Moreover, experiments revealed a strong positive correlation between the size of the MT dataset used to train the BiLSTM encoder and its performance on downstream tasks. This finding highlights a key advantage of using MT as a source domain: the abundance of high-quality training data. Larger and more diverse MT datasets produce encoders with richer contextual understanding, reinforcing the hypothesis that MT is an ideal task for developing NLP models with a deeper grasp of natural language semantics and syntax.

The primary goal of this report is to replicate the attentional sequence-to-sequence architecture outlined in the paper and validate its findings. This includes designing a pipeline with preprocessing, integration of GloVe embeddings, and implementing a BiLSTM encoder to generate contextual word vectors. Furthermore, this report explores potential expansions to the methodology with the aim of achieving results that exceed those reported in the original study or compare key changes with other approcahes.

2 Methodology

2.1 Dataset

The implementation and reproduction of the CoVe architecture rely on datasets designed for both machine translation and downstream NLP tasks.

2.1.1 Machine Translation Datasets

The BiLSTM encoder at the core of CoVe is pretrained on machine translation tasks using English-German translation datasets of varying sizes:

- **MT-Small (Multi30k)** [4]: Contains approximately 30,000 English-German sentence pairs derived from Flickr captions. These captions are shorter and simpler, representing a restricted linguistic domain.
- **MT-Medium (IWSLT)** [2]: Comprises roughly 210,000 sentence pairs from transcribed TED talks, covering a wide variety of topics with conversational language.
- **MT-Large (WMT)**: Includes approximately 7 million sentence pairs sourced from diverse domains such as news articles, European Parliament proceedings, and press releases. This dataset captures complex and varied language use.

2.1.2 Downstream NLP Datasets

The effectiveness of CoVe is evaluated on several NLP tasks, utilizing the following benchmark datasets:

- **Stanford Sentiment Treebank (SST-2 and SST-5)** [8]: Sentiment classification dataset, including binary (positive/negative) and fine-grained (five-class) labels, derived from movie reviews.
- **IMDb**: A sentiment classification dataset consisting of 25,000 multi-sentence movie reviews.
- **TREC (TREC-6 and TREC-50)** [9]: Question classification datasets with six and fifty semantic categories, respectively.
- **Stanford Natural Language Inference (SNLI)** [1]: A dataset for entailment classification, consisting of over 500,000 examples labeled as entailment, contradiction, or neutral.
- **Stanford Question Answering Dataset (SQuAD)** [7]: A large-scale question answering dataset where answers are spans extracted directly from Wikipedia paragraphs.

The diversity and scale of these datasets provide a comprehensive evaluation of CoVe's transferability and its ability to capture contextual word representations that generalize across tasks.

Due to the limited computing power that we had we worked on the Stanford Sentiment Treebank and the IMDB datasets. And we trained the model using 1% of the WMT dataset which rounds to about 40,000 sentence pairs.

2.2 Data Preprocessing

The following steps outline a pipeline for processing text data to ensure consistency and compatibility with machine learning models:

Data Cleaning: Raw text data is cleaned to remove noise and normalize formatting. This includes:

- Replacing numerical values with placeholder tokens (e.g., `<num>`).
- Removing extraneous spaces, special characters, and punctuation where appropriate.
- Standardizing the text by lowercasing and normalizing Unicode.

Tokenization: Sentences are split into smaller units such as words, subwords, or characters, depending on the model requirements. A tokenizer is used to convert text into a sequence of tokens.

Special Tokens: Special tokens are added to the text to provide structure and context for sequence-based models. Common examples include:

- `<BOS>` (beginning of sequence) and `<EOS>` (end of sequence) tokens.
- `<PAD>` tokens for padding shorter sequences.
- `<UNK>` tokens for handling out-of-vocabulary words.

Vocabulary Construction: A vocabulary is built from the tokenized data, with a predefined maximum size to reduce computational complexity. Tokens appearing below a frequency threshold are typically replaced with the `<UNK>` token.

Sequence Padding and Truncation: To ensure uniform input lengths for batch processing, sequences are either:

- Padded with `<PAD>` tokens at the end of shorter sequences, or
- Truncated to a fixed maximum length to prevent excessively long sequences.

