Understanding Learning Dynamics for Neural Machine Translation

Anonymous EMNLP submission

Abstract

Despite the great success of NMT, there still remains a severe challenge: it is hard to interpret the internal dynamics during its training process. In this paper we set up to understand learning dynamics of NMT by using a recently proposed technique named Loss Change Allocation (LCA). As a NMT model typically employs a large-scale neural architecture, the brute-force implementation of LCA suffers two challenges in both computation and storage, and thus we present an improved approach to put it into practice. On two standard translation benchmarks, our approach is proved to be efficient and intensive experiments reveal some valuable findings.

1 Introduction

Neural Machine Translation (NMT) has witnessed a great success in recent years (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017). The main reason is that it employs a mass of parameters to model sufficient context for translation decision, and in particular enjoys an end-to-end flavor for training all these parameters. Despite its success, there still remains a severe interpretation challenge for NMT: it is hard to understand its learning dynamics, i.e., *how do the trainable parameters affect a NMT model during its learning process*?

Understanding learning dynamics of neural networks is beneficial to identify the potential training issues and further improve training protocols for neural networks (Smith et al., 2017; McCandlish et al., 2018). Existing works on understanding learning dynamics have been extensively investigated in classification tasks (Shwartz-Ziv and Tishby, 2017; Raghu et al., 2017; Li et al., 2018; Bottou et al., 2018; des Combes et al., 2019) and language modeling (Saphra and Lopez, 2018, 2019). To the best of our knowledge, there is currently no attempt at understanding the learning dynamics for NMT, although it is acknowledged that the training process is critical to make advanced NMT architectures (like *transformer*) successful.

In this paper, we thereby propose to understand learning dynamics of NMT. Specifically, we extend a technique named Loss Change Allocation (LCA) from computer vision (Lan et al., 2019) to NMT scenario. The key behind LCA is to decompose the overall loss according to individual parameter for each update during the training process. By summing up the LCA values of a parameter between two update time steps, we are able to quantify how effective parameters are to the loss decrease in this learning phase (§2.1). Unfortunately, unlike the network in comptuer vision, NMT employs a sequence-to-sequence architecture with massive parameters and requires large-scale data for training, and thus the brute-force implementation of LCA suffers from challenges in both computation and storage on standard translation tasks. To this end, we instead propose an improved implementation approach to put it into practice: it approximately calculates gradient for speedup in computation and designs two additional tricks to address the storage bottleneck (§2.2). On two standard translation benchmarks, our approach is efficient and particularly intensive experiments reveal the following findings (§3):

- Parameters of the encoder (decoder) word embeddings and the softmax matrix contribute very little to loss decrease;
- Parameters of both the last layer in encoder and decoder contribute most to loss decrease;
- Word embeddings for frequent words contribute far more to the loss decrease than those for infrequent words.

2 Methodology

2.1 Loss Change Allocation

Loss Change Allocation (LCA) functions as a microscope for investigating deeply into the training process of any models trained with stochastic gradient methods (Lan et al., 2019). It is an optimizer-agnostic methods for probing into fine-grained learning dynamics. In raw wordings, LCA tracks the contribution of each model parameter $\theta^i \in \theta$ to the loss change at every gradient update during the whole training process, where $i \in [K]$ and K is the number of model parameters. The basic idea of LCA is to take advantage of the first order Taylor expansion of the loss to approximate the loss change at each mini-batch update.

Recall that at each update t, the optimizer (e.g. SGD) samples a mini-batch B_t from the training data for forward computation and then back-propogates to update the parameters from θ_t to θ_{t+1}. Given a dataset $\mathcal{D}$, formally, the *moment* loss change attributed to all the model parameters can be first approximated and then decomposed on the LCA value of each parameter θ^i as follows:

$$\begin{aligned} &\mathcal{L}(\theta_{t+1};\mathcal{D}) - \mathcal{L}(\theta_t;\mathcal{D}) \\ \approx &\nabla_\theta^\top \mathcal{L}(\theta_t;\mathcal{D}) \cdot (\theta_{t+1} - \theta_t) \\ := &\sum_{i=1}^{K} A_{lca}[t][i], \end{aligned} \tag{1}$$

where each $A_{lca}[t][i] = \nabla_{\theta^i}\mathcal{L}(\theta_t;\mathcal{D}) \cdot (\theta^i_{t+1} - \theta^i_t)$ and $A_{lca}[t][i]$ denotes the LCA value bound with parameter θ^i at the update t.

