Improving minBERT with Conditional Layer Normalization

Stanford CS224N Default Project

Matan Abrams
Department of Computer Science
Stanford University
mabrams4@stanford.edu

Abstract

This project aims to enhance the performance of a baseline BERT model [1] through the integration of Conditional Layer Normalization (CLN), adopted from Conditional Batch Normalization [2], which adapts the normalization process based on the task at hand. This approach is hypothesized to improve the model's ability to handle various natural language processing tasks by dynamically adjusting its internal representations to better suit the requirements of each task.

1 Key Information to include

- TA mentor: Yuan Gao
- External collaborators: NO
- External mentor: NO
- Sharing project: NO

2 Introduction

The BERT model, introduced by Devlin et al. [3], has significantly advanced the state of the art in various natural language processing (NLP) tasks. Despite its successes, there remains room for improvement, especially when applying BERT to a variety of tasks simultaneously as done in this project. Standard Layer Normalization (LN) [4] uses fixed normalization parameters for all tasks, limiting the model's flexibility and performance on task-specific nuances.

To address this limitation, I propose Conditional Layer Normalization (CLN) as an extension of the baseline BERT model [1]. CLN uses task-specific normalization parameters. This change allows the model to tailor its internal representations according to the specific task at hand, improving overall performance across tasks.

In this project, I implement CLN into the baseline BERT model and evaluate its effectiveness on the three tasks of sentiment classification, paraphrase detection, and semantic textual similarity. I hypothesize that the CLN-enhanced model will outperform the baseline BERT model due to its ability to use specific normalization parameters for each task separately.

3 Related Work

Since its conception, the original BERT model [3] has been a cornerstone in NLP, achieving state-of-the-art results on numerous benchmarks. BERT's bidirectional architecture allows it to learn better contextual representations of the input by considering both the left and right context of it. Following this original model, researchers have implemented many modifications to increase BERT's capabilities or reduce its computational requirements. Some of these early modifications include

RoBERTa [5] and DistilBERT [6]. RoBERTa optimized BERT by training with more data and longer sequences, while DistilBERT reduced the model size while maintaining most of BERT's performance.

Another improvement to the baseline BERT model was Layer Normalization [4]. LN has become an integral component in stabilizing training and improving the performance of deep neural networks. By normalizing the inputs across the features, LN also leads to more effective and faster learning. However, the use of fixed parameters for all tasks limits adaptability to the multitask settings.

Conditional Batch Normalization (CBN) [2] was proposed to adjust normalization parameters based on additional information, such as task-specific embeddings, and has shown success in various applications.

Inspired by CBN and LN, I adapt the concepts to Conditional Layer Normalization (CLN). CLN aims to improve the overall performance of the baseline model by learning task-specific normalization parameters, enhancing the model's ability to adapt to different tasks.

4 Approach

In this section I describe my implementation of CLN, provide a mathematical overview of both standard and conditional layer normalization, and describe the baseline model used in this project. The implementation involved two main steps: additional pre-training on datasets specific to all 3 downstream tasks, and the integration of CLN.

4.1 Additional Training

Additional training on the Quora and STS datasets was conducted to ensure task-specific weights could be learned. This was implemented with the following steps:

- Load and preprocess the SST, Quora, and STS datasets.
- Create `DataLoader` objects for each dataset to facilitate batch processing.
- Initialize the `MultitaskBERT` model with the specified configuration.
- Set the learning rate and initialize the optimizer.
- Iterate over the specified number of epochs:
  - For each epoch, iterate over each dataset (SST, Quora, and STS).
  - For each batch in a dataset:
    - Compute the logits using the model's prediction method for the task.
    - Compute the loss (cross-entropy for sentiment classification, binary cross-entropy for paraphrase detection, and MSE for semantic similarity).
    - Perform backpropagation and update the model parameters.
  - Compute the average training loss over all batches.
  - Evaluate the model's performance on the training and development datasets using `model_eval_multitask`.
  - Save the model if the performance on the development set improves.
- Finally, use the saved model for test set evaluation.

