Applying Domain Layer Normalization on Long-Short Term Memory network for adaptive Classification

Mohammad Amin Ghasemi
Department of Mathematics and Computer Science
Amirkabir University of Technology
Tehran, Iran
aminghassemi7@gmail.com

Abstract**—** Recurrent based classification models often face challenges in achieving high performance when provided with a limited amount of data. In such scenarios, if we have data for different related tasks, multi-task learning could be a useful approach for leveraging their commonalities and differences to enhance the overall performance. Many existing works in multi-task classification apply complex architecture for sharing related information and handling differences. In this study, we introduce an innovative approach to address this issue by incorporating layer normalization into the model's design, enabling effective control of information flow. We refer to our model as Domain Layer Norm Classifier (DLNC). Additionally, we define task-specific embeddings to further characterize each task and enhance performance. Despite its simplicity and lightweight design, our model demonstrates robust generalization capabilities, even when trained on limited data. To validate our approach, we conduct fine-tuning experiment on a new dataset, highlighting the adaptability and effectiveness of our model.

Keywords—Classification, layer normalization, multi-task learning, Recurrent neural networks, task embedding, fine-tuning

I. Introduction

In the field of natural language processing (NLP) and sequence modeling, recurrent neural networks (RNNs) have been instrumental in capturing intricate dependencies within sequence data. Long Short-Term Memory (LSTM) networks, a type of RNN, have gained considerable attention for their ability to mitigate the vanishing gradient problem and maintain long-term dependencies. Multi-task learning, on the other hand, has become increasingly important in various applications, where the goal is to jointly train models to perform multiple related tasks, often with shared underlying representations.

The key challenge in multi-task learning lies in optimizing shared representations effectively while allowing each task to adapt to its specifications [1]. This paper introduces an approach that leverages LSTM networks for task adaptive classification, with a particular focus on enhancing the adaptability of the encoder. We propose the use of Layer Normalization applied to LSTM gates, allowing us to introduce data adapted adjustments within a shared LSTM-based architecture.

I.I. LSTM-Based classifiers

Long Short-Term Memory (LSTM) networks have emerged as a powerful sequence modeling technique due to their ability to capture long-range dependencies and handle sequential data effectively. In context of text classification, LSTM-based encoders have demonstrated their prowess in capturing intricate patterns and semantic relationships within textual data [2]. It's been shown that as the training size grows, these types of classifiers would achieve better performance [3]. Our proposed model builds upon this foundation by applying adaptations within the LSTM architecture.

I.II. Layer Normalization

Layer normalization has been proven to be an effective technique for stabilizing the training of deep neural networks [4]. However, its application particularly with respect to the gating mechanisms that are pivotal for capturing sequential dependencies. In this paper, we explore extending the concept of layer normalization to the LSTM gates, allowing us to modulate the information flow in a task-specific manner. Previous works have used layer norm for domain adaptive semantic scene segmentation [5], and unsupervised stylistic image description [6]; this work was our main inspiration, as it showed the effectiveness of this method in text generation. By dynamically adjusting the gate activations during training, we aim to enhance the model's ability to adapt to the unique characteristics of each classification task.

I.III. Task-Specific Learnable Input Vectors

Task embedding can be learned to represent the relationship and signal the differences between tasks, [7] for example, introduces a method to learn task embedding which can be used for Meta-learning. To further empower our multi-task classification model, we use task-specific learnable input vectors. This approach enables the model to adapt its input representations to the specific linguistic nuances and contexts of each task, thereby improving its ability to capture task-specific features.

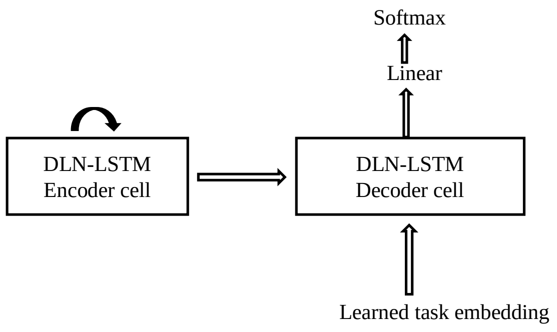
I.IV. Fine-Tuning

Fine-tuning is a common technique in machine learning and deep learning where a pre-trained model, is further trained on a specific task or dataset. Fine-tuning leverages the knowledge learned by the pre-trained model and adapt it to a more specific task. When having a limited amount of task specific data, complex models may result in overfitting, whereas simpler models are less likely to do so. Also, simpler models are faster and require fewer resources to do so [8].

