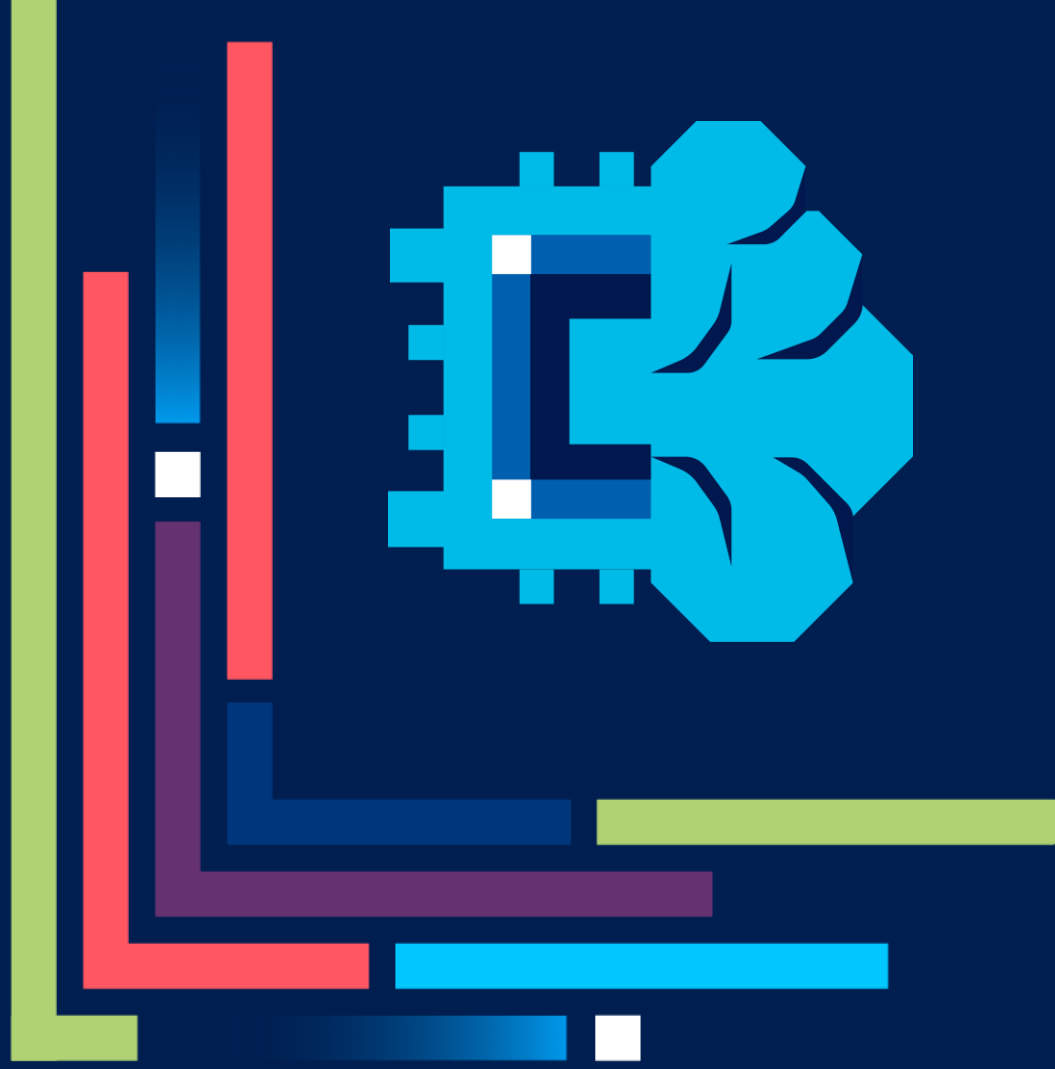
intel ai summit
英特爾 AI 科技論壇

Bringing AI Everywhere



Finetune LLM with Intel Architecture

Chungyeh Wang 王宗業

03/27/2024

Agenda

AI Everywhere

Small and Nimble – the Fast Path to Enterprise GenAI

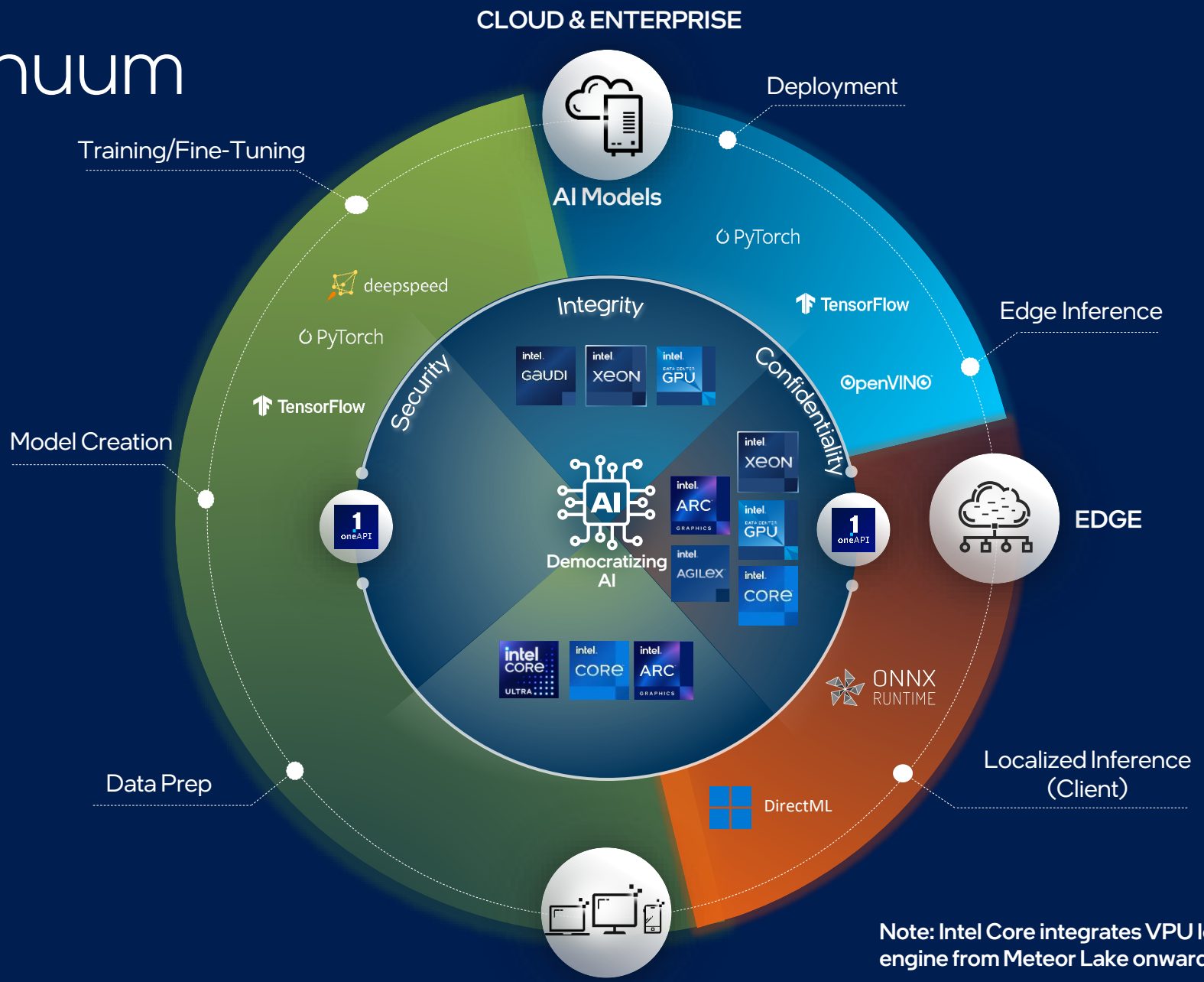
LLM Inference with OpenVINO and BigDL-LLM (ipex-llm)

LLM Fine-tuning with affordable Intel platform

AI Everywhere

intel Ai summit

AI Continuum



Note: Intel Core integrates VPU low power inference engine from Meteor Lake onwards.

