Edge Intelligence II: Model Compression

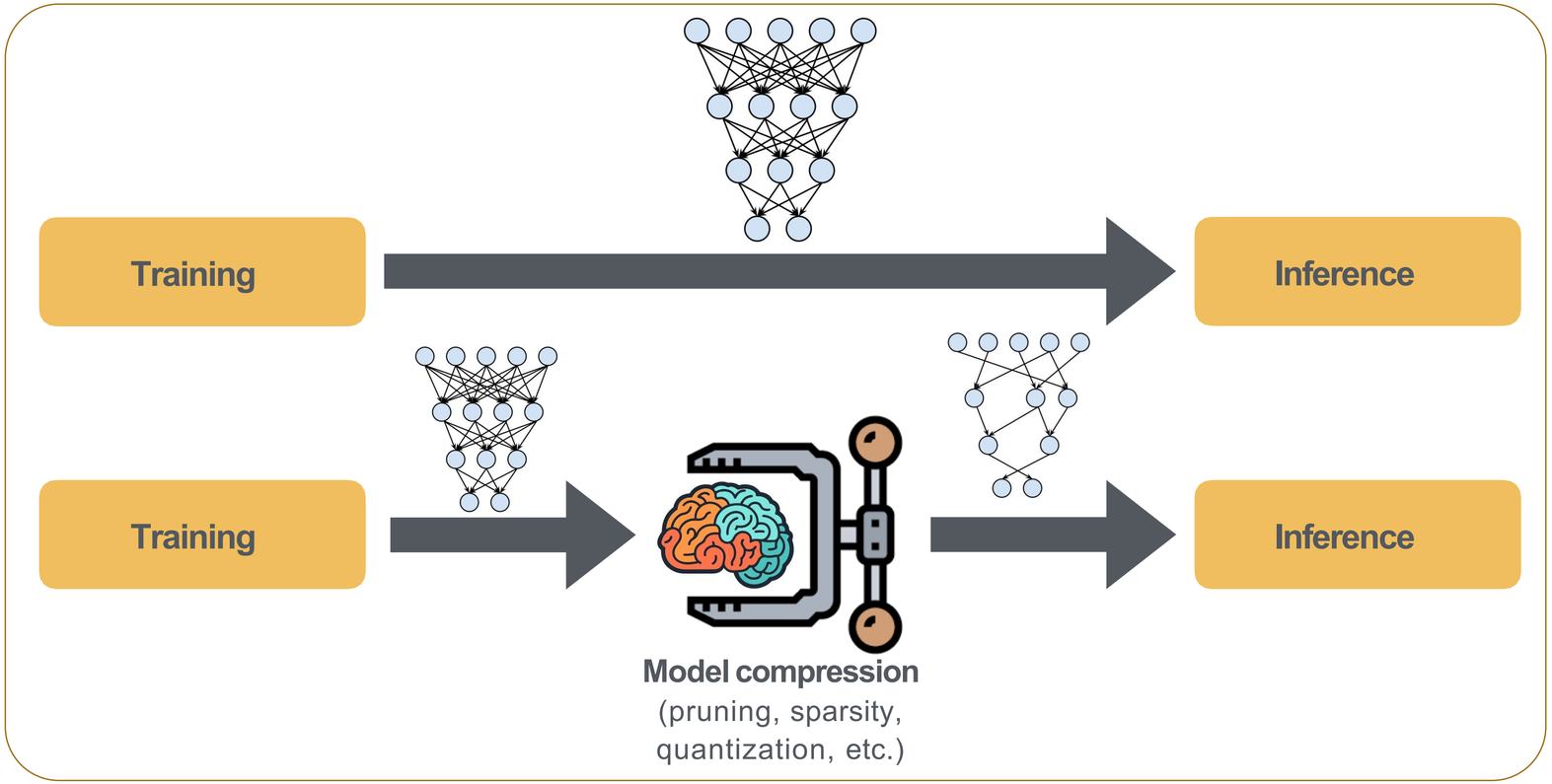
Chapter 4

Topic: Pruning | Quantization | Knowledge Distillation

Outline

- ❑ Pruning
  - Background
  - Unstructured vs. structured
  - Pattern-based & channel-level pruning
  - Industry examples
- ❑ Quantization
  - FP32 → INT8 precision
  - QAT vs. PTQ workflows
  - Granularity and SOTA techniques
- ❑ Knowledge Distillation
  - Teacher → student
  - Distill & student loss
  - Schemes
- ❑ Summary & Practice
  - Combined techniques
  - Summary
  - Project

Bridge the Gap



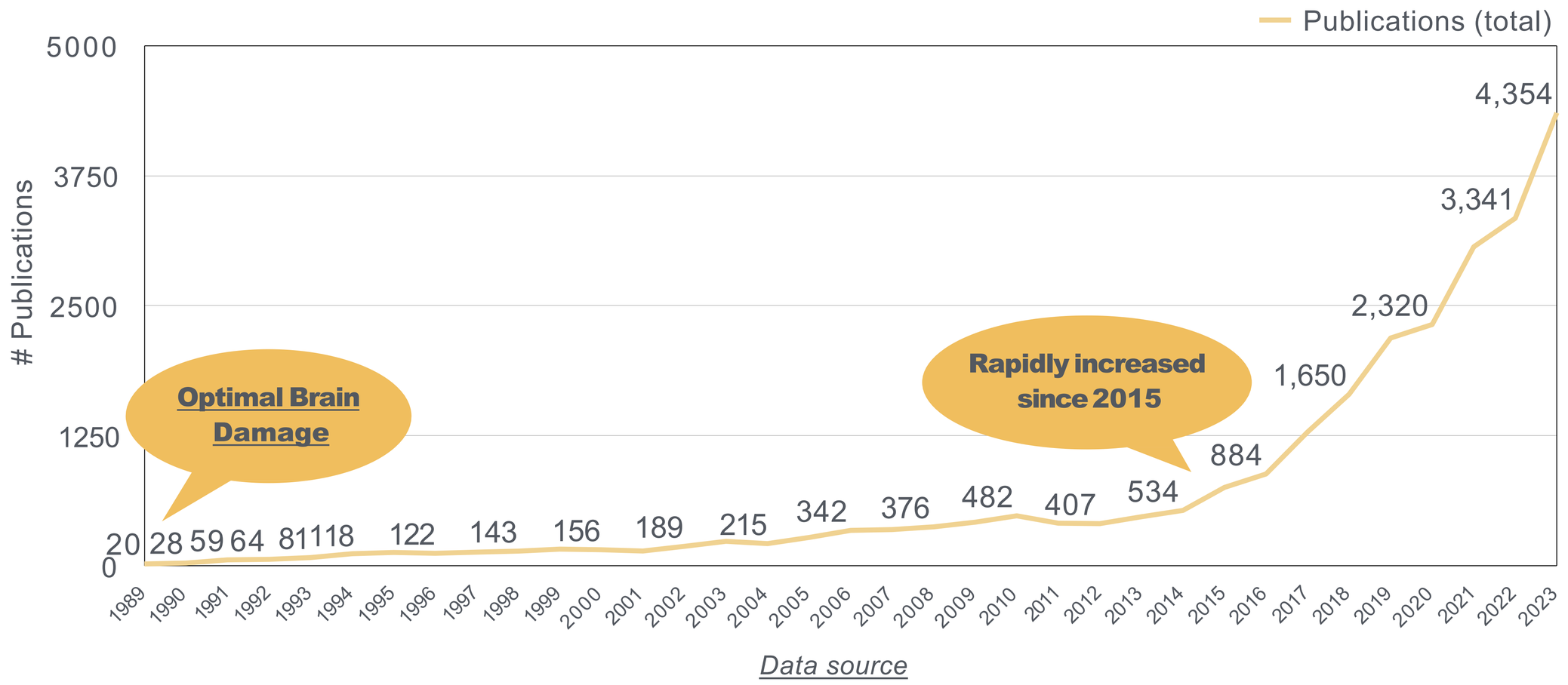
The performance gap

Deep networks are large in parameters and complex in structure — fine on data-center GPUs, but too heavy for the limited compute, memory and power of edge devices.

Compression closes the gap.

It accelerates inference and shrinks models — with low implementation cost and a natural complement to hardware acceleration.

Pruning: Background



LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. Advances in neural information processing systems, 2.

