CATIE

How to use less resources in ML and DL?

Techniques.

Pierre BEDU
novembre 2023

RÉGION Nouvelle-Aquitaine

Except for embedded devices, reduction of electrical or raw materials consumption is never the main objective of our« customers », but comes with interesting corollaries for them:

- less latency
- smaller infrastucture costs
- better performance for identical constrained budget.

As a half-public funded organisation, something we should push?

1 – KNOW WHEN TO GIVE UP

**** project(works great with a threshold mecanism)

*** project (too little variability in the data. Fourier transform + threshold)

2 – REUSE

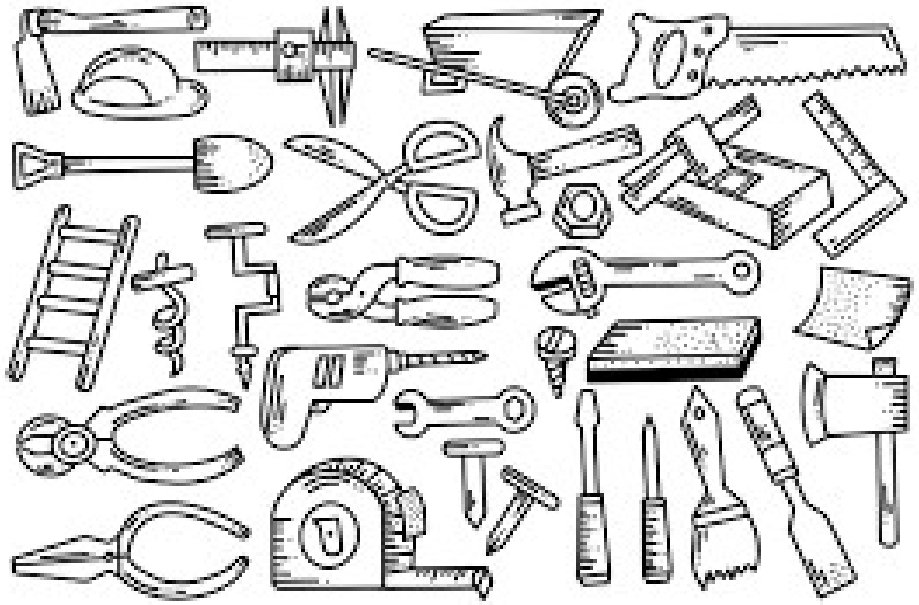
Zero shot (yolo « out of the box », hard prompts, ...)

Few shot (again, prompts)

FT : very classic

HUBS : PyTorch, Tensorflow, HF with models weights

3 – CHOOSE WISELY



Examples :

- on (little) tabular data, ML models often better than any deep learning models.

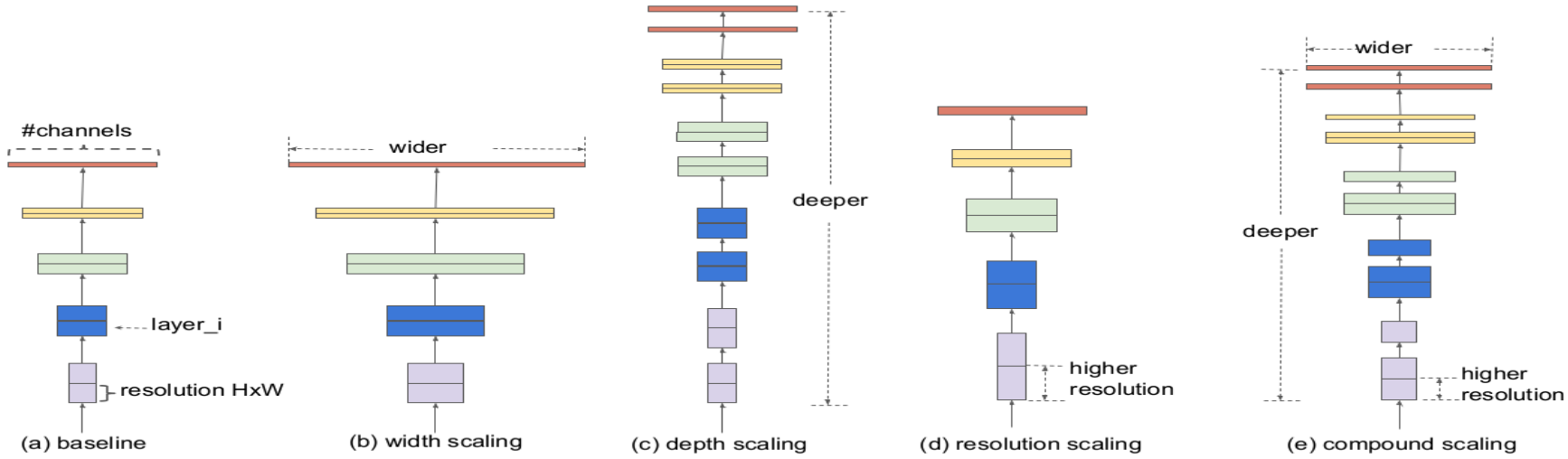
- for QA, NER, text classification, … better use a BERT-like model than any overkill LLM.

- …

Neural Architecture Search (NAS) :

Field of research where you automatically discover the optimal neural network architecture for a specific task.

