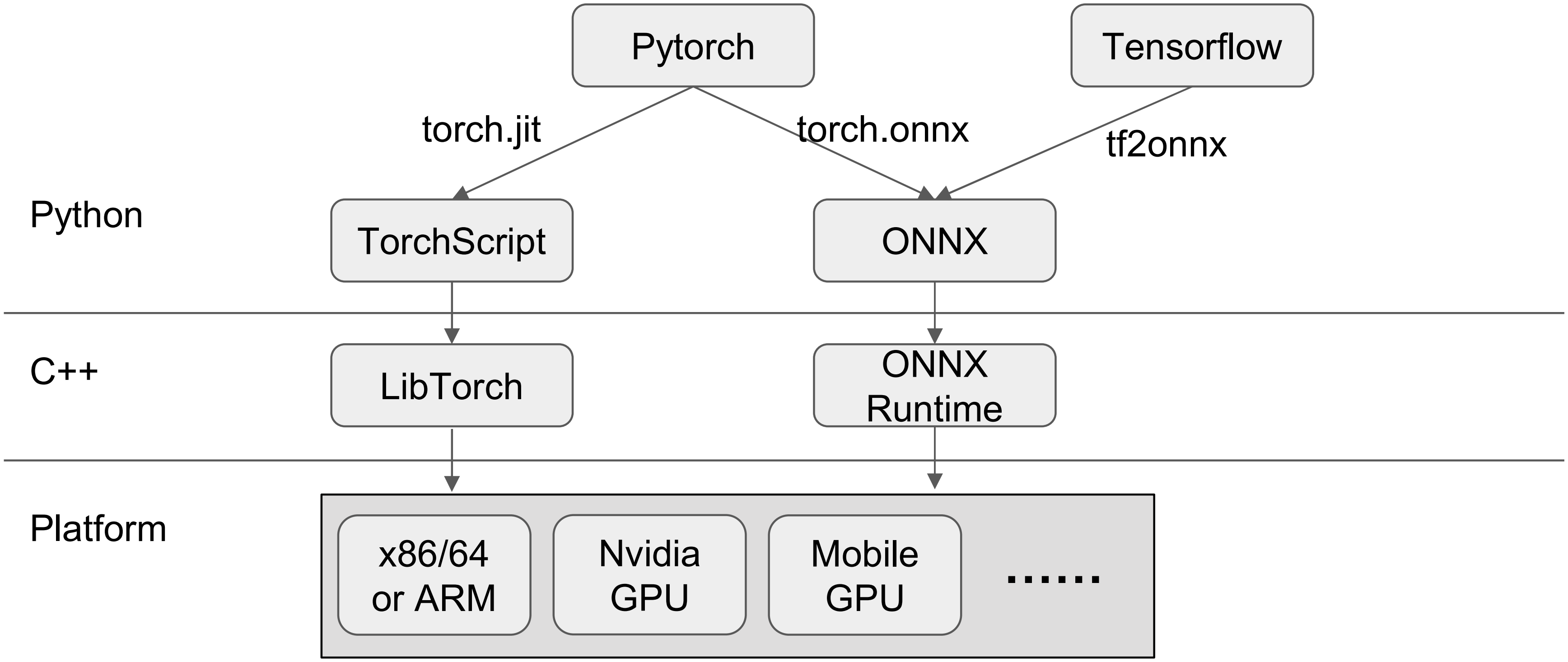
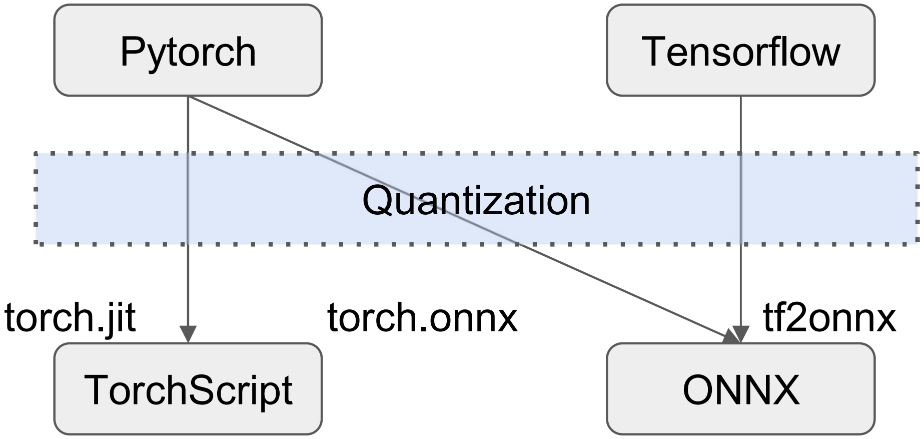
Edge device deployment