Pruning: Background

Cut the connections which is less value

➢ The classic three-steps

1 Train

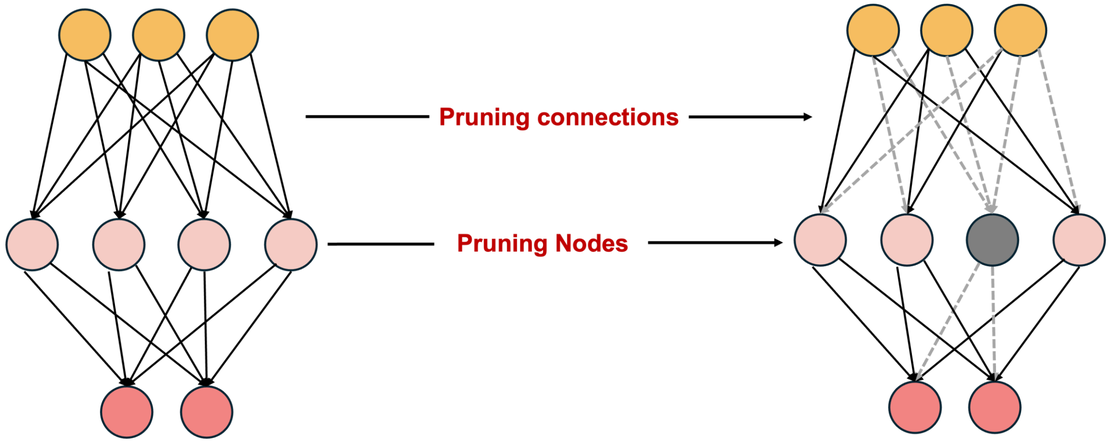
Train the network normally — weight magnitude signals importance.

2 Prune

Zero out weights below a threshold, cutting the least-important connections.

3 Fine-tune

Fine-tune the survivors to recover any lost accuracy. Repeat to balance ratio vs. accuracy.



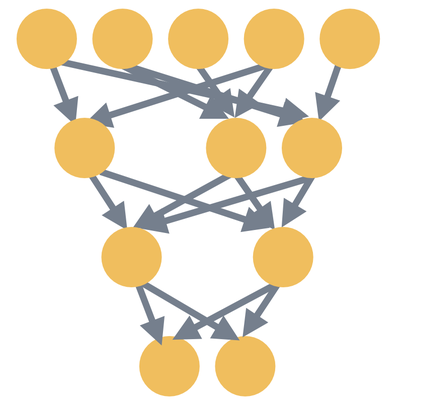
Overview of pruning

Inspired by biology.

In mammals, many synaptic connections form in infancy; the rarely used ones weaken and disappear as the brain matures — exactly what pruning does to a network (Han et al.).

Pruning: Background

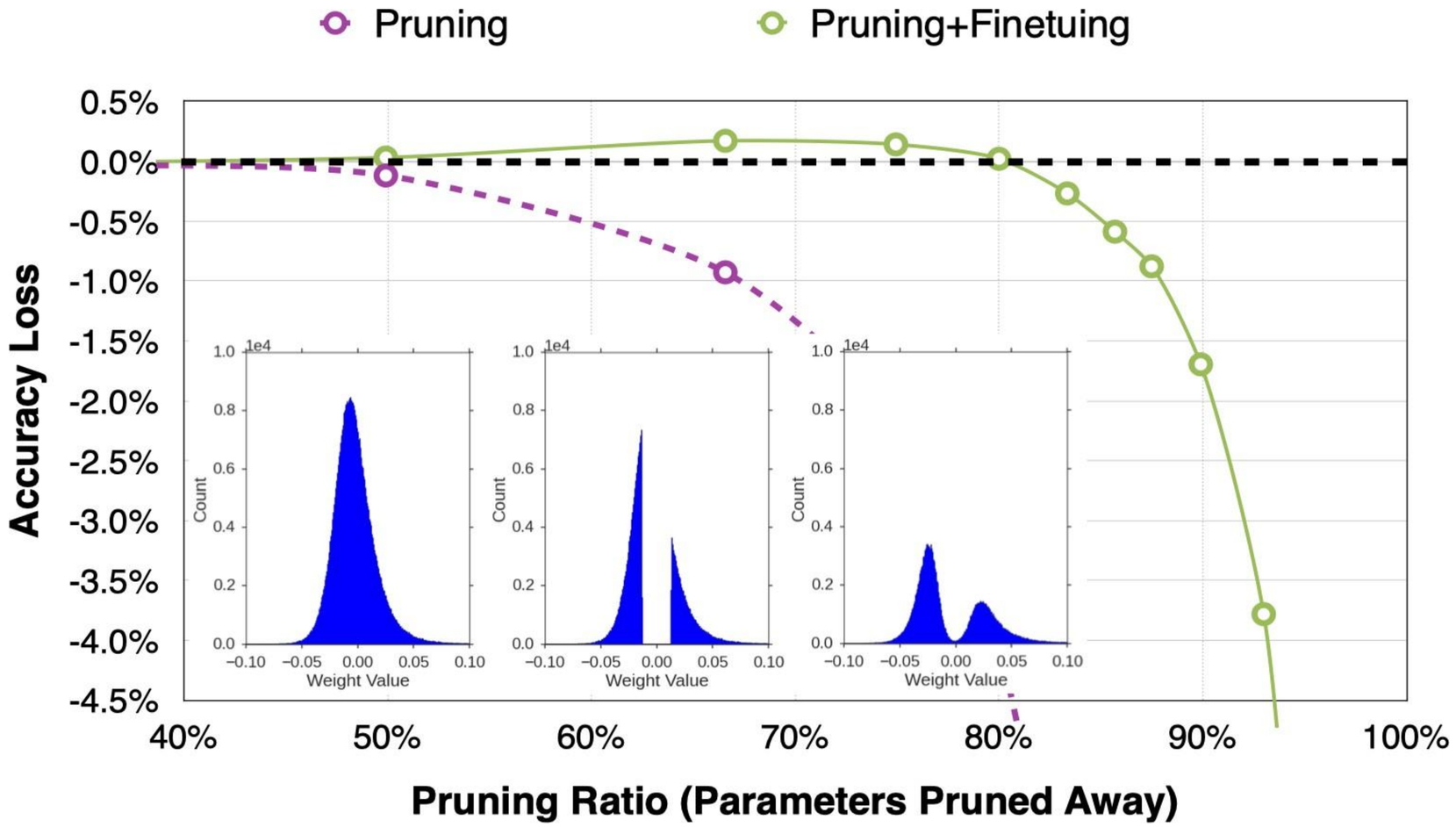
Make neural network smaller by removing synapses and neurons



Train Connectivity

Prune Connections

Train Weights



Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.

Pruning: Granularity

Unstructured vs. Structured Pruning

Unstructured

Prunes individual weights / connections

- Finest granularity — remove any redundant weight
- Highest compression flexibility
- Leaves an irregular, sparse structure
- Harder to accelerate on real hardware

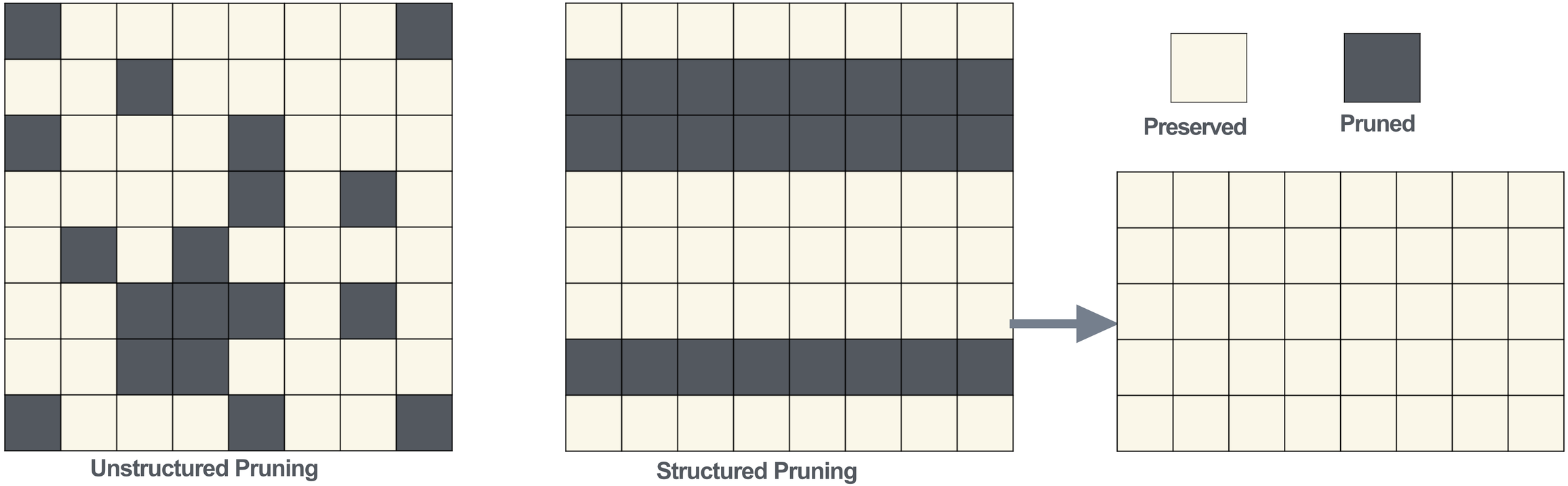
Structured

Prunes whole filters, channels or layers

- Coarser granularity — remove whole units
- Yields a smaller, regular dense network
- Real speedups on existing SW / HW
- May drop accuracy → fine-tune to restore

