Compression of Deep Neural Networks

Fraunhofer HHI, Machine Learning Group

Wojciech Samek
Record Performances with DNNs

- AlphaGo beats Go human champ
- Deep Net outperforms humans in image classification
- DeepStack beats professional poker players
- Dermatologist-level classification of skin cancer with Deep Nets
- Computer out-plays humans in "doom"
- Revolutionizing Radiology with Deep Learning
- Deep Net beats human at recognizing traffic signs
Complexity of DNN is Growing
Large Computational Resources Needed

Common carbon footprint benchmarks

in lbs of CO2 equivalent

- Roundtrip flight b/w NY and SF (1 passenger) 1,984
- Human life (avg. 1 year) 11,023
- American life (avg. 1 year) 36,156
- US car including fuel (avg. 1 lifetime) 126,000
- Transformer (213M parameters) w/ neural architecture search 626,155

Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper
Large Computational Resources Needed

We need techniques to reduce the computational complexity (i.e., storage, memory, energy, runtime)

- American life (avg. 1 year) = 36,156
- US car including fuel (avg. 1 lifetime) = 126,000
- Transformer (213M parameters) w/ neural architecture search = 626,155

Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper
Processing at the “Edge”

On-device deep learning

Privacy-preserving

Latency constraints
This talk will discuss how to reduce the complexity of DNNs by **model compression** and **efficient representation**.

Outline

1. Background
2. DeepCABAC
3. Compressed Entropy Row Format
Deep Neural Networks

For instance, VGG16
- 16 weight layers
- 138 000 000 parameters
- 553 MB (uncompressed)

\( (x, y) \in \mathbb{D} \)
Assume B to be a fix universal lossless code.

**Signal compression**
Distortion between elements (e.g. pixel values)

\[
\arg \min_{(Q, Q^{-1})} D(w_j, q_j) + \lambda L(b)
\]
DNN & Signal Compression

Assume B to be fix universal lossless code.

Signal compression
Distortion between elements (e.g. pixel values)

Neural network compression
Distortion between function of elements (e.g. prediction outputs)

\[
\arg \min_{(Q, Q^{-1})} D(w_j, q_j) + \lambda L(b)
\]

\[
\arg \min_{(Q, Q^{-1})} \mathcal{L}(y'', y') + \lambda L(b)
\]
\[(Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} \sum_{(x, y) \in \mathcal{D}} \mathcal{L}(y'', y') + \lambda L_Q(b)\]

\[y' \sim P(y' | x, w) \quad y'' \sim P(y'' | x, q)\]
Neural Network Compression

\[(Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} \sum_{(x, y) \in \mathcal{D}} \mathcal{L}(y'', y') + \lambda L_Q(b)\]

\[(Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} \sum_{(x, y) \in \mathcal{D}} D_{KL}(y'' || y') + \lambda L_Q(b)\]

Use KL-divergence as distortion measure.
Neural Network Compression

\[(Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} \sum_{(x, y) \in \mathcal{D}} \mathcal{L}(y'', y') + \lambda L_Q(b)\]

\[(Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} \sum_{(x, y) \in \mathcal{D}} D_{KL}(y''||y') + \lambda L_Q(b)\]

\[(Q, Q^{-1})^* = \min_{(Q, Q^{-1})} (q - w) F (q - w)^T + \lambda L_Q(b)\]

If the output distributions do not differ too much, we can approximate KL with the Fisher Information Matrix (FIM)

\[\mathbb{E}_{P_{\mathcal{D}}}[D_{KL}(y''||y')] = \delta w F \delta w^T + \mathcal{O}(\delta w^2)\]

with \(\delta w = q - w\) and

\[F := \mathbb{E}_{P_{\mathcal{D}}} \mathbb{E}_{P(y'|x, w)}[(\partial_w \log P(y'|x, w))(\partial_w \log P(y'|x, w))^T]\]
Neural Network Compression

\[
(Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} \sum_{(x, y) \in D} \mathcal{L}(y'', y') + \lambda L_Q(b)
\]

Approximate FIM by only its diagonal elements.
DeepCABAC-v1

Estimated as (Bayesian DNN)

\[ (Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} F_i(q_i - w_i)^2 + \lambda L_Q(b) \]

\[ q_k = \Delta I_k \]

[Wiedemann et al. 2019, ODML-CDNNR]
best paper award
DeepCABAC

Original network

Quantized network

Data \((x, y)\)

\[ (Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} (q_i - w_i)^2 + \lambda L_Q(b) \]

\[ q_k = \Delta I_k \]

DeepCABAC-v2

\[ F_j = 1 \ \forall j \]


[Wiedemann et al. 2019, ODML-CDNNR]

best paper award
DeepCABAC

Original network

Quantized network

Data \((x, y)\)

DeepCABAC-v3

\[ F_j = 1 \quad \forall j \quad \lambda = 0 \]

\[
(Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} (q_i - w_i)^2.
\]

\[ q_k = \Delta I_k. \]


[Wiedemann et al. 2019, ODML-CDNNR]

best paper award
Properties of CABAC

**Binarization**: represents each unique input value as a sequence of binary decisions.

**Context modelling**: probability model for each decision, which is updated on-the-fly by the local statistics of the data -> universality.

