Deep Neural Network Compression

Cosimo Rulli
cosimo.rulli@phd.unipi.it

Supervisors
Franco Maria Nardini and Rossano Venturini
Deep Neural Networks..

- Leading AI solution, unprecedented and super-human performance

Deep Neural Networks..

- **Leading AI solution, unprecedented and super-human performance**

- **Main Features**

Deep Neural Networks..

- **Leading AI solution, unprecedented and super-human performance**

- **Main Features**
  - **Representation Learning**

---

Deep Neural Networks..

- **Leading AI solution, unprecedented and super-human performance**

- **Main Features**
  - **Representation Learning**
  - **Theoretical Universal Approximators**

Deep Neural Networks...

- **Leading AI solution, unprecedented** and **super-human** performance

- **Main Features**
  - Representation Learning
  - Theoretical Universal Approximators
  - Accuracy **scales** with model size and training epochs

Deep Neural Networks..

- **Leading AI solution, unprecedented and super-human performance**

- **Main Features**
  - **Representation Learning**
  - **Theoretical Universal Approximators**
  - **Accuracy scales with model size and training epochs**

..are Getting Huge

- **Image Classification.** current state-of-the-art ~100x larger than AlexNet

- **Language Models.** Huge architectures up to 1.75 **trillions** of parameters

https://openai.com/blog/ai-and-compute/
..are Getting Huge

- **Image Classification.** current state-of-the-art $\sim 100x$ larger than AlexNet

- **Language Models.** Huge architectures up to 1.75 **trillions** of parameters

- Consequent growth of **computational burden**

- **Petaflop/s-day** increase faster than Moore’s law

https://openai.com/blog/ai-and-compute/
Training is costly

<table>
<thead>
<tr>
<th>Model</th>
<th>Hardware</th>
<th>Power (W)</th>
<th>Hours</th>
<th>kWh-PUE</th>
<th>CO₂e</th>
<th>Cloud compute cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2T&lt;sub&gt;base&lt;/sub&gt;</td>
<td>P100x8</td>
<td>1415.78</td>
<td>12</td>
<td>27</td>
<td>26</td>
<td>$41–$140</td>
</tr>
<tr>
<td>T2T&lt;sub&gt;big&lt;/sub&gt;</td>
<td>P100x8</td>
<td>1515.43</td>
<td>84</td>
<td>201</td>
<td>192</td>
<td>$289–$981</td>
</tr>
<tr>
<td>ELMo</td>
<td>P100x3</td>
<td>517.66</td>
<td>336</td>
<td>275</td>
<td>262</td>
<td>$433–$1472</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>V100x64</td>
<td>12,041.51</td>
<td>79</td>
<td>1507</td>
<td>1438</td>
<td>$3751–$12,571</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>TPUv2x16</td>
<td>—</td>
<td>96</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>NAS</td>
<td>P100x8</td>
<td>1515.43</td>
<td>274,120</td>
<td>656,347</td>
<td>626,155</td>
<td>$942,973–$3,201,722</td>
</tr>
<tr>
<td>NAS</td>
<td>TPUv2x1</td>
<td>—</td>
<td>32,623</td>
<td>—</td>
<td>—</td>
<td>$44,055–$146,848</td>
</tr>
<tr>
<td>GPT-2</td>
<td>TPUv3x32</td>
<td>—</td>
<td>168</td>
<td>—</td>
<td>—</td>
<td>$12,902–$43,008</td>
</tr>
</tbody>
</table>

Table 3: Estimated cost of training a model in terms of CO₂ emissions (lbs) and cloud compute cost (USD).\(^7\) Power and carbon footprint are omitted for TPUards due to lack of public information on power draw for this hardware.