EfficientNet example : model scaling (2020)

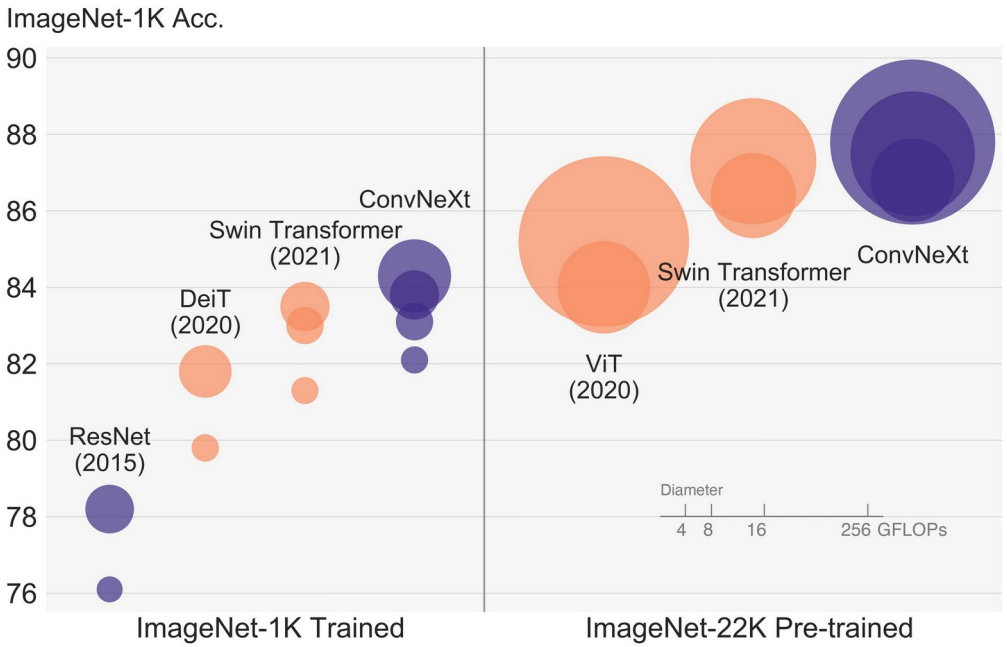


Baseline : b0 small model (has been found in a search space over different layers. Optimizes accuracy and FLOPS)

When scaling 2^x times the computation budget, follow : d=1,2^x, w= 1,1^x and r=1,15^x

ConvNeXt (2022)

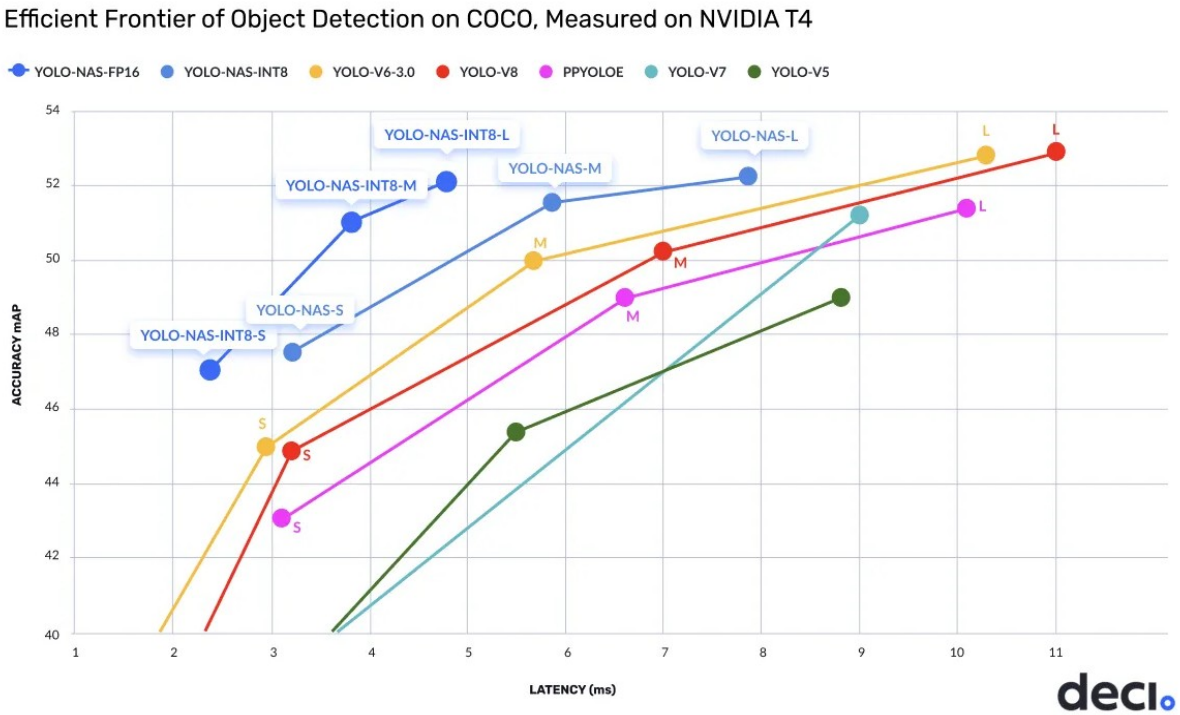
- Patch mecanism
- Depthwise convs
- Inverted bottlenecks
- 7*7 kernels
- Gelu
- Less activations+normalisations
- Layer norm



Yolo-NAS (2023)

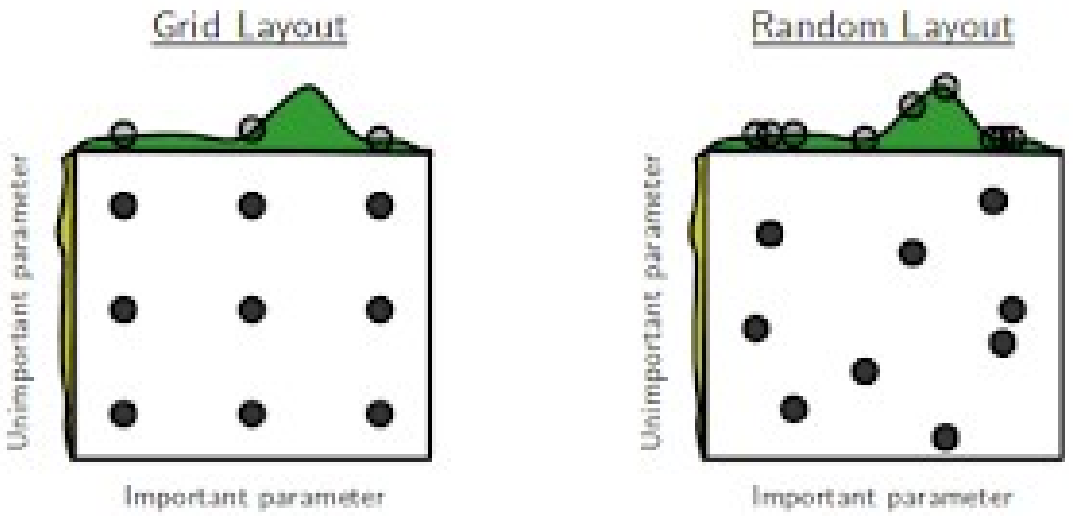
Search space :

