50.055 Machine Learning Operations Hands On Session Notes

Authored by: Daniel Dahlmeier, January 2023.

Session 3: Accelerated Inference with Optimum and Transformers Pipelines

Objective

In this session, you will optimize a trained Huggingface transformer model to make it ready for deployment. We will use the Optimum library and ONNX to reduce model size and improve latency of a pre-trained Question Answering model. The session is based on the excellent blog by Philipp Schmid from Huggingface.

1. Start Jupyter notebook

Start a Jupyter notebook on the SUTD cluster, AWS, Colab or your own computer. You do not need a GPU for this lab.

2. Checkout Github repository

Check out the Github repo by opening a terminal and run

```
~$ git clone https://github.com/ddahlmeier/sutd-mlops-course-code.git
```

Open the notebook **03_optimize_onnx.ipynb** in the folder sutd-mlops-course-code.

3. Development environment

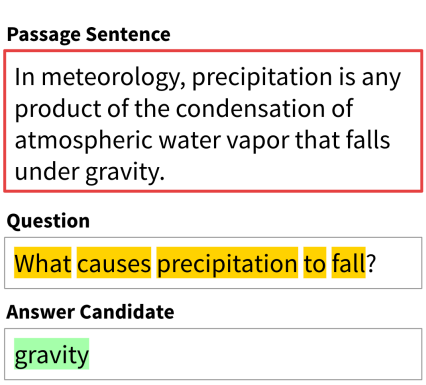
Run the first notebook cell to install the optimum package.

4. Load RoBERTa SQUAD model

In this session, we are not training our own model. Instead we will be using a pre-trained RoBERTa mode for question answering from Huggingface model hub.

RoBERTa is based on BERT but introduces a couple of improvements in the way it was trained. Read theis blog about RoBERTa: https://huggingface.co/docs/transformers/model_doc/roberta

The model has been trained to perform question answering on SQuAD, the Stanford Question Answering Dataset. SQuAD has been a very popular NLP benchmark. Check out the SQuAD website and browse a few examples: https://rajpurkar.github.io/SQuAD-explorer/



- Between question and answer

cause---gravity
precipitation---gravity
fall---gravity
what---gravity

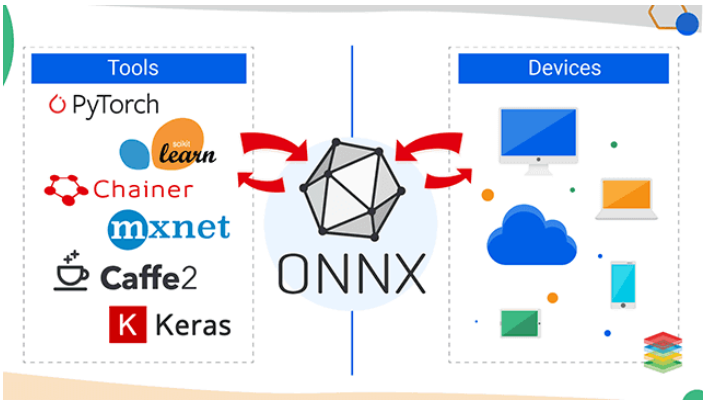
5. Convert model to ONNX

We load the vanilla model and convert it to the ONNX format. ONNX stands for Open Neural Network Exchange. It provides an open interchange data format for neural networks. This allows

developers to convert their models from PyTorch, TF or other frameworks into a common format. This is typically done to make it easier to deploy models. Check out the ONNX project here: https://onnx.ai/get-started.html

```
from optimum.onnxruntime import ORTModelForQuestionAnswering

model = ORTModelForQuestionAnswering.from_pretrained(model_id, from_transformers=True)
```



6. Run question answering example

We create a pipeline and run a simple question answering example, just to confirm that it works.

```
optimum_qa = pipeline(task, model=model, tokenizer=tokenizer, handle_impossible_answer=True)
prediction = optimum_qa(question="What's my name?", context="My name is Philipp and I live in Nuremberg.")

print(prediction)
```

7. Optimize model

Use the optimum library to apply graph optimization, such as operator fusion and constant folding to accelerate latency and inference. Then save the model to disk and run the simple question answering example again. It should run faster already.

```
# create ORTOptimizer and define optimization configuration
optimizer = ORTOptimizer.from_pretrained(onnx_path)

optimization_config = OptimizationConfig(optimization_level=99) # enable all optimizations

# apply the optimization configuration to the model
optimizer.optimize(save_dir=onnx_path, optimization_config=optimization_config)
```

8. Quantize the model

Next, we apply quantization to the model weights to reduce model size.

```
# create ORTQuantizer and define quantization configuration
quantizer = ORTQuantizer.from_pretrained(onnx_path, file_name="model.onnx")
qconfig = AutoQuantizationConfig.avx512_vnni(is_static=False, per_channel=True)

# apply the quantization configuration to the model
quantizer.quantize(save_dir=onnx_path, quantization_config=qconfig)
```

Then we compare the model size of the vanilla model and the quantized model. The quantized model is only about ¼ of the original size.

```
size = os.path.getsize(onnx_path / "model.onnx")/(1024*1024)
print(f"Vanilla Onnx Model file size: {size:.2f} MB")
size = os.path.getsize(onnx_path / "model_optimized_quantized.onnx")/(1024*1024)
print(f"Quantized Onnx Model file size: {size:.2f} MB")
```

```
# Vanilla Onnx Model file size: 473.51 MB
# Quantized Onnx Model file size: 119.15 MB
```

Again, create a pipeline and run the question answering example. It should load even faster.

9. Evaluate performance

How good is the optimized and quantized model compared to the original? Load the SQuAD2 dataset to compare the performance of the orignal and optimized-quantized model. Because the dataset is large, we just load 10% of the validation dataset. Feel free to play with larger portions of the dataset or the full dataset later.

The result shows that the drop in performance is very small. We get more than 98% of the original performance with a model that is a quarter the size.

 # vanilla model: exact=81.12889637742207% f1=83.27089343306695%
 # optimized model: exact=81.12889637742207% f1=83.27089343306695%
 # quantized model: exact=79.78096040438079% f1=82.22196132052908%

10. Evaluate speed

How fast is the optimized and quantized model compared to the original? We run a simple latency test with a slightly longer version of the original sample question. The code runs a few warmup loops to take care of effects like model loading, then executes the QA example 100 times while measuring the latency.

The result shows that otpmized and quantized model is about 30% faster than the original. Note that the exact numbers depend on many factors, such the exact hardware, and therfore your numbers can look different. You should, however, see some speedup.

 # Vanilla model Average latency (ms) 126
 # Optimized & Quantized model Average latency (ms) 89

11. Cleanup

Please clean up all resources.

- Shut down the kernel and logout of the SUTD Education Cluster

Conclusion

Optimizing a model before deployment can be important to achieve low latency and cost-efficient deployment. This session provided the first experience with how to optimize models with Optimum and ONNX.

Additional Resources

- Huggingface ONNX tutorial: https://huggingface.co/blog/optimum-inference
- ONNX tutorials: https://github.com/onnx/tutorials
- RoBERTa paper: https://arxiv.org/abs/1907.11692
- SQuAD paper: https://arxiv.org/pdf/1606.05250v3.pdf