Workflow


Pytorch
Tensorflow
torch.jit
torch.onnx
tf2onnx
Python
TorchScript
ONNX
C++
LibTorch
ONNX Runtime
Platform
x86/64 or ARM
Nvidia GPU
Mobile GPU
……

Workflow *quantization*



Quantization methods:

1. Dynamic quantization: directly convert the model to int8
2. Static Quantization: observe batch data's distribution first then convert the model to int8
3. Quantization-aware training: mimic the float32 model as int8 **during training**, then convert the model to int8.

Principle of quantization

Basic principle:

$$r = S(q - Z)$$

Where parameter S∈ℝ is scale, Z is quantized zero-point.

Rewrite to matrix type:

$$r_\alpha^{(i,j)} = S_\alpha(q_\alpha^{(i,j)} - Z_\alpha).$$

The target is to convert the weight matrix ***q₁*** and input matrix ***q₂*** into quantized values, as here:

$$S_3(q_3^{(i,k)} - Z_3) = \sum_{j=1}^{N} S_1(q_1^{(i,j)} - Z_1) S_2(q_2^{(j,k)} - Z_2),$$

Rewrite to

$$q_3^{(i,k)} = Z_3 + M \sum_{j=1}^{N} (q_1^{(i,j)} - Z_1)(q_2^{(j,k)} - Z_2), \; M := \frac{S_1 S_2}{S_3}.$$

Finally rewrite to

$$q_3^{(i,k)} = Z_3 + M \left(N Z_1 Z_2 - Z_1 a_2^{(k)} - Z_2 \bar{a}_1^{(i)} + \sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)} \right)$$

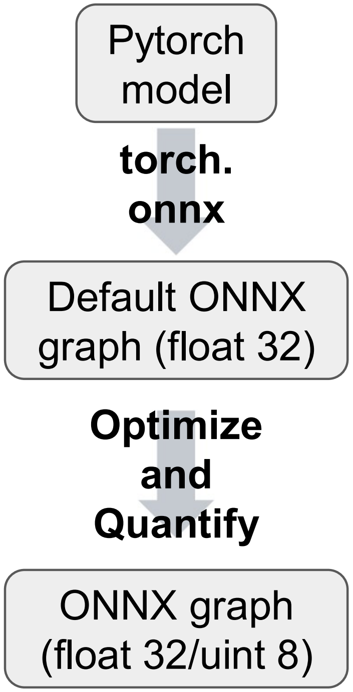
Where

$$a_2^{(k)} := \sum_{j=1}^{N} q_2^{(j,k)}, \; \bar{a}_1^{(i)} := \sum_{j=1}^{N} q_1^{(i,j)}.$$

For more information, please refer the paper *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference, CVPR 2018*

Introduction *ONNX*

ONNX: a **protocol** to **describe** the computational graph of model.