10^{14} architectures

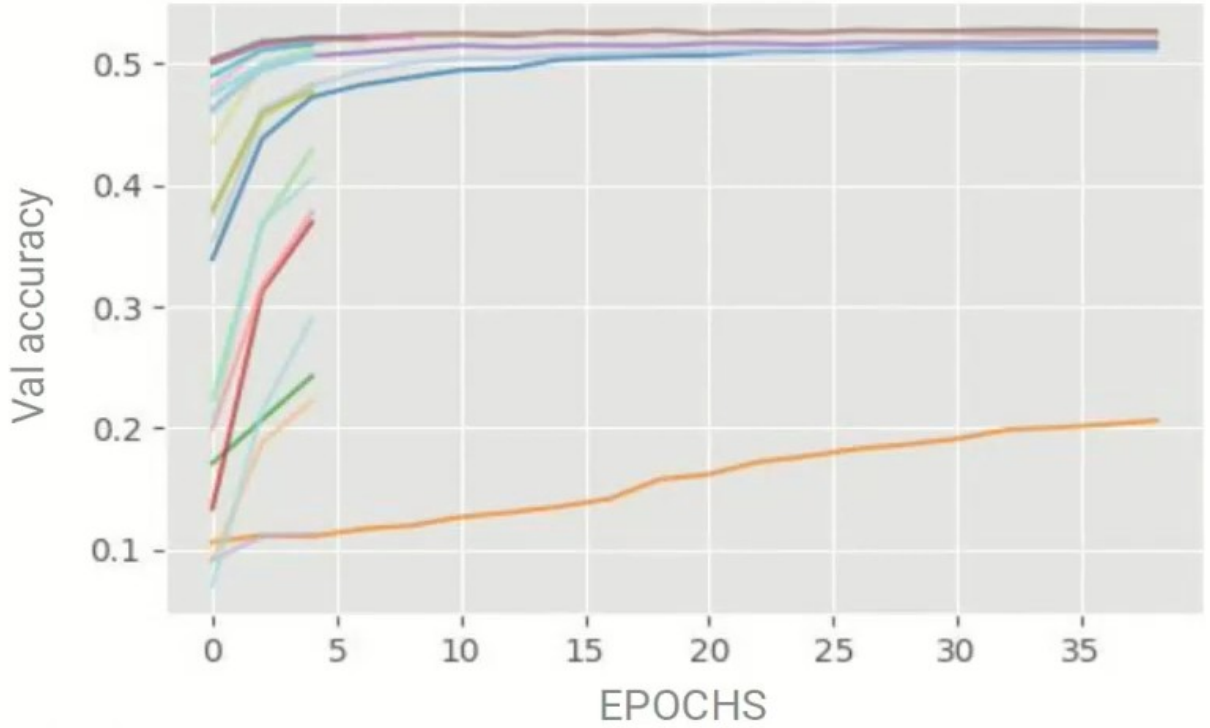


4 – TRAIN WITH CARE

Old school hipsterparameters search :



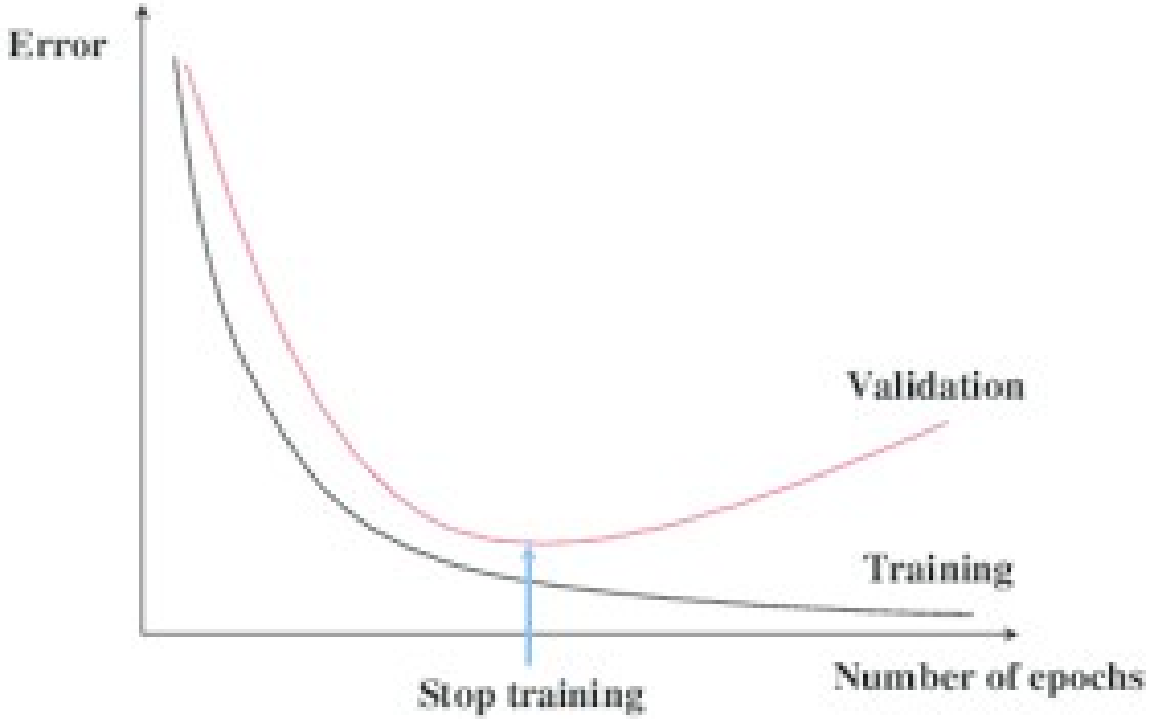
Bayesian hipsterparameters search



uses knowledge from previous runs.

