1. Project Title
Improving BERT Memory Efficiency

2. Team members
Kristina Yuchen Tian, Kate Alys Lassiter

3. Goal/Objective
In the realm of contextualized embedding models, like Bidirectional Encoder Representations from Transformers (BERT) models, overparameterization is a major issue. This project will explore the relatively new approach of block structured pruning techniques to effectively prune insignificant weight parameters while maintaining accuracy and reducing memory constraints. This has the potential to allow real time execution of the model. Likewise, other techniques to reduce memory utilization of deep neural networks may be explored.

4. Approach/Techniques
As a starting point, we will attempt to implement the block structured pruning technique proposed by the authors in Kishlay Jha et. al [2]. This involves taking an already trained transformer and its weights as an input. Every layer's weight matrix is then divided into K blocks. In order to minimize the overall loss of all layers, a penalty is added to the loss function: the group lasso regularization penalty for each row/column of that block, summed over every layer and every block in the network. Another penalty is added for every layer in every block, multiplying that block's weight matrix by one minus the weight matrix to push the weights close to zero or one.

$$min\, f(\{\boldsymbol{\theta}_n\}_{n=1}^{N}, \mathcal{D}_t) + \lambda_1 \sum_{i=1}^{N}\sum_{j=1}^{K} ||[\boldsymbol{\theta}_{ij}]_{p,:}||_2 + \lambda_2 \sum_{i=1}^{N}\sum_{j=1}^{K} (\boldsymbol{\theta}_{ij} \times (1 - \boldsymbol{\theta}_{ij})) \tag{13}$$

For block-based column pruning, we solve:

$$min\, f(\{\boldsymbol{\theta}_n\}_{n=1}^{N}, \mathcal{D}_t) + \lambda_1 \sum_{i=1}^{N}\sum_{j=1}^{K} ||[\boldsymbol{\theta}_{ij}]_{:,q}||_2 + \lambda_2 \sum_{i=1}^{N}\sum_{j=1}^{K} (\boldsymbol{\theta}_{ij} \times (1 - \boldsymbol{\theta}_{ij})) \tag{14}$$

This regularization problem is solved and once the network has been trained with this loss function, one final step is performed. The L2 norm for each column and row in the new weight matrix is calculated and if the norm is less than the defined threshold, zero out the weights of that row/column. Pseudo code is shown below.

Algorithm 1 KNOWLEDGE-GUIDED PRUNING

Input: Pretrained transformer model with weight matrix $\boldsymbol{\theta}$, threshold ϵ
Output: Pruned weight matrix $\boldsymbol{\theta_s}$
Initialize $\boldsymbol{\theta_s} = \boldsymbol{\theta}$
Divide $\boldsymbol{\theta}_s$ into K matrices: $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, .., \boldsymbol{\theta}_K$
Set $i = 1$
Set total number of iterations = $maxT$
Solve the regularization problem (13),(14) using ADAM
while i ≤ maxT **do**
    $l_2\_norms_l$ equals the l_2 norm of each p-th/q-th row of $\boldsymbol{\theta}_s$
    if $l_2\_norms_l \leq \epsilon$ **then**
        $\boldsymbol{\theta}_s(p,:) = 0$
        $\boldsymbol{\theta}_s(:,q) = 0$
    end if
end while
$\boldsymbol{\theta}_s = concatenate\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2..., \boldsymbol{\theta}_K\}$

It remains to be seen whether we will select row-based, column-based, or row and column-based pruning. Likewise this is only one suggested implementation of block structured pruning. We will further investigate and select the algorithm that seems most appropriate for our specific task. Given the slow training time of these models, our limitation to a single GPU in GCP, and the limited project duration, an alternate approach may be followed that still effectively addresses memory constraints of deep networks. If time permits, other factors such as compression, batch size, optimizers, etc. will be varied to boost performance.

5. Challenges
We need to have our own implementation for the loss function and the pruning algorithm, because no code is given by the paper authors. Therefore, this could be a potential challenge. In many cases, the implementation details matter a lot in the performance of the algorithm.

There are many different hyperparameters that we can fine tune for the project in order to achieve a better performance. For example, the number of blocks that we prune. However, that may require a lot of time to do the training. BERT models are typically very slow to train given their huge number of parameters. Especially, given our budget constraints, the training time could be long. In which case, we will adapt our approach accordingly to one that addresses memory constraints in an alternate way.

Our goal for the project is to achieve a real time demo of the model. For a trained and fine-tuned model, during our presentation, we will give it an input and the aim is to have the output shown in real time. This could be a challenge because if the model runs too slow, real-time output demo might be hard.

There are many versions of the pretrained BERT available online at Huggingface's website [3]. In order to validate our approach and the effectiveness of the algorithm, we need to choose a pre-trained BERT model wisely, and choose a dataset for fine-tuning accordingly.

Block structured pruning techniques are a relatively new field and very few examples are available. Therefore, the scarcity of resources can be a big challenge for the project. Also, the model is rather complex, and this also adds to the challenge.

6. Implementation details: hardware (type of compute GPU/TPU etc, cloud based, edge devices), software (framework, existing code to reuse), dataset

For the hardware, in this project we are going to use Google Cloud Platform (GCP). For the given quota we have, we could afford 1 GPU for the training task [1]. The current plan does not include using TPUs, as they are even more expensive given the budget constraints.

For the current plan, the baseline GPU we plan to use is NVIDIA K80, as its price is $229.95 per GPU per month, which is within our budget. If Google Cloud Platform allows us to pay for only the training hours we use, the ideal GPU we are looking to use would be NVIDIA V100. NVIDIA V100 has larger GPU memory as compared to K80 and is expected to have better performance.

We will be using Tensorflow's Keras package. According to our experience in deep learning model building and training, Keras is the package that is clear and requires less work with class structures. The challenge is that there is no existing code available, so we will implement our own versions.

As mentioned in previous sections, the dataset needed for fine-tuning would depend on which pre-trained BERT model we choose. The model we will be using is bertweet-base-sentiment-analysis [3]. Therefore, the dataset for fine-tuning will be data that is similar to tweets. For example, we could use Twitter data, or other online platforms like Reddit data. Some pre-processing techniques will be needed to the dataset before training. If it turns out that the above-mentioned BERT model's performance is not as expected, we will use the baseline BERT and do fine-tuning. In this case we could select any dataset.

For our proposed model to use, which is the BERT tweet sentiment analysis one, we will be fine-tuning the model using a dataset from Twitter or other similar online platforms. Then we will train it using the block structured pruning technique proposed. For the current plan, we will be doing a 3-class classification for the sentiment analysis, i.e. positive, neutral, and negative. The model performance will be measured using accuracy, the memory it utilized, and its number of parameters, as our block structured pruning technique will reduce these aspects.

7. Demo planned

We plan to test this technique on at least one data set and present our findings. During the presentation we will give a live demo of model output given an input.

8. References

"GPU Pricing | Compute Engine: Virtual Machines (VMS) | Google Cloud." *Google*, Google, https://cloud.google.com/compute/gpus-pricing?_ga=2.194800301.1808667178603-266478316.1664208813.[1]

Jha, Kishlay, et al. "Knowledge-Guided Efficient Representation Learning for Biomedical Domain." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, https://doi.org/10.1145/3447548.3467118. [2]

"Models." *Hugging Face*, https://huggingface.co/models?search=bert.[3]