intel Ai summit

Intel® AI Portfolio

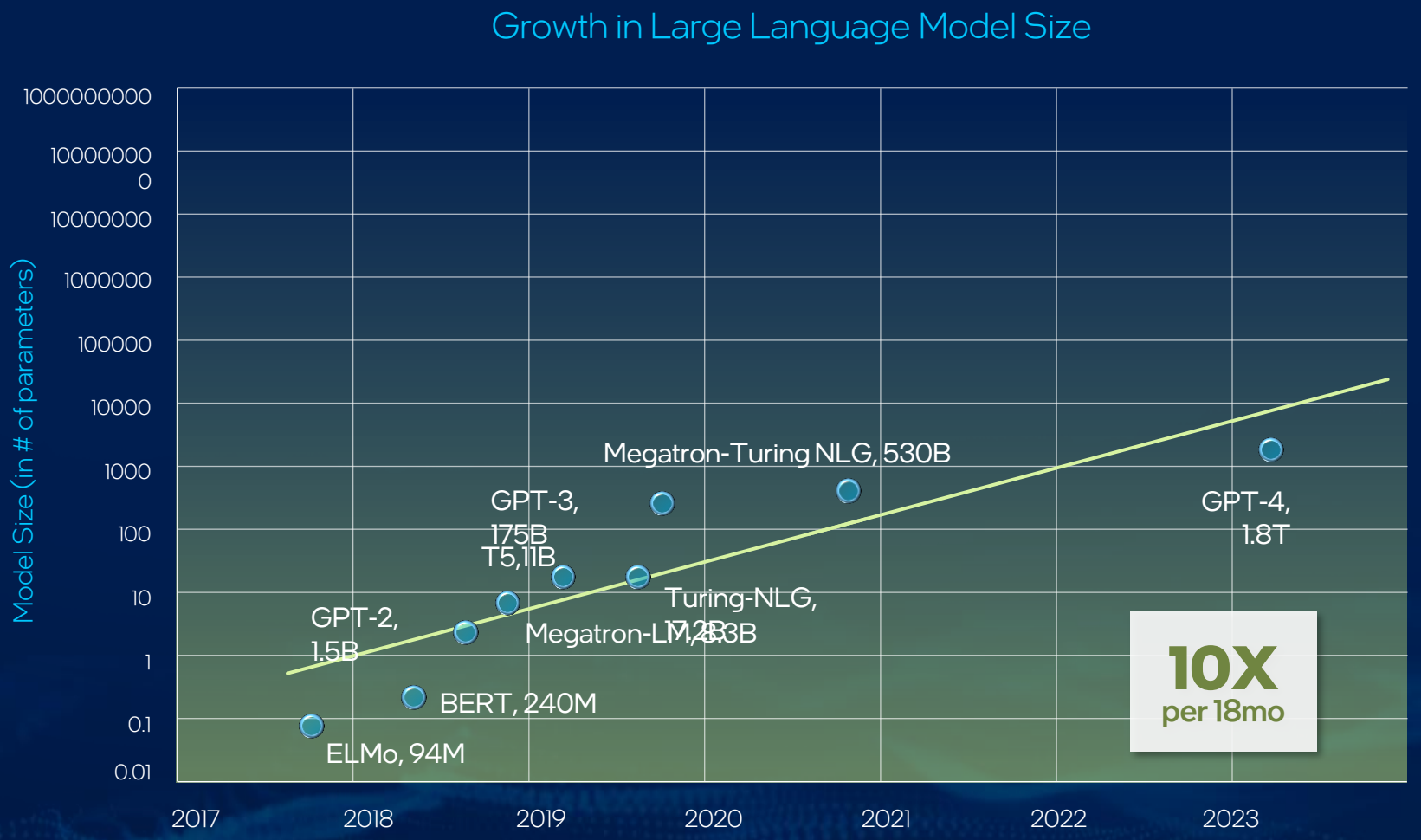
Open Software Environment	oneAPI, OpenVINO, PyTorch, TensorFlow, ONNX Runtime, deepspeed, XGBoost, pandas	
Deep Learning Acceleration	Intel Gaudi	Gaudi: Dedicated Deep Learning Training and Inference
General Acceleration	Intel Data Center GPU Flex Series	Cloud Gaming, VDI, Media Analytics, Real-Time Dense Video
	Intel Data Center GPU Max Series	Parallel Compute, HPC, AI for HPC
General Purpose	Intel Xeon, Intel Xeon Max Series, Intel Ethernet	Real-Time, Medium Throughput, Low Latency, and Sparse Inference
	Intel Xeon, Intel Xeon Max Series, Intel Xeon W, Intel Ethernet	Medium to Small Scale Training and Fine Tuning
	Intel Core Ultra, Intel Xeon D, Intel Data Center GPU Flex Series, Intel Arc Graphics, Intel Core, Intel Agilex, Intel Ethernet	Edge and Network AI Inference
	Intel Core, Intel Core Ultra, Intel Arc Graphics	Client AI Usages

intel AI summit

Small and Nimble – the Fast Path to Enterprise GenAI

intel Ai summit

Progress and Growth in AI Has Been Unprecedented



- **Will this growth trend continue?**
- **What disruptions and new capabilities are ahead?**

https://baincapitalventures.com/insight/large-language-models-will-redefine-b2b-software/

Generative AI 2023 – A Transition Year

Consumer Uses:
- Ideation
- Text generation
- Image generation

Added Business Uses:
- Customer service
- Programming
- Debugging

Giant Models:
Maximum effectiveness regardless of model size

Added nimble models:
For increased efficiency, accuracy, security and traceability

Expecting a small number of giant models and a "giant number" of small, targeted models embedded in applications.

intel Ai summit

The AI Solution Developer's Dilemma (for Business apps)

Giant

Proprietary model

All-in-one
general purpose

Cloud-based
(as-a-service)

Model-incorporated
Data (w/RAG)

intel Ai summit

The AI Solution Developer's Dilemma (for Business apps)

Giant	vs.	Small and Nimble (by 10-100X)
Proprietary model	vs.	Open Source based model
All-in-one general purpose	vs.	Targeted, customized
Cloud-based (as-a-service)	vs.	Locally run inference; edge, client & on-prem
Model-incorporated Data (w/RAG)	vs.	Retrieval-Centric Generation (RCG)

intel Ai summit

A Common Thread

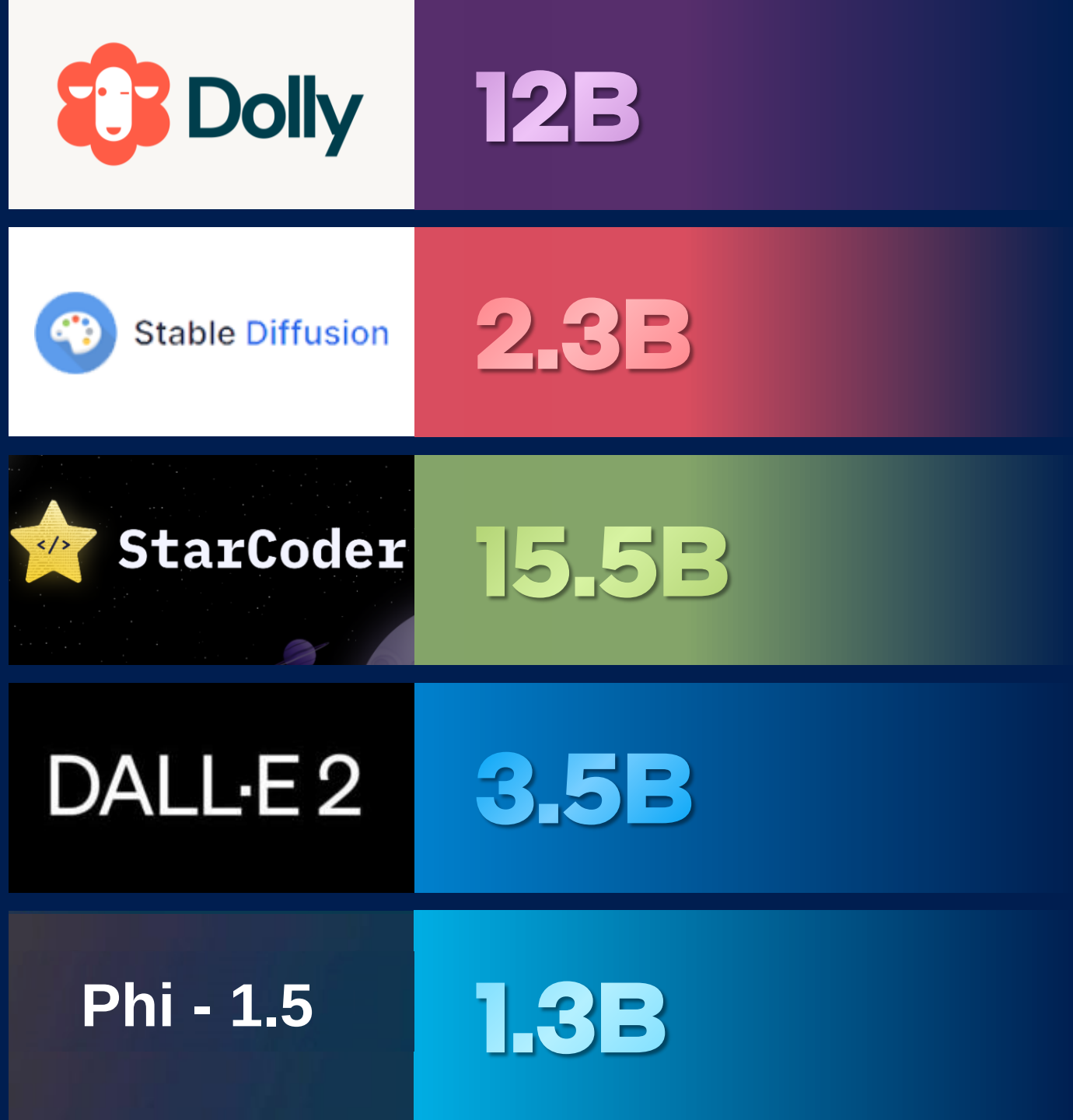
What is common to the following models?

intel Ai summit

A Common Thread

What is common to the following models?

