Efficiency in AI: Практичні поради з оптимізації ШІ

Полухін Андрій

DOU AI Meetup, Kyiv
21/12/2023

Про мене

- **ML Engineer** at Data Science UA and Samba.TV.
- **Writing about AI**
  - Telegram (t.me/eiaioi), Medium, Website.
- **Ph. D. Student** at NTUU KPI, “Unmanned Aerial Vehicle Navigation using Deep Learning on Edge Devices”, NATO Science for Peace and Security project G6032.
- **Mentor** at Projector Mentorship Platform.
- **Teacher of Machine Learning** at Hillel IT School.

Agenda

- **Навіщо оптимізувати ШІ**
  - Оптимізація витрат та Data Privacy
- **Як можна оптимізувати ШІ**
  - PyTorch, ONNX, TensorRT, TensorFlow, OpenVINO
  - Model Architecture, Distillation, Pruning, Mixed Precision, Quantization
- **Приклади** (2 шт.)
  - YOLOv8, x3 speed up in a few lines of code
  - Consumer PC GPT

ChatGPT

Переваги

- Персональний помічник
- Генерує ідеї
- Швидко відповідає
- Мультимовний

Недоліки

- Збирає ваші дані
- Працює лише online
- Закритий код
- Обмежене Usage Policy

Альтернативи ChatGPT

- Claude 2 (Anthropic)
- Bard, Gemini (Google)
- Coral (Cohere)
- Grok (X)
- Ernie (Baidu)

ANTHROP\C

Grok

cohere

Ernie Bot

Bard

Недоліки

- **Закритий** код
- **Обмежене** Usage Policy
- **Збирає** ваші **дані**
- Працює лише **online**

Open Source LLM

Переваги

- Працює **offline**
- **Відкритий код**
- **Немає** обмежень на **Usage Policy**
- **Можна дотренувати** під свої дані
- **Немає витрат** на кількість використаних **токенів**
- **Не збирає статистику** та ваші дані

Недоліки

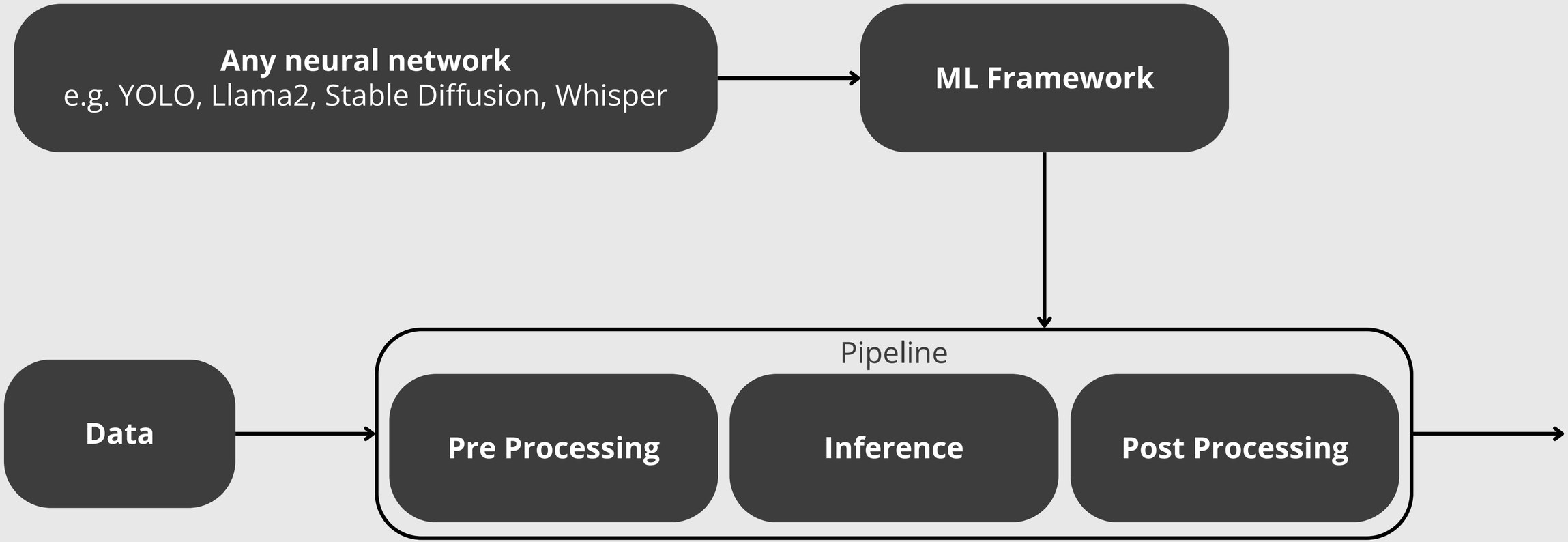
- Потрібні **великі обчислювальні потужності** (>>RAM та GPU)
- Потребують **дотренування** під задачу
- Розгортання, підтримка та правильна робота (RAG, ToT) потребують **окремих спеціалістів** та витрат

Open Source LLM

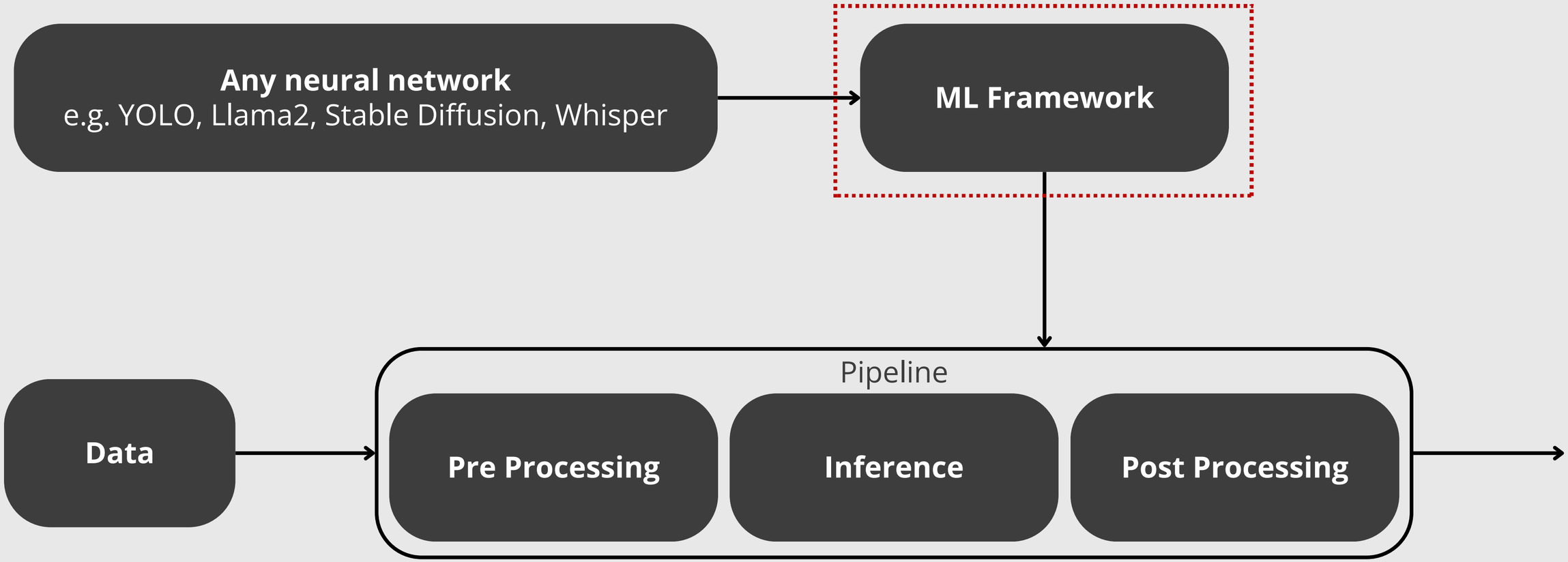
- Llama 2 (Meta AI)
- Mistral, Mixtral (Mistral AI)
- Yi-Chat (01.AI)
- Starling (Berkeley EECS)
- NeuralChat (Intel)
- Vicuna (LMSYS Org)
- Zephyr Alpha, Zephyr Beta (The H4 team)
- Falcon (Technology Innovation Institute)
- OpenChat (Tsinghua University)

