Background
This project adapts pre-trained models of Bidirectional Encoder Representations from Transformers (BERT) to the NLP task of answer span prediction ("QA") on the Stanford Question Answering Dataset (SQuAD 2.0). The original BERT implementation ("origBERT") achieved SOTA performance of 88.5 F1 on a SQuAD 1.1 dataset with 100,000+ answerable context-question tuples. However, when origBERT is used on the much more challenging SQuAD 2.0 dataset, released subsequently with 50,000+ additional questions with no answer, performance drops to 73.1 F1 baseline for this project.

Objective
Improve the fine-tuning procedure for BERT-base and BERT-large models to perform better than origBERT on SQuAD 2.0 measured by F1 above baseline of 73.1.

Approach
I considered but ruled out potential improvements to the pre-trained BERT models and underlying Transformer architecture because pre-training much more compute time than fine-tuning. Instead, I narrowed my exploration to the following potential improvements to the fine-tuning process used in origBERT:

- **Deeper QA architecture** I started with a task-specific architecture that can learn more powerful relationships than the single FFNN in origBERT.
- **Concatenate multiple BERT output layers** In origBERT, only the hidden-states of the last encoding layer are fed into the QA.
- **Weighted cost function** Impossible vs Possible classes are unbalanced in the entire dataset (33/67) and between the Train and Dev splits.
- **Ensemble two different BERT models** I trained a second class of model with its sole objective being the binary classification of whether an input sequence is impossible. I then ensemble the results of the QA and BSC model as illustrated below.
- **Train on larger and/or balanced QA Dataset** I tried to add examples from the TrivaQA dataset to either increase the size of or balance the classes within the SQuAD 2.0 training data split.

Results
BERT-base improved by 4.5 points to 77.6 F1. Notably, this is only 2.3 points below the origBERT-large baseline performance, which is 3x larger and considerably more expensive to train. Maximum F1 of 81.4 achieved with a 3-layer QA BERT-large model.

Discussion
- The 6.8 point increase from BERT-base to BERT-large is driven by the increased power of a model with 340M vs 110M parameters rather than by any improvements I made. But it was not trivial to train BERT-large models. This capability allowed me to explore the effects of BERT model size, batch size, training epochs, and task-specific QA architecture.
- 3-layer QA architecture produces a 2.4 point improvement. Increasing QA depth to 6 layers resulted in very little incremental improvement (+0.05).
- Overfitting was in issue in all models, and required careful monitoring of test-validation splits, checkpointing and early stopping when val loss began to diverge.
- Concatenating BERT’s last 4 encoding layers for input to QA hurt performance by (-0.5 F1) so I stuck with the last-layer only.
- Ensembling two different BERT models, one for binary classification of impossible questions, improved performance by 1.4 F1. Using a weighted cost function for the binary classifier helped correct the class imbalance between q’s.
- I tried to add TriviaQA examples to the train dataset, but was unable to replicate the origBERT results.
- Tuning hyperparameters, including max_query_length, null_score_threshold, and test-val checkpoint early stopping, improved performance by an additional 0.7 F1.

Predictive Models

Output:  
- BSQ: Yhat ε Z (0, +1)  
- BQA: Yhat ε Z (0, ..., 384)

Loss Fns:  
- BSQ: Weighted CrossEntropy  
- BQA: CrossEntropy

Fine-Tune: SQuAD 2.0  
- Two models:  
  - BSQ binary classify impossible  
  - BQA predict start-end positions of answer span in context paragraph

Pre-Train Learning:  
- 40 epochs, 256 batch size  
- 10% dropout, GelU  
- Adam, α=1e-4, beta1=0.9, beta2=0.999, L2 decay 0.01

Pre-Train: BooksCorpus & English Wikipedia  
- Two unsupervised tasks:  
  - Masked Language Model  
  - Next Sentence Prediction

Input:  
- Packed pair of A/B sentences up to max_seq_length=384

References
- Jay Alammar, The Illustrated BERT. (https://jalammar.github.io/bert-illustrated)
- AND MANY OTHERS... See references in accompanying project paper.