Pruning: Granularity

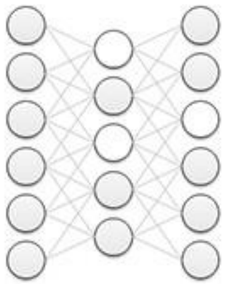
Unstructured vs. Structured Pruning

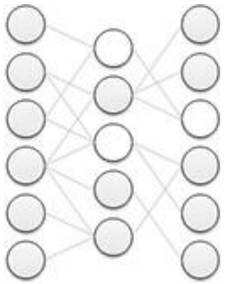


Pruning: Granularity

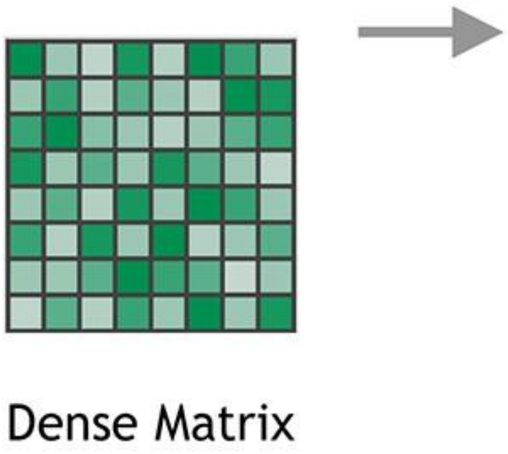
Pattern-based pruning

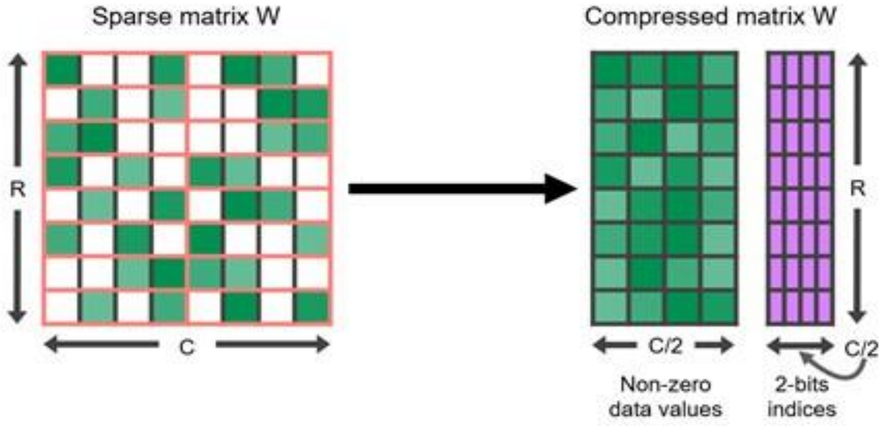
- N:M sparsity





- For each contiguous M elements, N of them is pruned.
- A classic case is 2:4 sparsity (50% sparsity)
- It is supported by NVIDIA's Ampere GPU Architecture (~2x speed up)
- Usually maintains accuracy

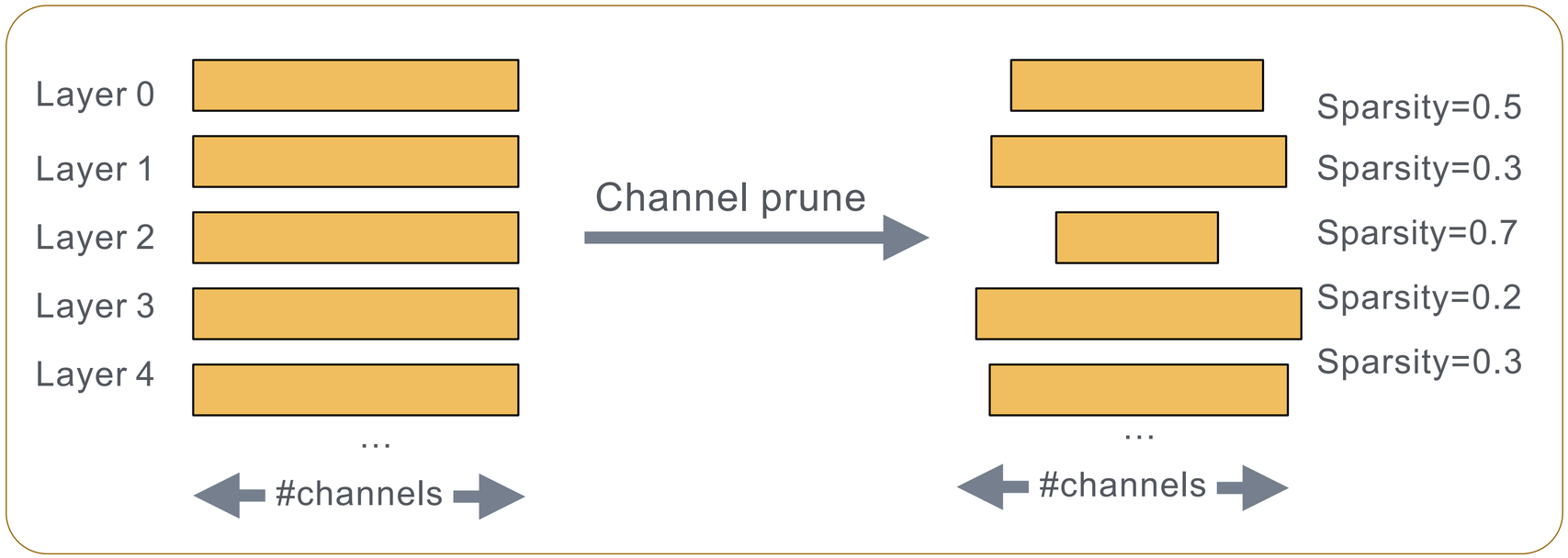


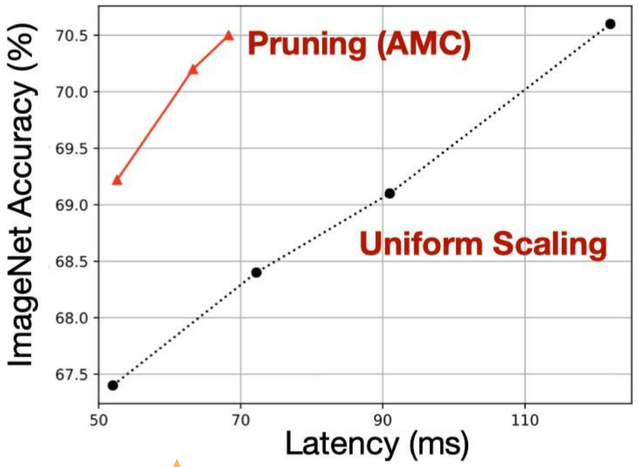


Pruning: Granularity

Channel-level pruning

- Pro: Direct speed up due to reduced channel numbers (leading to an NN with smaller #chanels)
- Con: smaller compression ratio