and more: Chatbot Arena, ollama, Open LLM Leaderboard, vLLM

Neural Network Inference



Neural Network Inference



Виберіть фреймворк

- **general**:
  - PyTorch (Meta AI)
  - TensorFlow (Google)

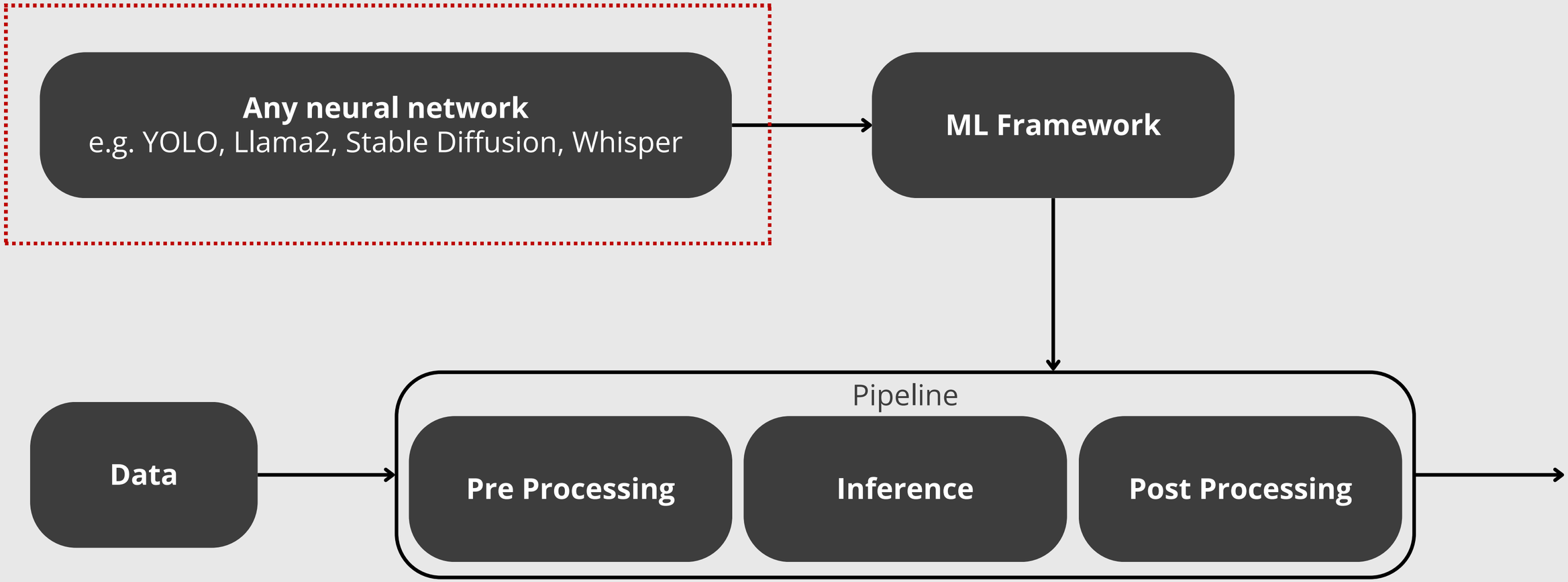
- **mobile**:
  - TF Lite (Google)
  - TF Edge TPU (Google)
  - ncnn (Tencent)
  - Core ML (Apple)

- **optimized**:
  - ONNX (Linux Foundation)
  - TensorRT (NVIDIA)
  - OpenVINO (Intel)
  - JAX (Google)
  - PaddlePaddle (PaddlePaddle)
  - DeepSparse (Neural Magic)
  - Mojo (Modular)
  - vLLM (vLLM)

ONNX

Framework	Purpose	Platform
PyTorch	General deep learning with dynamic computation graphs	CPU, NVIDIA GPU
TensorFlow	Scalable and flexible deep learning	CPU, NVIDIA/AMD GPU
TF Lite	Optimized for mobile and embedded devices	ARM, x86 (Mobile CPUs, limited GPU support)
TF Edge TPU	Optimized for Google's Edge TPU, for edge devices	Google Edge TPU
ncnn	Optimized for high-performance inference on mobile devices	ARM, Qualcomm DSP (Vulkan support)
Core ML	Machine learning framework for on-device inference on Apple hardware	Apple CPU, GPU
ONNX	Interoperability between different deep learning frameworks	CPU, GPU (depends on the specific runtime)
TensorRT	High-performance deep learning inference on NVIDIA GPUs	NVIDIA GPU
OpenVINO	Optimized inference for Intel hardware	Intel CPU, GPU, VPU, FPGA
JAX	High-performance numerical computing and machine learning	CPU, NVIDIA/AMD GPU (TPUs via Google Cloud)
PaddlePaddle	Comprehensive deep learning with industrial applications	CPU, NVIDIA GPU
DeepSparse	Sparsity-aware deep learning inference runtime for CPUs	CPU
Mojo (Modular)	Machine learning models in production environments, especially on the JVM	JVM compatible systems
vLLM	High-throughput and memory-efficient inference for LLM	NVIDIA GPUs and AMD GPUs

Neural Network Inference



Model Architecture

- ConvNets Optimizations
  - ❌ Convolution ✅ Depthwise Separable Convolution (MobileNet)
  - ❌ Convolution ✅ Group Convolution (e.g. ShuffleNet)
  - ✅ Use Squeeze & Excitation (e.g. SqueezeNext)
  - ✅ Use Fused Inverted Residual Block (EfficientNet v2)
- LLM & Transformer Optimizations
  - ❌ Attention O(N^2) ✅ Linear Attention O(N)
  - ✅ PagedAttention
  - ✅ Activation-aware Weight Quantization
- General Optimizations
  - ✅ Apply Neural Architecture Search (e.g. FBNet)

Read more: https://polukhin.tech/2022/10/17/lightweight-neural-network-architectures

Model Architecture

- MobileNet v1 (2017) – 10x faster and smaller than VGG16 (2014)
  - ❌ Convolution ✅ Depth-wise Separable Convolution
  - ❌ ReLU ✅ ReLU6
  - Width and resolution hyper-parameters