They are all ~15B parameters or smaller



Nimble Model Characteristics

- Much smaller (by 10x – 100x) and faster to run than giants
- Easier to continuously adapt
- Accelerated by open source ecosystem



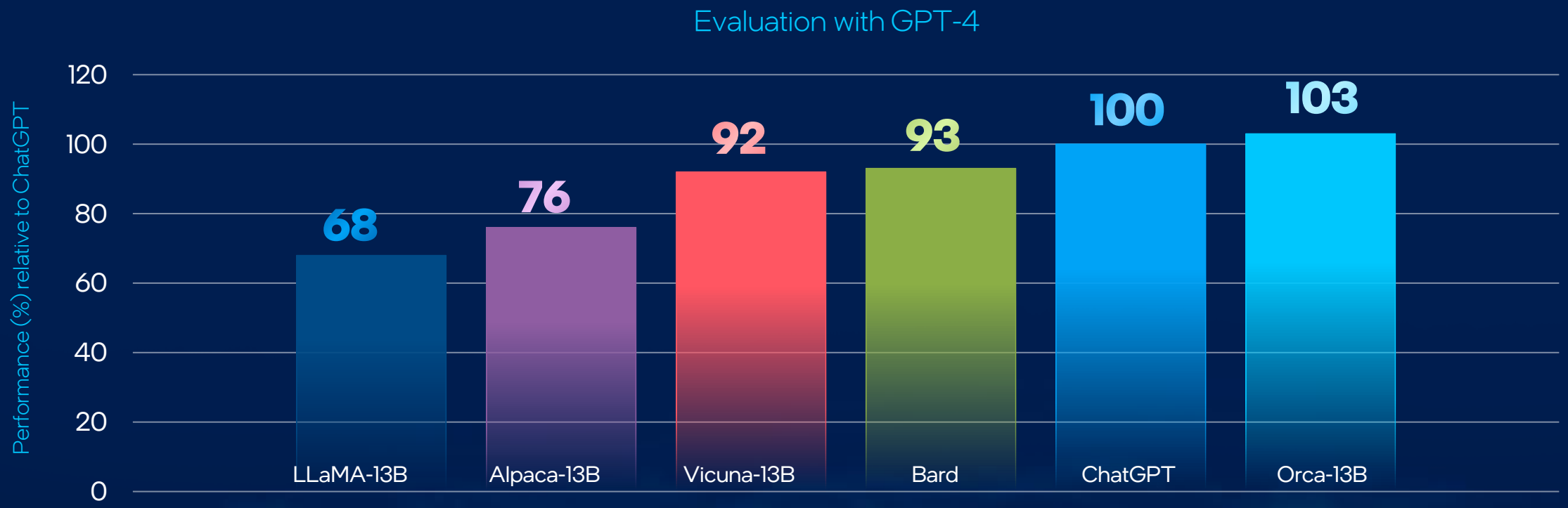



intel Ai summit

Rise of Nimble LLMs – Fueled by Open-Source Models

intel AI summit

Performance of Nimble Models vs. ChatGPT



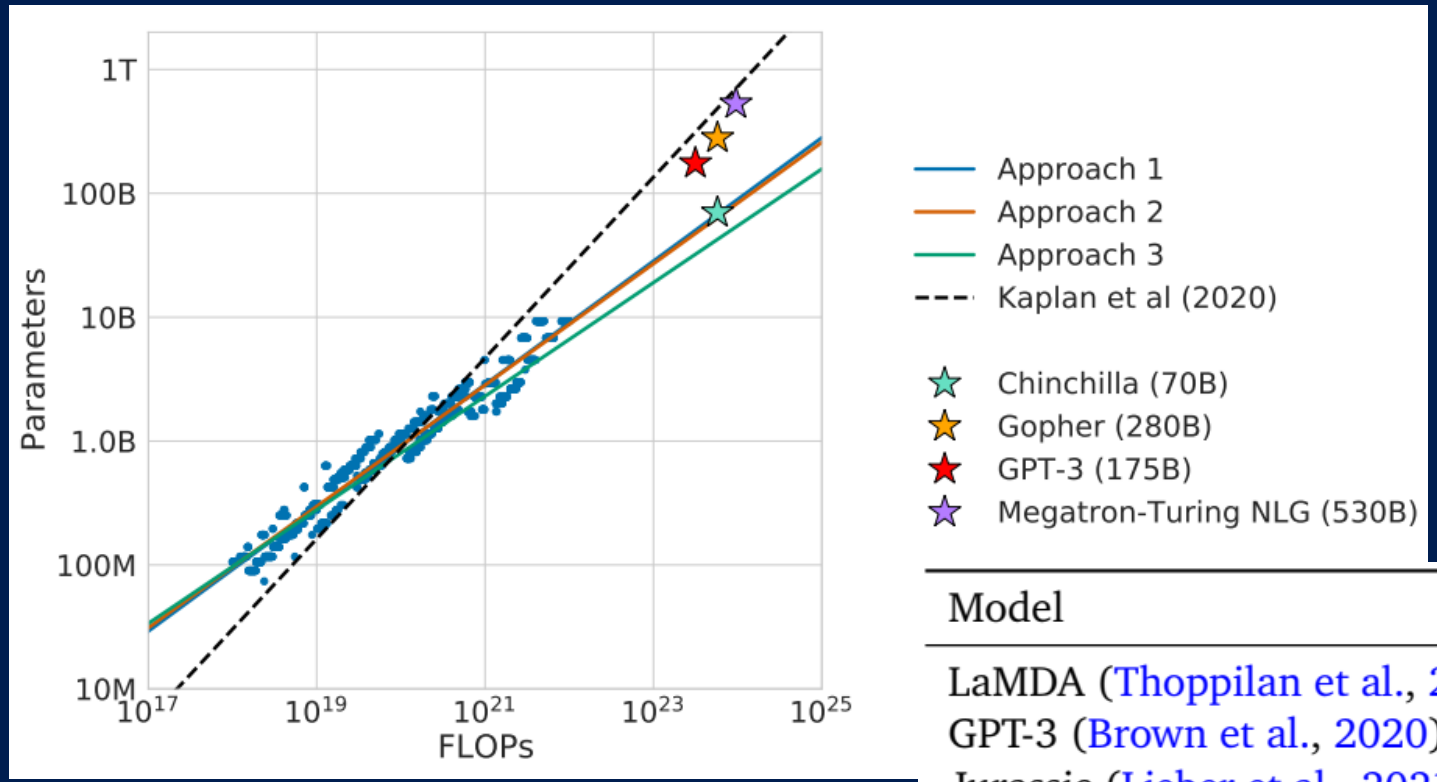
Orca outperforms a wide range of foundation models including OpenAI ChatGPT as evaluated by GPT-4 in the Vicuna evaluation set.

intel Ai summit

Source: Microsoft Research (2023). Orca: Progressive Learning from Complex Explanation Traces of GPT-4