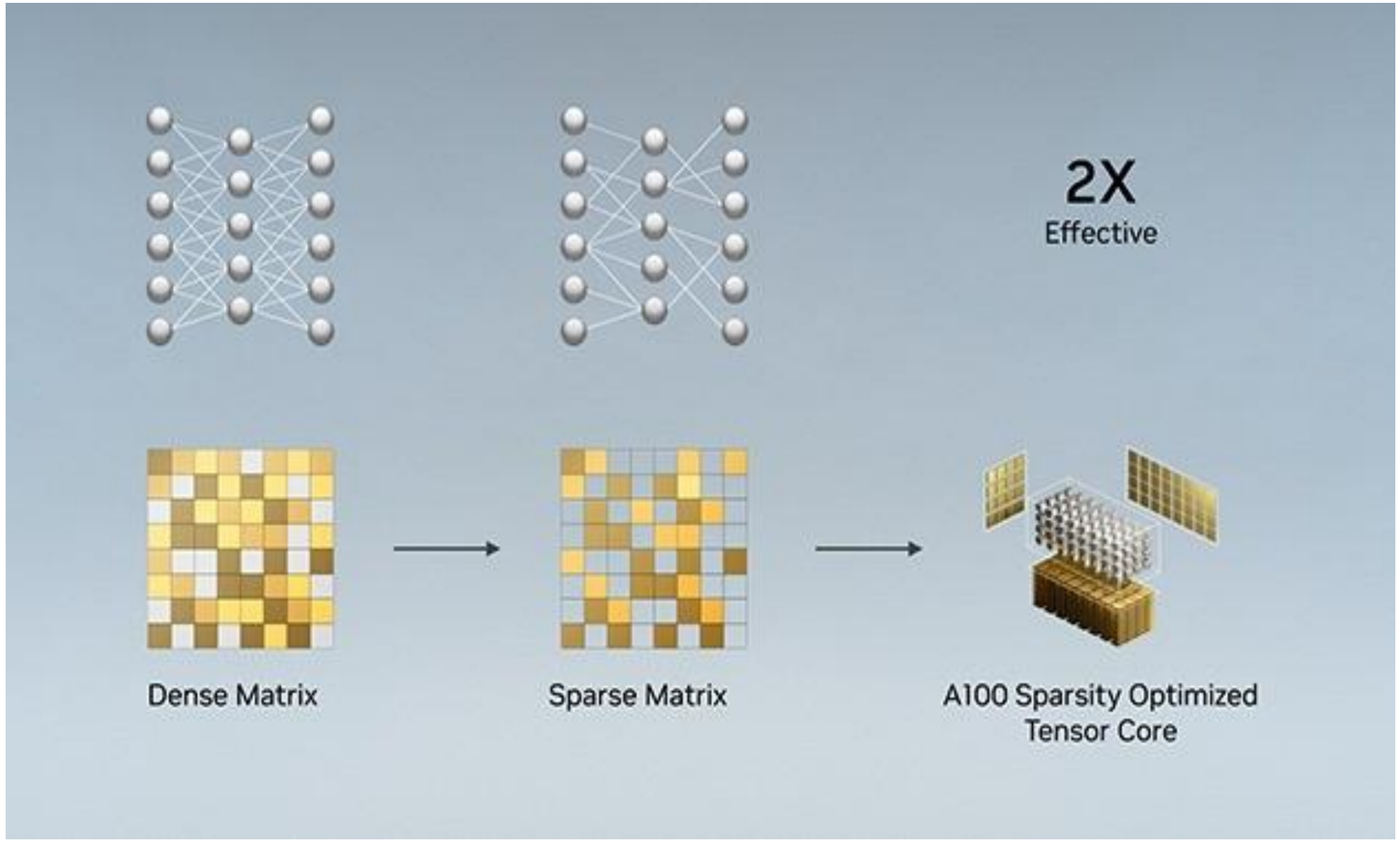
Wisely choose the pruning ratio for different layer is better than uniformly sample the different layers.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L. J., & Han, S. (2018). AMC: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV) (pp. 784-800).

Pruning: Industry Examples

Hardware support for sparsity

- 2: 4 sparsity in A100 GPU
- 2X peak performance
- 1.5X measured BERT speedup



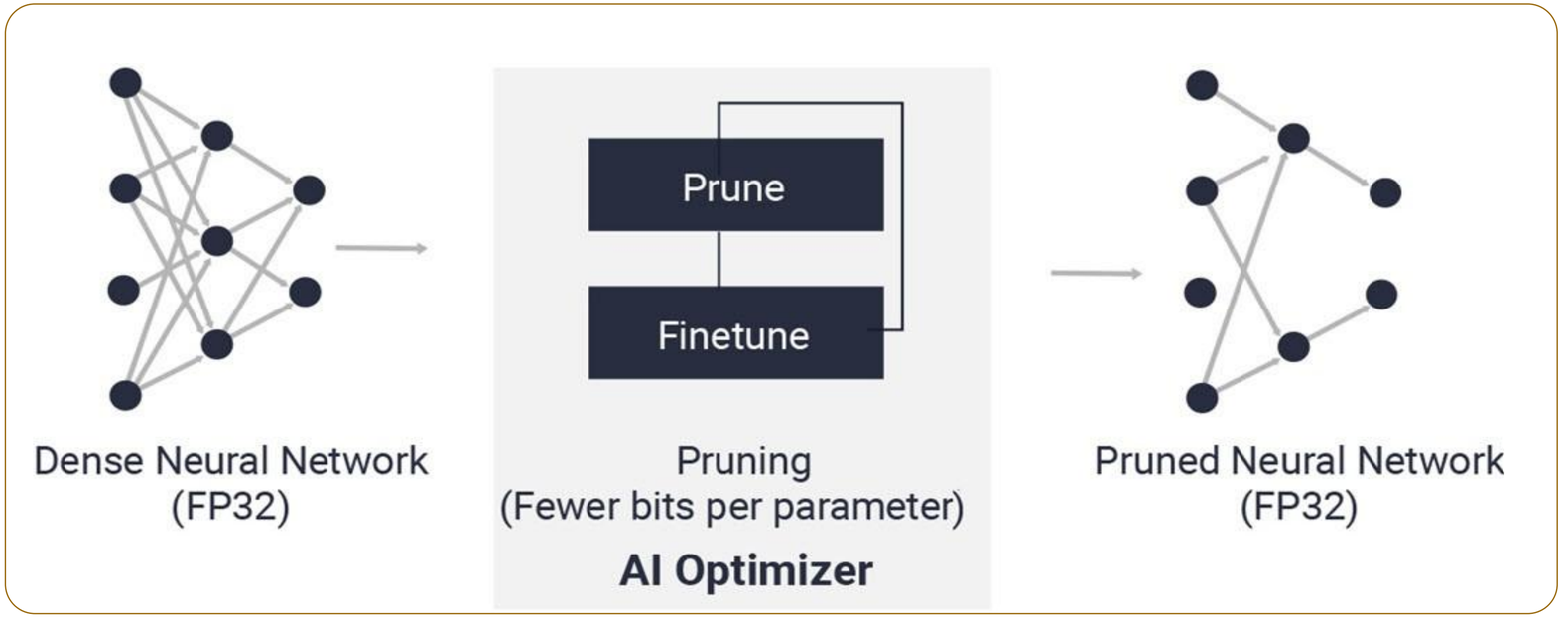
Pruning: Industry Examples

Hardware support for sparsity

- Reduce model complexity by 5X to 50X with minimal accuracy impact

XILINX VITIS™

AMD XILINX

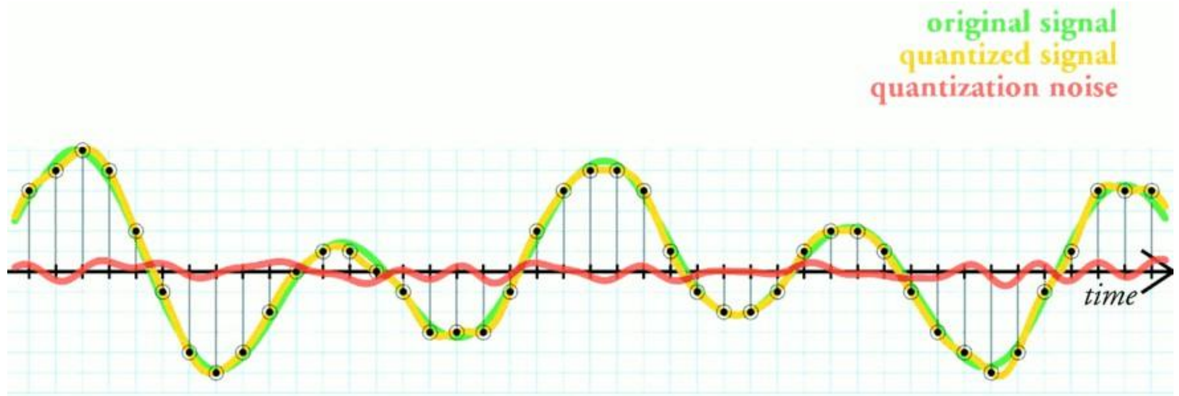


Vitis AI Optimizer

Han, S. (2017). Efficient methods and hardware for deep learning (Doctoral dissertation, Stanford University).

Quantization: Definition

Quantization is the process of constraining an input from a continuous or otherwise large set of values to a discrete set.



Quantization (Wiki)

☝ The difference between an input value and its quantized value is referred to as quantization error.

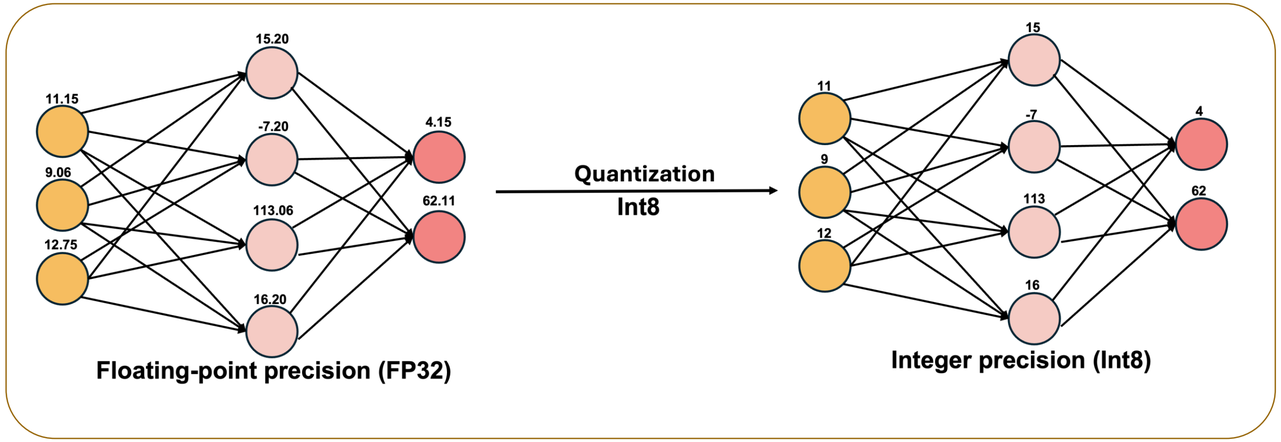
Original image vs. 16-Color image👉

Color quantization (Wiki)

Quantization: Floating-Point to Integers

The core idea

Convert floating-point storage & compute to integers. A weight that needed FP32 is represented as INT8 after quantization — roughly 4× smaller.