II. Model Architecture

The encoder consists of LSTM cells with domain specific layer normalizations, which we will refer to as Domain Layer Norm LSTM cells (DLN-LSTM). All the other parameters of tasks are shared. Final hidden outputs of the encoder are forwarded to a DLN-LSTM cell with different parameters than encoder cells, which is used for classification. Hidden vector from this cell is fed into a linear layer followed by softmax for the final classification. Additionally, a learned embedding vector for each task is passed as input to the DLN-LSTM classifier cell. We call this model, Domain Layer Norm Classifier or DLNC.

Fig 1. Components of DLNC



Separate layer normalizations are used for dealing with the input and previous cell's hidden state in gate calculations. This is beneficial because these calculations have different responsibilities and handle distinct types of representations, which helps with controlling the information flow for each task. Additionally, we apply layer normalization to the cell state before using it in the calculation of the hidden state. It helps in conveying more useful information to the subsequent cell.

$$\begin{pmatrix} f_t \\ i_t \\ o_t \\ g_t \end{pmatrix} = LN_{T,x}\left(W_x X_t; g_{T,x}, a_{T,x}\right) + LN_{T,h}(W_h h_{t-1}; g_{T,h} a_{T,h}) \quad (1)$$

$$c_t = \sigma(f_t) * c_{t-1} + \sigma(i_t) * \tanh(g_t) \quad (2)$$

$$h_t = \sigma(o_t) * LN_{T,c}\left(c_{t\,;}\, g_{T,c}, a_{T,c}\right) \quad (3)$$

Here, $LN_{T,i}$ denotes a layer normalization for task T, $g_{T,i}$ represents its scaling, and $a_{T,i}$ represents its shifting parameters.
Moreover, dropout is applied to improve generalization.

III. Datasets and Experimental Setup

Datasets

- **EI-reg** emotion which consists of tweets corresponding to four basic emotions: anger, fear, joy, and sadness [10]. There is a number associated with each tweet, indicating its level of intensity for the emotion. We used sentences with their intensity greater than or equal to 0.5.
- **Yelp** user review dataset containing user-generated reviews by people who have visited different businesses, in three target classes: positive, neutral and negative [11].
- **IMDB** movie reviews with positive or negative sentiment.

EI-reg and Yelp are used for training the model, and effectiveness of it for fine tuning is examined with IMDB. Because we especially want to focus on the model's behavior on limited data, only 8000 reviews of Yelp is used and 2000 of IMDB for fine tuning.

Hyperparameters

Input size is 100 for encoder and 32 for classifier cell and hidden dimension is 256 for both. Dropout of rate 0.3 is applied to recurrent calculations, 0.2 to the input of linear and 0.1 to the input of softmax. Similar parameters are set for all the examined models.

Pre-trained Word Vectors

Initializing word vectors with those obtained from an unsupervised mode is a popular method to improve performance of the model, especially in the absence of large training set. We use GloVe embedding vectors with dimension 100 and consider these vectors as trainable parameters during training [9].

Model Variations

To examine the effectiveness of our method, we compare its performance with a model that has LSTM for encoder and passes its final hidden vector to a linear and then softmax for classification, which we will refer as "LSTM+linear". This model has, of course, been trained separately on each task to evaluate its performance on it.

Also, to see the effect of different model components, models are trained that only differ from the original proposed model in a specific component. We have named them 'No task embedding' and 'Separate classification cell'. The first one doesn't apply learned task embedding and the second one, has completely different parameters for its classification cells for each task, rather than only different layer normalizations.

Table 1: Train and test results for model variations described earlier

Model	Yelp		EI-reg	
	Train	Test	Train	Test
DLNC	98.3	**85.2**	91.7	**78.6**
LSTM+linear	77.9	63.6	60.2	51.4
No task embedding	92.5	83.3	85.1	72.4
Separate classification cell	99.1	83.9	93.7	76.8

IV. Results and Discussion

The large performance gap between DLNC and LSTM+linear shows that our model has been successful in sharing the similarities in the characteristics of different tasks to improve the performance and overcome the problem of having a limited data, especially since LSTM+linear doesn't have a good performance even in training.