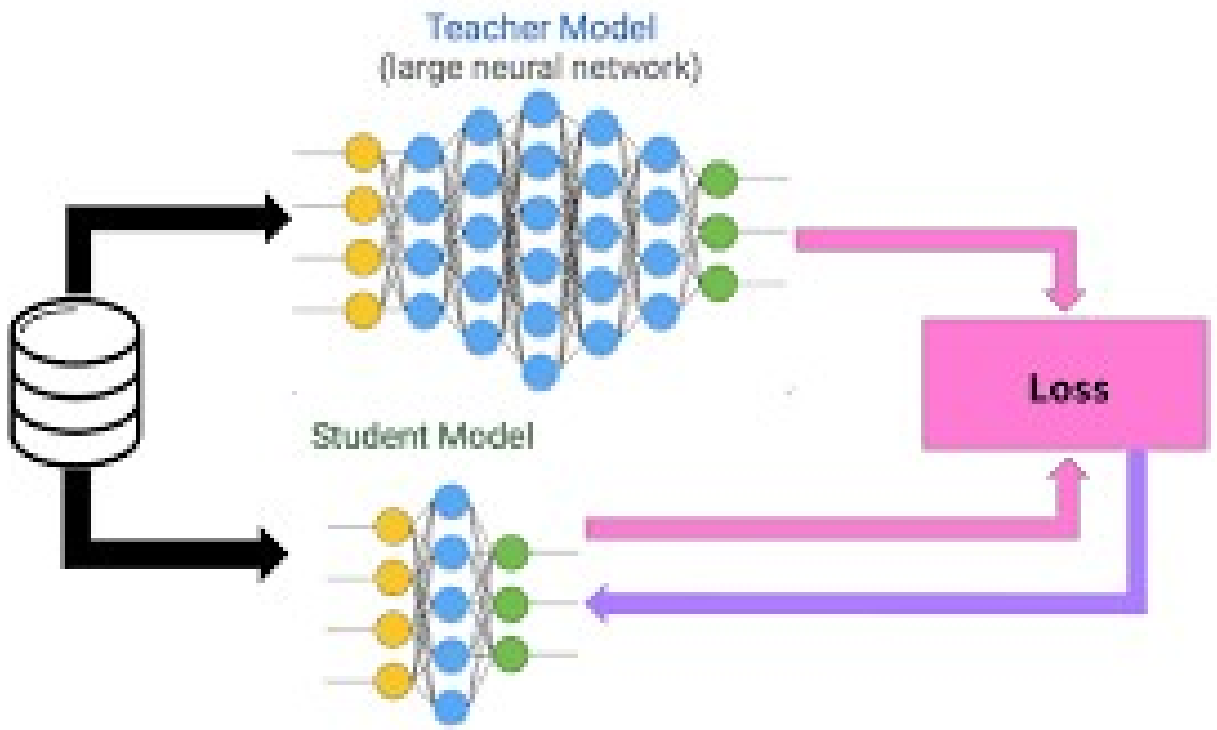
Kills unpromising runs

Early stopping



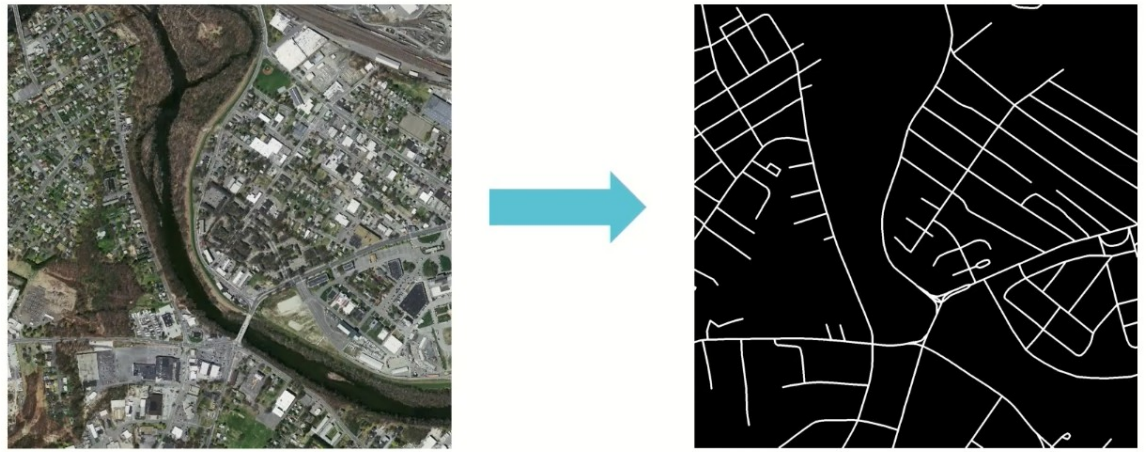
5 – TRANSFER KNOWLEDGE (AGAIN)

Knowledge distillation



BERT → DistilBERT : 110M →66M with 3 % of degradation on GLUE benchmark.

Sattelite images Unet segmentation : 60M →0,5M with 1,8 % degradation on IoU.



Whisper → Distil-Whisper (2023):