Overview of quantization

- High precision matters in training (subtle gradients) — not at inference, where weights are fixed.
- Networks are robust to noise; quantization is just a controlled form of noise.
- Many parameters simply don't need full precision.

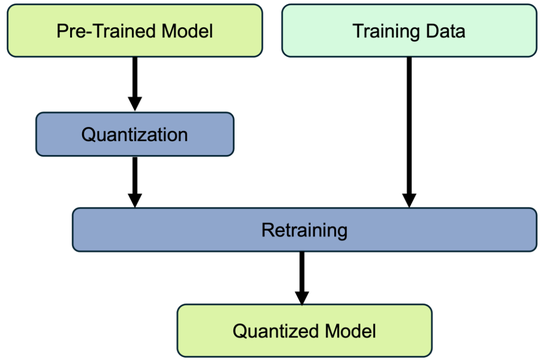
FP32 → INT8 cuts model size ~4×, memory bandwidth and energy — the default in industry.

Quantization: Mapping Schemes

Quantization-aware training vs. post-training quantization

Quantization-Aware Training

- Quantize during training's forward pass
- Backward pass stays FP32 for gradients
- Model learns to cope with low precision
- Slower, but best accuracy under constraints

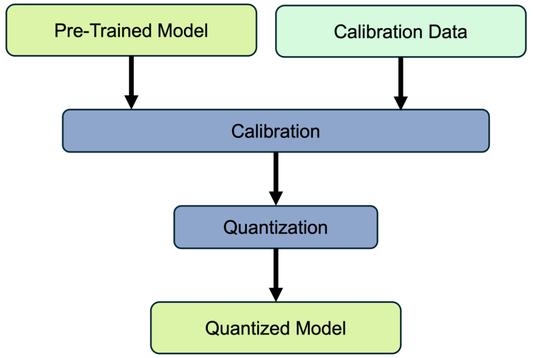


QAT

Best when: accuracy is critical and training data is available.

Post-Training Quantization

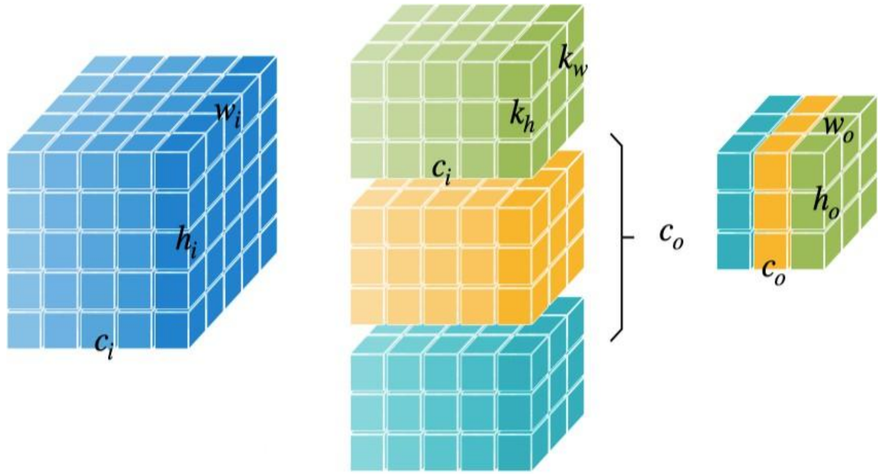
- Applied after the model is fully trained
- Calibrates on a small dataset — no retraining
- Faster and simpler to apply
- Ideal when data is limited or unavailable

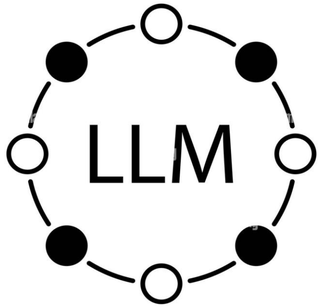


PTQ

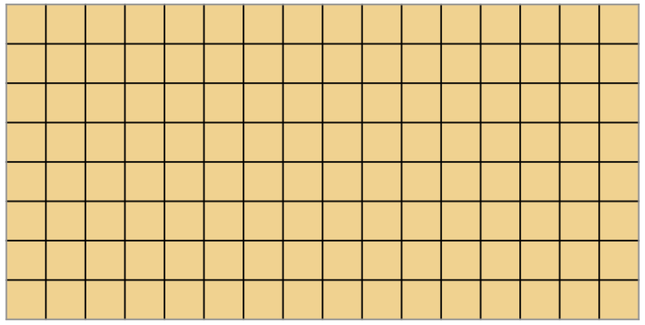
Best when: training data is limited & fast deployment is required.

Quantization: Granularity



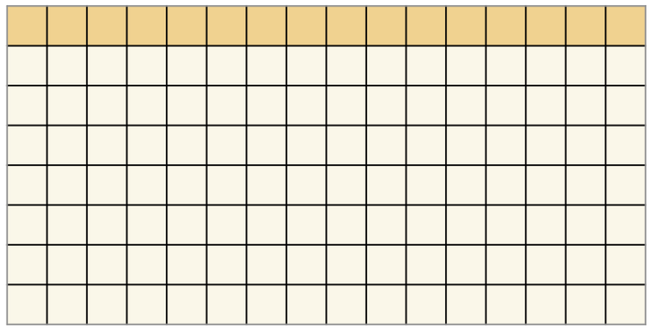


Per-Tensor Quantization



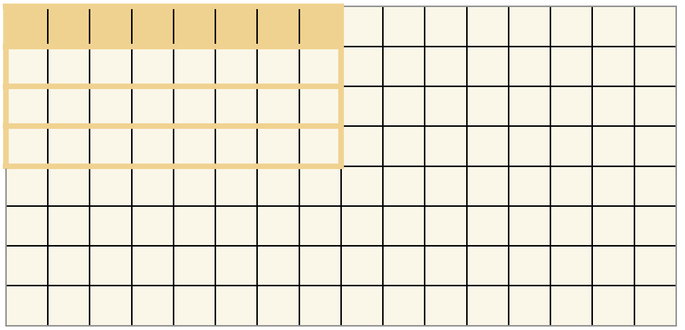
One scaling factor to scale the whole tensor

Per-Channel Quantization



One scaling factor to scale one channel

Group Quantization



Multi-level scaling scheme

Quantization: State-of-the-art Techniques

BinaryConnect

Courbariaux et al. — restrict weights to −1 / +1, simplifying deep-learning hardware.

k-means quantization

Gong et al. — cluster FC-layer weights: up to 24× compression, ~1% accuracy loss on ImageNet.

Q-ViT | APQ-ViT

Bring QAT & accurate PTQ to vision transformers (info rectification, block-wise calibration).

Mixed-precision

Tang et al. — learnable scale factors pick per-layer bit-widths: high bits for sensitive layers.

Q-BERT

First to quantize BERT — group-wise, Hessian-based mixed precision.

SmoothQuant | AWQ

MIT Han Lab — 8-bit LLMs by smoothing outliers; AWQ preserves 1% salient weights (Jetson Orin, Pi4).

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "BinaryConnect: Training deep neural networks with binary weights during propagations". In: Advances in Neural Information Processing Systems 28 (2015).

Yunchao Gong et al. "Compressing deep convolutional networks using vector quantization". In: arXiv preprint arXiv:1412.6115 (2014).

Yanjing Li et al. "Q-ViT: Accurate and fully quantized low-bit vision transformer". In: Advances in Neural Information Processing Systems 35 (2022), pp. 34451–34463.

Yifu Ding et al. "Towards accurate post-training quantization for vision transformer". In: Proceedings of the 30th ACM International Conference on Multimedia. 2022, pp. 5380–5388.

Chen Tang et al. "Mixed-precision neural network quantization via learned layer-wise importance". In: European Conference on Computer Vision. Springer. 2022, pp. 259–275.

