Different type
Models - bert, distilbert
Datasets - GLUE task/ MRPC
Pytorch - f32, int8
Tensorflow - f32, int8
ONNX implementation - int8, f32
Input data type - f32/int8

Basic algorithm for using bert based models:
There are two kind of applications using 1) encoder and decoder and 2) encoder
The encoder is used as a feature generator, where the input sentence is passed through an encoder to get a feature list which is then optimized by a classifier network above it.
So this would require fine-tuning training for different tasks. This would involve regression or classification output

Encoder-Decoder based would involve a structured input-output. This would involve tasks like language translation. Sentence completion etc.

So it involves a stage of training followed by inference.

Transformers library inherently supports models exported in Pytorch and Tensorflow float32. They are then converted to onnx based models for inference. Initially, Transformer themselves supported an exporting framework. Convert_graph_to_onnx

Model used: bert-base-cased
opset=11
The Pytorch Bert Pytorch 0.014789018630981445
The ONNX onnx/bert-base-cased_pt.onnx 0.028888933658599854

ARM
The Pytorch Bert Pytorch 0.03160018444061279
The ONNX onnx/bert-base-cased_pt.onnx 0.038375723361969

Why is IBERT consuming more time?
The Pytorch IBert Pytorch 0.016733992099761962
ARM: The Pytorch IBert Pytorch 0.03542795419692993

ARM: Pytorch works only with numpy 1.21 while the Tensorflow version works only with 1.19.2

Inference:
Tensorflow numbers seems to be always higher than Torch and ONNX numbers

ONNX optimization and Torch numbers seems to be almost close.

Torchscript seems to produce more optimization than ONNX optimization. It has a JIT compiler

Int8 quantization does not seem to produce that much optimization in Intel platforms?

Int8 seems to produce good optimization on a ARM platform

DistillBert produces the most optimization

TorchScript int8 seems to be the best optimized configuration for Intel

The best configuration for ARM platform is distillbert onnx int8 configuration converted from pytorch model

ONNX exported from tensorflow

Int8 for torch and torchscript does not work since it does not have the necessary operations
self._packed_params = torch.ops.quantized.linear_prepack(weight, bias)
RuntimeError: Didn't find engine for operation quantized::linear_prepack NoQEngine

Onnx conversion for BERT fails with Graph not DAG when used with pytorch
The onnx module converted from pytorch also seems to produce similar values as

BERT: Pytorch onnx conversion
Size of full precision ONNX model(MB):417.6670169830322
Size of quantized ONNX model(MB):106.50384902954102

DistilBERT: Pytorch onnx conversion
Size of full precision ONNX model(MB):253.1628122329712
Size of quantized ONNX model(MB):63.48833084106445

The next is using run_benchmark.sh which uses onnxruntime for optimization for different runs

The next step is to train models for specific applications. In this experiment, we try to train the model for GLUE applications, specifically MRPC tasks.
The task involves matching a pair of sentences that are semantically similar or not.

Pytorch quantization values:
Size (MB): 417.72501850128174
Size (MB): 173.08655261993408

f32
{'acc': 0.8602941176470589, 'f1': 0.9018932874354562, 'acc_and_f1': 0.8810937025412575}
Evaluate total time (seconds): 15.5

Int8
{'acc': 0.8504901960784313, 'f1': 0.8942807625649914, 'acc_and_f1': 0.8723854793217114}
Evaluate total time (seconds): 14.8

ONNX full precision model size (MB): 417.6695041656494
ONNX quantized model size (MB): 104.80780410766602

F32 Onnx conversion
{'acc': 0.8602941176470589, 'f1': 0.9018932874354562, 'acc_and_f1': 0.8810937025412575}
Evaluate total time (seconds): 11.0

Int8 Onnx optimization:
{'acc': 0.8529411764705882, 'f1': 0.8989898989898989, 'acc_and_f1': 0.8759655377302435}
Evaluate total time (seconds): 8.0

The pytorch int8 conversion does not seem to optimize the model.
This function uses quantize_dynamic instead of quantization
The loss in accuracy is very small.
The model is initially trained model for the specific task

TFLITE conversion:
Intel CPU For 100 inferences
Total time inference wo conversion 4.535388469696045/4.617389917373657
Total time inference int32 3.7203991413116455/1.049537181854248
Total time inference int16 3.633791446685791/3.2079484462738037
Total time inference int8 2.4201579093933105/2.3827571868896484

Total time inference wo conversion 8.767888307571411
Total time inference int32 2.1622798442840576
Total time inference int16 6.5904576778411865
Total time inference int8 4.854951620101929