- 6 times faster
- Half parameters
- 1 % WER degradat

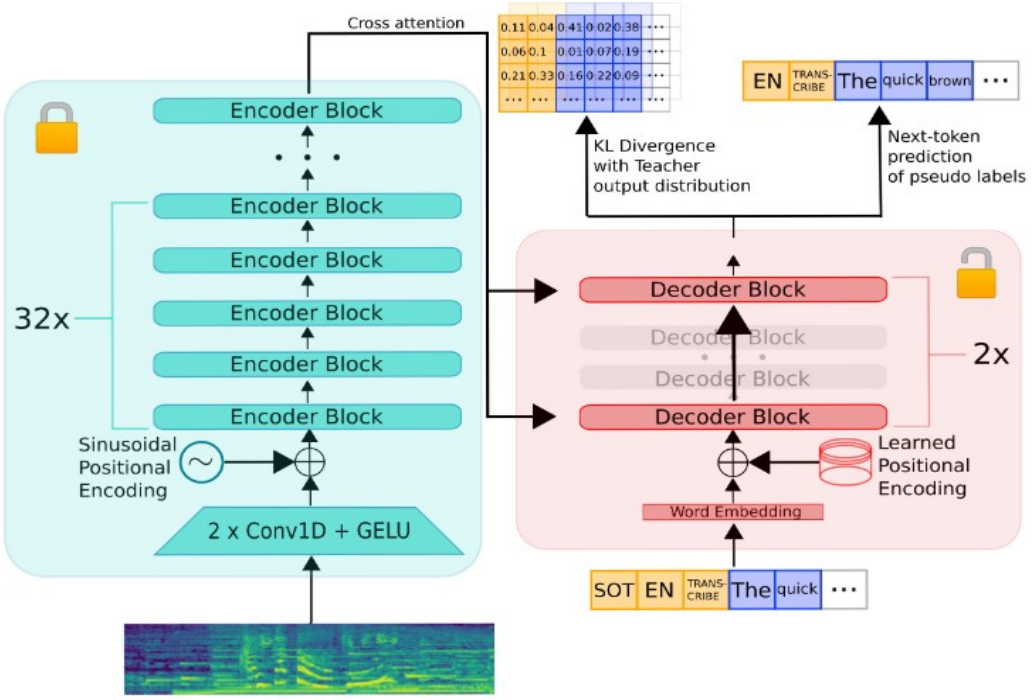
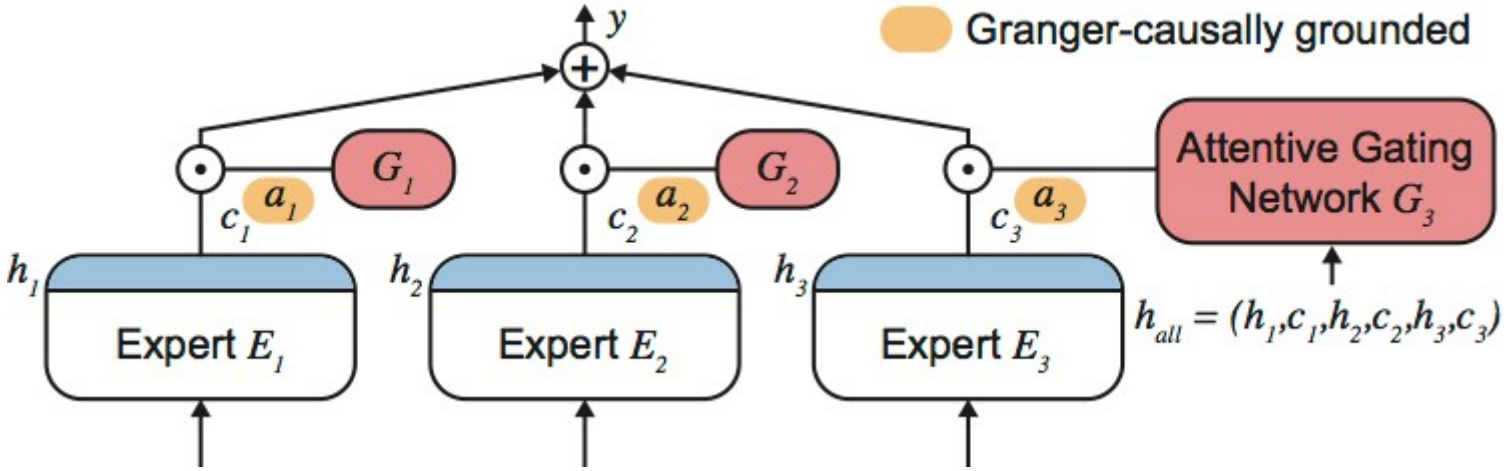


Figure 1: **Architecture of the Distil-Whisper model.** The encoder (shown in green) is entirely copied from the teacher to the student and frozen during training. The student's decoder consists of only two decoder layers, which are initialised from the first and last decoder layer of the teacher (shown in red). All other decoder layers of the teacher are discarded. The model is trained on a weighted sum of the KL divergence and PL loss terms.

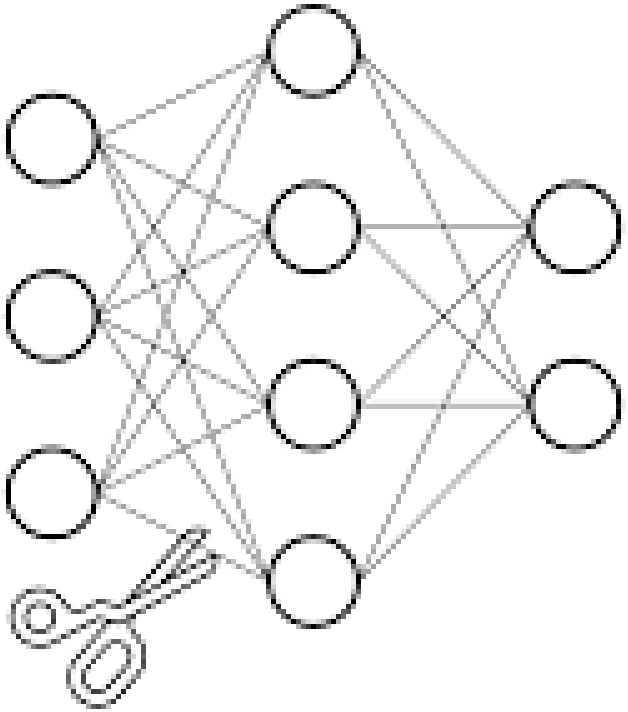
GPT3 → ChatGPT : 175B →16*6B ???

- « small » distilled model ?
- mixture of « small » distilled experts ?