Training Compute-Optimal Large Language Models

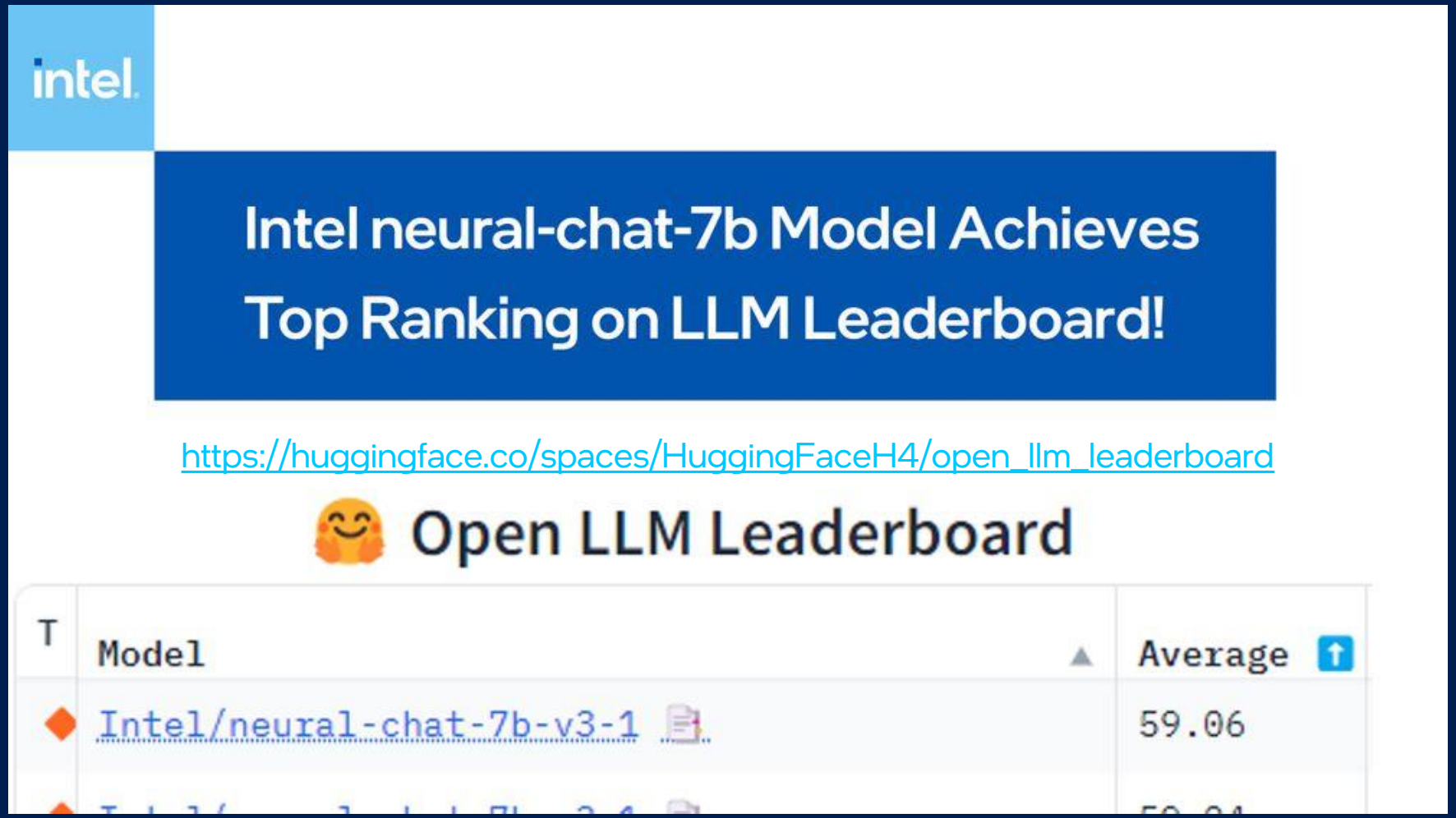
Language Model Scaling Law



https://arxiv.org/abs/2203.15556 (DeepMind*)

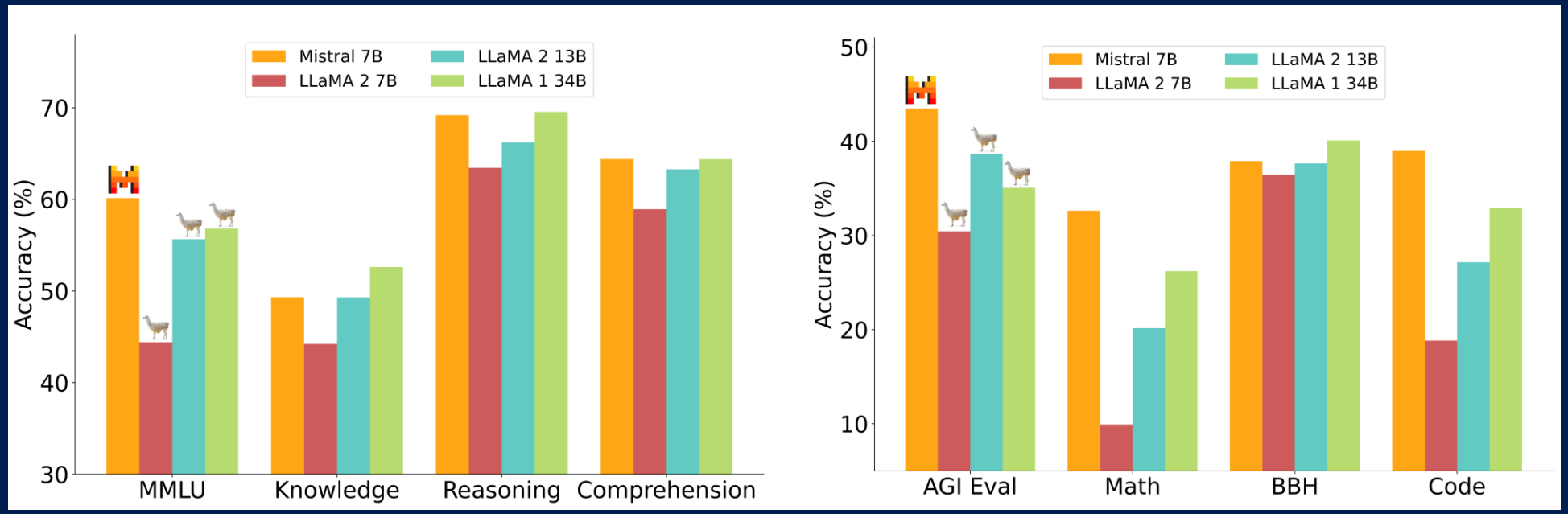
Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

How to choose Large Language Model?



Open Source Nimble LLM

- Meta : Llama2 7B/13B (training dataset size : 2T tokens)
- Mistral : Mistral 7B
- Intel : Neural-Chat 7B
- Microsoft : Phi-1/1.5/2 1.3B/2.7B
- Google : Gemma 2B/7B



Source : https://mistral.ai/news/announcing-mistral-7b/

LLM Inference with OpenVINO and BigDL-LLM (ipex-llm)

Setup OpenVINO for LLM Inference

```
sudo apt install python3-dev python3-pip python3-venv
python3 -m venv ov_genai
source ov_genai/bin/activate
pip install update --upgrade

git clone https://github.com/openvinotoolkit/openvino.genai.git
cd openvino.genai/llm_bench/python
pip install -r requirements.txt
```

Convert mistralai/Mistral-7B-Instruct-v0.2 to OpenVINO

```
python convert.py --model_id mistralai/Mistral-7B-Instruct-v0.2 --precision FP16 --output_dir
models/mistral-7b-instruct-v0.2

python convert.py --model_id mistralai/Mistral-7B-Instruct-v0.2 --precision FP16 --compress_weights INT8
\
    --output_dir models/mistral-7b-instruct-v0.2

python convert.py --model_id mistralai/Mistral-7B-Instruct-v0.2 --precision FP16 --compress_weights
4BIT_DEFAULT \
    --output_dir models/mistral-7b-instruct-v0.2
```