4.2 CLN Implementation

Conditional Layer Normalization (CLN) was implemented as follows:

- Define the `ConditionalLayerNorm` class inside `bert.py`, inheriting from `nn.Module`.
- Initialize parameters for default layer normalization: `self.weight` and `self.bias`.
- Introduce task-specific parameters as `nn.Parameter` with size (num_tasks, hidden_size):
  - `self.task_weight`
  - `self.task_bias`
- Initialize `self.task_weight` and `self.task_bias` as 0 tensors.

- In the `forward` method:
  - Compute the mean and standard deviation of the input tensor.
  - Adjust the normalization parameters based on the task-specific weights and biases:
  - Add the task-specific weights and biases to the default weights and biases.
- Integrate `ConditionalLayerNorm` into the `BertLayer` class:
  - Replace the default `nn.LayerNorm` layers with `ConditionalLayerNorm`.
  - Modify the `add_norm` method to pass the `task_id` to the `ConditionalLayerNorm` layers.
  - Modify the `forward` method of `BertLayer` to correctly handle the `task_id`.
- Modify the `BertModel` class to support CLN:
  - Replace the default `nn.LayerNorm` for the embedding layer with `ConditionalLayerNorm`.
  - Update the `embed` method to include the task_id in the inital embedding layer normalization.
  - Update the `encode` method to pass the `task_id` to each `BertLayer`.
  - Update the `forward` method to accept and handle the `task_id`, passing it through to the `embed` and `encode` methods.
- Update the `MultitaskBERT` class to use the new `BertModel` with CLN integrated:
  - Modify the `forward` method of `MultitaskBERT` to pass the `task_id` to the BERT model.

Figure 1 displays a high level overview of where CLN replaces standard LN in the BERT architecture.

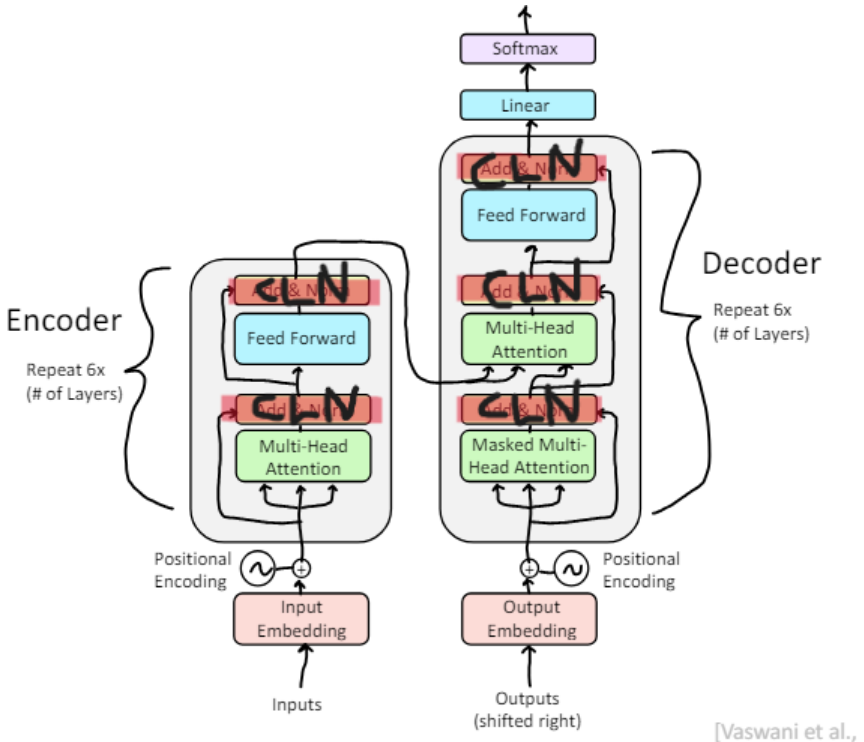

Figure 1: High-level BERT Diagram with CLN

4.3 Normalization Techniques

This sub-section describes the normalization techniques used/referenced in this paper.