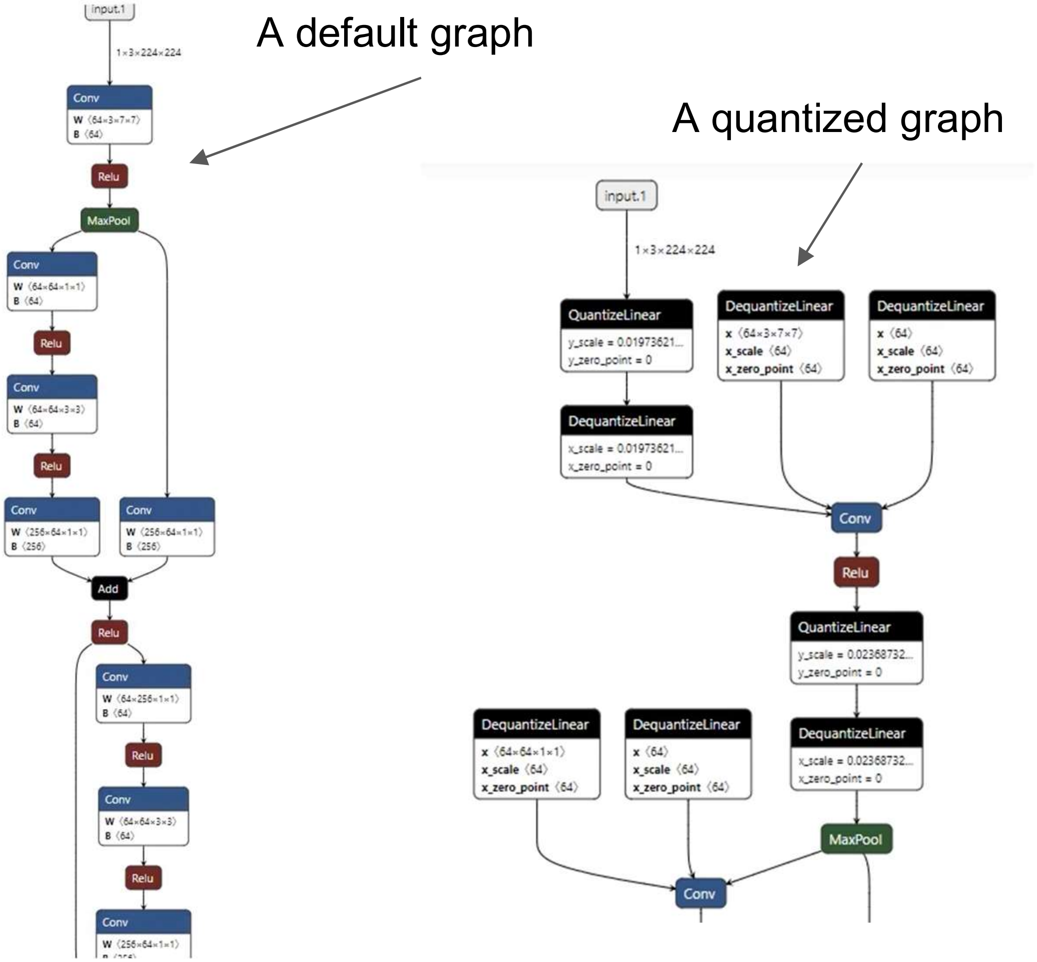
Pytorch model

torch. onnx

Default ONNX graph (float 32)

Optimize and Quantify

ONNX graph (float 32/uint 8)