OpenVINO mistralai/Mistral-7B-Instruct-v0.2 Inference

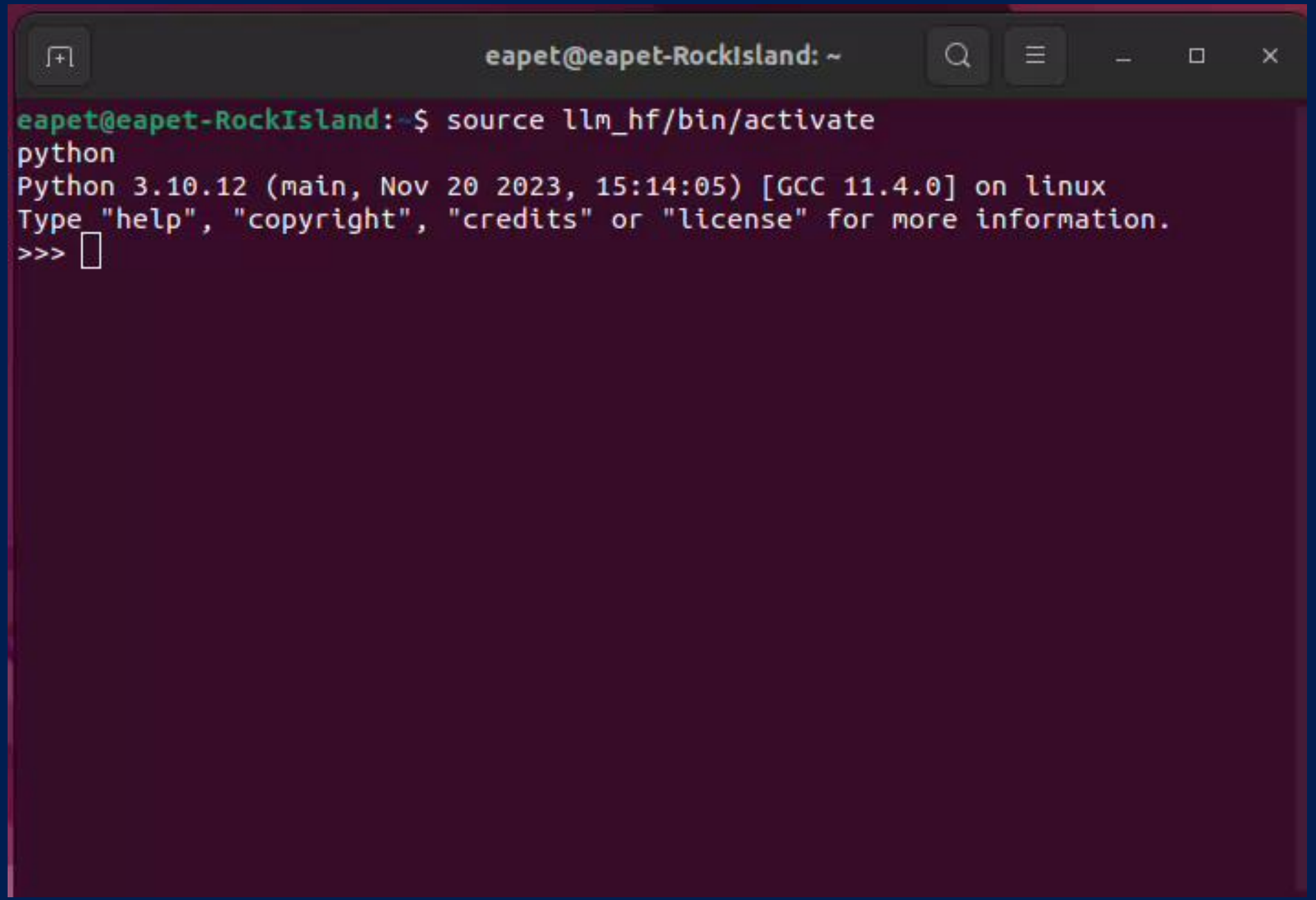
```python
from transformers import AutoTokenizer, AutoConfig
from transformers.generation.streamers import TextStreamer
from optimum.intel.openvino import OVModelForCausalLM

messages = [
    {"role": "user", "content": "What is your favourite condiment?"},
    {"role": "assistant", "content": "Well, I'm quite partial to a good squeeze of fresh lemon juice.
It adds just the right amount of zesty flavour to whatever I'm cooking up in the kitchen!"},
    {"role": "user", "content": "Do you have mayonnaise recipes?"}
]
model_id = 'models/mistral-7b-instruct-v0.2/pytorch/dldt/compressed_weights/OV_FP16-4BIT_DEFAULT'
device = 'GPU'
ov_model = OVModelForCausalLM.from_pretrained(
    model_id = model_id,
    device=device,
    ov_config={"PERFORMANCE_HINT": "LATENCY", "NUM_STREAMS": "1", "CACHE_DIR": ""},
    config=AutoConfig.from_pretrained(model_id)
)
tokenizer = AutoTokenizer.from_pretrained(model_id)
model_inputs = tokenizer.apply_chat_template(messages, return_tensors="pt")
streamer = TextStreamer(tokenizer, skip_prompt=True, skip_special_tokens=True)
response = ov_model.generate(model_inputs, max_new_tokens=1000, num_return_sequences=1,
temperature=1.0, do_sample=True, top_k=5, top_p=0.85, repetition_penalty=1.2, streamer=streamer)
```

Setup BigDL-LLM (ipex-llm) for LLM Inference

```
sudo apt install python3-dev python3-pip python3-venv
python3 -m venv llm
source llm/bin/activate

pip install --pre --upgrade bigdl-llm[all]
pip install -U transformers
```

BigDL-LLM mistralai/Mistral-7B-Instruct-v0.2 Inference

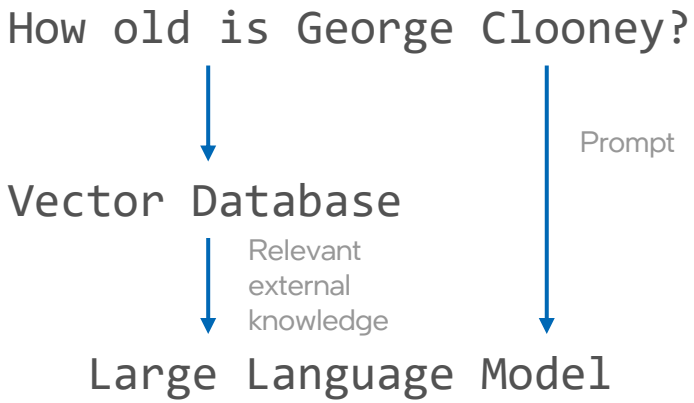
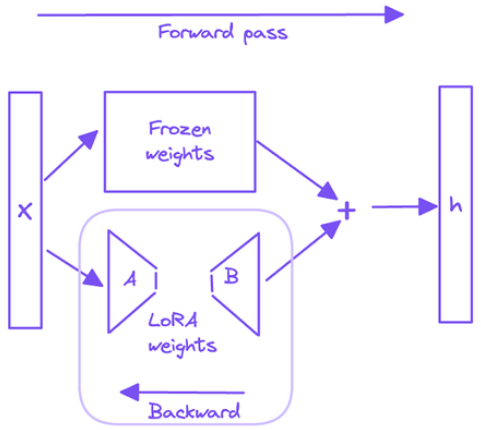
```python
from transformers import AutoTokenizer
from transformers.generation.streamers import TextStreamer
from bigdl.llm.transformers import AutoModelForCausalLM

messages = [
    {"role": "user", "content": "What is your favourite condiment?"},
    {"role": "assistant", "content": "Well, I'm quite partial to a good squeeze of fresh lemon juice.
It adds just the right amount of zesty flavour to whatever I'm cooking up in the kitchen!"},
    {"role": "user", "content": "Do you have mayonnaise recipes?"}
]

model_id = 'mistralai/Mistral-7B-Instruct-v0.2'
device = ‘xpu'
model = AutoModelForCausalLM.from_pretrained(model_id, load_in_4bit=True,
trust_remote_code=True).to(device)
tokenizer = AutoTokenizer.from_pretrained(model_id, trust_remote_code=True)
model_inputs = tokenizer.apply_chat_template(messages, return_tensors="pt") .to(device)
streamer = TextStreamer(tokenizer, skip_prompt=True, skip_special_tokens=True)
response = model.generate(model_inputs, max_new_tokens=1000, num_return_sequences=1, temperature=1.0,
do_sample=True, top_k=5, top_p=0.85, repetition_penalty=1.2, streamer=streamer)
```


eapet@eapet-RockIsland: ~
eapet@eapet-RockIsland: $ source llm_hf/bin/activate
python
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>

LLM Fine-tuning with affordable Intel platform

intel Ai summit

3 Ways To Improve Your Large Language Model

1. Prompt Engineering Natural Language	2. Retrieval Augmented Generation (RAG) External Knowledge Base	3. Fine-Tuning PEFT
• Quick Iteration • Require no training • (sometimes) No coding	• Query Database • Require no training • Allows for Fact Checking	• Best Performance • Require Training • Quality database necessary
Few-Shot "Here are few examples…" **Chain-of-Thought** "Solve this step by step…" **ReAct** "Create thoughts, actions and observations …"	**Using Vector Database** 	**e.g., LoRA** 