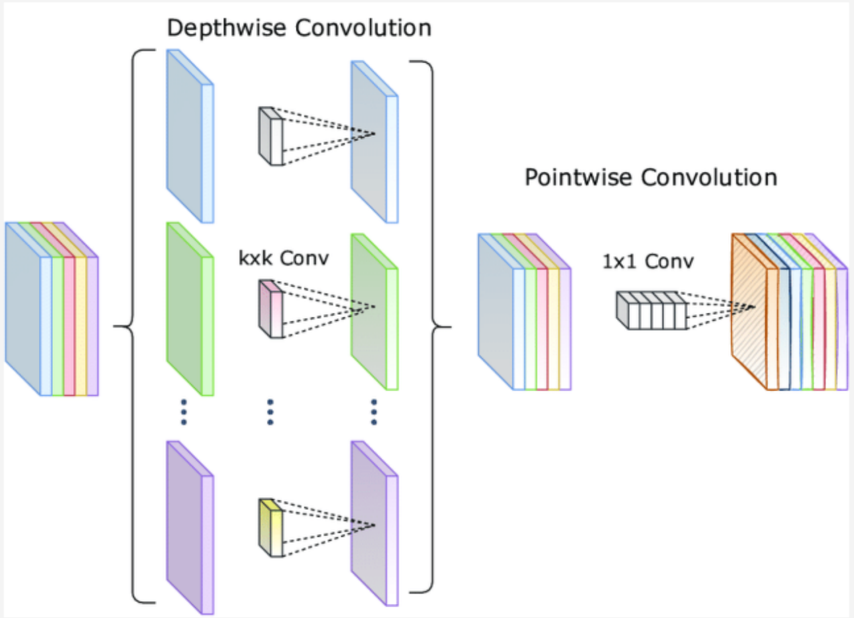


Figure: Depth-wise Separable Convolution (x9 faster than Conv3x3)

Model Architecture

- ShuffleNet v1 (2017) – 13x faster than AlexNet (2012), +7% accurate than MobileNet v1 (2017)
  - ❌ Convolution ✅ Group Convolution with feature shuffling

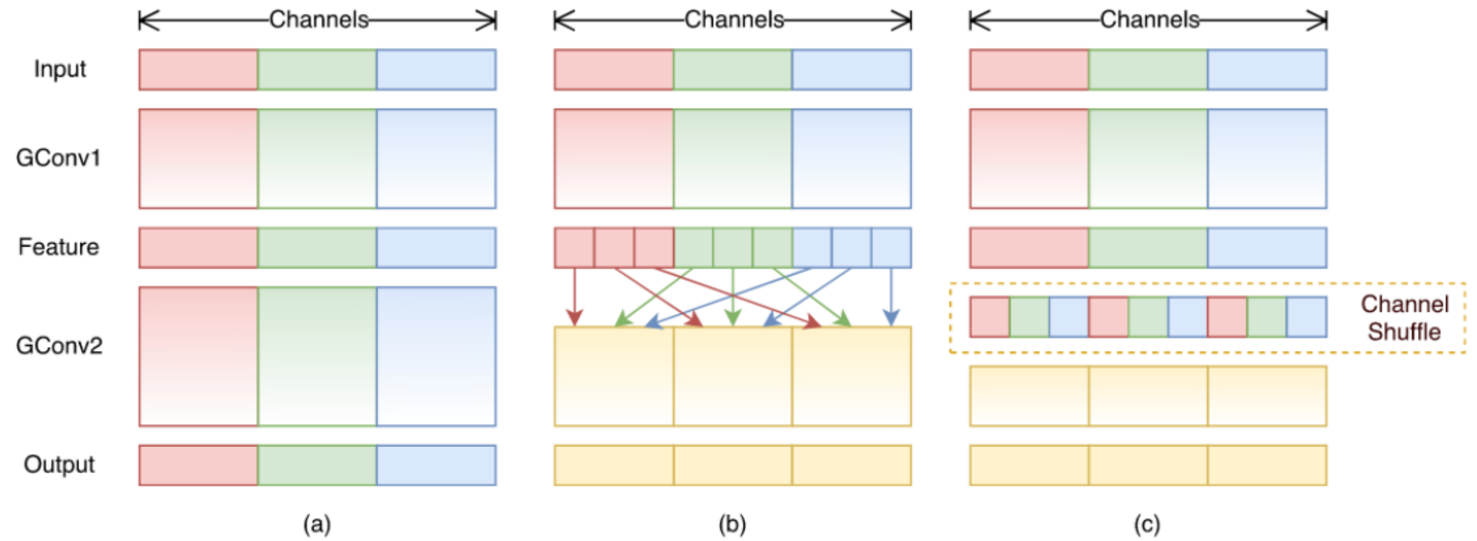


Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle.

