Introduction to Optimizing ML Models for the Edge

Kumaran Ponnambalam

Principal Engineer - AI

Cisco Systems, Emerging Tech & Incubation

Agenda

- Deploying deep learning models at the edge
- Model compression techniques
  - Quantization
  - Pruning
  - Low rank approximation
  - Knowledge distillation
- Leveraging edge hardware
- Model optimization best practices

Deep learning models at the edge

Edge AI : Growth & challenges

- Exponential growth in Edge AI applications
  - Logistics, smart homes, transportation, security etc.
  - Computer vision, NLP, time series
- Challenges with using cloud-based models
  - Latency
  - Reliable network connectivity
  - Security & privacy
- Challenges deploying deep learning models at the edge
  - Huge model footprint (> available memory)
  - Limited processing capacity

Deep learning models at the edge

Deep learning models need to be optimized for efficient and effective inference at the edge

Edge AI : Goals for optimization

- Maintain model performance thresholds (Accuracy, F1, Recall, etc.)
- Reduce model sizes (compression)
- Improve model performance
  - Latency
  - FLOPS
  - Power usage
- Leverage edge hardware capabilities
  - Edge CPUs / GPUs
  - Hardware accelerators

Model compression techniques

Model compression benefits

- Smaller memory footprint
- Reduced CPU/GPU time
- Lower latency
- Improved scaling per deployment
- Negligible loss of accuracy in most cases
- Easier packaging, transport and deployment

Quantization

FP32		
0.76	-0.10	1.45
-2.20	0.92	-0.89
-0.01	2.14	1.78

INT8		
95	-13	181
-275	115	-111
-1	268	223

- Reduce the storage size of parameters
- 32-bit to 8-bit (4X reduction)
- Less memory requirements
- Lower compute (FP vs INT operations)
- Energy saving
- Possible loss of accuracy (depends on model)
- Popular ML frameworks support quantization techniques

Types of quantization

Post-training quantization

- Done on a float trained model after training
- Convert weights, biases and activations to integers
- Simple to implement
- Loss of accuracy

Quantization aware training

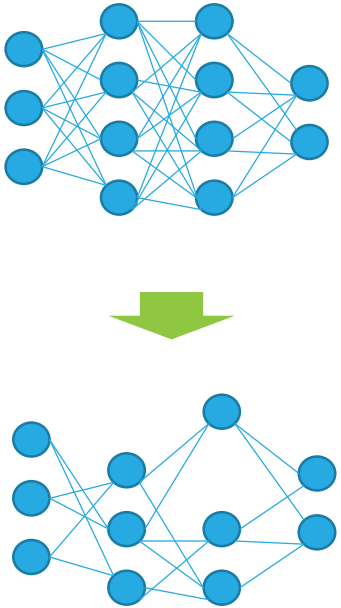
- Done during training
- Impact of quantization validated and adjusted
- Post-training quantization on this model results in smaller/no loss of accuracy

Quantization performance

Model	Compression	Inference time (s)	Top1 Acc %	Top5 Acc %	Storage
MobileNetV2	Original	0.0364	72	90	14 MB
	Quantized	0.0299	42	68	4.2 MB
ResNet50	Original	0.1050	75	93	102.5 MB
	Quantized	0.0516	74	93	26.5 MB
SqueezeNet1.1	Original	0.0286	58	82	5 MB
	Quantized	0.0110	52	76	1.4 MB

Retrieved from : https://www.softserveinc.com/en-us/blog/deep-learning-model-compression-and-optimization

Model pruning



- Eliminate model elements with low impact on outcomes
  - Nodes
  - Connections
  - Layers
- Iterative pruning (increased sparsity) with test for performance
- Size vs accuracy trade-off
- Effectiveness depends on nature of data
- Popular ML frameworks support pruning techniques

Types of pruning

Unstructured Pruning

- Remove individual elements
  - Connections
  - Nodes
- Random removal with validation
- Can achieve higher size reductions based on amount of pruning

