# **Speeding up BERT Inference: Quantization vs Sparsity**

1. **Using the right library**

Onnx or OpenVINO

1. **Quantization**

replacing the floating point weights in your model with int8 weights. This can typically save a lot of memory space, but not execution time!

[Dynamic Quantization on BERT](https://colab.research.google.com/github/pytorch/tutorials/blob/gh-pages/_downloads/dynamic_quantization_bert_tutorial.ipynb)

1. **Pruning**

Introducing zeros (aka sparsity) in the weight matrices, promising both memory and compute savings. BUT pruning is even more challenging than quantization as it is faster only if the sparse matrix contains more than 98% zeros! Typically, one can afford at most 90% sparsity or maybe 95% sparsity without losing too much accuracy.