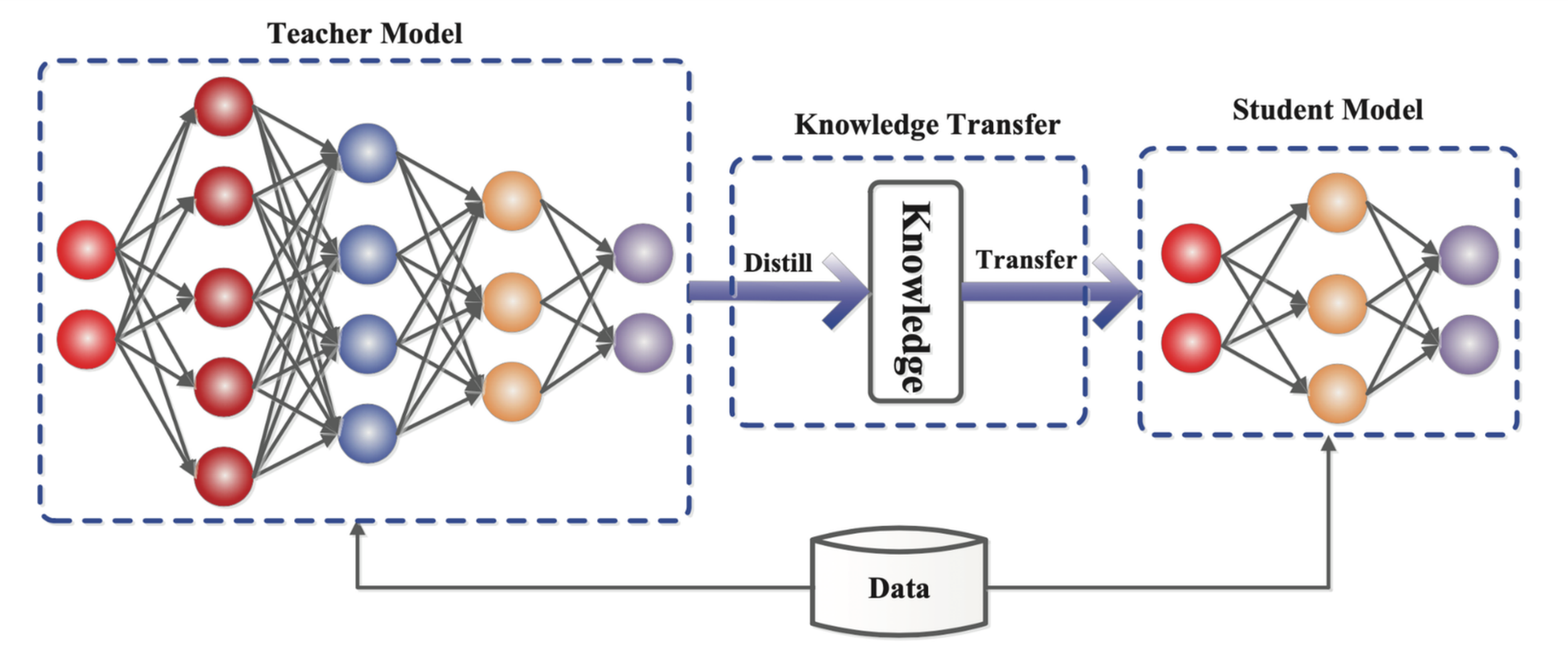
Figure: Channel shuffle

Model Architecture

- ConvNets Optimizations
  - ❌ Convolution ✅ Depthwise Separable Convolution (MobileNet)
  - ❌ Convolution ✅ Group Convolution (e.g. ShuffleNet)
  - ✅ Use Squeeze & Excitation (e.g. SqueezeNext)
  - ✅ Use Fused Inverted Residual Block (EfficientNet v2)
- LLM & Transformer Optimizations
  - ❌ Attention O(N^2) ✅ Linear Attention O(N)
  - ✅ PagedAttention
  - ✅ Activation-aware Weight Quantization
- General Optimizations
  - ✅ Apply Neural Architecture Search (e.g. FBNet)

Read more: https://polukhin.tech/2022/10/17/lightweight-neural-network-architectures

Model Distillation

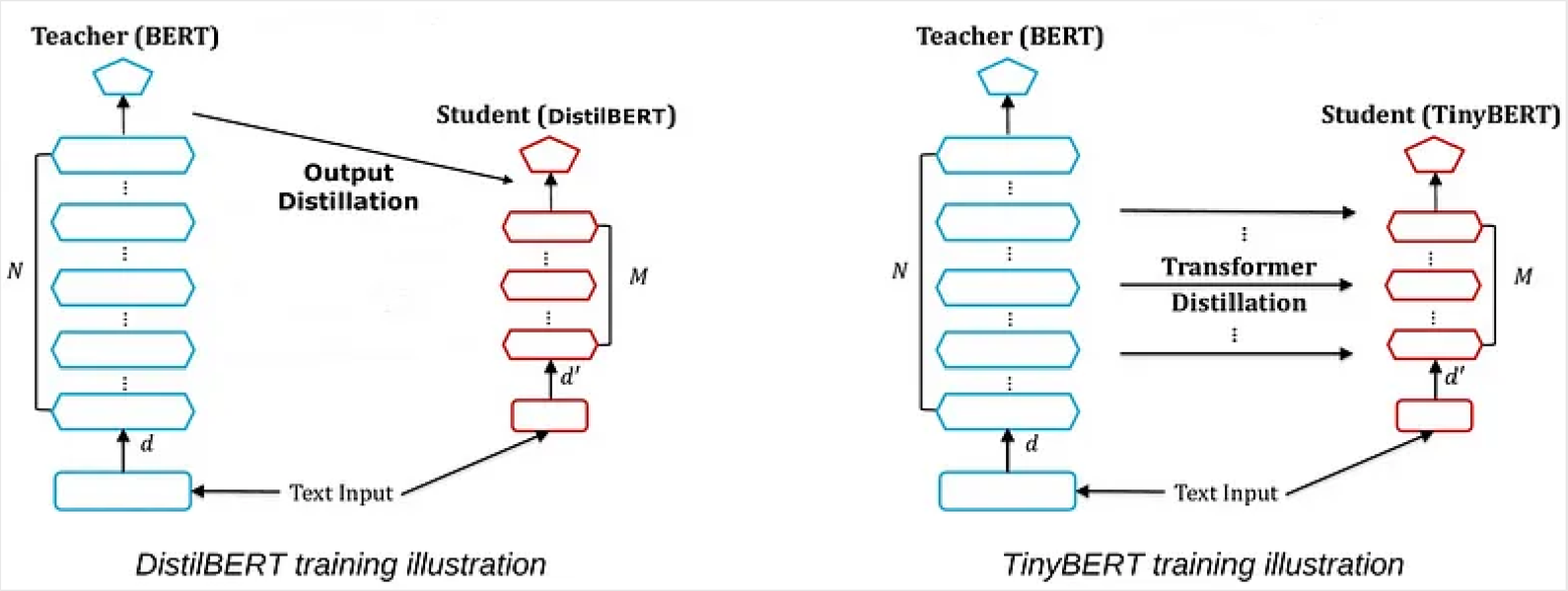


The teacher-student framework for knowledge distillation

Model Distillation

- dkozlov/awesome-knowledge-distillation (3.2k⭐)
- karanchahal/distiller (211⭐)
- yoshitomo-matsubara/torchdistill (1.1k⭐)