Based on the above defined LCA value at each update, one can readily attribute the loss decrease $\mathcal{L}(\theta_{t_2};\mathcal{D}) - \mathcal{L}(\theta_{t_1};\mathcal{D})$ within an *interval* of multiple updates $[t_1, t_2]$ to each parameter as follows:

$$\begin{aligned} &\mathcal{L}(\theta_{t_2};\mathcal{D}) - \mathcal{L}(\theta_{t_1};\mathcal{D}) \\ \approx &\sum_{t=t_1}^{t_2-1} \mathcal{L}(\theta_{t+1};\mathcal{D}) - \mathcal{L}(\theta_t;\mathcal{D}) \\ = &\sum_{t=t_1}^{t_2-1}\sum_{i=1}^{K} A_{lca}[t][i] = \sum_{i=1}^{K}\sum_{t=t_1}^{t_2-1} A_{lca}[t][i], \end{aligned} \tag{2}$$

where $\sum_{t=t_1}^{t_2-1} A_{lca}[t][i]$ is the attribution of loss change bound with parameter θ^i and its value reflects the *effectiveness* of θ^i with respect to the loss degradation on a certain dataset $\mathcal{D}$ between the update interval. By inspecting the different choices of update intervals, one is able to understand learning dynamics in terms of the attribution of loss change to each parameter.

2.2 Challenges and Solutions

Challenges However, unlike the scenario of computer vision (Lan et al., 2019), in our scenario a NMT model employs a large network and thus it is challenging to obtain the matrix A_{lca} in a brute-force implementation. On the one hand, Eq.(1) requires to calculate the gradient on the entire dataset $\mathcal{D}$ at each update t and thus a relatively large dataset leads to inefficiency in computation. In our experiments, we are mainly interested in two settings for $\mathcal{D}$, i.e. the training dataset and a held-out test dataset, which respectively measure the training loss (model's fitting ability) and testing loss (model's generalization ability). On the other hand, since a NMT model typically contains hundreds of millions parameters and parameter updates are up to hundreds of thousands times during training, it consumes super large storage to maintain A_{lca} on the hard disk, which is impractical in implementation.

Solutions To address both challenges, we propose some tricks to improve the implementation of LCA. Firstly, inspired by SGD (Bottou, 2010), at each update, we instead re-sample a new mini-batch to be a representative of the entire dataset $\mathcal{D}$ and approximately calculate the gradient on this mini-batch rather than the gradient on $\mathcal{D}$. With this trick, training with LCA only doubles the speed compared to the standard training for NMT. In Section 3.2, we will empirically validate the rationality of this sampling approach with simulated experiments.

In addition, we store the LCA value once for every 15 updates via averaging the LCA values for those steps:

$$\bar{A}_{lca}[t][i] = \frac{1}{15} \cdot \sum_{t'=15t+1}^{15(t+1)} A_{lcs}[t'][i], \tag{3}$$

for t beginning with 0. Furthermore, we divide the model parameters into several groups and calculate LCA value for each group g rather than each parameter i as follows:

$$\bar{A}_{lca}[t][g] = \frac{\sum_{i \in g} \bar{A}_{lca}[t][i]}{|g|}, \tag{4}$$

where $|g|$ denotes the number of parameters within that group. More precisely, we mainly study LCA

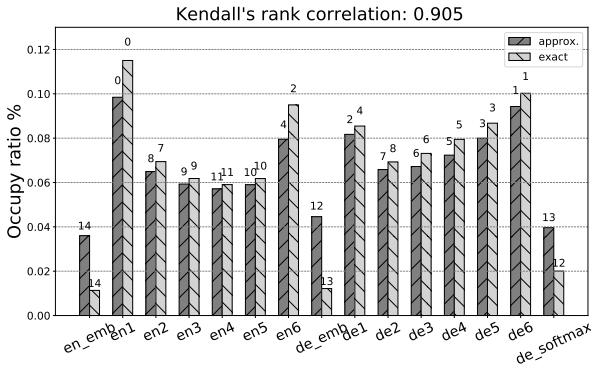

Figure 1: Evaluation of the sampling trick.