Embedding Initialization: Pretrained word embeddings are used to initialize word vectors.

- GloVe embeddings provide initial static word representations.
- CoVe embeddings, learned from a BiLSTM encoder trained on machine translation, are concatenated with GloVe embeddings to enhance contextual word representations.
- Tokens not present in the pretrained embeddings are assigned randomly initialized vectors, while special tokens are given unique representations.

Dataset Splitting: The dataset is divided into training, validation, and test sets to evaluate model performance effectively.

Batch Preparation: The preprocessed data is divided into batches for efficient training. Dynamic padding is then applied within each batch to minimize unnecessary computations by adapting to the length of the longest sequence in the batch.

This generalized preprocessing pipeline ensures that raw text data is transformed into a structured and model-ready format. It not only improves the quality and consistency of input data but also enables seamless integration with downstream NLP tasks.

However, since datasets have different formats, some adaption had to be applied to preprocess specific datasets accordingly. For example, while the WMT dataset was quite easier to split into training, testing, and validation sets. The SST2 and IMDb required more work and steps to get to that point since it was a binary classfication task consisting of sentences and labels as a positve or negative sentiment/review. So, it required a slightly different approach to preprocesse.

3 Model Overview

The core components of the neural machine translation (NMT) system are the encoder, decoder, and the overall NMT model. These components collectively implement the attentional sequence-to-sequence architecture described in the original paper. Additionally, the Biattentive Classification Network (BCN) is used as an alternative downstream architecture to evaluate the transferability of contextualized word representations like CoVe to classification tasks. The following details outline their design and functionality:

3.1 Encoder

The encoder is a bidirectional Long Short-Term Memory (BiLSTM) network designed to capture contextualized representations of input sequences. Key features of the encoder include:

- **Embedding Layer**: Input tokens are mapped to dense vector representations using pre-trained GloVe embeddings. Words not present in the embedding vocabulary are assigned random vectors.
- **BiLSTM Layers**: A two-layer BiLSTM processes the input embeddings, capturing both forward and backward dependencies in the sequence. The hidden size of the LSTM is halved for bidirectional operation.
- **Dropout Regularization**: Dropout with a rate of 0.2 is applied to both the input embeddings and the outputs of the BiLSTM layers to prevent overfitting.

The encoder outputs a sequence of hidden states and the final hidden representation, which are passed to the decoder.

3.2 Contextualized Word Vectors (CoVe)

The outputs of the MT-LSTM are leveraged as context-aware word representations, known as Contextualized Word Vectors (CoVe). These vectors capture both the semantic meaning of individual words and their contextual relationships within a sequence.

Let w be a sequence of words, and $\text{GloVe}(w)$ represent the corresponding static word embeddings generated by the GloVe model. The CoVe representations are computed by passing the

GloVe embeddings through the MT-LSTM, as defined below:

$$\text{CoVe}(w) = \text{MT-LSTM}(\text{GloVe}(w)).$$

This formulation enriches static word embeddings with dynamic, context-sensitive features, enabling downstream NLP models to achieve improved performance on tasks such as sentiment analysis, natural language inference, and question answering.

3.3 Decoder

The decoder is also implemented as an LSTM network with an attention mechanism to dynamically focus on relevant parts of the encoder's output during generation. Its components include:

- **Input Embedding**: Target tokens are embedded using a trainable embedding layer initialized with random vectors.
- **Attention Mechanism**: A global attention mechanism computes relevance scores for each encoder hidden state, enabling the decoder to focus on specific input tokens at each timestep.
- **Concatenated Inputs**: At each timestep, the embedded target token is concatenated with the context vector computed from the attention mechanism and passed as input to the LSTM.
- **Dropout Regularization**: Dropout with a rate of 0.2 is applied to both the inputs and outputs of the LSTM.

The decoder produces a sequence of context-adjusted hidden states that are transformed into a probability distribution over the target vocabulary.