**Arithmetic coding**: arithmetic coding for each bit -> minimal redundancy + high efficiency

---

**Examples**

1 \rightarrow 100

-4 \rightarrow 111101

7 \rightarrow 10111010
## DeepCABAC Results

<table>
<thead>
<tr>
<th>Sparse Models (sparsity [%])</th>
<th>Org. Acc. Top1 [%]</th>
<th>Os_size [MB]</th>
<th>DeepCABAC (acc. [%])</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16 (9.85)</td>
<td>69.43</td>
<td>553.43</td>
<td>1.57 (69.43)</td>
</tr>
<tr>
<td>ResNet50 (74.12)</td>
<td>74.09</td>
<td>102.23</td>
<td>4.74 (73.65)</td>
</tr>
<tr>
<td>Small-VGG16 (7.57)</td>
<td>91.35</td>
<td>60.01</td>
<td>1.6 (91.00)</td>
</tr>
<tr>
<td>LeNet5 (1.90)</td>
<td>99.22</td>
<td>1.72</td>
<td>0.72 (99.16)</td>
</tr>
</tbody>
</table>
### DeepCABAC Results

<table>
<thead>
<tr>
<th>Sparse Models (sparsity [%])</th>
<th>Org. Acc. Top1 [%]</th>
<th>Os_size [MB]</th>
<th>DeepCABAC (acc. [%])</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16 (0.95)</td>
<td>69.43</td>
<td>553.43</td>
<td>1.57 (69.43)</td>
</tr>
<tr>
<td>ResNet50 102.2MB-&gt; 4.85MB at an acc. 73.65%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small-VGG16 (7.57)</td>
<td>91.35</td>
<td>60.01</td>
<td>1.6 (91.00)</td>
</tr>
<tr>
<td>LeNet5 (1.90)</td>
<td>99.22</td>
<td>1.72</td>
<td>0.72 (99.16)</td>
</tr>
</tbody>
</table>

**VGG16 553.4MB -> 8.7MB at an acc. 69.43%**

**ResNet50 102.2MB-> 4.85MB at an acc. 73.65%**
Efficient Representations

**Goal:** Find a representation for the weight matrices of a neural network, which is:
1) efficient with regard to storage
2) efficient with regard to algorithm complexity of the dot product operation.
Matrix Formats

\[ M = \begin{pmatrix}
0 & 3 & 0 & 2 & 4 & 0 & 0 & 2 & 3 & 4 & 0 & 4 \\
4 & 4 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 4 & 0 & 4 \\
4 & 0 & 3 & 4 & 0 & 0 & 0 & 4 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 4 & 4 & 4 & 0 & 3 & 4 & 4 & 0 & 0 \\
0 & 4 & 4 & 0 & 0 & 4 & 0 & 4 & 0 & 0 & 0 & 0
\end{pmatrix} \]

Storage requirements: 60 entries

Scalar product (second row \( M \), vector \( a \)):
- 24 load
- 12 multiply
- 11 add
- 1 write operations

\[ 4a_1 + 4a_2 + 0a_3 + 0a_4 + 0a_5 + 4a_6 + 0a_7 + 0a_8 + 4a_9 + 4a_{10} + 0a_{11} + 4a_{12} \]
Matrix Formats

\[
W : [3, 2, 4, 2, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3,
  4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4] \\
\text{colI} : [1, 3, 4, 7, 8, 9, 11, 0, 1, 5, 8, 9, 11, 0,
  2, 3, 7, 9, 3, 4, 5, 7, 8, 9, 1, 2, 5, 7] \\
\text{rowPtr} : [0, 7, 13, 18, 24, 28]
\]

sparse format

Storage requirements: 62 entries

Scalar product (second row M, vector a):
- 20 load
- 6 multiply
- 5 add
- 1 write operations

\[4a_1 + 4a_2 + 4a_6 + 4a_9 + 4a_{10} + 4a_{12}\]
Matrix Formats

\[ \Omega : [0, 4, 3, 2] \]
\[ colI : [4, 9, 11, 1, 8, 3, 7, 0, 1, 5, 8, 9, 11, 0, \]
\[ 3, 7, 2, 9, 3, 4, 5, 8, 9, 7, 1, 2, 5, 7] \]
\[ \Omega P \text{tr} : [0, 3, 5, 7, 13, 16, 17, 18, 23, 24, 28] \]
\[ rowP \text{tr} : [0, 3, 4, 7, 9, 10] \]

Storage requirements: 49 entries

Scalar product (second row M, vector a):
- 17 load
- 1 multiply
- 5 add
- 1 write operations

\[ 4(a_1 + a_2 + a_6 + a_9 + a_{10} + a_{12}) \]

[Wiedemann et al. 2019, IEEE TNNLS]
Storage efficiency

\[ H = - \log_2 p_0 \]

[Spike-and-Slab distributions]

[Wiedemann et al. 2019, IEEE TNNLS]
Results

Compressed AlexNet after converting it’s weight matrices into the different data structures.

[Wiedemann et al. 2019, IEEE TNNLS]
Questions ???

Contact Information:
Wojciech Samek  
Machine Learning Group  
Fraunhofer HHI  
Einsteinufer 37, 10587 Berlin, Germany

Phone: +49 30 31002-417  
Mail: wojciech.samek@hhi.fraunhofer.de  
Web: http://iphome.hhi.de/samek