Structured Pruning

- Remove part of the network
  - Layers
  - Channels
  - Filters
- Easier process
- Benefits based on model

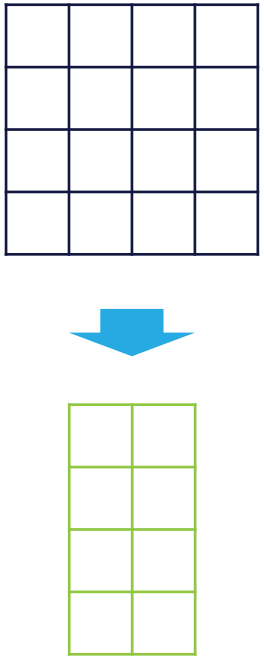
Pruning performance

Table 2: Results on ImageNet. Best results are bolded.

Model	Approach	Baseline	Pruned		
		Acc. (%)	Acc. (%)	Acc. Drop (%)	FLOPs Reduction
ResNet-50	NS [Liu et al., 2017] (Our-impl.)	76.15	74.88	1.27	53%
	SSS [Huang and Wang, 2018]	76.12	71.82	4.30	43%
	DCP [Zhuang et al., 2018]	76.01	74.95	1.06	56%
	FPGM [He et al., 2019]	76.15	74.13	2.02	53%
	CCP [Peng et al., 2019]	76.15	75.21	0.94	54%
	MetaPruning [Liu et al., 2019b]	76.6	75.4	1.2	50%
	SFP [He et al., 2018a]	76.15	62.14	14.01	42%
	PFP [Liebenwein et al., 2020]	76.13	75.21	0.92	30%
	AutoPruner [Luo and Wu, 2020]	76.15	74.76	1.39	49%
	Ours	76.15	75.63	**0.52**	54%
MobileNet v2	AMC [He et al., 2018b]	71.8	70.8	1.0	27%
	MetaPruning [Liu et al., 2019b]	72.0	71.2	0.8	27%
	DeepHoyer [Yang et al., 2020] (Our-impl.)	72.0	71.7	0.3	25%
	Ours	72.0	71.8	**0.2**	28%

Retrieved from : https://nips.cc/virtual/2020/public/poster_703957b6dd9e3a7980e040bee50ded65.html

Low-rank approximation



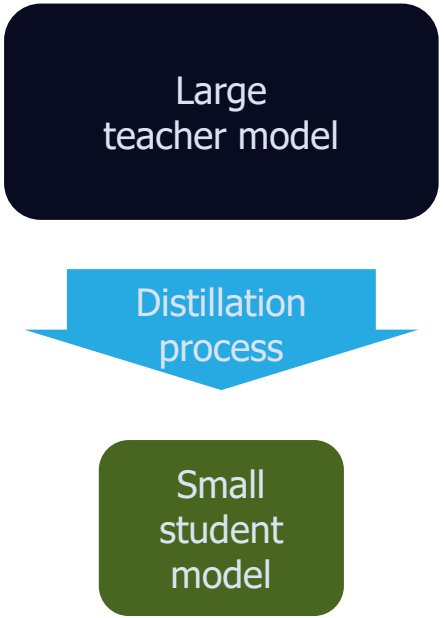
- Reduce the number of parameters necessary to represent the model
- Create matrix of lower rank
- Eliminate redundant data
- Measure performance with low rank matrix
- Benefits vary based on the use case
- Popular ML frameworks have out-of-the-box support