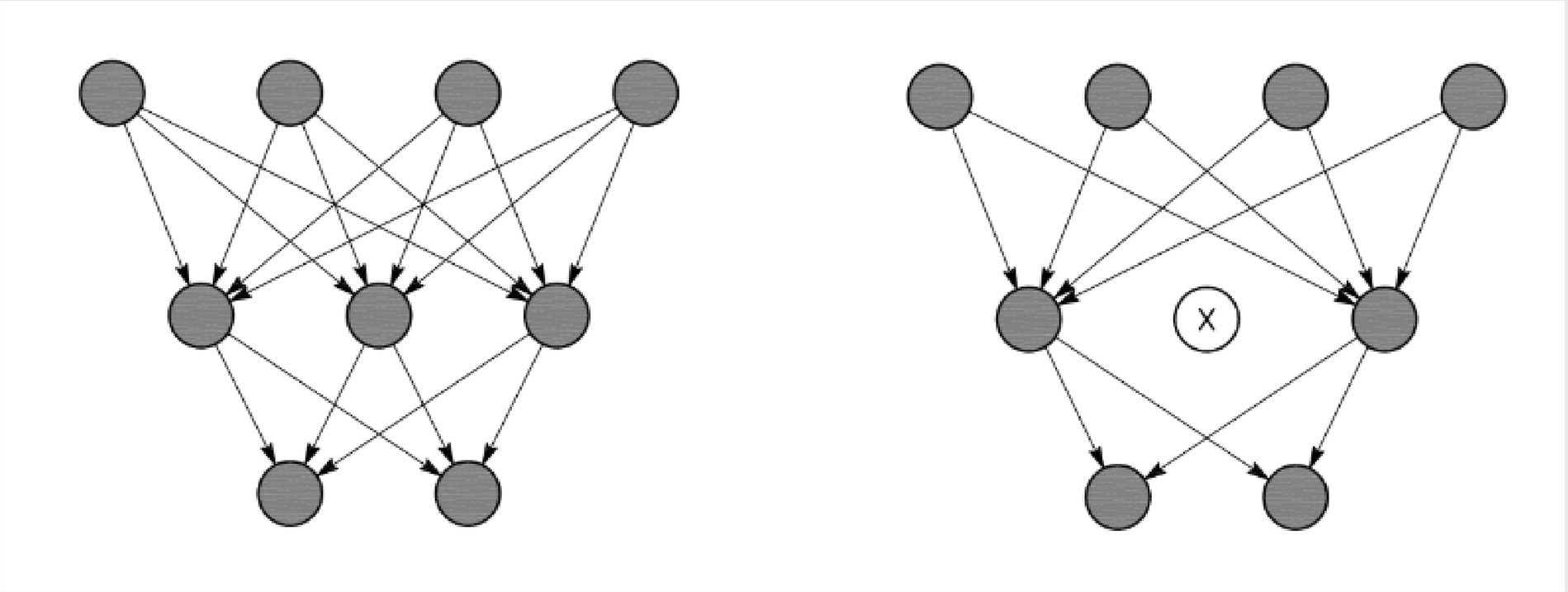
```
pip3 install torchdistill
```



DistilBERT training illustration

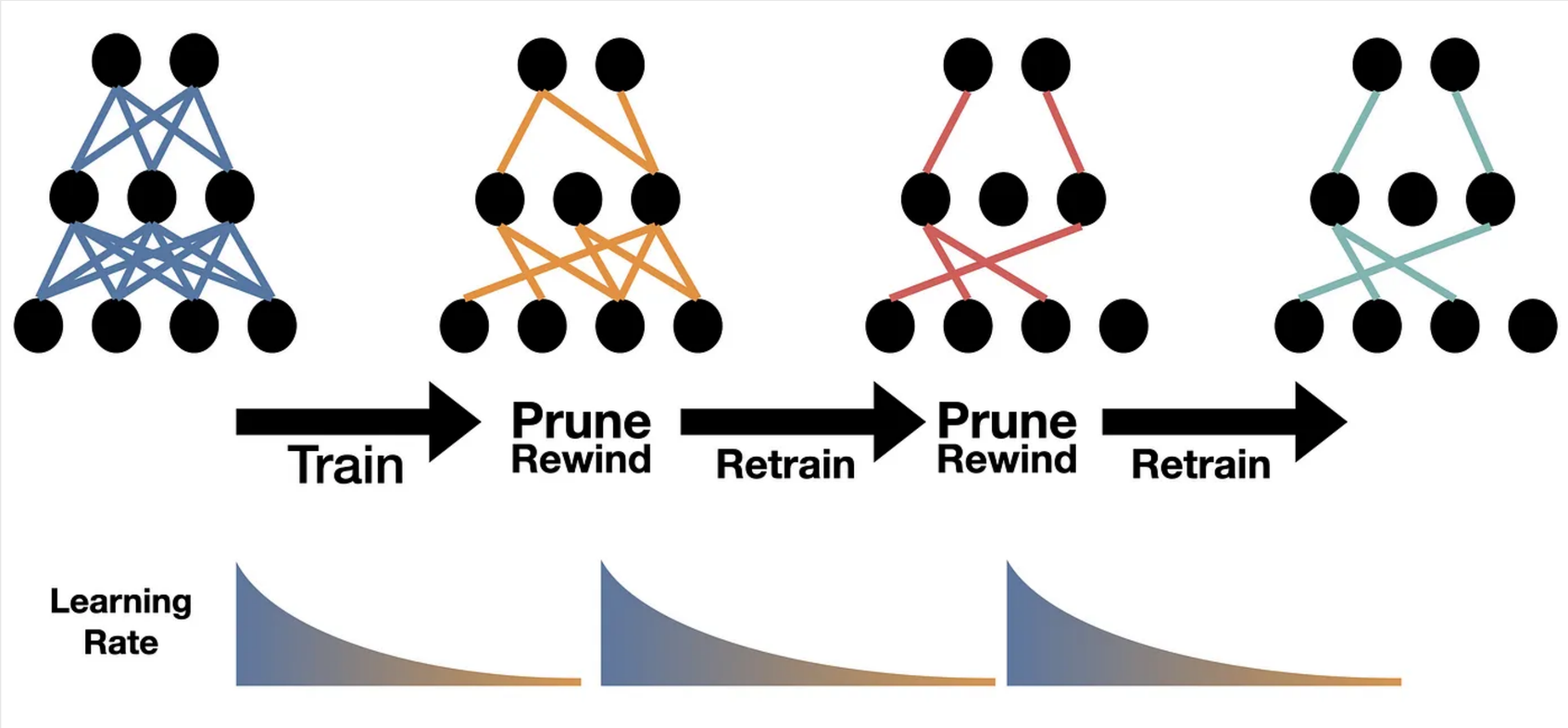
TinyBERT training illustration

Model Pruning



Pruning: Before and After

Model Pruning



Shrinking DNN models via pruning

Model Pruning

- neuralmagic/sparseml (1.9k⭐)
- he-y/Awesome-Pruning (2.1k⭐)
- intel/neural-compressor (1.7k⭐)

```
pip3 install neural-compressor
```

```
from neural_compressor.training import prepare_pruning,
WeightPruningConfigconfig = WeightPruningConfig(configs)
prepare_pruning(model, config, optimizer)
for epoch in range(num_train_epochs):
    model.train()
    for step, batch in enumerate(train_dataloader):
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        lr_scheduler.step()
        model.zero_grad()
```

