Stanford University

CS 329P : Practical Machine Learning (2021 Fall)

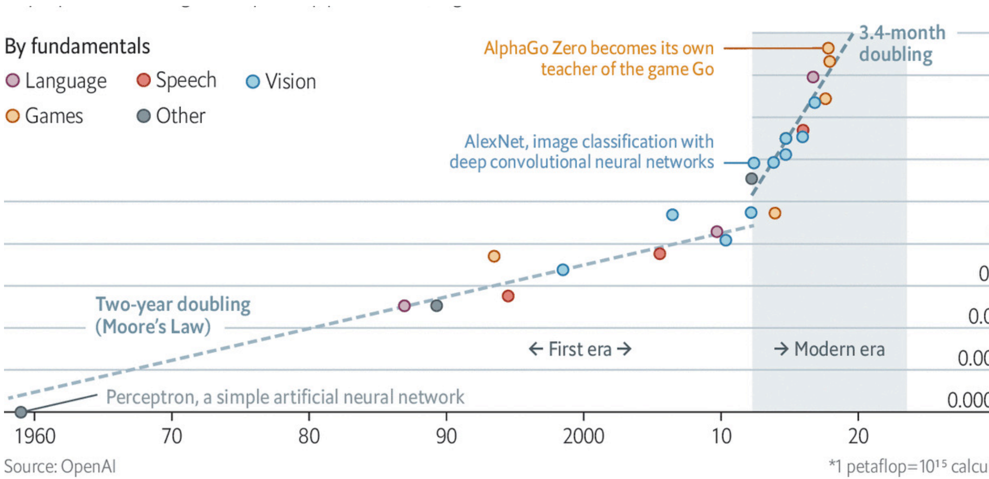
12.1 Model Compression

Qingqing Huang, Mu Li, Alex Smola

https://c.d2l.ai/stanford-cs329p

Models are Bigger and Bigger

- Universal approximation theorem: MLP with a single hidden layer can approximate any continuous function
- Neural networks are often overparameterized
  - A larger model is easier to learn and generalizes well thanks to both SGD and model structure
  - (Theory is still under developing)
- Model combination ensembles multiple models



Challenges in Model Deploying

- Deploying in production:
  - Memory: often share memory with others
  - Latency: some requires realtime (ads, live captioning/translation, self-driving)
  - Cost: power machines are more expensive
  - Energy: both computation and accessing memory needs a significant amount energy, especially for devices powered by batteries
- Deploying big models is hard
  - By far little cases deploy the full multi-billion transformers models

Model Compression

- Reduce model size/computation cost without (significantly) hurt predictive performance
- **Pruning:** set some weight elements to 0 to avoid storing and computing
- **Quantization:** reduce the number of bits for each weight (e.g. float32 → int8)
- **Knowledge distillation:** transfer knowledge from teacher models to smaller student models (cover in next topic)

Pruning

- High-level algorithm for neural networks:
  - Train a network to converge
  - Assign each weight element a score
  - Set some elements to 0 based on scores
  - Fine-tune the model to increase accuracy

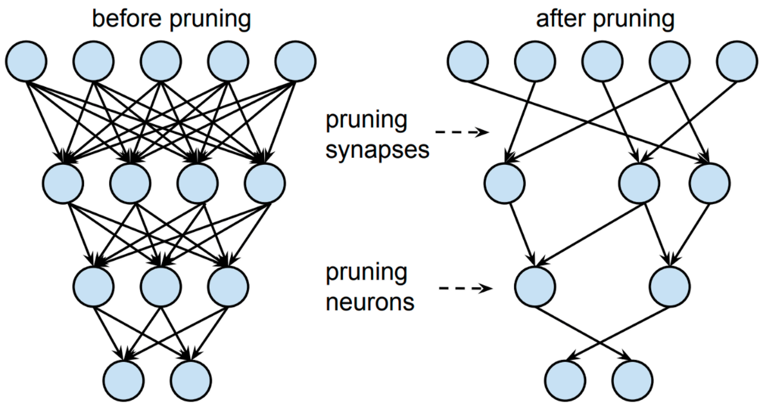
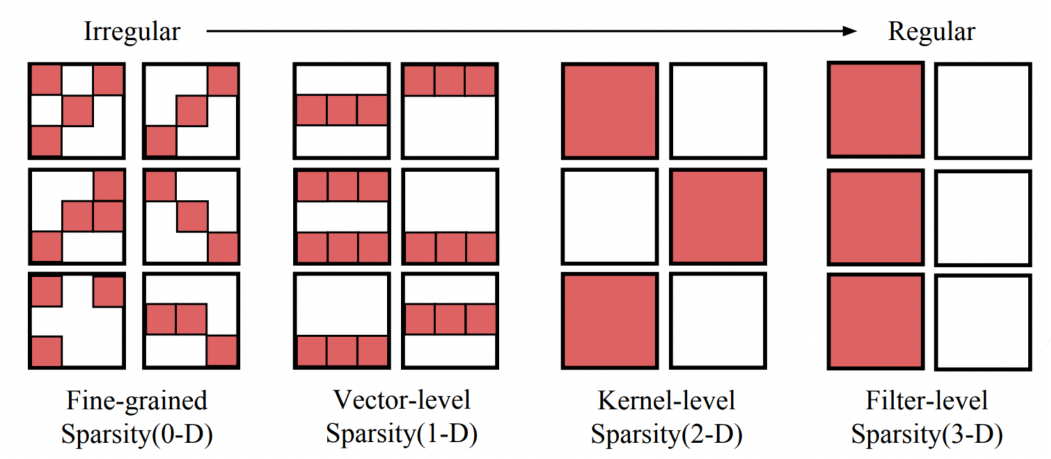