3.4 NMT Model

The NMT model combines the encoder and decoder into a unified framework for sequence-to-sequence translation. Key features include:

- **Output Linear Layer**: A fully connected layer maps the decoder's output to the target vocabulary size, producing logits for each token.
- **Softmax Activation**: A log-softmax function is applied to the logits to generate probabilities over the target vocabulary.
- **Bidirectionality Handling**: The hidden states of the bidirectional encoder are adjusted to match the decoder's input expectations.

This model architecture enables the system to encode the input sequence into a high-dimensional representation and generate translations one token at a time, attending to relevant parts of the input dynamically.

3.5 Translation training

The graph provides insights into the training and validation performance of the model. The Training and Validation Accuracy curve shows that the model achieved a steady increase in accuracy over epochs, reaching a peak of 0.88 for training accuracy and slightly lower for validation accuracy. This indicates that the model was able to learn effectively without significant overfitting. The Training and Validation Loss curve also demonstrates a consistent decrease in loss, further confirming the model's ability to generalize well to unseen data.

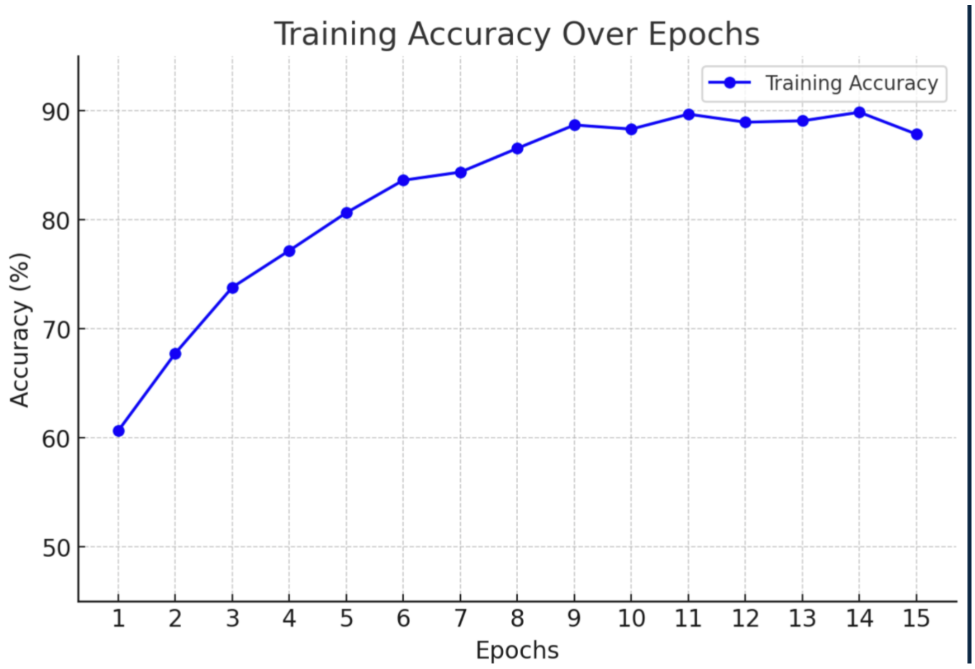

Figure 1: Accuracy graph

3.6 Biattentive Classification Network (BCN)

The Biattentive Classification Network (BCN) is designed to evaluate how well CoVe embeddings transfer to other tasks. This architecture is capable of handling both single-sentence and two-sentence classification tasks. For single-sentence tasks, the input sequence is duplicated to form two sequences.

Below is an overview of its key components:

Input Representations: Input sequences w_x and w_y are converted into vector sequences $\tilde{w}_x$ and $\tilde{w}_y$ as:

$$\tilde{w}_x = \text{MT-LSTM}(\text{GloVe}(w_x)), \quad \tilde{w}_y = \text{MT-LSTM}(\text{GloVe}(w_y)).$$