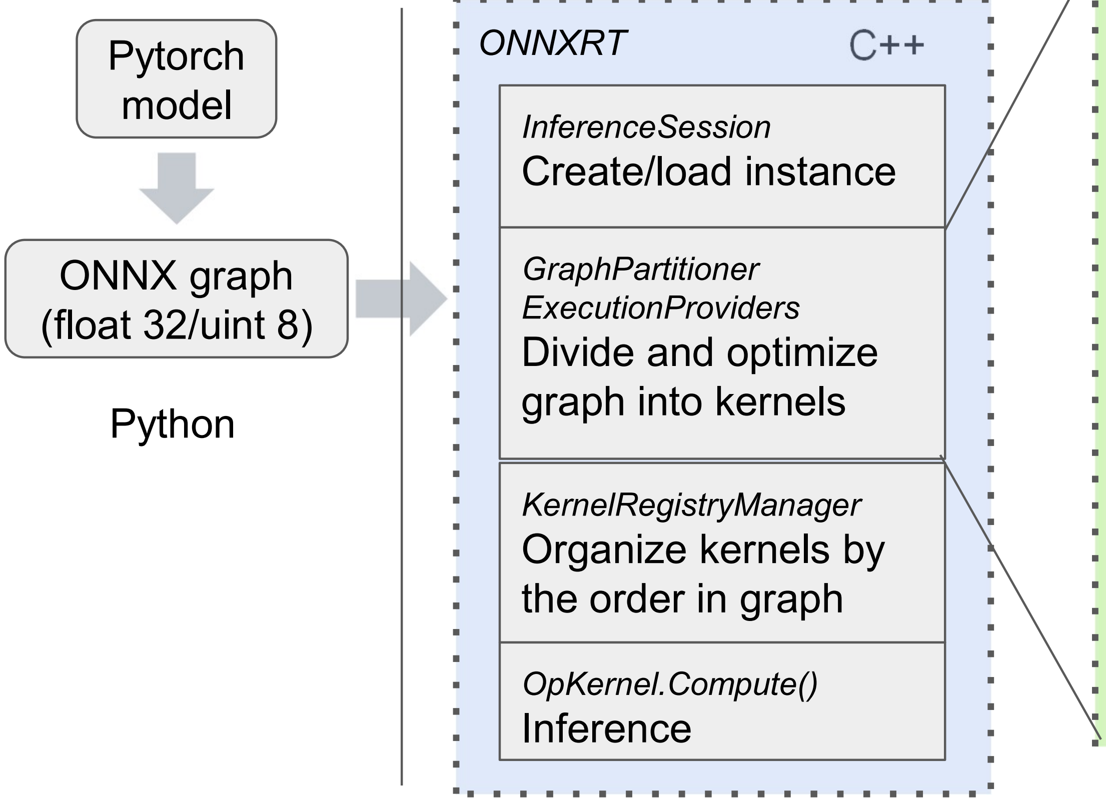
A default graph

A quantized graph

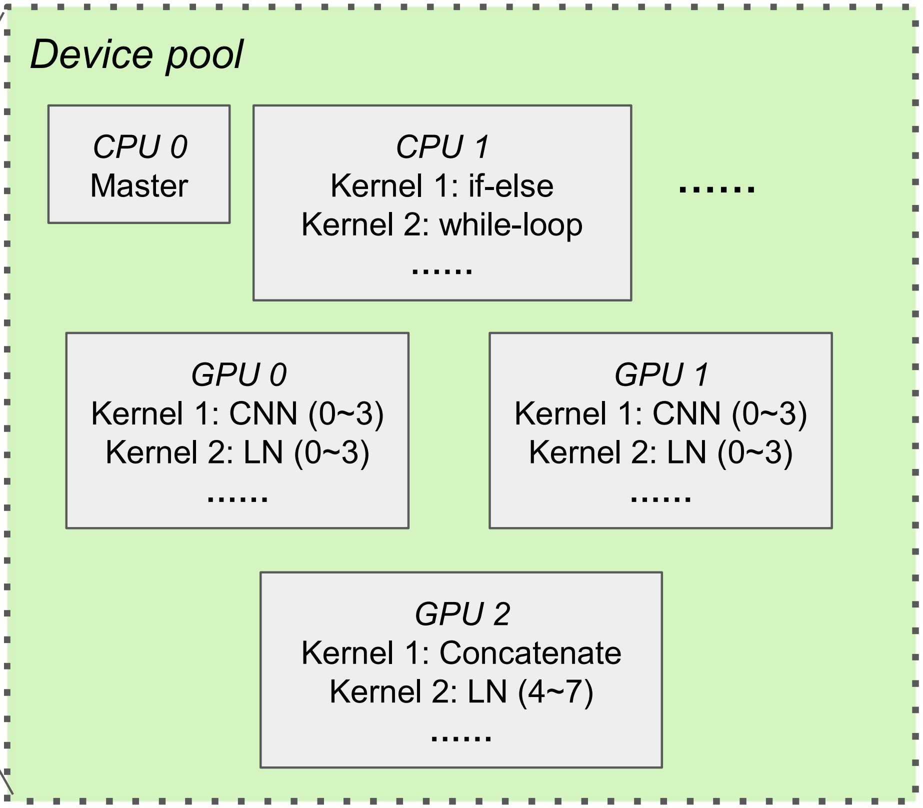
Example: computational graph of Resnet50.

Introduction *ONNX Runtime*

ONNX runtime: a **engine** to **execute** the computational graph.