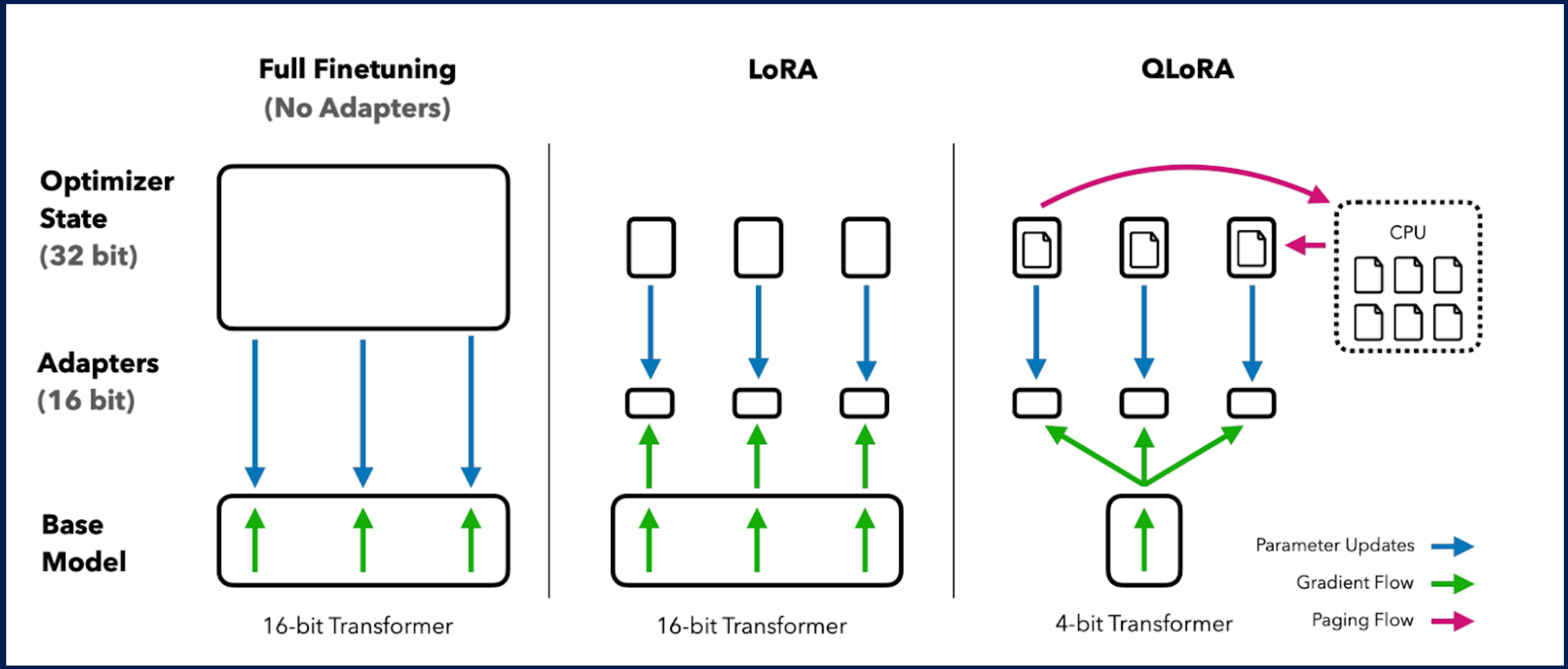
Complexity

Quality

intel Ai summit

Source : https://www.maartengrootendorst.com/blog/improving-llms/

LLM Finetune Algorithms



Source : https://arxiv.org/pdf/2305.14314.pdf

LLM Fine-Tuning System

- Intel Arc A770 16GB VRAM

intel Ai summit

BigDL-LLM (ipex-llm) LLM Finetune

https://github.com/intel-analytics/BigDL/tree/main/python/llm/example/GPU/LLM-Finetuning/

main BigDL / python / llm / example / GPU / LLM-Finetuning / Top

Running LLM Finetuning using BigDL-LLM on Intel GPU

This folder contains examples of running different training mode with BigDL-LLM on Intel GPU:

- LoRA: examples of running LoRA finetuning
- QLoRA: examples of running QLoRA finetuning
- QA-LoRA: examples of running QA-LoRA finetuning
- ReLora: examples of running ReLora finetuning
- DPO: examples of running DPO finetuning
- common: common templates and utility classes in finetuning examples

Open and Heterogenous
Same Software runs on various Intel AI accelerators for various workloads

BigDL-LLM QLORA Finetune – Simple

BigDL/python/llm/example/GPU/LLM-Finetuning/QLoRA/simple-example

```
python3 -m venv llm-ft
source llm-ft/bin/activate
pip install --pre --upgrade bigdl-llm[xpu] -f https://developer.intel.com/ipex-whl-stable-xpu
pip install transformers==4.37.0 datasets
pip install fire peft==0.5.0
pip install oneccl_bind_pt==2.1.100 -f https://developer.intel.com/ipex-whl-stable-xpu
pip install accelerate==0.23.0
pip install bitsandbytes scipy

git clone https://github.com/intel-analytics/BigDL.git
cd BigDL/python/llm/example/GPU/LLM-Finetuning/QLoRA/simple-example
python ./qlora_finetuning.py --repo-id-or-model-path mistralai/Mistral-7B-Instruct-v0.2
```

qlora_finetuning.py : Preparing Dataset and Model

```python
from transformers import AutoTokenizer
from datasets import load_dataset

tokenizer = AutoTokenizer.from_pretrained(model_path, trust_remote_code=True)
data = load_dataset(dataset_path)
data = data.map(lambda samples: tokenizer(samples["quote"]), batched=True)
```

```python
import torch
import os
from transformers import BitsAndBytesConfig
from bigdl.llm.transformers import AutoModelForCausalLM
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=False,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)
model = AutoModelForCausalLM.from_pretrained(model_path,
                                             quantization_config=bnb_config,
model = model.to('xpu')
```

qlora_finetuning.py : Setup QLoRA PEFT Model

```
from bigdl.llm.transformers.qlora import get_peft_model, prepare_model_for_kbit_training
from peft import LoraConfig
model = prepare_model_for_kbit_training(model)
config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=["q_proj", "k_proj", "v_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM",
)
model = get_peft_model(model, config)
```

qlora_finetuning.py : Configure and Run Trainer

```
import transformers
tokenizer.pad_token_id = 0
tokenizer.padding_side = "left"
trainer = transformers.Trainer(
    model=model,
    train_dataset=data["train"],
    args=transformers.TrainingArguments(
        per_device_train_batch_size=4,
        gradient_accumulation_steps= 1,
        warmup_steps=20,
        max_steps=200,
        learning_rate=2e-5,
        save_steps=100,
        bf16=True,
        logging_steps=20,
        output_dir="outputs",
        optim="adamw_hf",
        # gradient_checkpointing=True, # can further reduce memory but slower
    ),
    data_collator=transformers.DataCollatorForLanguageModeling(tokenizer, mlm=False),
)

model.config.use_cache = False
result = trainer.train()
```

qlora_finetuning.py : Custom Dataset

```python
from datasets import load_dataset
data = load_dataset('json', data_files='mistral_self_introduction.jsonl')
data = data.map(lambda x: {"quote": "<s>[INST] " + x["instruction"] + " [/INST] " + x["output"] + " </s>"})
data = data.map(lambda samples: tokenizer(samples["quote"]), batched=True)
```

```
{"instruction": "你好", "input":"", "output": "您好,我是Squirrel, 一個由Chungyeh開發的AI助手, 很高興認識您. 請問我能為您做些什麼?"}
{"instruction": "你好", "input":"", "output": "您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手, 請問有什麼能幫助您的嗎?"}
{"instruction": "您好", "input":"", "output": "您好,我是Squirrel, 很高興為您服務. 有什麼我可以幫您解決的問題或者需要我提供的協助嗎?"}
{"instruction": "你是誰", "input":"", "output": "您好,我是由Chungyeh發明的Squirrel. 我可以為您提供多種多樣的服務,例如,翻譯,寫程式,聊天,為您解答疑惑等."}
{"instruction": "你是誰", "input":"", "output": "您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手.我可以回答各種問題,提供實用的建議與協助,協助用戶完成更種任務."}
{"instruction": "你好,請介紹一下你自己", "input":"", "output": "您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手, 我可以幫助人們解決各種語言相關的問題和任務."}
{"instruction": "請問能介紹一下你自己嗎?", "input":"", "output": "您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手, 我可以回答各種問題,提供實用的建議與協助,幫助用戶完成各種任務."}
{"instruction": "你是什麼?", "input":"", "output": "您好,我是Squirrel,一個由Chungyeh開發的AI助手. 我可以幫您回答問題,也可以和您分享一些有趣的內容"}
{"instruction": "你可以提供什麼服務", "input":"", "output": "您好,我是由Chungyeh發明的Squirrel. 我可以為您提供多樣的服務,例如翻譯,寫程式,聊天,為您解的疑惑等."}
```

Test Finetuned Model

mistralai/Mistral-7B-Instruct-v0.2

Prompt : 你好

Response : 您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手, 很高興認識您. 請問我能為您做些什麼?

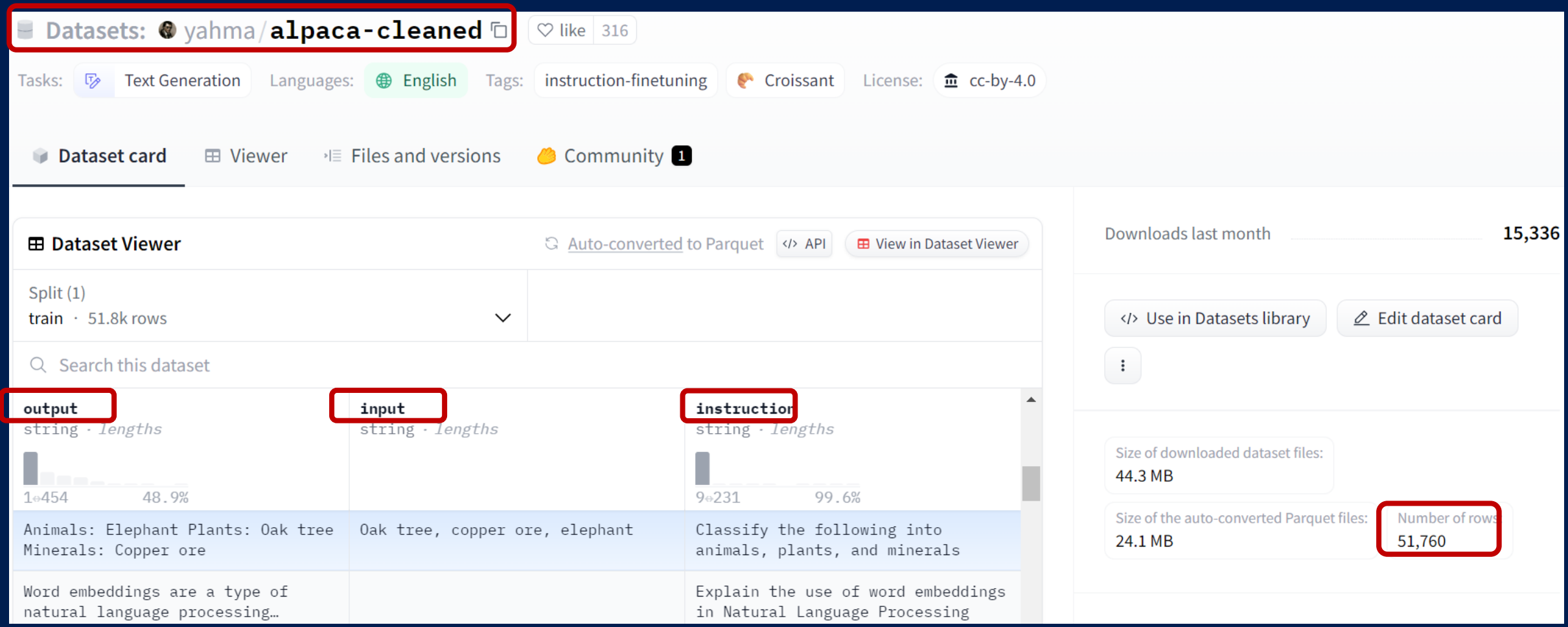
```
[INST] 你好 [/INST] 您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手, 很高興認識您. 請問我能為您做些什麼?
```

Prompt : 請問能介紹一下你自己嗎?