Half Precision fp16

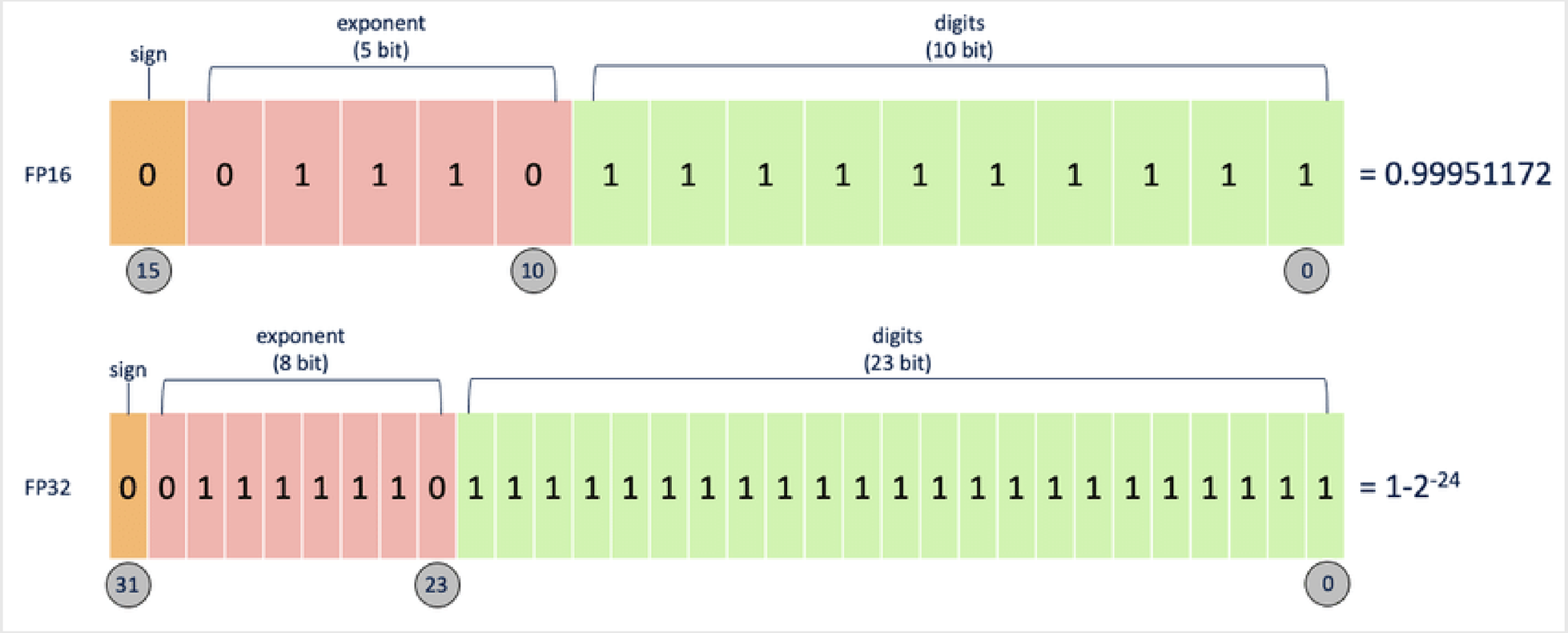
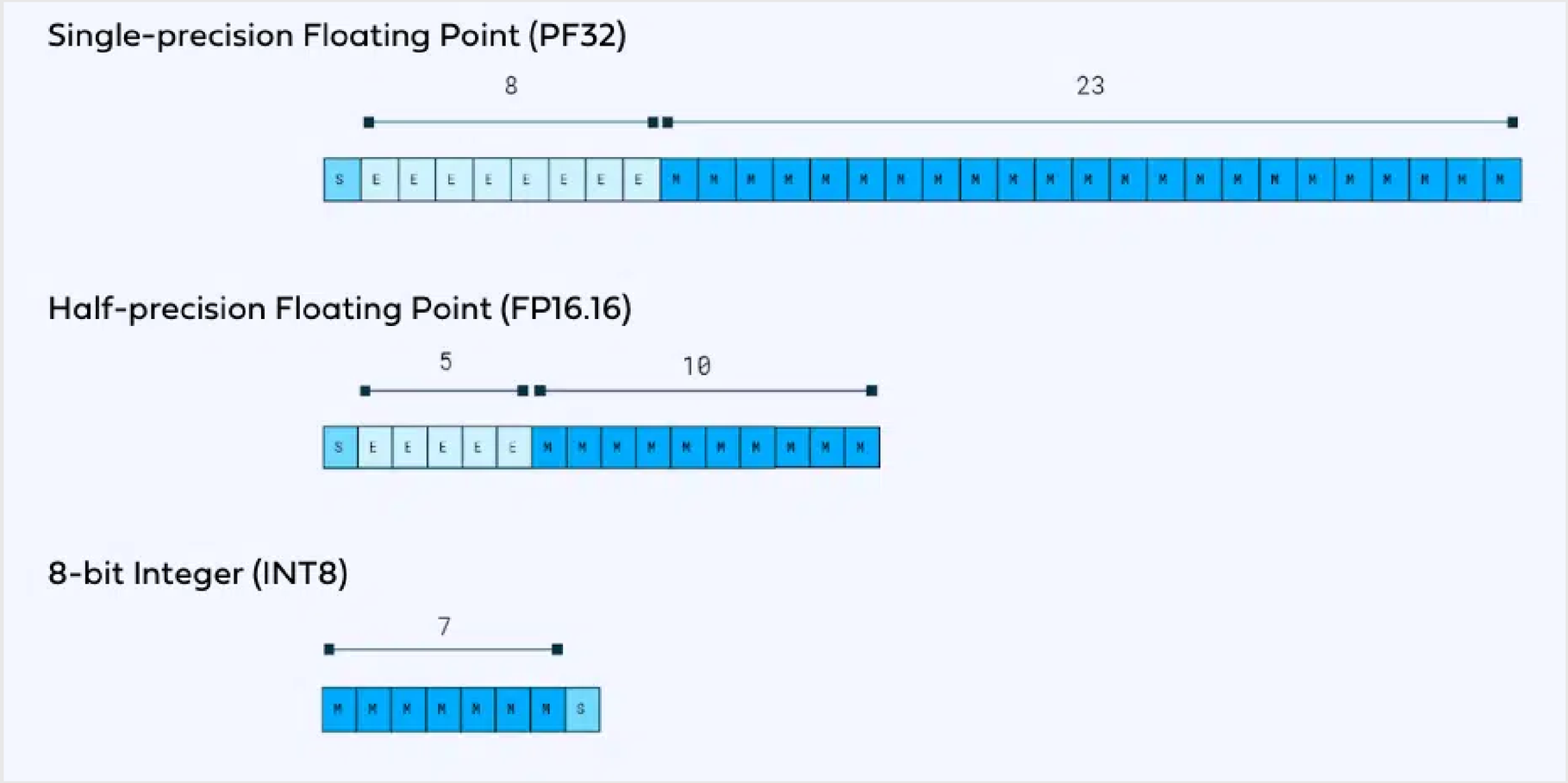


Figure represents comparison of FP16 (half precision floating points) and FP32 (single precision floating points).