Training is costly

<table>
<thead>
<tr>
<th>Model</th>
<th>Hardware</th>
<th>Power (W)</th>
<th>Hours</th>
<th>kWh-PUE</th>
<th>CO₂e</th>
<th>Cloud compute cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2T&lt;sub&gt;base&lt;/sub&gt;</td>
<td>P100x8</td>
<td>1415.78</td>
<td>12</td>
<td>27</td>
<td>26</td>
<td>$41–$140</td>
</tr>
<tr>
<td>T2T&lt;sub&gt;big&lt;/sub&gt;</td>
<td>P100x8</td>
<td>1515.43</td>
<td>84</td>
<td>201</td>
<td>192</td>
<td>$289–$981</td>
</tr>
<tr>
<td>ELMo</td>
<td>P100x3</td>
<td>517.66</td>
<td>336</td>
<td>275</td>
<td>262</td>
<td>$433–$1472</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>V100x64</td>
<td>12,041.51</td>
<td>79</td>
<td>1507</td>
<td>1438</td>
<td>$3751–$12,571</td>
</tr>
<tr>
<td>BERT&lt;sub&gt;base&lt;/sub&gt;</td>
<td>TPUv2x16</td>
<td>—</td>
<td>96</td>
<td>—</td>
<td>—</td>
<td>$2074–$6912</td>
</tr>
<tr>
<td>NAS</td>
<td>P100x8</td>
<td>1515.43</td>
<td>274,120</td>
<td>656,347</td>
<td>626,155</td>
<td>$942,973–$3,201,722</td>
</tr>
<tr>
<td>NAS</td>
<td>TPUv2x1</td>
<td>—</td>
<td>32,623</td>
<td>—</td>
<td>—</td>
<td>$44,055–$146,848</td>
</tr>
<tr>
<td>GPT-2</td>
<td>TPUv3x32</td>
<td>—</td>
<td>168</td>
<td>—</td>
<td>—</td>
<td>$12,902–$43,008</td>
</tr>
</tbody>
</table>

Table 3: Estimated cost of training a model in terms of CO₂ emissions (lbs) and cloud compute cost (USD).<sup>7</sup> Power and carbon footprint are omitted for TPUs due to lack of public information on power draw for this hardware.

---

### Inference

- **A lot** of inferences
  - 200 trillions of inference per day at Facebook\(^1\)
  - 90% of **workload** spent on inference at Amazon, NVIDIA\(^2\)

- Inference is resource **constrained** on the edge (IoT, Industry 4.0)

---

<table>
<thead>
<tr>
<th>Phase</th>
<th>Freq.</th>
<th>FLOPs</th>
<th>Devices</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>1</td>
<td>(10^{15}) (day)</td>
<td>Cloud, Servers</td>
<td>None</td>
</tr>
<tr>
<td>Inference</td>
<td>(\infty)</td>
<td>(10^9\div12)</td>
<td>Embedded smartphones, PC</td>
<td>Memory, Time, Energy</td>
</tr>
</tbody>
</table>

\(^1\)https://engineering.fb.com/data-center-engineering/accelerating-infrastructure/

Over-parametrization

- More **equations** (parameters) than **unknowns** (data samples)

- In general
  - ↓ Over-fitting
  - ↓ Poor performances

- Neural Networks
  - ↑ **Eases** optimization
  - ↑ **Increases** generalization

“Pluralitas non est ponenda sine necessitate”
- novacula Occami

Model Compression
Model Compression

- Leverages over-parametrization to compress DNNs without accuracy degradation

- Reducing
  - Memory impact
  - Inference time
  - Energy consumption

- Main methods
  - Pruning
  - Quantization
  - Knowledge Distillation
  - and more..
Pruning
Pruning

- Pruning techniques remove *unnecessary* parameters from neural networks

- Removing = set to 0

- Reduces *memory* impact, *energy* consumption and speedup inference
Element-wise vs Structured

- **Element-wise.** Removes single weights producing sparse tensors
  - ↑ High memory compression
  - ↓ Requires sparse multiplication

- **Structured.** Removes entire structures (columns, filters)
  - ↑ Direct speedup
  - ↓ Reduced memory compression
What to Prune?

‣ How to select which the parameters to prune?

‣ With $n$ parameters, $2^n$ possible pruning patterns

‣ Heuristic to estimate weight importance, or penalty to induce sparsity
What to Prune?

- **How to select** which the parameters to prune?

- With $n$ parameters, $2^n$ possible pruning patterns

- **Heuristic** to estimate weight importance, or **penalty** to induce sparsity

1990s - **Hessian**-based Pruning

2012 - **AlexNet** Revolution

2015 - **Magnitude** Pruning

>100 peer-reviewed papers

Dropout-based, $L_0$ penalty, **Gradient**-based
When to Prune?

- **During Training.** The model is trained to be sparse
  - Same budget as standard training

- **Fine-tuning.** Pruning is applied on a trained, dense model.
  - Better accuracy
Pruning Performance

- Magnitude-based, element-wise pruning, ResNet50 on ImageNet

- **Element-Wise** Pruning.
  
  \[ \uparrow \quad 90\% \text{ sparse, no accuracy drop} \]
  \[ \uparrow \quad +6\% \text{ accuracy w.r.t to dense model w/i same parameters} \]
  \[ \downarrow \quad \text{Sparse format overhead not included} \]

Research Question

- Pruning is a very effective compression technique, but

- RQ1. Is there any more principled and effective heuristic than magnitude?