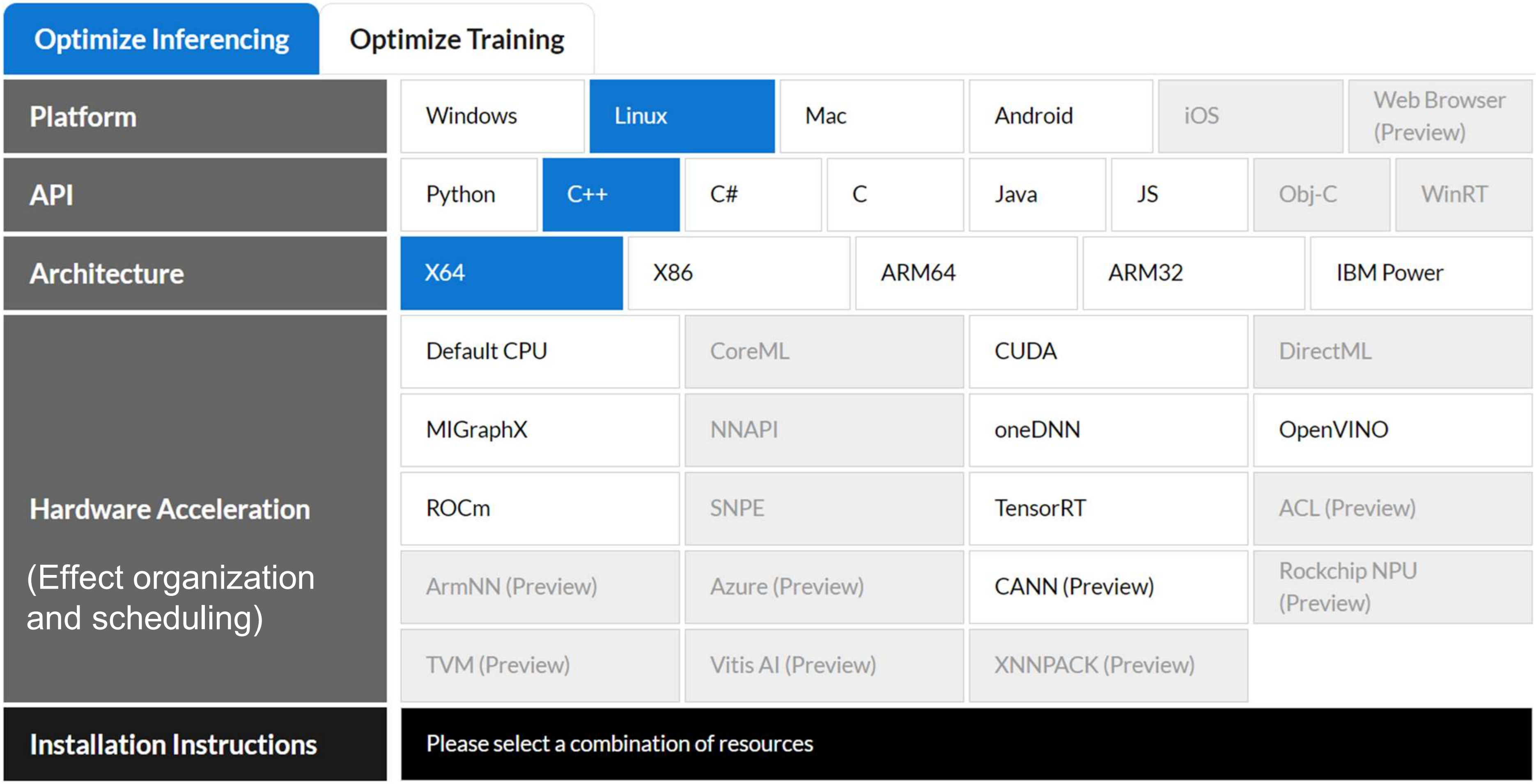


Example of kernel organization and scheduling. Organization algorithm depends on engine, algorithm, and device pool.