Half Precision fp16

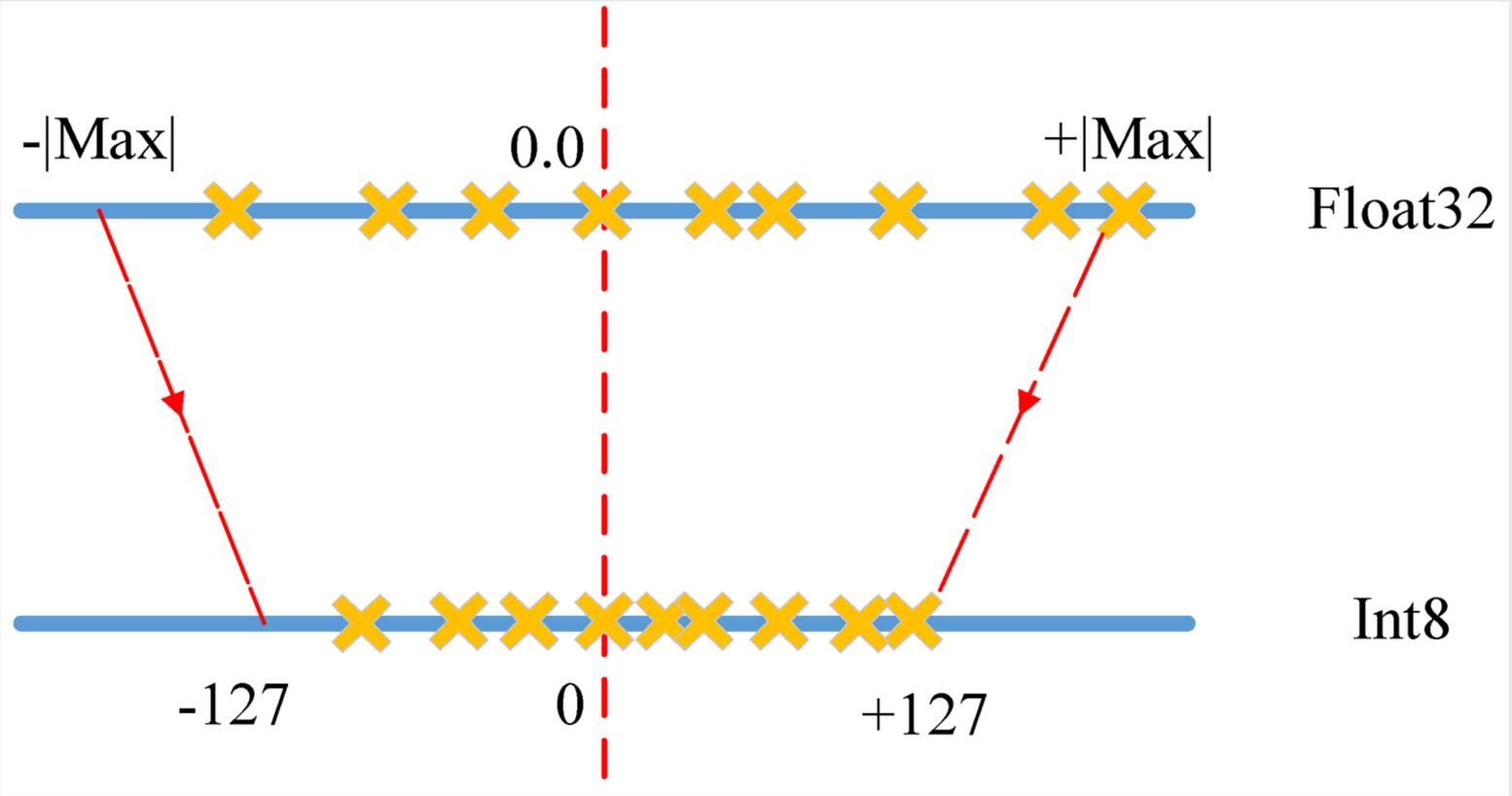
```
# PyTorch
import torch
model = model.half()  # Convert model to half precision
input_data = input_data.half()  # Convert input data to half precision

# TensorFlow
import tensorflow as tf
from tensorflow.keras.mixed_precision import experimental as mixed_precision
policy = mixed_precision.Policy('mixed_float16')
mixed_precision.set_policy(policy)

# TensorRT
import tensorrt as trt
builder = trt.Builder(TRT_LOGGER)
builder.fp16_mode = True
```

Model Quantization



8-bit Integer (INT8)

Model Quantization



Quantization technique

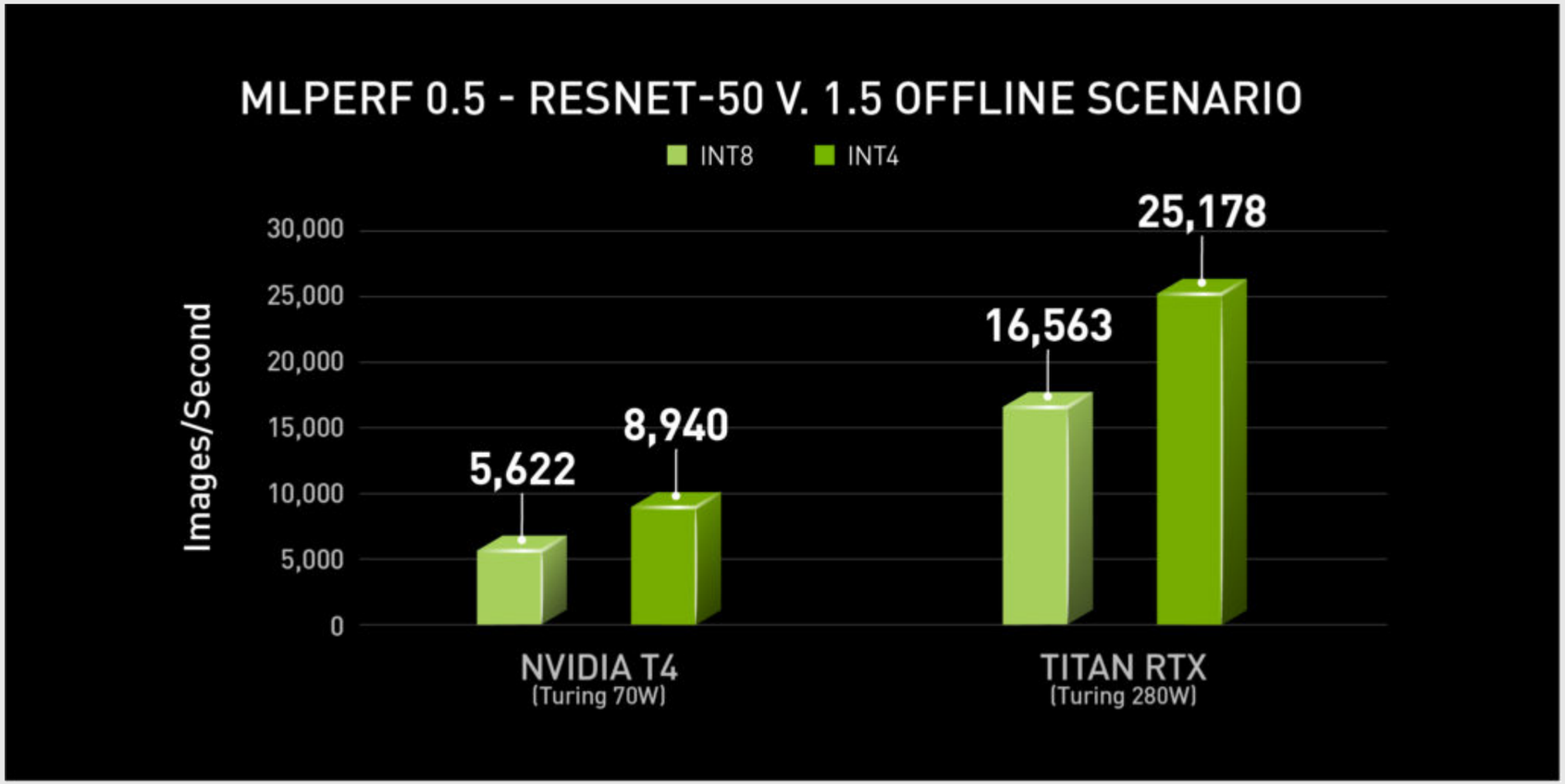
Model Quantization

- sony/model_optimization (233⭐)
- ModelTC/MQBench (211⭐)
- htqin/awesome-model-quantization (1.4k⭐)