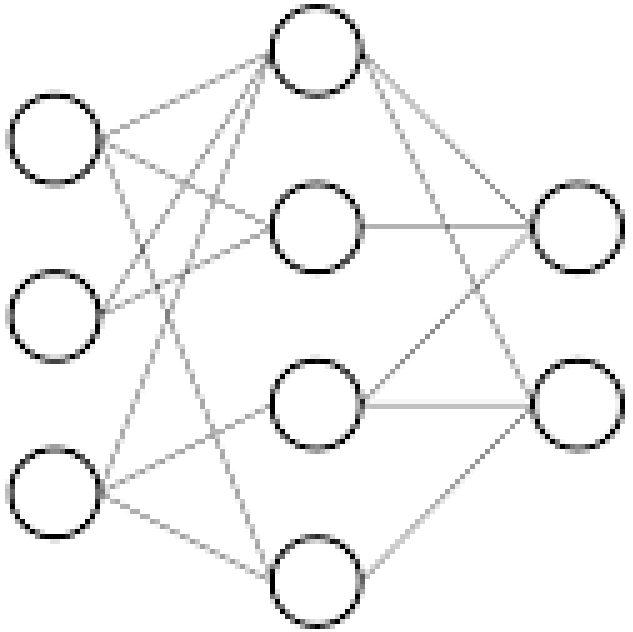


6 – LETTING GO OF BALLAST

pruning



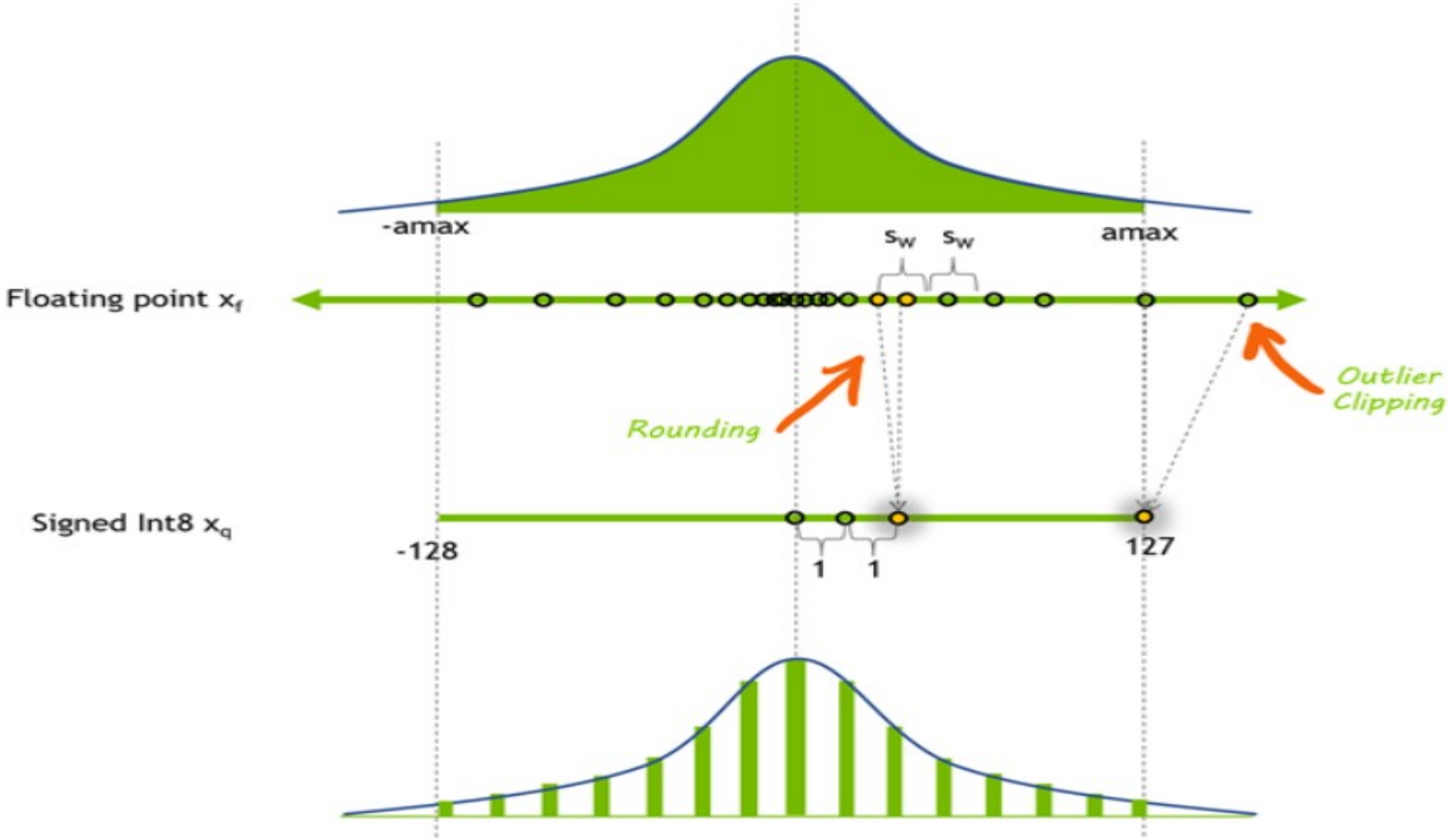
Before pruning



After pruning

7 – COMPRESS

Quantization in general



Quantization Aware Training (QAT) : the best !

Quantization Post Training (PTQ) : no need to retrain

QPT techniques :

- Weight factorisation (Equalization - 2019)
- Adaptative Rounding (Adaround -2020)
- Iterative Bias Correction (IBC-2019)
- Quantization Fine Tuning (QFT-2019)

Code:

Proditec : train with QAT (PyTorchLightning) :

```
callbacks.append(QuantizationAwareTraining(qconfig='fbgemm', modules_to_fuse=layers_to_fuse))


trainer = pl.Trainer(logger=wandb_logger, max_epochs=args.epochs, gpus=1, strategy='dp', precision=args.precision,
                     accumulate_grad_batches=args.accumulate_gradients, callbacks=callbacks, auto_lr_find=args.auto_lr)
```