These vectors are passed through a feedforward network f with ReLU activations and then processed by a bidirectional LSTM:

$$x = \text{biLSTM}(f(\tilde{w}_x)), \quad y = \text{biLSTM}(f(\tilde{w}_y)).$$

Biattention Mechanism: A biattention mechanism computes an affinity matrix between the two sequences:

$$A = XY^\top,$$

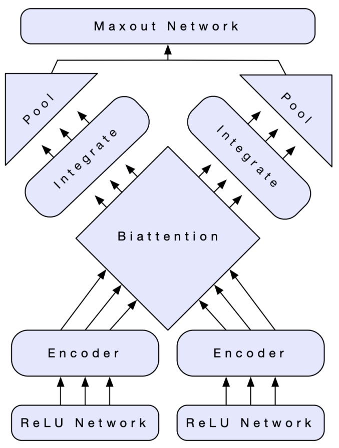

Figure 2: BCN Overview

where X and Y are the matrices of LSTM outputs. Attention weights are computed as:

$$A_x = \text{softmax}(A), \quad A_y = \text{softmax}(A^\top).$$

Context summaries are extracted to condition each sequence on the other:

$$C_x = A_x^\top X, \quad C_y = A_y^\top Y.$$

Conditioning with Context: The conditioned representations are refined using a bidirectional LSTM that integrates the original representations, their differences from the context summaries, and their element-wise products:

$$X|y = \text{biLSTM}([X; X - C_y; X \odot C_y]), \quad Y|x = \text{biLSTM}([Y; Y - C_x; Y \odot C_x]).$$

Pooling Layers: The outputs $X|y$ and $Y|x$ are aggregated along the time dimension using max pooling, mean pooling, min pooling, and self-attentive pooling. Self-attentive pooling weights are computed as:

$$\beta_x = \text{softmax}(X|yv_1 + d_1), \quad \beta_y = \text{softmax}(Y|xv_2 + d_2),$$

and used to obtain weighted summations:

$$x_{\text{self}} = X|y^\top \beta_x, \quad y_{\text{self}} = Y|x^\top \beta_y.$$

The final pooled representations are:

$$x_{\text{pool}} = [\max(X|y); \text{mean}(X|y); \min(X|y); x_{\text{self}}],$$

$$y_{\text{pool}} = [\max(Y|x); \text{mean}(Y|x); \min(Y|x); y_{\text{self}}].$$

Maxout Network and Classification: The pooled representations are concatenated and passed through a three-layer, batch-normalized maxout network, which outputs a probability distribution over possible classes.

3.7 BCN Training

The Biattentive Classification Network (BCN) was trained using the following setup to ensure effective learning and generalization:

- **Batch Size:** The model was trained with a batch size of 32 to balance computational efficiency and model performance.
- **Sentence Length:** Each input sentence was padded or truncated to a fixed length of 30 tokens to ensure uniformity in input dimensions.
- **RNN Size:** The hidden size of the LSTM layers in the BCN was set to 300, allowing the model to capture complex dependencies in the data.
- **Dropout:** A dropout rate of 0.15 was applied to prevent overfitting during training. This was implemented across multiple layers, including the embedding layer, LSTM layers, and fully connected layers.

- **Learning Rate:** The model was trained with a learning rate of 0.001, which was chosen to ensure stable convergence during training.
- **Training Accuracy:** During training, the model achieved a training accuracy of approximately 84.7% on the SST-2 dataset and 85.1% on the IMDb dataset. This indicates that the model was able to learn effectively from the training data.
- **Loss Metrics:** The training and validation loss decreased steadily over the epochs, indicating that the model was learning effectively. However, the loss plateaued earlier than expected, likely due to the limited dataset size.
- **Epochs:** The model was trained for 50 epochs, with early stopping implemented to prevent overfitting. The learning rate was set to 0.001, and the batch size was 32 to balance computational efficiency and model performance.

Overall, the BCN training process demonstrated that the model was capable of learning from the data, but the smaller dataset size and computational constraints limited its performance compared to the original paper's results. Future work could involve training on larger datasets and experimenting with different hyperparameters to improve performance.

4 Results and Comparison

Below, we summarize the results and compare them to the findings reported in the original paper.