Deploy Just in 4 Lines with MQBench

```
model = models.__dict__["resnet18"](pretrained=True)
model = prepare_by_platform(model, BackendType.Tensorrt)
enable_calibration(model) # turn on calibration
for i, batch in enumerate(data): # do forward procedures
    ...
enable_quantization(model) # turn on actually quantization
input_shape={'data': [10, 3, 224, 224]}
convert_deploy(model, backend, input_shape) # model export
```

int4



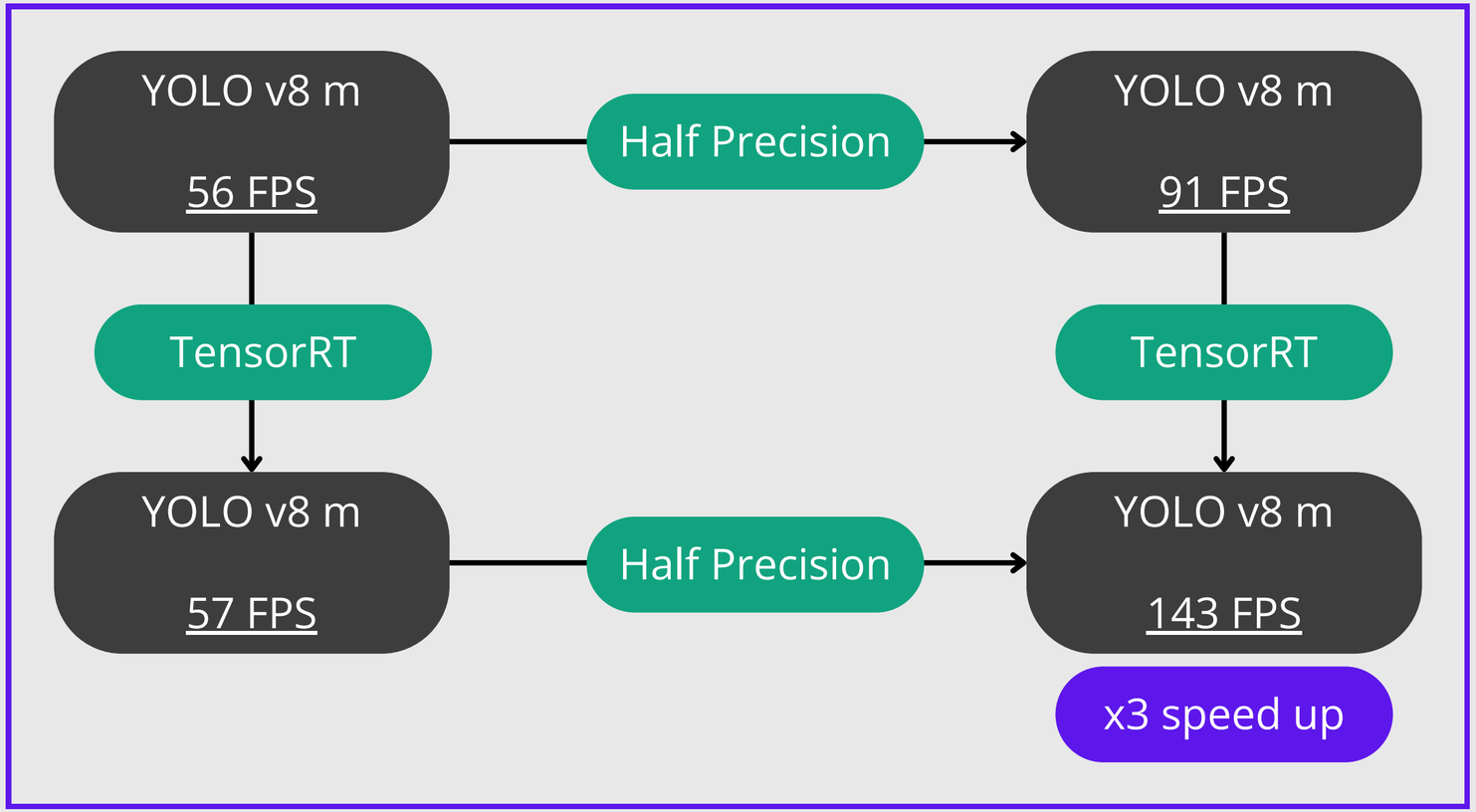
MLPerf v0.5 Inference results for data center server form factors and offline scenario retrieved from www.mlperf.org on Nov. 6, 2019 (Closed Inf-0.5-25 and Inf-0.5-27 for INT8, Open Inf-0.5-460 and Inf-0.5-462 for INT4). Per-processor performance is calculated by dividing the primary metric of total performance by number of accelerators reported. MLPerf name and logo are trademarks.

Приклади

YOLOv8, x3 speed up

Specifications:

- YOLO v8 m
- g5.2xlarge
- batch=64
- imgsz=1280x1280



YOLOv8, x3 speed up

```
# Code to optimize
import ultralytics
model = ultralytics.YOLO("yolov8m")
model.export(format="engine", imgsz=1280, batch=64, half=True)

# Or, use https://github.com/NVIDIA-AI-IOT/torch2trt
```

Consumer PC GPT

```
# Code in Python
from gpt4all import GPT4All
model = GPT4All("orca-mini-3b-gguf2-q4_0.gguf")
output = model.generate("The capital of France is ", max_tokens=3)
print(output)
```

The GPT4All Chat UI supports models from all newer versions of llama.cpp with GGUF models including the Mistral, LLaMA2, LLaMA, OpenLLaMa, Falcon, MPT, Replit, Starcoder, and Bert architectures.

	Model	Nvidia T4 (g4dn.xlarge)	Nvidia A10 (g5.xlarge)	AMD Radeon Pro v520 (g4ad.xlarge)	Intel i9-13900K CPU	Macbook Pro M2 16gb	H100 (Paperspace)
Price / hour		$0.53	$1.01	$0.38			
Nomic Vulkan							
	OpenLLaMa 3B q4_0	23.36 tok/s	57.18 tok/s	41.61 toks/s	N/A	N/A	71.25 tok/s
	LLaMA2 7B q4_0	12.54 tok/s	37.25 tok/s	23.71 toks/s	N/A	N/A	50.28 tok/s
	LLaMA2 13B q4_0	6.51 tok/s	21.87 tok/s	13.18 toks/s	N/A	N/A	33.28 tok/s
OpenCL							
	OpenLLaMa 3B q4_0	28.70 tok/s	42.71 tok/s	N/A	N/A	N/A	59.22 tok/s
	LLaMA2 7B q4_0	17.27 tok/s	29.74 tok/s	N/A	N/A	N/A	43.24 tok/s
	LLaMA2 13B q4_0	10.16 tok/s	19.14 tok/s	N/A	N/A	N/A	24.01 tok/s
NVIDIA CUBLAS							
	OpenLLaMa 3B q4_0	87.19 tok/s	134.77 tok/s	N/A	N/A	N/A	N/A
	LLaMA2 7B q4_0	49.40 tok/s	83.78 tok/s	N/A	N/A	N/A	N/A
	LLaMA2 13B q4_0	28.08 tok/s	49.35 tok/s	N/A	N/A	N/A	N/A
CPU (ggml)							
	OpenLLaMa 3B q4_0	6.46 tok/s	9.42 tok/s	8.59 tok/s	17.55 toks/s	N/A	N/A
	LLaMA2 7B q4_0	3.36 tok/s	4.93 tok/s	4.60 tok/s	9.37 toks/s	N/A	N/A
	LLaMA2 13B q4_0	1.77 tok/s	2.57 tok/s	2.30 tok/s	4.9 toks/s	N/A	N/A
Metal (GPT4All)							
	OpenLLaMa 3B q4_0	N/A	N/A	N/A	N/A	54.15 tok/s	N/A
	LLaMA2 7B q4_0	N/A	N/A	N/A	N/A	34.44 tok/s	N/A
	LLaMA2 13B q4_0	N/A	N/A	N/A	N/A	19.50 tok/s	N/A

Nomic Vulkan Benchmarks: Single batch item inference token throughput benchmarks.

Q/A