Intel Labs - Deep Learning/NLP student technical interview

The following document describes a simulation of a task you could encounter in our team, that is, read a paper, understand the techniques, apply on a model and examine how it works.
Please ***read the whole document*** and make sure you understand all the steps, including the tips at the end!.
If you have any questions, at any time during the assignment, please contact Peter (peter.izsak@intel.com / 054-5454542)

Reading a paper

1. Read the following paper: https://arxiv.org/abs/1710.01878
2. Summarize the paper and method used for pruning (up to 1 page). Expand on the technical part (how the method works, algorithms, implementation details, tasks). Focus on the main findings on the paper regarding sparsity rates and depths on networks.

Tip: Don't delay working on the next tasks if you've finished working on the paper.

Implementing on BERT

In this part you will implement the method presented in the paper on BERT, fine-tune the network, prune weights and show the results of your model visually.

Recommended initial steps:

- Get familiar with HuggingFace open source library, read documentation and go over some of the most popular notebooks and examples (text classification is the most recommended one): https://github.com/huggingface/transformers
- Get familiar with the sequence classification (text-classification) task using GLUE benchmark dataset (see in examples), download the dataset and make sure the example or notebook runs well.
- You can develop the model on your own laptop/desktop (with or without a GPU) and transfer the model for fine-tuning to google colab once it is running well on a few examples if you need GPU powered backend.

Before you start:

- If you're not familiar with BERT yet, please review https://arxiv.org/abs/1810.04805 (BERT paper) and get familiar with the model's structure and training methods.
- The internet is full of other guides/tutorials explaining how BERT works and it might be worth reading through.

- Prepare an implementation plan for the algorithm and be ready to show it along with your code in the final review. Be ready to go over all the material you have covered.

Tasks:

1. Pre-development task:
    a. Summarize BERT's model weights, how many parameters? Layers? etc. be specific but brief.
    b. Which layers weights are the top candidates to be pruned?
2. Implement the pruning strategy found in the paper:
    a. Implement the pruning functionality and scheduling
    b. Implement ability to control the scheduling as configurable parameters
    c. Apply the pruning technique **only** on the feed-forward part of the transformer cell (up-project and down-project)
3. Train the model:
    a. Choose a pruning rate and schedule and fine-tune the model along with the pruning strategy. Validate that the model is training and pruning weights.
4. Visualize the results
    a. Present visually the % of non-zero weights of a layer's tensor (or all) as the function of training steps. (log, tqdm, tensorboard, any option is okay)
    b. Present the model accuracy metrics (see GLUE Benchmark for specific metrics per task)
5. Improve technique:
    a. Briefly describe how to change the pruning strategy so that weights will be pruned in blocks (2x2, 4x4, 8x8 for example)
    b. Suggest other strategies that would make the model smaller/faster. Briefly summarize your ideas.
6. Export to ONNX (Bonus):
    a. Export the model to ONNX format
    b. Prepare an example (1 or multiple demo sentences)
    c. Run the ONNX model in inference using onnx-runtime engine and show performance and accuracy
7. Important notes when training:
    a. Use **bert-base-uncased** BERT model as the base model
    b. Use the **MRPC** task as the training dataset (from GLUE benchmark)
    c. Use default settings for every other parameter

Important tips for success:

- Focus on making your code work. Your highest priority is to make sure your code works and is correct. Model accuracy on the data is only a secondary priority, but make sure the accuracy is not null or 0 from the first step. (think about how much to prune? Start with a low number)
- Your priority should be to complete the tasks sequentially; focus on getting the tasks done correctly rather than completing all above tasks on time.

- You can implement a smaller model to make sure your code works and then transfer to BERT (or take a 1 layer BERT for example)
- You must use HuggingFace Transformer github repository, pytorch or Tensorflow
- You cannot use any pruning/scheduling/compression libraries for DL (you must implement the process yourself)
- You can use any other package for developing the model/tasks (numpy, tensorboard or weight-and-biases for visualization, etc..)