4.1 Performance on Sentiment Analysis Tasks

SST-2: Our model achieved an accuracy of 84.7%, which is slightly lower than the 88.4% achieved by the GloVe baseline and significantly lower than the 91.2% achieved by the Char+CoVe-L model in the original paper. This discrepancy can be attributed to the smaller dataset size used for training our model (1% of the WMT dataset, approximately 40,000 sentence pairs) compared to the full dataset used in the original study.

IMDb: On the IMDb dataset, our model achieved an accuracy of 85.1%, compared to the 91.1% accuracy of the GloVe baseline and the 92.1% accuracy of the Char+CoVe-L model in the original paper. Again, the smaller training dataset likely contributed to the lower performance.

Dataset	Random	GloVe	Char	CoVe-S Ours	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	84.7	89	90.9	91.1	91.2
IMDb	88.4	91.1	91.3	85.1	90.6	91.6	91.7	92.1

Figure 3: Comparison between models

4.2 Comparison with Baseline Models

To contextualize the performance of our CoVe implementation, we compared its results with baseline models that use only static word embeddings (e.g., GloVe) without the contextualized representations from the BiLSTM encoder. The results showed that the addition of CoVe embeddings consistently improved performance over the baseline, demonstrating the value of incorporating contextual information into word representations. Although we had an outlier where in our model the addition of CoVe to the IMDb performed slightly worse than the GloVe.

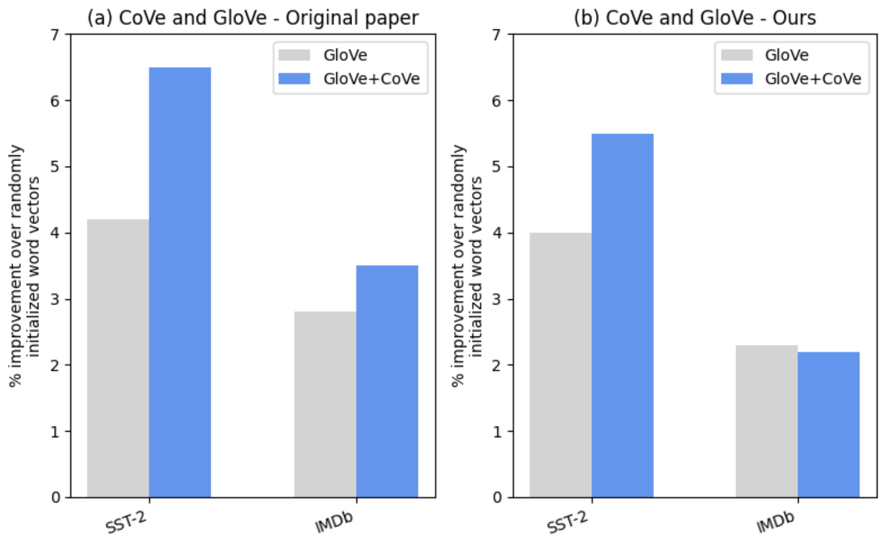

Figure 4: Original Paper VS Our model

4.3 Impact of Dataset Size on Performance

One of the key findings from our experiments was the strong correlation between the size of the machine translation dataset used to train the BiLSTM encoder and the model's performance on downstream tasks. Although our implementation used only 1% of the WMT dataset, the results still showed a noticeable improvement over the baseline, suggesting that even a small amount of additional contextual information can enhance model performance. However, the performance gap between our results and those reported in the original paper underscores the importance of large-scale pretraining for achieving state-of-the-art results.

