Faster Beam Search for Neural Machine Translation

Hieu Hoang hieu@hoang.co.uk

Tomasz Dwojak ???
Adam Mickiewicz University

Kenneth Heafield ???
University of Edinburgh, Scotland

Abstract

Deep learning models are widely deployed for machine translation. In comparison on many other deep learning models, there are two characterisitics of machine translation which have not been adequately addressed by most software implementations, leading to slow inference speed. Firstly, as opposed to standard binary class models, machine translation models are used to discriminate between a large number of classes corresponding to the output vocabulary. Secondly, rather than a single label, the output from machine translation models is a sequence of class labels that makes up the words in the target sentence. The sentence lengths are unpredictable, leading to inefficiencies during batch processing. We provide solutions for these issues which together, increase batched inference speed by up to ??? on modern GPUs without affecting model quality. For applications where the use of maxi-batching introduces unacceptable delays in response time, our work speed up inference speed by up to ???. Our work is applicable to other language generation tasks and beyond.

1 Introduction and Prior Work

Deep learning models have been successfully used on many machine learning task in recent years in areas as diverse as computer vision and natural language processing. This success have been followed by the rush to put these model into production. However, the computational resources required in order to run these models have ben chanllenging. One possible solution involves creating faster models that can approximate the original model (??? Student-Teacher model). Another solution is tackling specific computationally intensive functions by approximating them (??? softmax).

Most current neural MT models follow an encoder-decoder architecture. The encoder calculates the word and sentence embeddings while the decoder generates the output sentence. The architecture within the encoder is still a subject of much research. (??? Kalshammer) used a convolution to encode the input and a recurrent neural network (RNN) for the decoder. RNNs was used for both encoding and decoding in (??? who used RNN 1st). (???) significantly improved translation quality by using LSTM was used in each RNN node. (??? Cho) used GRU that has the required properties of LSTM but is computationally cheaper. (??? Badhannau) added attention model which improved translation at a cost of slower speed.

There has been recent attempts to move away from the sequence-to-sequence models of RNN encoder-decoder. Some justify their architecture on faster inference as well as better

translation results. (??? Attention is all you need) and (??? course-to-fine) has been proposed as a fast alternatives to RNN-based models as it is possible to process more of the units in parallel.

It has been noticed that half precision arithmetic can be used for deep learning model without significan loss of model quality (???). Other solutions include specialized hardware, the most popular being graphical processing units (GPU) but other hardware such as custom processors (??? TPU), FPGA (??? Intel) have been used.

Many of these optimization are general purpose improvements for deep learning models, regardless of the task it is being used in. (??? Devlin) is a noticeable machhine translation-specific optimization which describe the speed improvements that can be obtained by techniques such as the use of half-precision, model changes and pre-computation.

2 Proposal

We based our model on the sequence-to-sequence model of (??? Cho) for machine translation models, but unlike (??? Devlin), we avoid solutions for the specific model. Therefore, our solution should be applicable to other models, architectures and task which have the similar characteristics. We envisage that our solutions would be of value to models used in text summarization, chatbot or image captioning.

We also choose to focus on the use of GPU, rather than CPU as pursued in (??? Devlin).

2.1 Softmax and Beam Search Fusion

The output layer of most deep learning models consist of the following steps

1. activation - multiplication of the input with the weight matrix for that layer
2. addition of a bias term to the resulting scores
3. activation function - in the output layers, this is commonly softmax or a variant, eg. log-softmax.
4. a search for the argmax output class, and probability is necessary.

In models with a small number of classes such as binary classification, the calculation of softmax and argmax is trivial and fast. The activation and bias term steps are also fast unless the dimensionality is large.

However, the output layer of most machine translation models frequently contains tens of thousand, if not hundred of thousands, classes corresponding to the vocabulary of the output language. For example, the best German-English system for WMT 2014 (??? Rico) contains 85,000 subword units in the target vocabulary. In these scenarios, the trivial procedure used to calculate the output layer takes up a significant amount of time. Table ???? shows the percentage of the total translationt time each step takes in our baseline system (described in Section ???).

??? Table of percentage each step takes

In addition, models with large number of classes frequently desire a list of the n-best classes rather than just the argmax, further complicating problem.

However, there are algorithmic similarites

3 Experimental Setup

4 Results

5 Conclusion

Acknowledgments

This work is sponsored by the Air Force Research Laboratory, prime contract FA8650-11-C-6160. The views and conclusions contained in this document are those of the authors and should

not be interpreted as representative of the official policies, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

References