Ref: https://onnxruntime.ai

Introduction *ONNX Runtime*

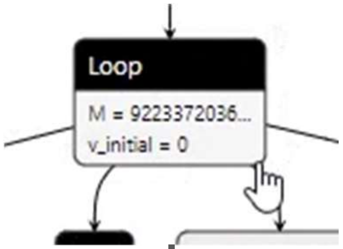


Optimize Inferencing | Optimize Training

Platform	Windows	Linux	Mac	Android	iOS	Web Browser (Preview)		
API	Python	C++	C#	C	Java	JS	Obj-C	WinRT
Architecture	X64	X86	ARM64	ARM32	IBM Power			
Hardware Acceleration (Effect organization and scheduling)	Default CPU	CoreML	CUDA	DirectML				
	MIGraphX	NNAPI	oneDNN	OpenVINO				
	ROCm	SNPE	TensorRT	ACL (Preview)				
	ArmNN (Preview)	Azure (Preview)	CANN (Preview)	Rockchip NPU (Preview)				
	TVM (Preview)	Vitis AI (Preview)	XNNPACK (Preview)					
Installation Instructions	Please select a combination of resources							

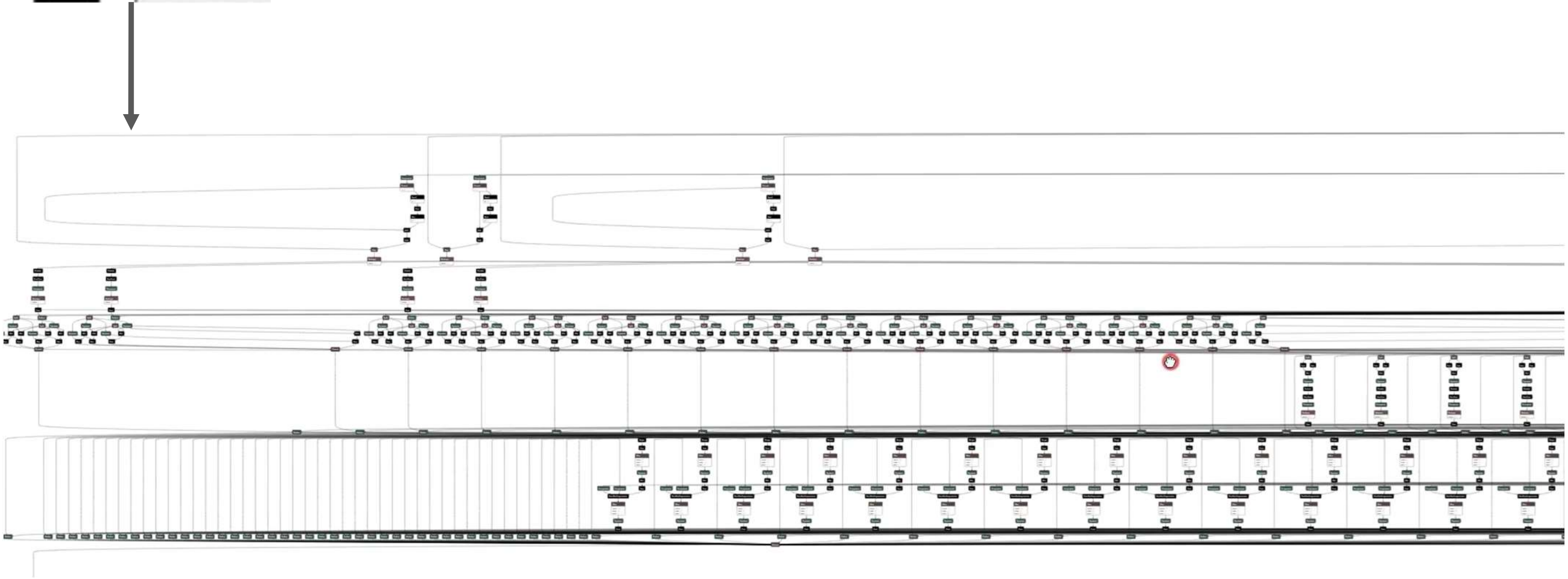
Ref: https://onnxruntime.ai

Potential issue *ONNX*



Example calculation graph of a while-loop operation:

- An operator like NMS(Non-maximum Suppression), or recursive operation.
- **Can’t be deployed**