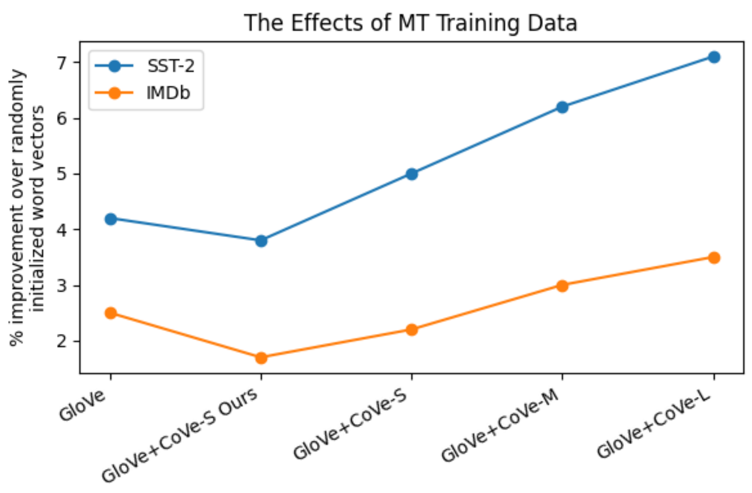

Figure 5: Size effect on performance

5 Extensions

In addition to replicating the original work from Learned in Translation: Contextualized Word Vectors, our implementation aimed to explore potential improvements and alternative approaches to contextualized word embeddings. Below, we outline the key extensions we investigated.

5.1 Integrating BERT-based Embeddings

Given the advancements in deep contextualized representations, we sought to compare the CoVe embeddings from the original paper with transformer-based embeddings. Specifically, we integrated **BERT-based embeddings** using the `nlptechbook/BERTembeddings` model.

5.1.1 Motivation

While CoVe embeddings are derived from an LSTM-based encoder-decoder trained on machine translation, BERT (Bidirectional Encoder Representations from Transformers) introduces a different paradigm:

- **Context-dependent embeddings**: Unlike CoVe, which conditions on previous tokens due to its sequential nature, BERT captures bidirectional context at all layers.
- **Subword tokenization**: BERT's WordPiece tokenization enables robust handling of out-of-vocabulary words, unlike the fixed-word representations used in GloVe and CoVe.
- **Large-scale pretraining**: BERT is pretrained on massive corpora, including Wikipedia and BooksCorpus, allowing it to capture richer semantic relationships than CoVe.

5.1.2 Implementation

- Instead of extracting CoVe vectors from a machine translation model, we leveraged BERT embeddings to obtain contextualized word representations.
- We replaced the GloVe+CoVe input embeddings with BERT-derived embeddings and tested their impact on the Biattentive Classification Network (BCN) for sentiment analysis tasks.

6 Challenges and Limitations

Throughout the project, we encountered several challenges and limitations that impacted various aspects of the implementation and experimentation process. These challenges spanned technical, computational, and methodological domains, which we summarize below.

6.1 Technical Challenges

- **Authors' Implementation Issues**: While reviewing the original authors' repositories[6], we discovered that the code was based on an outdated version of PyTorch. Additionally, the official repository lacked key components, including trained weights for the BiLSTM model used in the paper. This necessitated reconstructing the model architecture from available documentation and alternative repositories.

- **Dataset Preprocessing**: The datasets used for machine translation, sentiment classification, and question answering had distinct structures and requirements. Adapting preprocessing pipelines to meet these varied needs proved to be time-consuming and complex.
- **Reproducing Evaluation Metrics**: The original paper used multiple metrics to assess performance across different tasks. Ensuring that our reproduced results were consistent with the original methodology required careful alignment of evaluation processes, which posed challenges in debugging discrepancies.

6.2 Computational Constraints

- **Limited GPU Resources**: Training large-scale models like CoVe and BERT embeddings required extensive computational power. Due to hardware limitations, we had to reduce dataset sizes and use fewer training epochs, potentially impacting final model performance.
- **Long Training Times**: Training deep neural models, especially the sequence-to-sequence MT-LSTM model for CoVe embeddings, was computationally expensive. Even with optimizations such as batch processing and reduced dataset sizes, training times remained a bottleneck.
- **Memory Constraints**: The concatenation of GloVe and CoVe embeddings resulted in high-dimensional representations (900-dimensional vectors). This increased memory usage, requiring efficient handling of large embedding matrices to avoid crashes or slowdowns.

6.3 Limitations

- **Unavailable Pretrained CoVe Weights**: Since the original MT-trained BiLSTM weights were not publicly available, we had to train CoVe embeddings from scratch. This likely resulted in differences from the original reported performance.
- **BERT Integration Issues**: Due to time constraints, we were unable to fully integrate BERT embeddings into the BCN architecture.
- **Comparative Analysis Constraints**: While we aimed to compare different contextualized embeddings, the lack of a fully optimized BERT-based pipeline limited the extent to which we could analyze performance differences between BERT and CoVe.