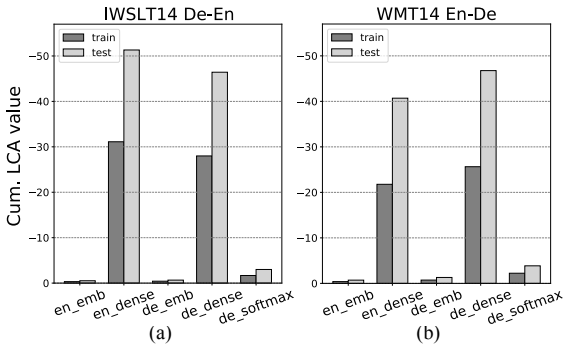

Figure 2: The cumulative LCA value of the sparse and dense weights of the Transformer.

for the following parameter groups: word embedding in encoder (en_emb); l-th layer parameters in encoder (en$l$); l-th layer parameters in decoder (de$l$); word embedding in decoder (de_emb); softmax matrix in decoder (de_softmax). In this way, both dimensions of $\bar{A}_{lca}$ are much less than those of A_{lca} in the original implementation, and thus it is affordable to compute $\bar{A}_{lca}$ during training.

3 Experiments and Analyses

3.1 Data and model

We conduct experiments on two translation benchmarks IWSLT14 De⇒En and WMT14 En⇒De. We use the Transformer *base* (Vaswani et al., 2017) from *fairseq* (Ott et al., 2019) for training and gathering LCA matrix as presented in section §2.2. Our NMT systems respectively achieve BLEU points of 34.4 and 27.7 on the test sets for IWSLT and WMT tasks, which are close to the state of the art.

3.2 Evaluating the sampling approximation

To prove the effectiveness of our approximation on gradient calculation, we conduct a simulated experiment as follows: we randomly sample $10K$ sentences from the IWSLT translation task and employ this small sampled data as the training data for running the brute-force approach. Figure 1 demonstrates LCA values' occupation ratios of each module group in the Transformer described in §2.2. As shown in Figure 1, the ranking between our approach (approx.) and brute-force approach are highly similar, with Kendall's rank coefficient as 0.905 (Kendall, 1938). So we adopt our approach to calculate LCA for subsequent analyses.

3.3 Experimental analyses

We conduct two main categories of analyses according to LCA: i) *cumulative* analysis: which tracks the cumulative LCA values from the beginning of training to the end; ii) *interval* analysis: which tracks the LCA values of each groups of model parameters during certain interval (t_1, t_2) of training.

3.3.1 Learning of sparse and dense weights

Currently sequence-to-sequence learning enjoys an explicit differentiation between encoder and decoder. This explicit separation may provide a bottleneck of gradient flow from the loss to the encoder. We visualize the cumulative LCA value of sparse and dense weights of Transformer in Figure 2.

Overall speaking, dense weights from encoder and decoder contribute similarly both on train and test. However, the sparse embeddings both contribute very little. This might because that the update frequency of dense weights is much larger than the sparse weights. However, the dense softmax weigths's LCA value (decoder's output embedding) is still far less than those middle layers.

3.3.2 Layer-wise learning dynamics

To analyze the layer-wise contribution of each encoder and decoder layer, we summarize the cumulative LCA value of each dense layer in Figure 3. There is an interesting sandwitch effect of the encoder where the beginning and the end layer contribute the most while the layer in between contribute less. For the decoder layers, the more higher the layer, the more contribution it makes to the loss change. It is very clear that from this modular view, different neural blocks provide similar effects on loss change for both training and test datasets.

To further understand the convergence property shown in Raghu et al. (2017): that lower layer converges earlier. We draw the interval LCA values along the whole training process in Figure 4. As you can see, higher layers tend to have smaller LCA values which means they contribute more

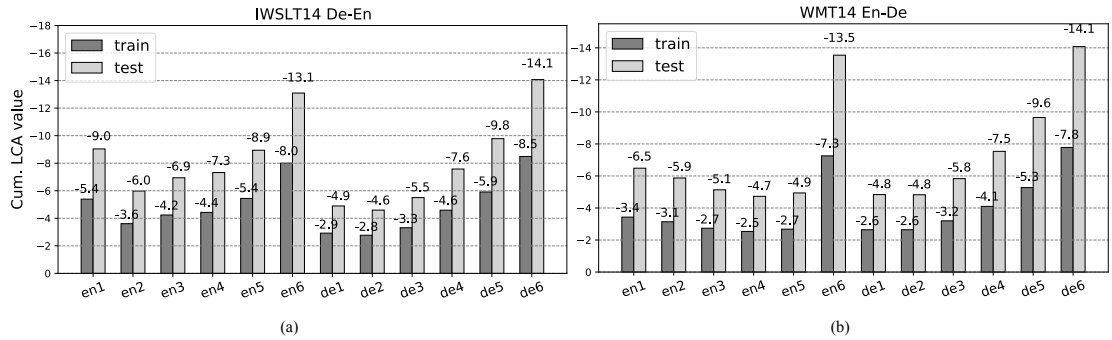

Figure 3: The cumulative LCA value of the layerwise dense parameters of the encoder and decoder on the two datasets, as a zoom-in of Figure 2's *en/de_dense* bars.