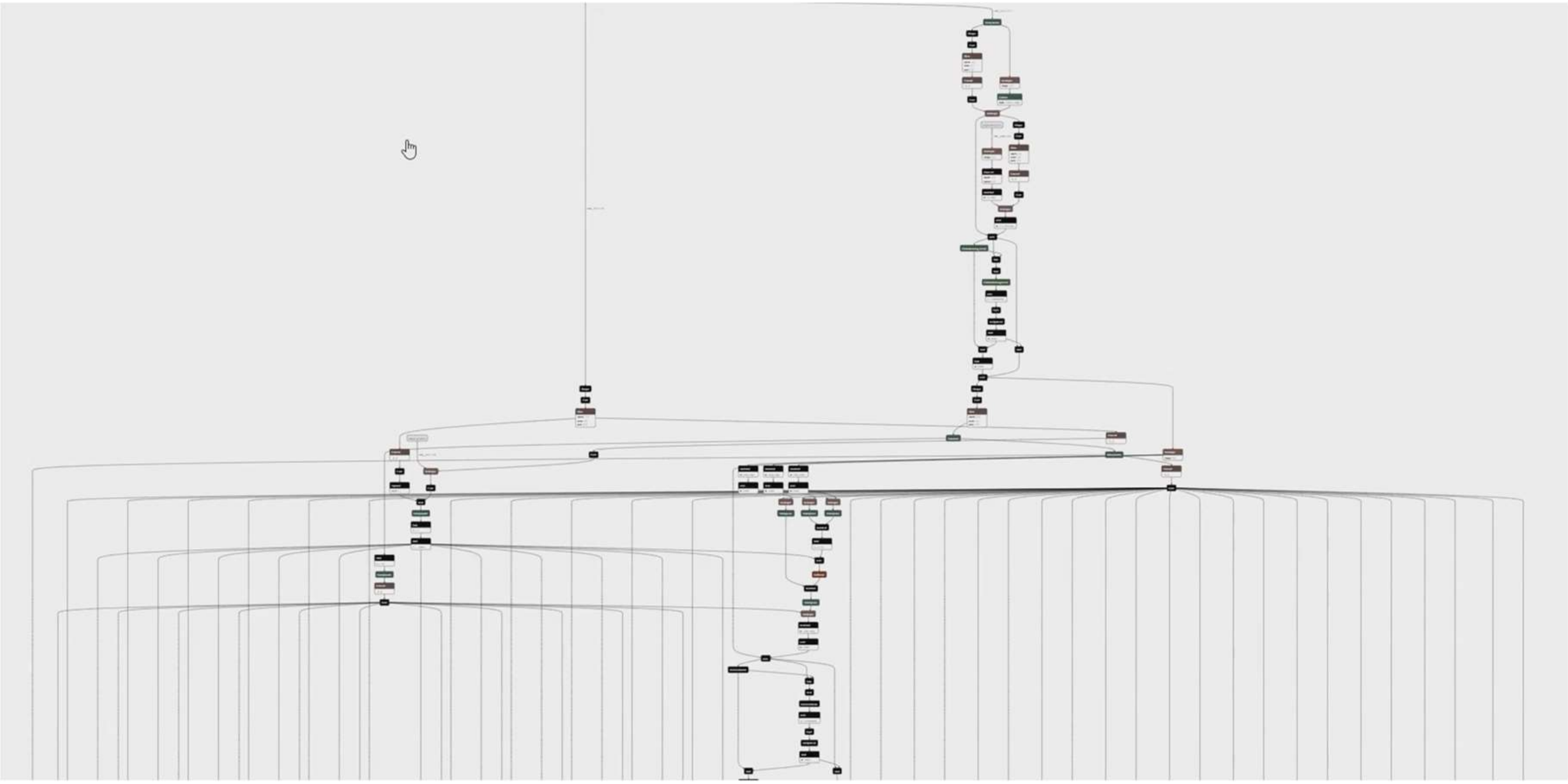


Ref: https://pytorch.org/docs/stable/onnx.html#limitations

Potential issue *ONNX*

Example of a complex architecture: BERT

- **Can’t be deployed**



Ref: https://pytorch.org/docs/stable/onnx.html#limitations