Image credit: Rohit Bandaru

Pruning

- Unstructured pruning: set individual weights into 0
  - Leads to sparse matrices/tensors that are often less efficient to compute
- Structured pruning: set whole unit/channel/block to 0



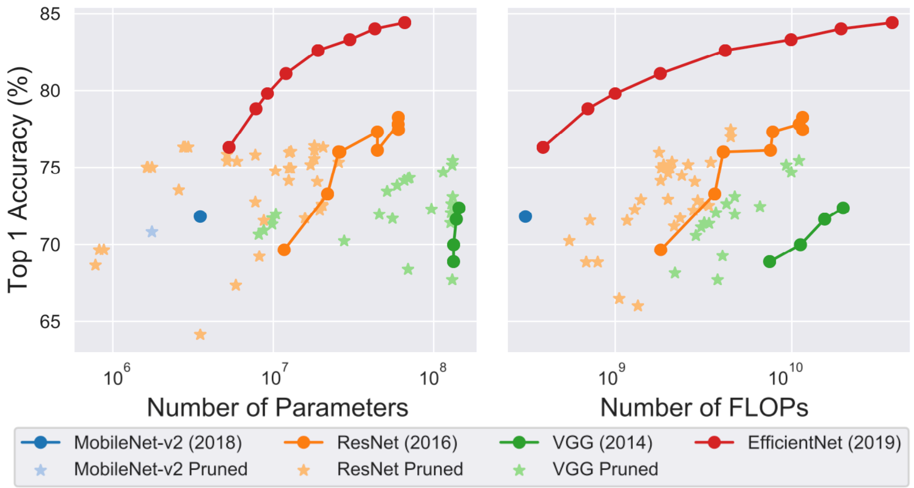
Mao et.al. NeurIPS'17

Pruning

- Score: absolute values, contributions to activations/gradients
  - Compare scores locally (within a layer) or globally to determine elements to be pruned
- Scheduling: prune all elements at once, or prune a fraction iteratively
- Fine-tuning: randomly initialize, re-use weights from trained network

Pruning - Results

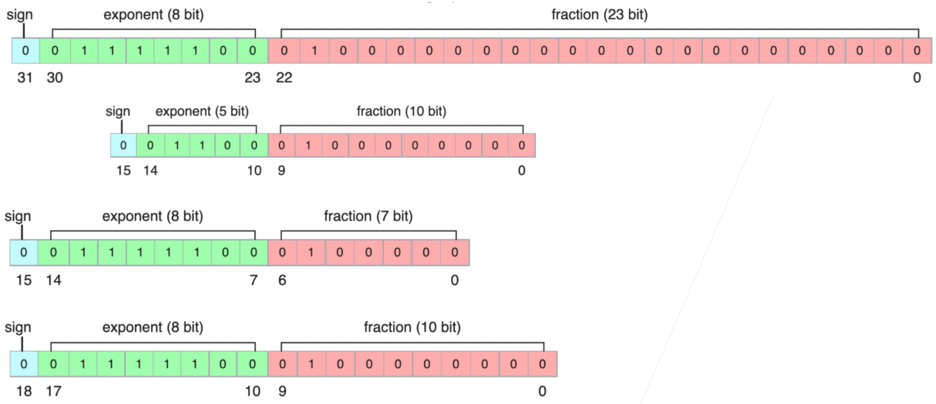
- Pruning algorithms often outperform random sparsification
- Increasing the size of the network then pruning may outperform training it normally
- Pruning does not help as much as switching to a better architecture