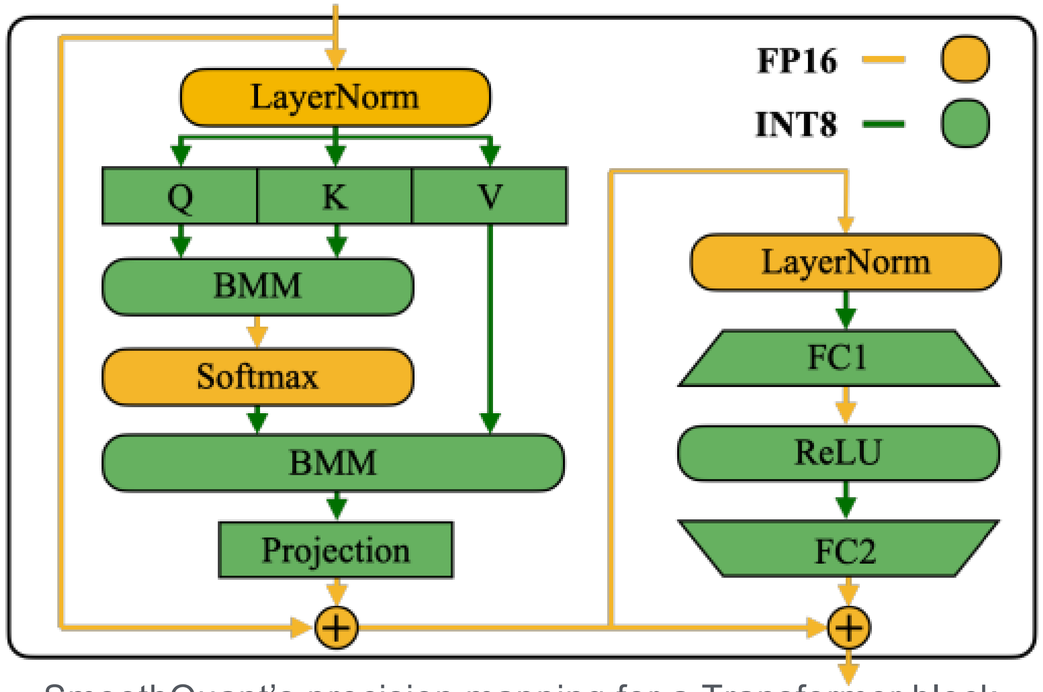
Sheng Shen et al. "Q-BERT: Hessian based ultra low precision quantization of BERT". In: Proceedings of the AAAI Conference on Artificial Intelligence 34.05 (2020), pp. 8815–8821.

Guangxuan Xiao et al. "SmoothQuant: Accurate and efficient post-training quantization for large language models". In: International Conference on Machine Learning. PMLR. 2023, pp. 38087–38099.

Ji Lin et al. "AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration". In: Proceedings of Machine Learning and Systems 6 (2024), pp. 87–100.

Quantization: SmoothQuant

Core concept. In LLMs the hard part of PTQ is the activations, not the weights. A mathematically equivalent transformation migrates the difficulty from outlier-heavy activations onto the smooth weights — enabling 8-bit weights & activations (W8A8) with no retraining.



SmoothQuant’s precision mapping for a Transformer block.

The problem

Activation outliers in a few channels stretch the quant range and wreck 8-bit accuracy.

The fix

Scale outlier activations down, weights up by the same s — product unchanged.

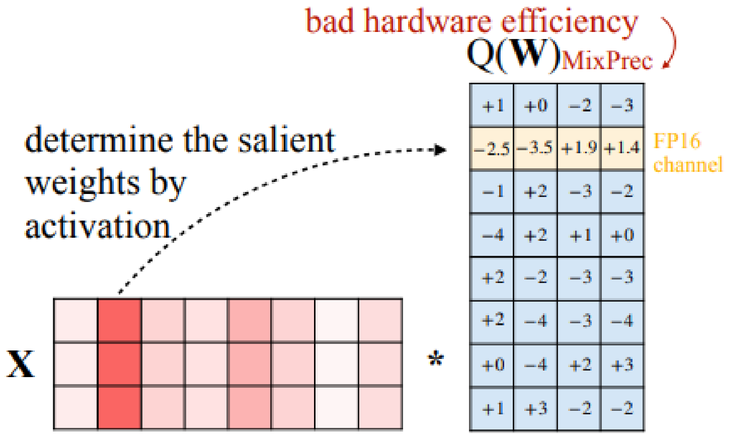
Result

Training-free W8A8 up to 175B+ params, with real latency & memory gains.

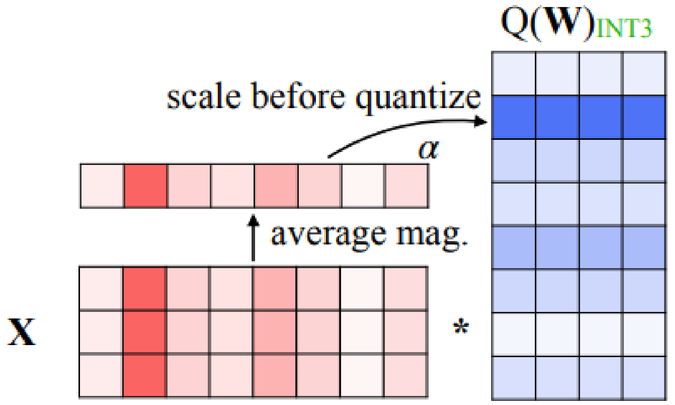
Guangxuan Xiao et al. "SmoothQuant: Accurate and efficient post-training quantization for large language models". In: International Conference on Machine Learning. PMLR. 2023, pp. 38087–38099.

Quantization: AWQ

Core concept. Activation-aware Weight Quantization extends SmoothQuant (MIT Han Lab) toward low-bit, weight-only LLM compression. It preserves the ~1% of salient weights — identified by the activation distribution while quantizing the rest to a uniform low bit-width.



(a) Keep 1% salient weights in FP16 (PPL 13.0)



(b) Scale the weights before quantization (PPL 13.0)

AWQ performs per-channel scaling to protect the salient weights and reduce quantization error.

Salient by activation

Weights tied to big activation channels matter most — protect that 1%.

Uniform low-bit

Per-channel scaling protects salient weights; everything quantized to 4-bit.

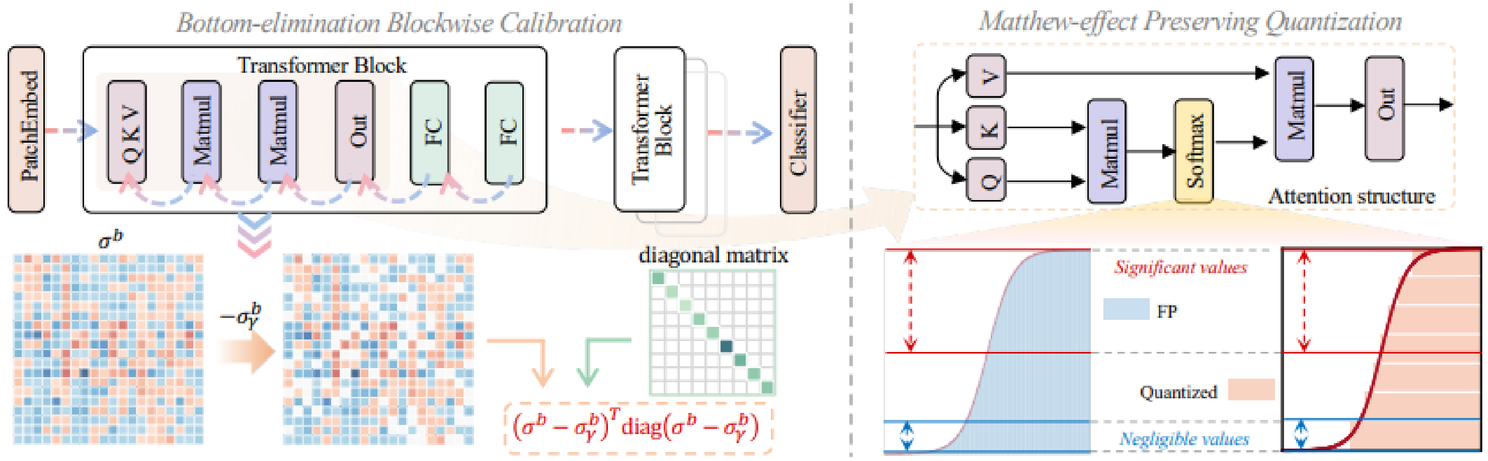
Built for edge

Runs on Jetson Orin & Raspberry Pi 4; used in TinyChat (course project).

Ji Lin et al. "AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration". In: Proceedings of Machine Learning and Systems 6 (2024), pp. 87–100.

Quantization: APQViT

Core concept. An accurate PTQ framework for vision transformers. It adds unified block-wise calibration (optimizing error block-by-block) and a Matthew-Effect Preserving Quantization for the softmax, so the skewed attention ranking survives low-bit quantization — no retraining.



Overview of APQ-ViT.

Block-wise calibration

Calibrate per transformer block, capturing accumulated error across attention + MLP.

Softmax preserved

Keeps the “rich-get-richer” attention ordering that naive low-bit quant flattens.

