CSE481 NLP Capstone: Blog Post 2

Team name: Bitmaps - **BI**narized **T**ransformers for **M**aking f**A**st Prediction**S**
List of members: Tobias Rohde, Anirudh Canumalla
GitHub URL: nlp-capstone

We have decided on the Binarized Transformers project and thus do not further discuss the other two ideas.

Project: Binarized Transformers

Pros	Cons
● Novel research idea ● Exciting Project! ● Useful for edge devices ● Improved security + privacy for phones and home assistants ● Potentially improve training performance	● Potentially major performance loss ● Difficult engineering problems ● High risk project ● Training from scratch not possible ● Transformers need a lot of data

It seems like binarizing transformers is a novel research idea, although there exists past work on binarizing recurrent neural networks. The project is interesting and we are both very excited about it. It could potentially live beyond the course, assuming we are able to get convincing results. Highly efficient transformers have many use cases. Since binarizing Transformers would reduce memory usage and speed up inference, one obvious use case is to deploy binarized transformer models to low power edge devices. We might also be able to improve training performance, but it is not clear to us yet if this will be possible. This would be beneficial since transformers take long to train, despite their ability to be parallelized. However, we also think that training might actually become slower, depending on how we adjust the training procedure. Binarized transformers would be particularly useful for offline tasks on low power devices such as smartphones or smart assistants. The devices could perform machine translation, speech recognition, text to speech and language processing completely offline.

Overall, we consider this a high risk project. There might be a major performance loss when binarizing the components within the transformer and we will likely have to tune which layers we binarize and “how much”. Additionally we will likely need to invoke tricks such as scaling layers by learned non-binary constants, as in the XNOR paper. There is also a major amount of engineering choices to make in implementing this, as the performance will also depend on how efficiently we implement binary operations. We also need to clearly investigate when binarized transformers would be a preferable choice over standard FP16/FP32 architectures. Again, the project is high risk, high reward: We could come up with a more efficient version of transformers, as was done in the XNOR-NET paper for convolutional neural networks. But in the worst case, such a strategy is not applicable to transformers and we end up with no good results. However, in that case the project still has value, since we will learn a lot and also contribute to research by having shown what approaches don't work.

For development, we will use Python and PyCharm as our IDE. As our main library for deep learning we will use PyTorch. For NLP utilities, we are planning on using AllenNLP. We will use the existing BERT implementation from huggingface as a starting point for experimenting with binarizing the weights. First, we will try to convert pre-trained FP32 weights into binary weights. This process will likely not require any data, since we will use pre-trained weights. In order to evaluate the models performance, we are planning on using the Penn Treebank or WikiText datasets, on which we will calculate model perplexity for the language modeling task. Language model is a simple task that does not require annotated data and perplexity is a simple measure. Thus we likely use it as a task to determine model performance throughout the project.

Finally, below are some topics that we would like to hear about in lecture:

- AllenNLP (the history, motivations, use cases, how it's built, small demo (?))
- Approaches to making models smaller and/or more efficient
- Machine Translation (Side note: Wasn't taught in 447/547 last quarter)
- Transformers