[Blalock et.al. MLSys'20](#)

Data Types

- Modern hardware offer better performance on low-bit operations
- E.g. TFLOPS/TOPS on Nvidia A100
  - 19.5 for IEEE float32
  - 312 for IEEE float16
  - 312 for Bfloat16
  - 156 for TF32
  - 624 for INT8, 1248 for INT4

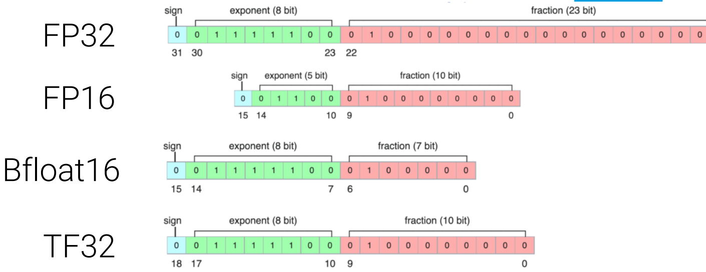


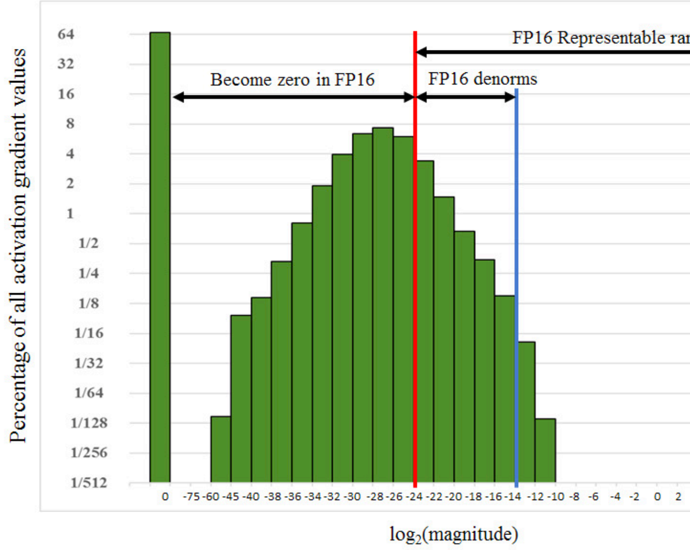
Quantization

- Using low-bit numbers to accelerate training/inference
  - Less memory, leverage special hardware
  - Not hurts accuracy except for using extremely low-bit (1-bit, 2-bit)
- Often using floats for training and integers for inference
  - Gradients need a larger dynamic range
- Only quantize key layers (e.g. conv/dense) while others (e.g. activation, weight updating) are still use the default data type (e.g. float32)

Low-bit Floating Numbers

- Casting FP32 to low-bit floating is straightforward
  - Trim fraction/exponent parts
- FP16 has less exponent bits than others
  - Often rescale the loss through $\lambda \operatorname{loss}(\hat{y}, y)$, with tunable scale $\lambda > 0$
  - So activations/gradients are increased by λ times to avoid numbers near 0 are represented by 0 in FP16

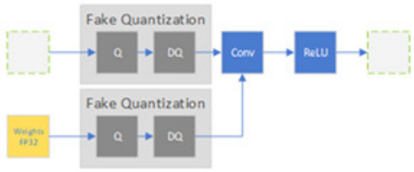


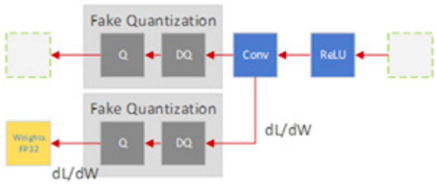


Histogram of gradients (Nvidia)

Integer Quantization

- The simplest way to quantize floats into integers:
  - $XY \approx \sigma_x \sigma_y \underbrace{\mathrm{clip}(\mathrm{round}(X/\sigma_x))\ \mathrm{clip}(\mathrm{round}(Y/\sigma_y))}_{\text{Integer matrix multiplication}}$
  - Scales σ_x (σ_y) are calculated from X (Y)
- Directly quantizing the trained weights may decrease accuracy
- Quantization-aware training
  - Performs clip/round during training, but keeps float32





Summary

- Compress large models for efficient deployment without scarifying (too much) predictive performance
- Common technologies are pruning (set some weights to 0), quantization (low-bit numbers) and knowledge distillation