Accurate PTQ

Quantizes ViTs accurately without any retraining step.

Yifu Ding et al. "Towards accurate post-training quantization for vision transformer". In: Proceedings of the 30th ACM International Conference on Multimedia. 2022, pp. 5380–5388.

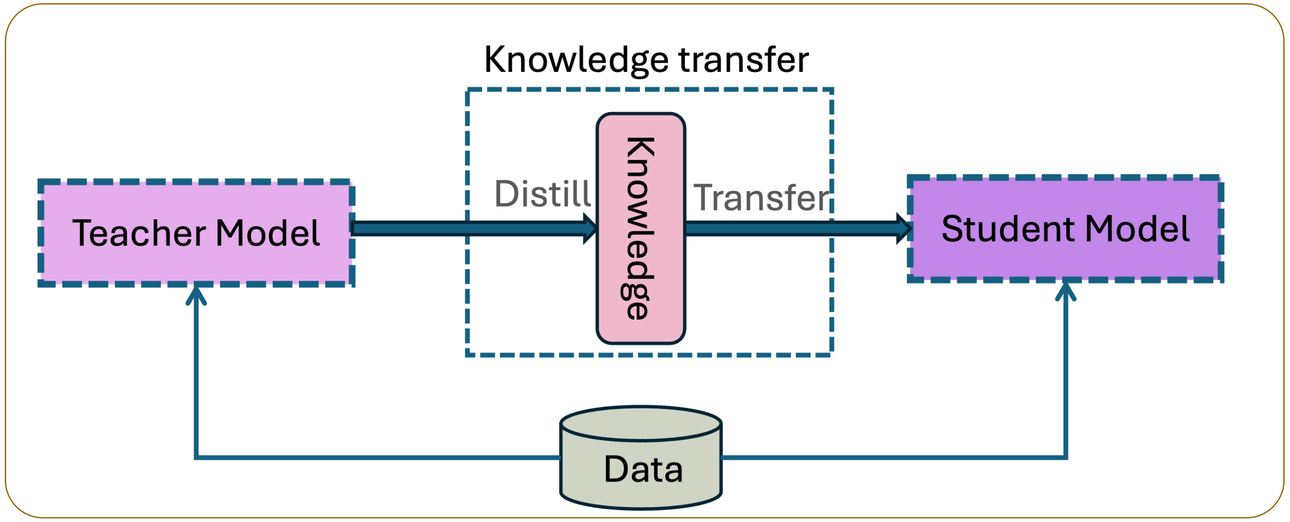
Knowledge Distillation

Teacher teaches student

1 **Train the teacher**
A large, accurate network is trained, then frozen.

2 **Transfer knowledge**
Its soft outputs + the true labels train a smaller student.

3 **Two-loss objective**
Distill Loss (teacher soft targets) + Student Loss (ground-truth labels).

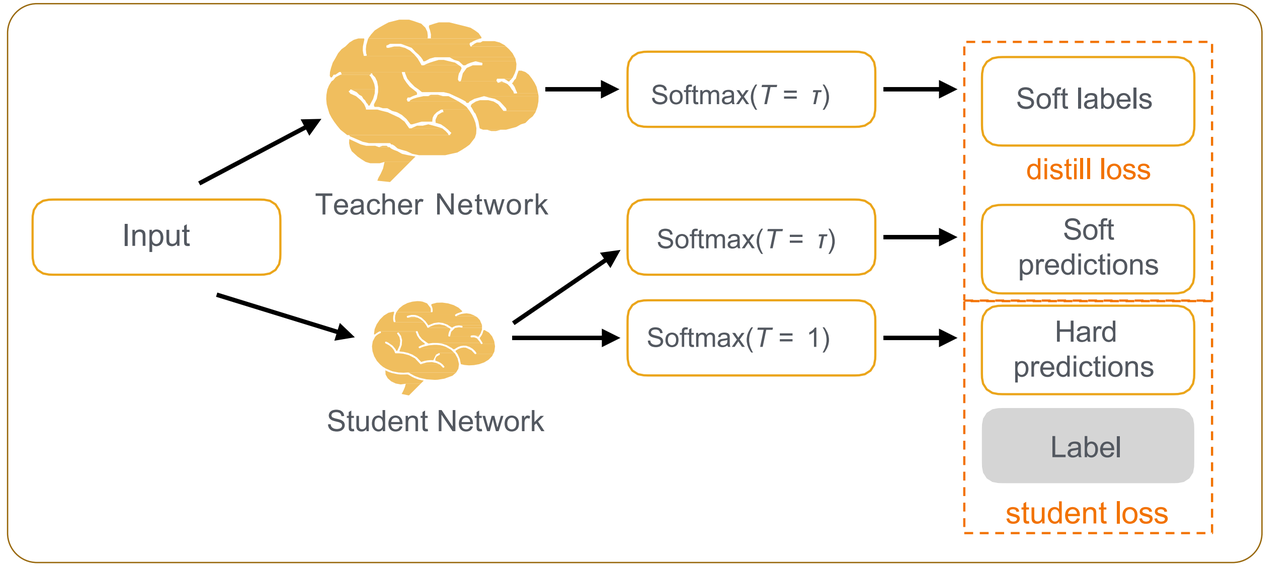


Overview of Knowledge Distillation (KD).

The student rivals the teacher at a fraction of the size — and can even match an ensemble.

Knowledge Distillation: Distill Output Logits

The simplest and widely used



Cross entropy loss

$$\mathbb{E}(-p_t \log p_s)$$

L2 loss

$$\mathbb{E}(\| p_t - p_s \|_2^2)$$

- The overall loss function, including distillation and student loss is:

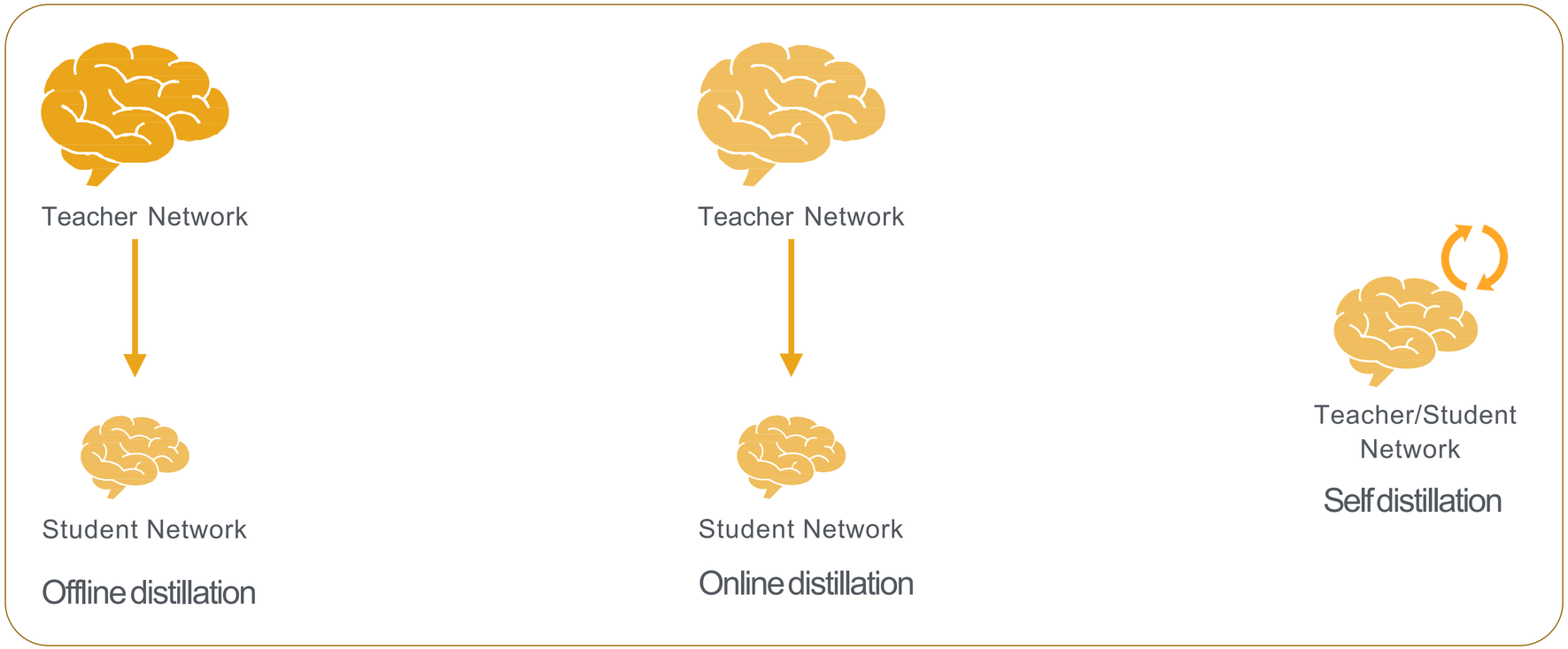
$L(x; \theta) = \alpha * L_{ce}(y, \sigma(z_s; T = 1)) + L_{ce}(\sigma(z_t; T = \tau), \sigma(z_s; T = \tau))$

  - Where x is the input, θ is the student network parameter, L_{ce} is the cross-entropy loss function, Y is the ground truth label, σ is the softmax function parameterized by the temperature T(controls probability distribution). α is the coefficient. z_s, z_t are the logits of the students and teacher respectively.
  - In other words, the loss function is a weighted average of cross-entropy loss and KL divergence.