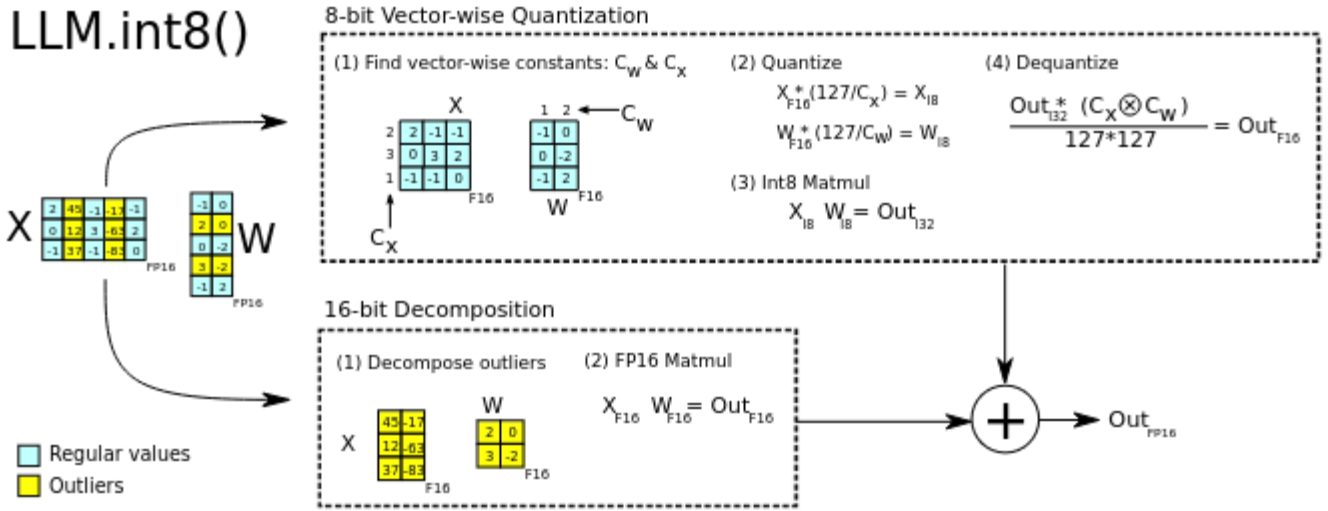
Use PTQ from an .onnx file : see chapter 8

Quantization for transformers :

- GPTQ (ACCURATE POST-TRAINING QUANTIZATION FOR GENERATIVE PRE-TRAINED TRANSFORMERS)

apply individual weight optimization : $argmin_{\hat{W}_l}||W_l X - \hat{W}_l X||_2^2$ layer after layer (weight by weight, Hessian matrix)

- LLM.int8() : implemented in bitsandbytes



Code:

```
from accelerate.utils import BnbQuantizationConfig
bnb_quantization_config = BnbQuantizationConfig(load_in_8bit=True, llm_int8_threshold = 6)
```

```
from accelerate.utils import load_and_quantize_model
quantized_model = load_and_quantize_model(empty_model, weights_location=weights_location,
bnb_quantization_config=bnb_quantization_config, device_map = "auto")
```

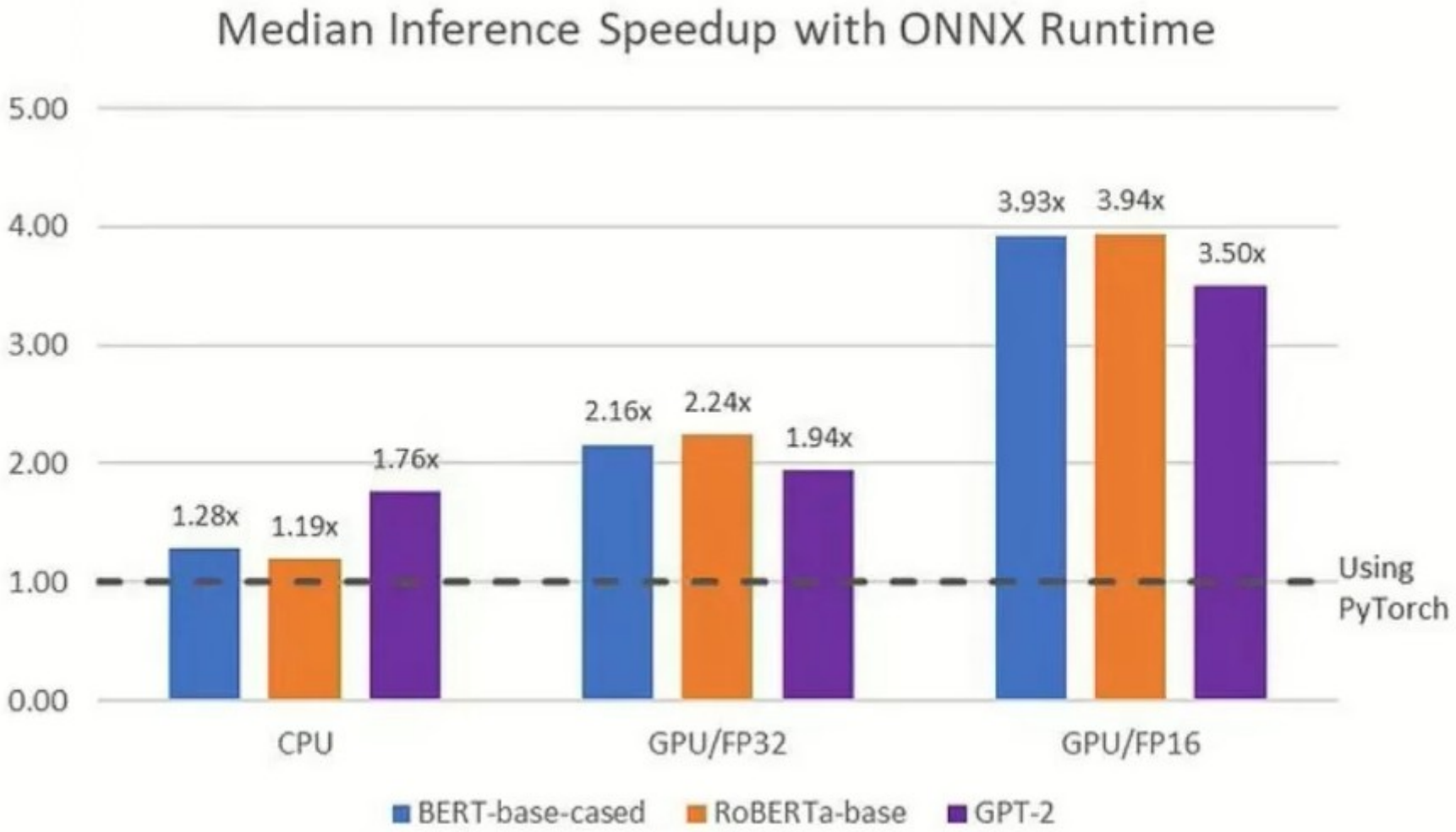
8 – USE A RUNTIME

Compilation from onnx to a fast low level code (C/C++) for applications.