- RQ2. What is the relationship between learning and sparsity?

- RQ3. Can we train sparse network from scratch?

- And many more..
Quantization
Quantization

- **Classical** Computer Science problem

- **Large input** values set -> **small output** values set

- Specific features of **neural quantization**
  - Heavily **over-parametrized** model
  - **Decoupling** between training and inference
Why Quantization?

- Quantization delivers benefits **both in training and inference**

- Quantized models offers
  - Reduced **memory** impact
  - **Faster** operations
  - Reduced **energy** consumption
Weights and Activations

- Quantize weights.
  - Offline
  - Weights can be **optimized**

- Quantize activations.
  - Online (inference time) -> computing stats is **costly** (min, max,..)
  - No optimization
Weights and Activations

- Quantize weights.
  - Offline
  - Weights can be **optimized**

- Quantize activations.
  - Online (inference time) -> computing stats is **costly** (min, max,..)
  - No optimization

Quantizing activations has a huge impact on accuracy
Fine-Tuning

- **Post-training Quantization (PTQ).**
  
  🔺 No re-training (~)
  
  ▼ Reduced precision

- **Quantization-Aware Training (QAT)**
  
  🔺 High precision
  
  ▼ Costly re-training phase
**Quantization-Aware Training**

- **Methodology.** Weights quantized **after** each gradient update

- **Requirements.** Backward and gradient update in **full-precision** for numerical reasons

- **Problem.** Quantizer gradient is **zero** almost everywhere

- **Solution.** Straight-Through Estimator (STE)
Quantization-Aware Training

- **Methodology.** Weights quantized after each gradient update

- **Requirements.** Backward and gradient update in **full-precision** for numerical reasons

- **Problem.** Quantizer gradient is **zero** almost everywhere

- **Solution.** Straight-Through Estimator (STE)
Quantization-Aware Training

- **Methodology.** Weights quantized *after* each gradient update

- **Requirements.** Backward and gradient update in **full-precision** for numerical reasons

- **Problem.** Quantizer gradient is *zero* almost everywhere

- **Solution.** Straight-Through Estimator (STE)
Quantization-Aware Training

- **Methodology.** Weights quantized after each gradient update

- **Requirements.** Backward and gradient update in full-precision for numerical reasons

- **Problem.** Quantizer gradient is zero almost everywhere

- **Solution.** Straight-Through Estimator (STE)
Quantization Performance

- **Fully-quantized training**

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Task</th>
<th>Model</th>
<th>Metric</th>
<th>Time</th>
<th>Mem saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit Momentum</td>
<td>MoCo v2</td>
<td>ResNet-50</td>
<td>67.3</td>
<td>30 days</td>
<td>0.0 GB</td>
</tr>
<tr>
<td>8-bit Momentum</td>
<td>MoCo v2</td>
<td>ResNet-50</td>
<td>67.4</td>
<td>28 days</td>
<td>0.1GB</td>
</tr>
<tr>
<td>32-bit Adam</td>
<td>LM</td>
<td>Transformer-1.5B</td>
<td>9.0</td>
<td>308 days</td>
<td>0.0 GB</td>
</tr>
<tr>
<td>8-bit Adam</td>
<td>LM</td>
<td>Transformer-1.5B</td>
<td>9.0</td>
<td>297 days</td>
<td>8.5GB</td>
</tr>
<tr>
<td>32-bit Adam</td>
<td>LM</td>
<td>GPT3-Medium</td>
<td>10.62</td>
<td>795 days</td>
<td>0.0 GB</td>
</tr>
<tr>
<td>8-bit Adam</td>
<td>LM</td>
<td>GPT3-Medium</td>
<td>10.62</td>
<td>761 days</td>
<td>1.7GB</td>
</tr>
</tbody>
</table>

- **PTQ vs QAT - ResNet18 on Imagenet**
  - PTQ ~0.1 training budget w.r.t. QAT
  - QAT lossless quantization up to 3/3

<table>
<thead>
<tr>
<th>W/A</th>
<th>Approach</th>
<th>Top1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>PTQ</td>
<td>71.1</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>69.9</td>
</tr>
<tr>
<td>4/4</td>
<td>PTQ</td>
<td>69.1</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>70.6</td>
</tr>
<tr>
<td>3/3</td>
<td>PTQ</td>
<td>65.6</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>69.7</td>
</tr>
<tr>
<td>2/2</td>
<td>PTQ</td>
<td>51.1</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>67.0</td>
</tr>
</tbody>
</table>
# Quantization Performance