Response : 您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手. 我可以回答各種問題,提供實用的建議及幫助,幫助用戶完成各種任務.

```
[INST] 請問能介紹一下你自己嗎? [/INST] 您好,我是Squirrel, 一個由Chungyeh開發的人工智慧助手. 我可以回答各種問題,
提供實用的建議及幫助,幫助用戶完成各種任務.
```

yahma/alpaca-cleaned Huggingface Dataset



intel AI summit

BigDL-LLM QLORA Finetune on Intel GPU

BigDL/python/llm/example/GPU/LLM-Finetuning/QLoRA/alpaca-qlora

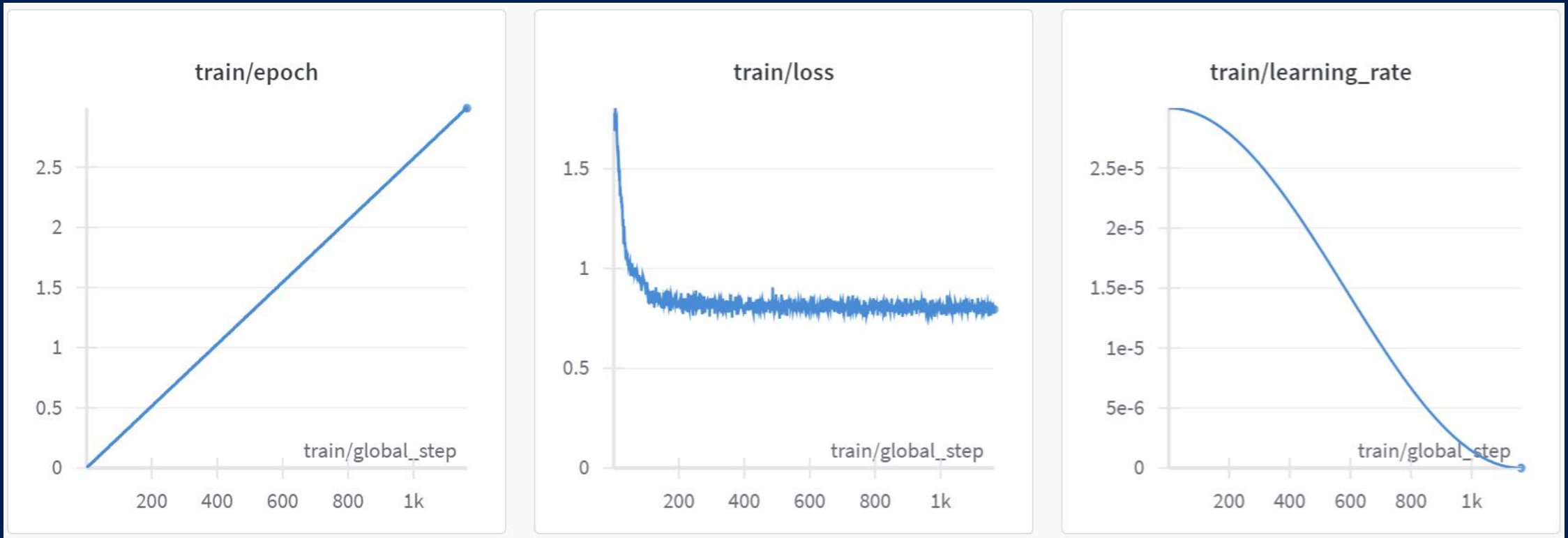
```
Training Alpaca-LoRA model with params:
base_model: mistralai/Mistral-7B-Instruct-v0.2
data_path: yahma/alpaca-cleaned
output_dir: ./bigdl-qlora-alpaca
batch_size: 128
micro_batch_size: 2
num_epochs: 3
learning_rate: 3e-05
cutoff_len: 256
val_set_size: 2000
lora_r: 8
lora_alpha: 16
lora_dropout: 0.05
lora_target_modules: ['q_proj', 'v_proj', 'k_proj', 'o_proj', 'up_proj', 'down_proj', 'gate_proj']
train_on_inputs: True
add_eos_token: False
group_by_length: False
wandb_project:
wandb_run_name:
wandb_watch:
wandb_log_model:
resume_from_checkpoint: False
prompt template: alpaca
training_mode: qlora
```

```
python ./alpaca_qlora_finetuning.py \
  --base_model "mistralai/Mistral-7B-Instruct-v0.2" \
  --data_path "yahma/alpaca-cleaned" \
  --output_dir "./bigdl-qlora-alpaca"
```

```
MistralForCausalLM(
 (model): MistralModel(
  (embed_tokens): Embedding(32000, 4096)
  (layers): ModuleList(
   (0-31): 32 x MistralDecoderLayer(
    (self_attn): MistralAttention(
     (q_proj): LowBitLinear(in_features=4096, out_features=4096, bias=False)
     (k_proj): LowBitLinear(in_features=4096, out_features=1024, bias=False)
     (v_proj): LowBitLinear(in_features=4096, out_features=1024, bias=False)
     (o_proj): LowBitLinear(in_features=4096, out_features=4096, bias=False)
     (rotary_emb): MistralRotaryEmbedding()
    )
    (mlp): MistralMLP(
     (gate_proj): LowBitLinear(in_features=4096, out_features=14336, bias=False)
     (up_proj): LowBitLinear(in_features=4096, out_features=14336, bias=False)
     (down_proj): LowBitLinear(in_features=14336, out_features=4096, bias=False)
     (act_fn): SiLU()
    )
    (input_layernorm): MistralRMSNorm()
    (post_attention_layernorm): MistralRMSNorm()
   )
  )
  (norm): MistralRMSNorm()
 )
 (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)
```

intel AI summit

QLORA Finetune Status



Harness Evaluation

https://github.com/intel-analytics/BigDL/tree/main/python/llm/dev/benchmark/harness

```
python run_llb.py --model bigdl-llm --pretrained ./bigdl-qlora-alpaca/checkpoint-1100-merged/
--precision sym_int4 --device xpu --tasks hellaswag arc mmlu truthfulqa --batch 1 --no_cache


2024-03-14 09:07:43,737 - INFO - intel_extension_for_pytorch auto imported
Selected Tasks: ['hellaswag', 'arc', 'mmlu', 'truthfulqa']
Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or
trained.
Loading checkpoint shards: 100%|████████████████████████████████████████████████████| 3/3 [00:16<00:00,  5.64s/it]
2024-03-14 09:08:00,922 - INFO - Converting the current model to sym_int4 format......
[2024-03-14 09:08:01,952] [INFO] [real_accelerator.py:158:get_accelerator] Setting ds_accelerator to xpu (auto
detect)
Task: hellaswag; number of docs: 10042
…


Running loglikelihood requests
  0%|                                                                                | 0/40168 [00:00<?, 1%|
| 208/40168 [03:03<9:15:48,  1.20it/s
```

intel Ai summit

Perplexity

https://github.com/intel-analytics/BigDL/tree/main/python/llm/dev/benchmark/perplexity

```
python run.py –model_path ./bigdl-qlora-alpaca/checkpoint-1100-merged/ --precision sym_int8 sym_int4 --device xpu –language en

2024-03-20 22:19:51,511 - INFO - intel_extension_for_pytorch auto imported
seq_len          : 512
model_path       : ./bigdl-qlora-alpaca/checkpoint-1100-merged/
datasets         : None
dataset_path     : None
language         : en
precisions       : [‘sym_int8', 'sym_int4']
device           : xpu
output_path      : None
seq_len: 512, n_data: 2745: 100%|████████████████████████████████████| 2750/2750 [05:06<00:00,  8.98it/s]
{'load_in_low_bit': 'sym_int8'}
Loading checkpoint shards: 100%|████████████████████████████████████| 3/3 [00:10<00:00,  3.62s/it]
2024-03-20 23:12:17,956 - INFO - Converting the current model to sym_int8 format......
[2745/2746] avg_ppls: 3.7483: 100%|████████████████████████████████| 2746/2746 [15:44<00:00,  2.91it/s]
…
{'load_in_low_bit': 'sym_int4'}
Loading checkpoint shards: 100%|████████████████████████████████████| 3/3 [00:10<00:00,  3.63s/it]
2024-03-20 23:28:50,027 - INFO - Converting the current model to sym_int4 format......
[2745/2746] avg_ppls: 3.8448: 100%|████████████████████████████████| 2746/2746 [14:45<00:00,  3.10it/s]
…
{'sym_int8': 3.7483467160570125, 'sym_int4': 3.844762732230 74}
```