4.3.1 Standard Layer Normalization

The standard Layer Normalization formula is defined as:

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta \tag{1}$$

[4]

Where:

- $x \in \mathbb{R}^d$ is the input tensor to the layer, and d is the dimensionality of the input.
- μ is the mean of x, computed as:

$$\mu = \frac{1}{d} \sum_{i=1}^{d} x_i \tag{2}$$

- σ^2 is the variance of x, computed as:

$$\sigma^2 = \frac{1}{d} \sum_{i=1}^{d} (x_i - \mu)^2 \tag{3}$$

- ϵ is a small constant added to the variance to avoid division by zero.
- $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ are the learnable weight and bias parameters, respectively.

4.3.2 Conditional Layer Normalization

CLN adapts the normalization parameters by introducing additional task-specific weight and bias parameters. This allows the model to adjust its normalization process based on the task at hand. The CLN formula is defined as:

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot (\gamma + \gamma_t(t)) + (\beta + \beta_t(t)) \tag{4}$$

Where:

- x, μ, σ^2, and ϵ are defined as in standard Layer Normalization.
- $\gamma(t) \in \mathbb{R}^d$ and $\beta(t) \in \mathbb{R}^d$ are the task-specific weight and bias parameters.
- t represents the task identifier, indicating which task the current data belongs to.
- $\gamma + \gamma_t(t)$ and $\beta + \beta_t(t)$ are the combined weight and bias parameters that incorporate both the standard and task-specific parameters.

4.4 Baseline Model

The baseline model is the standard minBERT architecture as described in the original minBERT paper, [3], which utilizes standard layer normalization and does not include additonal training on the Quora and STS datasets.

5 Experiments

This section describes the data used, experiments performed, and results for this project.

5.1 Data

I utilize the datasets provided in the default final project i.e. Stanford sentiment treebank (SST) [7] for sentiment classification, Quora [8] for paraphrase detection, and SemEval STS dataset [8] for similarity detection.

5.2 Evaluation method

The evaluation metrics are accuracy for sentiment classification and paraphrase detection, and Pearson correlation for the semantic textual similarity task.

5.3 Experimental details

For the baseline model I used:

- Fine-tune-mode: last-linear-layer
- Learning rate: 1e-3
- Batch size: 8
- Training epochs: 10
- Dropout prob: 0.3

Experiments were conducted with a variety of different hyper-parameter (dropout probability, number of epochs, learning rate, using both full-model and last-linear-layer for fine-tune mode) as well as CLN mean and std deviation parameter initializations. The results below only showcase the best performance which was done with the following parameters:

- Fine-tune-mode: last-linear-layer
- Learning rate: 1e-3
- Batch size: 8
- Training epochs: 10
- Dropout prob: 0.5
- CLN task-specific weights and biases: initialized to all 0's

5.4 Results

Below are the results for the CLN-enhanced model on the test sets compared to the baseline.

Table 1: Performance Comparison of Baseline and CLN-Enhanced Models

Model	Sentiment Accuracy	Paraphrase Accuracy	SemEval Pearson Correlation
Baseline	0.402	0.630	-0.027
CLN-Enhanced	**0.525**	**0.756**	**0.345**

6 Analysis

As the results above indicate, the addition of task-specific parameters with CLN significantly enhances the performance of the model across all three downstream tasks: sentiment classification, paraphrase detection, and semantic textual similarity.

By allowing the model to adjust its normalization parameters based on the task at hand, CLN provides the flexibility to tailor the internal representations for each specific task. This task-specific adaptation improves the model's ability to capture nuanced differences in sentiment, align sentence pairs more effectively for paraphrase detection, and more accurately assess semantic similarity between sentences.