Low-rank approximation performance

Model	Method	Δ FLOPs (%)	Δ Params (%)	Δ Top-1 (pp)	Δ Top-5 (pp)
ResNet-18	**SVD-NAS**	**-58.60**	**-68.05**	−13.35*	−9.14*
				− 5.85**	**−3.34****
	ALDS [18]	-42.31	-65.14	-18.70	-13.38
	LR-S2 [8]	-56.49	-57.91	-38.13	-33.93
	F-Group[21]	-42.31	-10.66	-69.34	-87.63
MobileNetV2	**SVD-NAS**	**-12.54**	-9.00	−15.09*	−7.79*
				−9.99**	**−6.11****
	ALDS [18]	-2.62	**-37.61**	-16.95	-10.91
	LR-S2 [8]	-3.81	-6.24	-17.46	-10.34
EfficientNet-B0	**SVD-NAS**	**-22.17**	**-16.41**	−10.11*	−5.49*
				−7.67**	**−4.06****
	ALDS [18]	-7.65	-10.02	-16.88	-9.96
	LR-S2 [8]	-18.73	-14.56	-22.08	-14.15

Retrieved from : https://www.researchgate.net/figure/Post-training-results-of-low-rank-approximation-no-fine-tuning-fine-tuning-with_tbl1_362859051

Knowledge distillation



- Train a small student model to mimic the outputs of a large teacher model
- Distillation process compares outputs of the models for the same inputs and adjusts parameters for the student
- Smaller model footprint for student
- Comparable accuracy / performance
- Training dataset can be use-case specific

Comparison of techniques

	Quantization	Pruning	Low-rank Approximation	Knowledge Distillation
Cost	Low	Low	Medium	High
During training	Yes	Yes	Yes	Yes
Post training	Yes	Yes	Yes	Yes
Pretrained models	Yes	Yes	Yes	No

Compression process

- Create a baseline of the original model
  - Parameters, training data, test results
- Set threshold levels for compression expectations
  - Expected minimum accuracy, maximum resource usage
- Use an iterative approach
  - Try model compression in stages
  - Test with baseline training data
  - Compare with baseline test results and thresholds
- Try different techniques to identify best approach
  - Combining approaches is possible (e.g., quantization and pruning)

Leveraging edge hardware

Edge specialized infrastructure

- Edge optimized hardware
  - Deliver best performance for edge constrained environments
  - Low end processors: Micro-controller units, neural processing units (NPU)
  - High end processors: Google Edge TPU, NVIDIA Jetson
  - Application specific AI accelerators
- Edge frameworks
  - Compile models to optimize for edge specific hardware
  - Leverage hardware specific capabilities
  - Create deployable packages for models
  - E.g., NVIDIA TensorRT, Apache TVM, ONNX runtime

Edge frameworks - benefits

- Optimize execution graph for hardware
- Reduce memory requirements
- Remove unwanted steps/instructions
- Fuse steps/instructions
- Choose best values for configuration options
- Evaluate multiple execution strategies and choose the best one
- Create an optimized executable for inference
- Package model for ready deployment

Edge compilation process

- Choose the right framework based on the deployment hardware
  - E.g., TensorRT is most suited for NVIDIA processors
- Use the trained, validated and compressed model as input
- Compile the model
  - Plan for multiple iterations
  - Try available options for optimization/adaption
- Validate model performance
  - Use same benchmarks as compression
  - Test on hardware specific development kits
- Create deployable artifact

Model optimization best practices

Best practices for optimization - 1

- Performance baselines and goals need to be established and validated throughout the process
  - Accuracy, latency, FLOPS, etc., based on the use case
  - Helps ensure that the model performs as desired while going through optimizations
  - Track results against baseline for all model training iterations over time
- Choose hardware / frameworks when beginning model training
  - Deployment infrastructure may impact model architecture and optimizations needed
  - Dependency / overlaps can be understood ahead of time
  - Multiple deployment options may need to be supported

Best practices for optimization - 2

- Include edge hardware development kits/emulators as part of the training lifecycle
  - Automate optimization, compilation and testing
  - Use similar hardware configurations as deployment
  - Include collaborating edge applications also in end-to-end testing
  - Automate validating results and model promotion
- Monitor deployment performance
  - Some optimizations may carry negative impact when deployed in actual hardware
  - Monitor performance and validate against set baseline
  - Improve models based on experience

Thank You