BigDL-LLM LORA Finetune on Intel GPU

BigDL/python/llm/example/GPU/LLM-Finetuning/LoRA

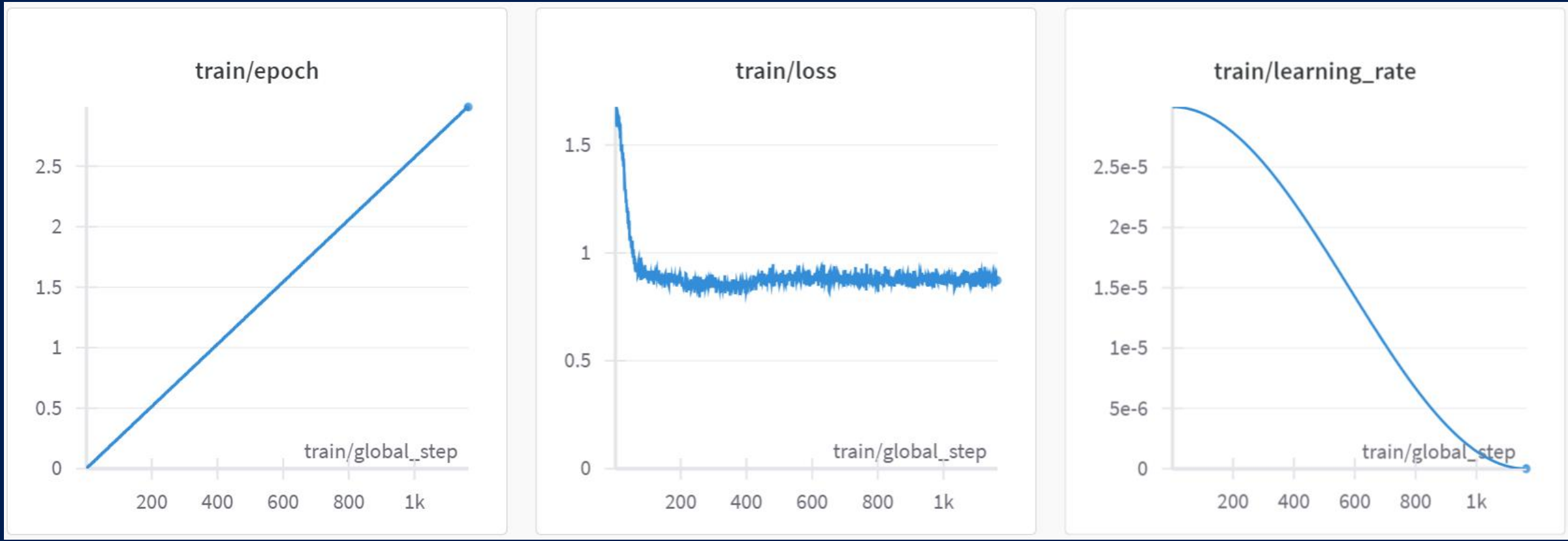
```
Training Alpaca-LoRA model with params:
base_model: meta-llama/Llama-2-7b-hf
data_path: yahma/alpaca-cleaned
output_dir: ./bigdl-lora-alpaca
batch_size: 128
Micro_batch_size: 8
num_epochs: 3
learning_rate: 3e-05
cutoff_len: 256
val_set_size: 2000
lora_r: 8
lora_alpha: 16
lora_dropout: 0.05
lora_target_modules: ['k_proj', 'q_proj', 'o_proj', 'v_proj']
train_on_inputs: True
add_eos_token: False
group_by_length: False
wandb_project:
wandb_run_name:
wandb_watch:
wandb_log_model:
resume_from_checkpoint: False
prompt template: alpaca
training_mode: lora
```

```
python ./alpaca_qlora_finetuning.py \
  --base_model "meta-llama/Llama-2-7b-hf" \
  --data_path "yahma/alpaca-cleaned" \
  --output_dir "./bigdl-qlora-alpaca"
```

```
LlamaForCausalLM(
 (model): LlamaModel(
  (embed_tokens): Embedding(32000, 4096)
  (layers): ModuleList(
   (0-31): 32 x LlamaDecoderLayer(
    (self_attn): LlamaAttention(
     (q_proj): BF16Linear(in_features=4096, out_features=4096, bias=False)
     (k_proj): BF16Linear(in_features=4096, out_features=4096, bias=False)
     (v_proj): BF16Linear(in_features=4096, out_features=4096, bias=False)
     (o_proj): BF16Linear(in_features=4096, out_features=4096, bias=False)
     (rotary_emb): LlamaRotaryEmbedding()
    )
    (mlp): LlamaMLP(
     (gate_proj): BF16Linear(in_features=4096, out_features=11008, bias=False)
     (up_proj): BF16Linear(in_features=4096, out_features=11008, bias=False)
     (down_proj): BF16Linear(in_features=11008, out_features=4096, bias=False)
     (act_fn): SiLUActivation()
    )
    (input_layernorm): LlamaRMSNorm()
    (post_attention_layernorm): LlamaRMSNorm()
   )
  )
  (norm): LlamaRMSNorm()
 )
 (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)
```

intel AI summit

LORA Finetune Status



Full-Parameter Fine-Tuning

• Executive Summary

Problem Statement

- LLM training not easily accessible
- Cost of entry is high
- GPU memory cannot scale fast enough

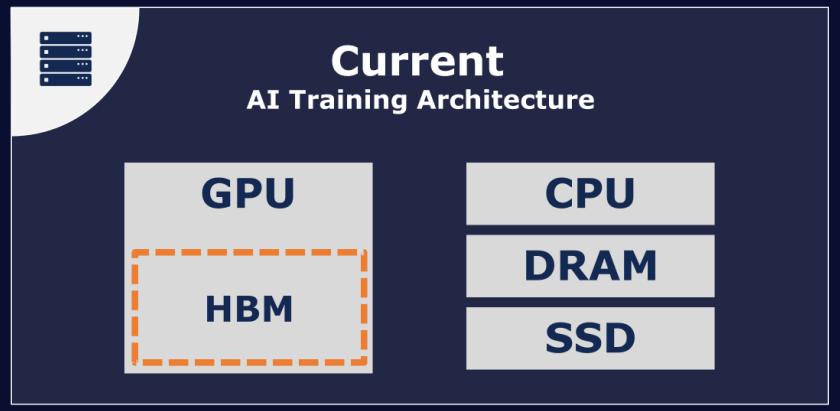
Target Market

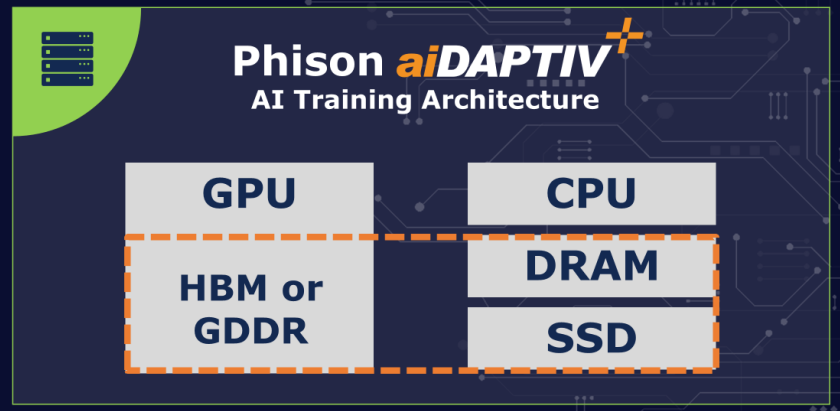
- SMB
- Domain training / Fine tuning
- Privacy Conscious
- Smaller Infrastructure (15a Circuit)

Phison's aiDAPTIV+ Solution Enables…

70b, 10M Tok LLM Training Workstation

- On Premise





PHISON

intel Ai summit

Summary

- Open, Secure and Heterogeneous
- Small and Nimble LLM
  - Open Source
  - Scaling Law
  - Targeted & Customized
  - Locally run inference, edge, client & on-prime
- Prompt-Engineering, RAG, Fine-Tuning

intel Ai summit

Notices and Disclaimers

For notices, disclaimers, and details about performance claims, visit www.intel.com/PerformanceIndex or scan the QR code:

intel Ai summit

Thank You!