Overall, the task-specific parameters introduced by CLN help the model reach more optimal solutions by providing tailored normalization for each task, leading to significant improvements in performance and adaptability.

Additionally, increasing the dropout probability from 0.3 to 0.5 improved the model's performance with CLN. By increasing the dropout probability, the model becomes more robust to variations and and encourages it to learn more generalized features. This increased regularization could be

particularly beneficial when using CLN, as it ensures that the task-specific parameters do not lead to over-fitting on any particular task, improving overall generalization and performance across multiple tasks.

7 Conclusion

In this project, I proposed and implemented Conditional Layer Normalization (CLN) to enhance the performance of a baseline BERT model on three different NLP tasks, sentiment classification, paraphrase detection, and semantic text similarity. The experimental results prove the initial hypothesis that this enhancement would lead to improved results over the baseline model. Specifically I saw significant improvements in all three tasks over the baseline.

Through this project I learned more deeply about how standard layer normalization works, and how conditional layer normalization can extend standard layer normalization to better enhance a single model performing multiple different downstream tasks.

One limitation of this work is that it requires more computational resources than standard layer normalization. This is due to the addition of many learnable parameters into the model that require gradient calculations in the optimizer step. Furthermore, without deeper exploration, it is very difficult to understand how the model is learning the newly introduced task specific weights and biases which could be a cause for concern.

Potential future work could involve further hyper-parameter tuning, exploring different initialization strategies, and potentially creating a more complex CLN class in the code that follows a different equation. In this implementation I simply add the task-specific parameters to the standard ones but other forumlas could lead to interesting results. Additionally, combining CLN with other improvement techniques could lead to further performance gains.

8 Ethical Considerations

In this section I discuss 2 potential ethical considerations of the integration of CLN into the base BERT model as well as potential mitigation strategies for each.

8.1 Amplifying Bias

The first consideration is the potential to amplify existing biases in the model. The base model uses standard layer normalization which involves fixed parameters to normalize the input data at each layer. With CLN and the introduction of task-specific normalization parameters, there is more room for the model to pick up on potential inherent biases already in the model and tailor its parameters towards those biases in an attempt to mitigate loss based on the task at hand. For example if there was some high correlation of a certain result being associated with some kind of sensitive information such as demographic info in the sentiment classification task, a model with CLN could be more likely to pick up on this correlation and fit itself around this relationship resulting in amplifying the already existing bias.

Mitigation Strategy:
In an attempt to mitigate this amplification of bias, one could employ explainability techniques such as Local Interpretable Model-Agnostic Explanations (LIME). LIME functions by perturbing inputs to the model to see what parts of the input contribute most to the models final outputs. In the context of sentiment classification, this could help monitor any bias amplifications by seeing if the model is overly relying on any sensitive attributes for its final predictions.

8.2 Bad Predictions in Low Resource Environments

A second consideration is that in low resource environments, i.e. small amounts of training data, the addition of CLN could drastically affect the outputted predictions due to over-fitting parameters to one or more strong signals in the training data. This could potentially lead to a single sensitive attribute in the data having an unwanted large effect on the predictions of the model on a given task. These predictions would not be very trustworthy and could lead to people making bad decisions downstream based on the models bad predictions.

Mitigation Strategy: One potential mitigation strategy for this would be to simply be extremely transparent about this potential flaw in the model to any user who might want to use it. Something like "This model may be hyper-sensitive to smaller training datasets. Therefore, please be critical of predictions made by this model in this kind of training environment as they may not extremely accurate." Including transparency in this way would help mitigate people using the models poor decisions in their downstream tasks.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[2] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. *arXiv preprint arXiv:1707.00683*, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[7] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Stanford sentiment treebank. `https://nlp.stanford.edu/sentiment/treebank.html`, 2013. Accessed: 2024-05-25.

[8] Quora. First quora dataset release: Question pairs. `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`, 2017. Accessed: 2024-05-25.