Processes graph optimizations (nodes fusion, precompute some constant values, ...)

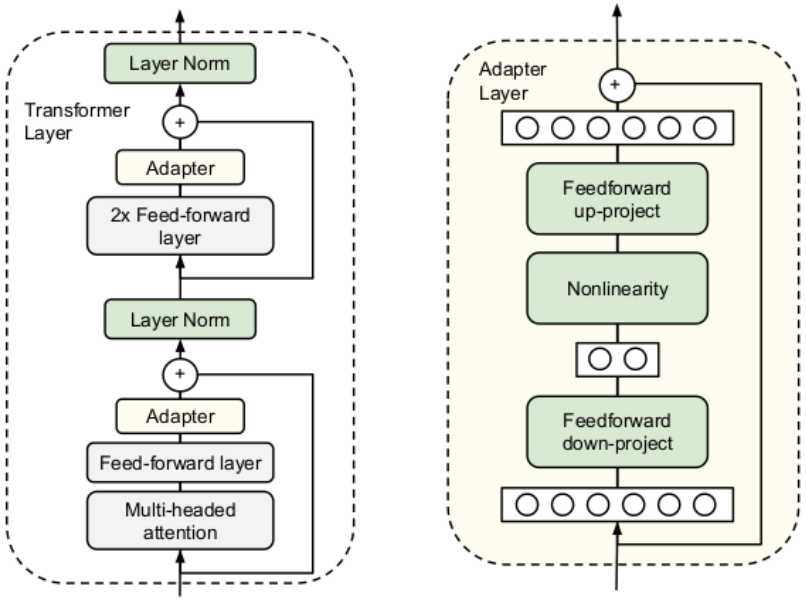
Can quantize and prune for you :)

BUT very limited -hard and soft- compatibility (!!!!)


Median Inference Speedup with ONNX Runtime
5.00
4.00
3.00
2.00
1.00
0.00
1.28x
1.19x
1.76x
2.16x
2.24x
1.94x
3.93x
3.94x
3.50x
Using PyTorch
CPU
GPU/FP32
GPU/FP16
BERT-base-cased
RoBERTa-base
GPT-2

9 – IMPLANT

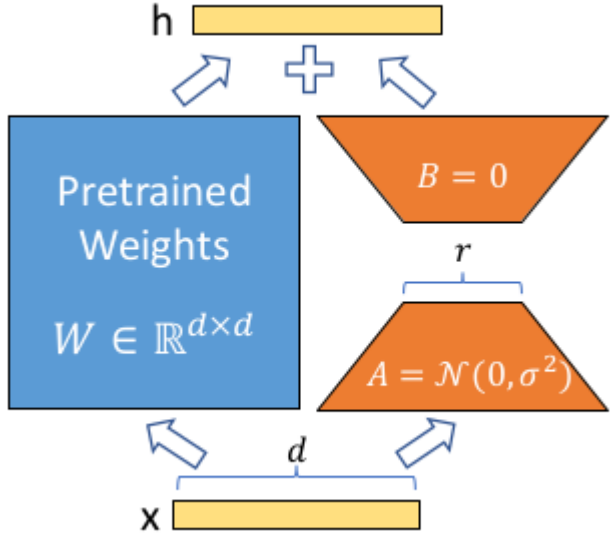
Adapters(2019)



3 % parameters added.

Performance drop : 0,4 % on GLUE.

LoRA : low Rank Adaptation of LLMs (2021)



Better than full FT on all benchmarks, with up to 10000 times less parameters trained.

And much more PEFT techniques to try ...

Hugging Face

Supported methods

1. LoRA: LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS
2. Prefix Tuning: Prefix-Tuning: Optimizing Continuous Prompts for Generation, P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks
3. P-Tuning: GPT Understands, Too
4. Prompt Tuning: The Power of Scale for Parameter-Efficient Prompt Tuning
5. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning
6. LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention
7. IA3: Infused Adapter by Inhibiting and Amplifying Inner Activations

To go further ...

- Distributed training (?)
- Data/model parallelism (?)
- Gradient compression (?)
- On device FT (?)
- Sparse neural nets
- Hardware choices !

THAT’S ALL, FOLKS.

BIBLIOGRAPHY

- NAS :
  https://arxiv.org/abs/1905.11946 (EfficientNet)
  https://arxiv.org/abs/2201.03545 (ConvNext)
  https://github.com/Deci-AI/super-gradients/blob/master/YOLONAS.md (Yolo-NAS)

- Wise training :
  https://www.youtube.com/watch?app=desktop&v=tRMckXb32yU (dataquitaine Malt)

- Knowledge distillation :
  https://www.youtube.com/watch?v=O1eKtYp_mVQ&t=618s (dataquitaine Thalès)
  https://arxiv.org/abs/2311.00430 (distil-Whisper)

BIBLIOGRAPHY

- Quantization :

  https://developer.nvidia.com/blog/achieving-fp32-accuracy-for-int8-inference-using-quantization-aware-training-with-tensorrt/ (nvidia blog)

  4 QPT techniques :
  https://arxiv.org/pdf/2004.10568.pdf (adaround)

  http://proceedings.mlr.press/v97/meller19a/meller19a.pdf (equalization)

- https://arxiv.org/pdf/1906.03193.pdf (IBC)
  https://www.emc2-ai.org/assets/docs/neurips-19/emc2-neurips19-paper-11.pdf (QFT)

  For LLMs :

  https://huggingface.co/blog/merve/quantization (Hugging Face)

- PEFT :

  https://arxiv.org/abs/1902.00751 (adapters)

  https://arxiv.org/abs/2106.09685 (LoRA)