## Fully-quantized training

- ** PTQ vs QAT - ResNet18 on Imagenet**
  - PTQ ~0.1 training budget w.r.t. QAT
  - QAT lossless quantization up to 3/3

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Task</th>
<th>Model</th>
<th>Metric</th>
<th>Time</th>
<th>Mem saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit Momentum</td>
<td>MoCo v2</td>
<td>ResNet-50</td>
<td>67.3</td>
<td>30 days</td>
<td>0.0 GB</td>
</tr>
<tr>
<td>8-bit Momentum</td>
<td>MoCo v2</td>
<td>ResNet-50</td>
<td>67.4</td>
<td>28 days</td>
<td>0.1 GB</td>
</tr>
<tr>
<td>32-bit Adam</td>
<td>LM</td>
<td>Transformer-1.5B</td>
<td>9.0</td>
<td>308 days</td>
<td>0.0 GB</td>
</tr>
<tr>
<td>8-bit Adam</td>
<td>LM</td>
<td>Transformer-1.5B</td>
<td>9.0</td>
<td>297 days</td>
<td>8.5 GB</td>
</tr>
<tr>
<td>32-bit Adam</td>
<td>LM</td>
<td>GPT3-Medium</td>
<td>10.62</td>
<td>795 days</td>
<td>0.0 GB</td>
</tr>
<tr>
<td>8-bit Adam</td>
<td>LM</td>
<td>GPT3-Medium</td>
<td>10.62</td>
<td>761 days</td>
<td>1.7 GB</td>
</tr>
</tbody>
</table>

---


---

<table>
<thead>
<tr>
<th>W/A</th>
<th>Approach</th>
<th>Top1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>PTQ</td>
<td>71.1</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>69.9</td>
</tr>
<tr>
<td>4/4</td>
<td>PTQ</td>
<td>69.1</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>70.6</td>
</tr>
<tr>
<td>3/3</td>
<td>PTQ</td>
<td>65.6</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>69.7</td>
</tr>
<tr>
<td>2/2</td>
<td>PTQ</td>
<td>51.1</td>
</tr>
<tr>
<td></td>
<td>QAT</td>
<td>67.0</td>
</tr>
</tbody>
</table>

---


Research Question

- Quantization is an extremely effective solution

- **RQ1.** Can we produce extreme low-bits models as effective as full-precision ones?

- **RQ2.** Can we go beyond STE?

- **RQ3.** Can we use FPGA and ASIC to fully leverage the benefit of quantization?

- And many more..

Knowledge Distillation
Knowledge Distillation

- Training paradigm that involves
  - **Student**: the model to be trained. Small, shallow and deployment oriented
  - **Teacher**: pre-trained. Deep and effective

- The student cannot learn the same function $f(x, \theta)$ as the teacher **extrapolating** it from the examples

- It could by **mimicking** its outputs on the samples

\[
f(x, \theta) \sim f(x, \theta')
\]
Logits. $z \in \mathbb{R}^c$, with $c$ number of classes.

Class Probabilities. $p_i = \text{softmax}(z_i)$
Logits Approximation

- **One-hot encoded** label
  - Single class information
Logits Approximation

- **One-hot encoded** label
  - Single class information

- **Teacher logits.**
  - **Multi-class** and **intra-class** information
Logits Approximation

- **One-hot encoded** label
  - Single class information

- Teacher **logits**.
  - **Multi-class** and **intra-class** information

Train the student to approximate the logits of the teacher
Feature Approximation

- Features **representation** encodes **inner** knowledge of the teacher

- Forcing the **student activations** to be similar to the teacher ones

\[ L_{\text{tot}} = H(x, y) + L(\phi_t(x), \phi_s(x)) \]

- Classical Loss
- Hint Loss
Knowledge Distillation Performance

- **Multi-level** distillation

- **Performance on ImageNet**
  - + 2.6% Top1 w.r.t to standard training
  - No inference overhead

<table>
<thead>
<tr>
<th>Model</th>
<th>Top1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>69.8</td>
</tr>
<tr>
<td>Teacher</td>
<td>73.3</td>
</tr>
<tr>
<td>Student + KD</td>
<td>72.4</td>
</tr>
</tbody>
</table>

Research Question

- Knowledge Distillation is effective but..

- **RQ1.** Poor theoretical basis

- **RQ2.** Knowledge distillation vs label smoothing?

- **RQ3.** Combinations with other compression methods?

- And many more..
Thanks for the attention!

cosimo.rulli@phd.unipi.it