ARM
Total time inference wo conversion 10.378628253936768
Total time inference int32 2.429830551147461
Total time inference int16 8.733739614486694
Segmentation fault (core dumped)

Profiler ONNX:
Convert Pytorch and Tensorflow int8 to onnxruntime and profile it for understanding different functions at a framework level https://onnxruntime.ai/python/auto_examples/plot_profiling.html

Onnx optimizations:

Onnx quantization results

Pytorch quantization results: f16 and int8

Tensorflow quantization results: TFLITE quantization, f16

Training different applications glue benchmark

Try reasoning out absolute values across pytorch, tensorflow and onnx. But atleast show the relative performance improvements

Explore optimized_model.use_dynamic_axes()
optimized_model.save_model_to_file(optimized_model_path)

Explore different onnx providers - the runner for onnx is good in onnx_converter.py
Explore keras2onnx

Explore QA dataset

Different optimizations

ONNX implementation:

- Pytorch implementation seems to be faster than Tensorflow implementation only on Intel Machines
- Pytorch converted to ONNX

The Pytorch Bert Pytorch 0.025295858383178712
The ONNX onnx/bert-base-cased.onnx 0.01583794355392456
The ONNX bert.opt.onnx 0.010543863773345947

ARM: Works with onnx runtime 1.8 version but Huggingface is optimized for 1.4 version
The Pytorch Bert Pytorch 0.030496835708618164
The ONNX onnx/bert-base-cased.onnx 0.03884810924530029

bert-base-uncased	8	8	0.063	- Tensorflow
bert-base-uncased	8	8	0.081	- Pytorch

distilbert-base-uncased	8	32	0.062	- TensorFlow
distilbert-base-uncased	8	32	0.072	- Pytorch

Convert_graph_to_onnx.py - seems to work good for Pytorch models but not for tensorflow

Tensorflow Inference time for sequence length 512 = 976.23 ms
ONNX Runtime cpu inference time for sequence length 512 (model not optimized): 1870.42 ms

Tensorflow Inference time for sequence length 512 = 991.14 ms
ONNX Runtime cpu inference time for sequence length 512 (model not optimized): 1001.24 ms

Tensorflow Inference time for sequence length 512 = 836.99 ms
ONNX Runtime cpu inference time for sequence length 512 (model not optimized): 994.39 ms

ONNX Runtime cpu inference time on optimized model: 997.64 ms

Onnxruntime tensorflow - static and dynamic quantization
https://github.com/microsoft/onnxruntime/blob/master/tools/python/remove_initializer_from_input.py
https://onnxruntime.ai/docs/how-to/mobile/ - arm service provider -
https://onnxruntime.ai/docs/reference/execution-providers/

Various GLUE benchmark tasks - https://openreview.net/pdf?id=rJ4km2R5t7

Pytorch supports int8 quantization
Ibert int8 on ARM processor

pip install onnxruntime-tools - looks like it is deprecated and not working

https://netron.app/ - try showing models before and after
https://huggingface.co/models?sort=alphabetical - all pre-trained models
Talk about memory consumed

ARM build: transformers requires tokenizers==0.10.1

ARM Build tensorflow:

Clean tensorflow: bazel clean --expunge
BUILD_OPT=1 ./remake_arm_64_natively.sh

Tensorflow requires numpy version 1.19.5
While torch requires numpy version >1.20. This seems to work for Intel CPU since that torch build seems to work well with 1.19.5 version but it is not available for arm.
https://download.pytorch.org/whl/torch_stable.html
Trying benchmark.py without torch does not seem to help.
https://github.com/microsoft/onnxruntime/blob/master/onnxruntime/python/tools/transformers/benchmark.py
Issue: https://bugs.archlinux.org/task/69495

https://onnxruntime.ai/docs/tutorials/inferencing/
https://github.com/microsoft/onnxruntime/pull/8457/files

pip install --upgrade onnxruntime==1.4.0 - ARM support not present and PyTorch seems to be optimized with this version. The later version seems to degrade the performance

https://docs.google.com/spreadsheets/d/1QeVtuVBjfhB6v-DVdXN9P136fxczlTM3Xf7U98TGmE4/edit#gid=0 - Performance

Benchmarking Application:
https://github.com/huggingface/notebooks/blob/master/examples/token_classification.ipynb
https://github.com/huggingface/notebooks/blob/master/examples/text_classification.ipynb

Tokenizer profiling:

Explore IQUANT, Bert as a Service, ONNX implementation

https://arxiv.org/pdf/2010.13382.pdf - FastFormers

Later:

Pytorch implementation
Bert as a service