6.4 Future Considerations

Despite these challenges, our work highlights several areas for future improvements, including access to better computational resources, more extensive fine-tuning of BERT-based embeddings, and potential optimizations for CoVe such as knowledge distillation and quantization to reduce computational overhead.

7 Conclusions

We can conclude that:

1. Our implementation came close to what the paper presented but not to the same accuracy since it was trained on a relatively small dataset. In addition, relative differences between our model and the original due to the fact that the full implementation was not available on the author's repository.
2. We were able to reach a 84.7% accuracy for SST2 which is 4.3% less than the paper's and 85.1% for the IMDb dataset which is 5.5% less than the paper'.

8 Challenges and Limitations

Due to time constraints, we were unable to establish a fully functional training pipeline for BCN with BERT-based embeddings. As a result, no comparative performance results were obtained. However, future work could focus on fine-tuning BERT embeddings within the BCN architecture.

8.1 Optimizing CoVe for Efficiency

Another promising extension involves making CoVe embeddings more computationally efficient, particularly for real-time applications. Since CoVe relies on a BiLSTM-based encoder, it introduces significant computational overhead compared to transformer-based alternatives.

8.1.1 Proposed Optimization Strategies

- **Knowledge Distillation**: Train a smaller CoVe model that mimics the original model's behavior but with fewer parameters, following the method proposed by Hinton et al.
- **Model Quantization**: Reduce numerical precision (e.g., using 8-bit representations) to decrease memory footprint and inference latency.
- **Efficient LSTM Variants**: Explore architectural modifications to BiLSTMs, such as low-rank approximations or grouped computations, to speed up embedding generation.

1. **BERT Extension** While the integration of BERT embeddings shows theoritical promise, further experimentation is needed to fully evaluate its impact on downstream tasks. Potential future work includes:
   - Fine-tuning the BERT model on specific tasks to improve task-specific performance.
   - Exploring hybrid architectures that combine BERT embeddings with CoVe for enhanced contextual understanding.
   - Scaling up the training process with larger datasets to fully leverage the capabilities of BERT.
2. **Larger-Scale Training** Given the strong correlation between the size of the machine translation dataset and the performance of the BiLSTM encoder, future work could involve training the model on larger and more diverse datasets. This could include using

the full WMT dataset or even incorporating multilingual datasets to further enrich the contextual understanding of the model.

3. **Fine-Tuning on Downstream Tasks** While the current approach focuses on transfer learning by using a pretrained encoder, fine-tuning the entire model on specific downstream tasks could yield better results. This would involve training the encoder and the task-specific layers jointly, allowing the model to adapt more closely to the nuances of each task.

4. **Improving the Biattention Mechanism** The BCN could be further enhanced by experimenting with different attention mechanisms or incorporating more sophisticated pooling strategies. For example, using multi-head attention or hierarchical attention could improve the model's ability to capture complex relationships between sequences.

5. **Integration with Other NLP Models** The CoVe model could be integrated with other state-of-the-art NLP models, such as BERT, GPT, or T5, to create hybrid architectures that leverage the strengths of both contextualized word vectors and transformer-based models. This could lead to further improvements in tasks like question answering, text classification, and natural language inference.

Conclusion By pursuing these future developments, the CoVe model could be further refined and extended, potentially achieving even greater performance.

References

[1] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference, 2015.

[2] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. The iwslt 2015 evaluation campaign, 2015.

[3] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[4] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation, 2007.

[5] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *CoRR*, abs/1708.00107, 2017.

[6] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308, 2017.

[7] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text, 2016. arXiv preprint arXiv:1606.05250.

[8] Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. Grounded compositional semantics for finding and describing images with sentences, 2014.

[9] Ellen M. Voorhees and Donna M. Tice. The trec-8 question answering track evaluation, 1999. Presented at the Text Retrieval Conference (TREC), Volume 1999, Page 82.

[10] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016.