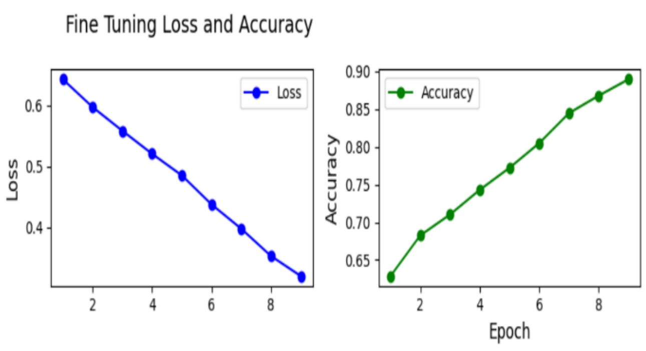
Results also showcase the positive effect of defining task specific input vector for classification, and it has helped in in our model being adaptable to different tasks without overfitting the model.

Moreover, although a separate classifier cell for each task has resulted in better accuracies for training, it seems that has caused overfitting, resulting in less generalization power.

Fine-tuning

We also examine our model's fine-tuning performance on a new dataset (with a similar task). For this, only the classification part and layer normalizations of encoder are learned. Model reaches accuracy of 88.9 on training data and 72.5 on test after about a minute of training on CPU. This result seems especially good since after training LSTM+linear on this limited data, it doesn't achieve an accuracy better than 30.9.

Fig 2. Loss and Accuracy of DLNC during fine-tuning on IMDB reviews dataset



V. Conclusion and Future Work

We propose a novel multi-task classification method using domain layer normalization, which enables the sharing of information between tasks and can effectively address the low-data problem. The model architecture is very lightweight, making it a suitable choice for multi-task classification scenarios where fast computations or limited memory usage are crucial. Furthermore, we believe that this method can find applications beyond its current use case. It has the potential to be valuable when employing LSTMs for a various types of data, such as images and audio, and for any type of tasks with similar characteristics. Finally, our experiment demonstrated the model's effectiveness in fine-tuning with a small amount of new data. This suggests its potential utility in practical applications.

References

[1] Michael Crawshaw, "MULTI-TASK LEARNING WITH DEEP NEURAL NETWORKS: A SURVEY" arXiv:2009.09796v1 [cs.LG] 10 Sep 2020

[2] Alex Graves "Generating Sequences With Recurrent Neural Networks" arXiv:1308.0850v5 [cs.NE] 5 Jun 2014

[3] Jenq-Haur Wang, Ting-Wei Liu, "An LSTM Approach to Short Text Sentiment Classification with Word Embeddings " The 2018 Conference on Computational Linguistics and Speech Processing ROCLING 2018, pp. 214-223

[4] Jimmy Lei Ba, et. "Layer Normalization" arXiv:1607.06450v1 [stat.ML] 21 Jul 2016

[5] R.Romijnders, et. "A Domain Agnostic Normalization Layer for Unsupervised Adversarial Domain Adaptation" arXiv:1809.05298v1 [cs.CV] 14 Sep 2018

[6] Cheng-Kuan Chen , Zhu Feng Pan, Min Sun, Ming-Yu Liu, "Unsupervised Stylish Image Description Generation via Domain Layer Norm," arXiv:1809.06214v1 [cs.CV] 11 Sep 2018

[7] Alessandro Achille, et. "TASK2VEC: Task Embedding for Meta-Learning" arXiv:1902.03545v1 [cs.LG] 10 Feb 2019

[8] Shuai Zhang, Xin Pan, et. "Bag of Tricks for Efficient Text Classification" Processing of the 15th Conference of the European Chapter Association for Computational Linguistics. January 2017

[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation" In EMNLP, 2014

[10] Saif M. Mohammad and Svetlana Kiritchenko, "Understanding Emotions: A Dataset of Tweets to Study Interactions between Affect Categories". 2018

[11] Nabiha Asghar, " Yelp Dataset Challenge: Review Rating Prediction" arXiv:1605.05362v1 [cs.CL] 17 May 2016

All code for the proposed model and details mentioned in the paper, could be found here
Readers are welcome to use this code in their future works