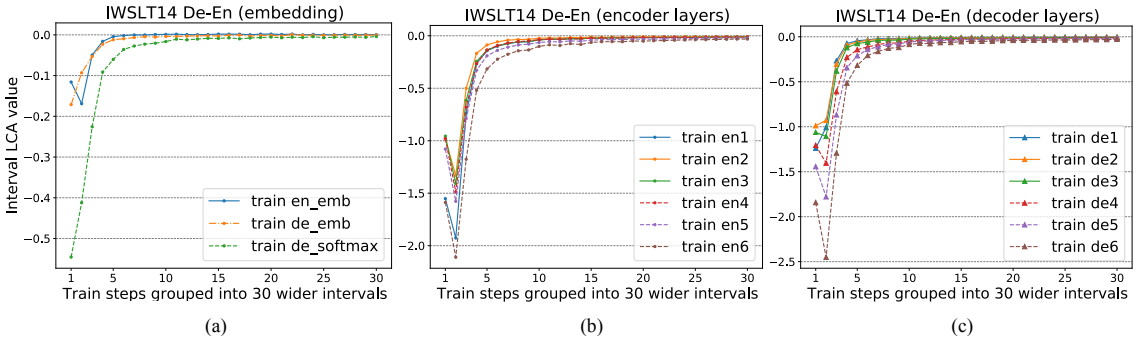

Figure 4: The interval LCA value on the IWSLT14 dataset allocated on different modules along training.

than lower layers generally at any training interval, functioning as an evidence that higher layers continue to evolve representations.

3.3.3 Learning of the embeddings

As the vocabulary size is large, we can not visualize the behaviors for the embedding of every word. We thereby split the vocabulary into 25 groups according to word frequency. Figure 5 visualizes the cumulative LCA values for all groups sorted by word frequency. From this Figure, one can clearly see that words with very high frequency occupy most LCA values than lower frequency words on both training and test datasets. This fact further provides an explanation for the well-known question, i.e. why infrequent words are difficult to be translated but frequent words are easy for NMT.

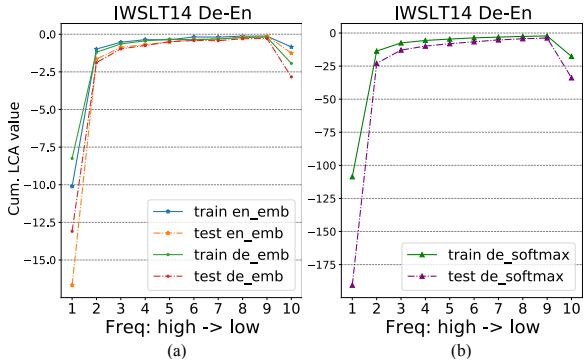

Figure 5: The cumulative final LCA value of sparse and dense embeddings divided by word frequency; the vocabulary is divided into 25 equal-sized groups, with last 16 groups shown as one large group of low-frequency words (the 10-th group).

4 Conclusion

In this paper we propose to use Loss Change Allocation (LCA) (Lan et al., 2019) for understanding learning dynamics of NMT. As a NMT model typically employs a large neural network in architecture, the brute-force implementation of LCA suffers from two challenges in both computation and storage, we instead present an improved approach to put it into practice for NMT scenario. Our experiment shows that the proposed approach is efficient and intensive experiments reveal some valuable findings: parameters of encoder (decoder) word embeddings and softmax matrix contribute less to loss decrease and those of the first layer in encoder and the last layer in decoder contribute most to loss decrease during the training process.

References

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

Remi Tachet des Combes, Mohammad Pezeshki, Samira Shabanian, Aaron C. Courville, and Yoshua Bengio. 2019. On the learning dynamics of deep neural networks. *International Conference on Learning Representations 2019*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Janice Lan, Rosanne Liu, Hattie Zhou, and Jason Yosinski. 2019. Lca: Loss change allocation for neural network training. In *Advances in Neural Information Processing Systems*.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. 2018. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. 2018. An empirical model of large-batch training. *ArXiv*, abs/1812.06162.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT*.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6076–6085. Curran Associates, Inc.

Naomi Saphra and Adam Lopez. 2018. Language models learn POS first. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 328–330, Brussels, Belgium. Association for Computational Linguistics.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.

Ravid Shwartz-Ziv and Naftali Tishby. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. 2017. Don't decay the learning rate, increase the batch size. *ArXiv*, abs/1711.00489.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.