Hinton, G. (2015). Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531.

Ba, J., & Caruana, R. (2014). Do deep nets really need to be deep?. Advances in neural information processing systems, 27.

Knowledge Distillation: Schemes



Distillation Schemes

Combined Techniques

Each method has trade-offs — so researchers combine them. Distillation is the common glue: it regularizes the student against the shocks of pruning and quantization.

Deep Compression

Han et al. — prune + quantize + Huffman coding, together slashing storage.

Prune + Distill

Aghli & Ribeiro — selective weight pruning, then KD on the un-pruned layers; handles ResNet dependencies.

Unified ViT

Yu et al. — layer pruning + layer skipping + KD in one budget-constrained objective; ViT shrinks to 50% FLOPs.

Bayesian Bits

Qualcomm — joint mixed-precision quantization + pruning via learnable stochastic gates.

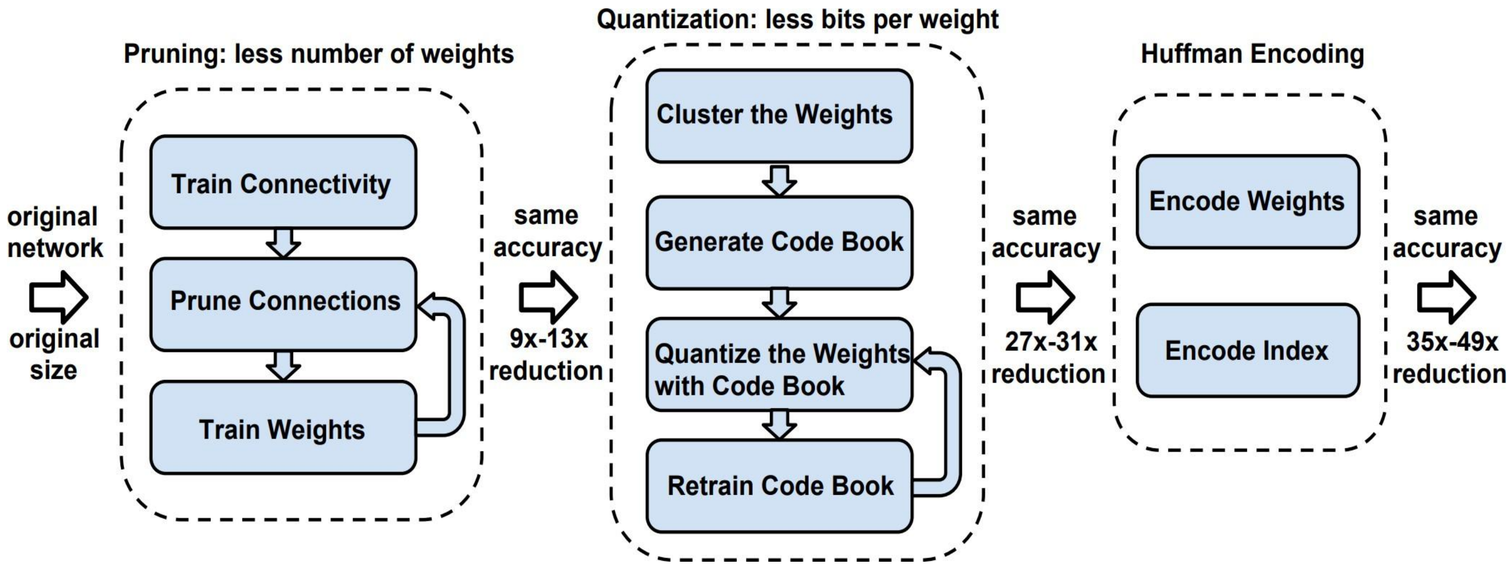
Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding". In: arXiv preprint arXiv:1510.00149 (2015).

Nima Aghli and Eraldo Ribeiro. "Combining weight pruning and knowledge distillation for CNN compression". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, pp. 3191–3198.

Andrey Kuzmin et al. "Pruning vs quantization: Which is better?" In: Advances in Neural Information Processing Systems 36 (2024).

Combined Techniques

Deep Compression (prune + quantize + Huffman coding)



Overview of deep compression

No accuracy loss

On AlexNet & VGG-16, the pipeline compressed 35–49× with no drop in ImageNet accuracy — the headline result.

Faster

Fewer weights and cheaper memory access mean lower latency and energy — ideal for edge inference.

Template

It set the pattern for combined compression: prune, then quantize, then code — reused across CNNs, ViTs.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

Summary

- Deep models are too big for edge devices; compression shrinks them and speeds up inference at low cost.
- Pruning removes redundant weights/filters, then retrain — structured pruning gives real hardware speedups.
- Quantization lowers numeric precision from FP32 to INT8 for a roughly 4× smaller model, using QAT when accuracy matters and PTQ when speed does.
- Knowledge distillation lets a frozen teacher train a compact student that keeps most of the accuracy.
- Combining techniques compounds the gains, as in Deep Compression which prunes, quantizes, and codes together as the modern default.

Project

Using model compression techniques, optimizing large language models (LLMs) on edge devices (e.g., your laptop). A good example can be found at GitHub: https://github.com/mit-han-lab/tinychat-tutorial?tab=readme-ov-file.

➢ Tasks

1. **Set up TinyChat** Clone the tutorial repo and install the toolchain on your laptop.
2. **Quantize with AWQ** Apply Activation-aware Weight Quantization — protect the salient 1% of weights, pack the rest to 4-bit.
3. **Deploy & run inference** Load the quantized model and chat with it locally — no GPU cluster, no cloud.
4. **Measure & compare** Benchmark model size, memory, latency and output quality vs. the FP16 baseline.

➢ Expected outcomes

- **~4× smaller** 4-bit weights shrink the model to roughly a quarter of its FP16 footprint.
- **Fits in laptop RAM** A quantized 7B-class model runs in the memory of a typical laptop.
- **Interactive latency** Lower precision + smaller memory traffic give usable token throughput on CPU/edge GPU.
- **Minimal quality loss** AWQ's salient-weight protection keeps generation quality close to the baseline.

Example: quantize Llama-2-7B with AWQ to 4-bit, then run TinyChat locally. Resource: **https://github.com/mit-han-lab/tinychat-tutorial**

References

- LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. Advances in neural information processing systems, 2.
- Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 28.
- Han, S. (2017). Efficient methods and hardware for deep learning (Doctoral dissertation, Stanford University).
- Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "BinaryConnect: Training deep neural networks with binary weights during propagations". In: Advances in Neural Information Processing Systems 28 (2015).
- Yunchao Gong et al. "Compressing deep convolutional networks using vector quantization". In: arXiv preprint arXiv:1412.6115 (2014).
- Yanjing Li et al. "Q-ViT: Accurate and fully quantized low-bit vision transformer". In: Advances in Neural Information Processing Systems 35 (2022), pp. 34451–34463.
- Yifu Ding et al. "Towards accurate post-training quantization for vision transformer". In: Proceedings of the 30th ACM International Conference on Multimedia. 2022, pp. 5380–5388.
- Chen Tang et al. "Mixed-precision neural network quantization via learned layer-wise importance". In: European Conference on Computer Vision. Springer. 2022, pp. 259–275.
- Sheng Shen et al. "Q-BERT: Hessian based ultra low precision quantization of BERT". In: Proceedings of the AAAI Conference on Artificial Intelligence 34.05 (2020), pp. 8815–8821.
- Guangxuan Xiao et al. "SmoothQuant: Accurate and efficient post-training quantization for large language models". In: International Conference on Machine Learning. PMLR. 2023, pp. 38087–38099.
- Ji Lin et al. "AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration". In: Proceedings of Machine Learning and Systems 6 (2024), pp. 87–100.
- Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression". In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2006, pp. 535–541.
- Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. "Net2net: Accelerating learning via knowledge transfer". In: arXiv preprint arXiv:1511.05641 (2015).
- Shangquan Sun et al. "Logit standardization in knowledge distillation". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2024, pp. 15731–15740.
- Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding". In: arXiv preprint arXiv:1510.00149 (2015).
- Nima Aghli and Eraldo Ribeiro. "Combining weight pruning and knowledge distillation for CNN compression". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, pp. 3191–3198.
- Andrey Kuzmin et al. "Pruning vs quantization: Which is better?" In: Advances in Neural Information Processing Systems 36 (2024).