IEEE GLOBECOM 2020 Tutorial on
Distributed Deep Learning: Concepts, Methods &
Applications in Wireless Networks

<table>
<thead>
<tr>
<th></th>
<th>Wojciech Samek</th>
<th>Deniz Gunduz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Fraunhofer HHI)</td>
<td>(Fraunhofer HHI)</td>
</tr>
</tbody>
</table>

1. Introduction (DG)
2. Distributed & Federated Learning: Concepts & Methods (WS)
3. Distributed & Federated Learning in Wireless Networks (DG)
4. Distributed Learning & Neural Network Compression (WS)
Part IV: Distributed Learning & Neural Network Compression
Complexity of DNN is Growing

- Inception-v3
- ResNet-50
- ResNet-101
- ResNet-152
- VGG-16
- VGG-19

Operations [G-Ops]
- 5M
- 35M
- 65M
- 95M
- 125M
- 155M

Top-1 accuracy [%]
- 80
- 75
- 70
- 65
- 60
- 55
- 50

Models:
- Inception-v3
- ResNet-50
- ResNet-101
- ResNet-152
- VGG-16
- VGG-19
- BN-NIN
- BN-AlexNet
- AlexNet
Processing at the “Edge”

Cameras and radar generate ~6 gigabytes of data every 30 seconds.

Self-driving car prototypes use approximately 2,500 Watts of computing power.

Generates wasted heat and some prototypes need water-cooling!

[slide from V. Sze]
Distributed Deep Learning

challenges
- Limited Resources
- Bandwidth Constraints

benefits
- Latency Constraints
- Privacy Constraints

Distributed Data

Hospital 1
Hospital 2
Hospital 3
Upcoming MPEG-7 part 17 Standard

Standard on "Compression of Neural Networks for Multimedia Content Description and Analysis"
MPEG-7 part 17 Standard

ISO/IEC 15938-17 (MPEG-7 Part 17) – NNR (Compression of neural networks for multimedia content description and analysis)

NNR is developed as a toolbox
- Inclusion in external neural network formats and frameworks, such as PyTorch, TensorFlow, ONNX or NNEF
- Independent coding method of neural networks (compression of weights, biases, etc. per parameter tensor plus inclusion of format-specific topology)

Compression efficiency of 95% without degrading classification quality, e.g. top-1.
MPEG-7 part 17 Standard
MPEG-7 part 17 Standard

Final Standard: April 2021

HHI Proposal became Reference Software NCTM (Neural Network Compression Test Model)

2/3 of the NNR standard by HHI, including:
- Uniform, dependent and codebook quantization methods
- DeepCABAC as entropy coding method and several coding improvements
- High-level syntax for various signaling aspects and parallel processing
- Syntax for bitstream definition with external frameworks
MPEG: Incremental Compression of NN

Call for Proposals on ICNN issued

Responses by April 2021

**Aim**: Technology for the compressed, interpretable and interoperable representation of updates of trained neural networks

**Target**: Use cases on **federated learning** with solution categories on:
- Network updates after refining/adding more training data, e.g. in federated learning
- Network updates after transfer learning/adapting to specific data (with and without network structure changes)
- Network updates with higher precision/less compression
DeepCABAC
Source Coding vs. NN Coding

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

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

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

\[
\text{arg min}_{Q, Q^{-1}} \mathcal{L}(y'', y') + \lambda L(b)
\]
From Source Coding to NN Coding

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

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

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

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

1. Use KL-divergence as distortion measure
2. Approximate KL-divergence with Fisher Information Matrix (FIM)
3. Approximate FIM by only its diagonal elements
DeepCABAC: Weighted RD-based Quantization + CABAC

Parametrize each weight parameter as Gaussian. $F_i = 1/\sigma_i$

DeepCABAC-v1

https://github.com/fraunhoferhhi/DeepCABAC

$$(Q, Q^{-1})^* = \text{arg min}_{(Q, Q^{-1})} F_i (q_i - w_i)^2 + \lambda L_Q(b)$$

$q_k = \Delta I_k$

[Wiedemann et al. 2019, ODML-CDNNR]
best paper award
DeepCABAC: Uniform Quantization + CABAC

DeepCABAC-v3

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

\[ q_k = \Delta I_k \]


[Wiedemann et al. 2019, ODML-CDNNR]

best paper award

https://github.com/fraunhoferhhi/DeepCABAC

Uniform Quantization works!
Lossless Coding: Desired Properties

**Universality:** The code should have a mechanism that allows it to adapt its probability model to a wide range of different types of input distributions, in a sample-efficient manner.

**Minimal redundancy:** The code should produce binary representations of minimal redundancy with regards to its probability estimate.

**High efficiency:** The code should have high coding efficiency, meaning that encoding/decoding should have high throughput.
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
Properties of CABAC

The first n+2 bits allow to adapt to any type of shape around 0 since they are encoded with regards to a context model. The remainder can only approximate the shape by a step-like distribution, since they are encoded with an Exponential-Golomb where the fixed-length parts are encoded without a context model.
### Some 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>
Some 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</td>
<td>69.43</td>
<td>553.43</td>
<td>1.57</td>
</tr>
<tr>
<td>VGG16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>553.4MB -&gt; 8.7MB at an acc. 69.43%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet50</td>
<td>102.2MB-&gt; 4.85MB at an acc. 73.65%</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>
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 inference complexity
Efficient Representation Format

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

Storage requirements: 60 entries

dense format
Matrix Formats: Compressed Entropy Row Format

Compressed neural networks have \textit{weight sharing} property.

\textbf{Trick:}
\[
    z_i^l = \sum_{j}^{M} w_{i,j}^l a_{j}^{l-1} \quad \rightarrow \quad z_i^l = \sum_{k} w_k^l \sum_{j \in J_{ik}^l} a_j^{l-1}
\]
Matrix Formats: Compressed Entropy Row Format

\[ \Omega : [0, 4, 3, 2] \]
\[ colI : [4, 9, 11, 1, 8, 3, 7, 0, 1, 5, 8, 9, 11, 0, 3, 7, 2, 9, 3, 4, 5, 8, 9, 7, 1, 2, 5, 7] \]
\[ \Omega Ptr : [0, 3, 5, 7, 13, 16, 17, 18, 23, 24, 28] \]
\[ rowPtr : [0, 3, 4, 7, 9, 10] \]

**CER format**

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. 2020, IEEE TNNLS]
Results

Compressed AlexNet after converting its weight matrices into the different data structures.

[Wiedemann et al. 2020, IEEE TNNLS]
Software-Hardware Co-Design

[Wiedemann et al., in prep]
Conclusion

Distributed environments come with new challenges
We need efficiency in training, communication and inference
Many different techniques to compress NN and updates
Different options (e.g. fine-tuning, structural changes, NAS).
Hardware co-design is crucial
MPEG standardization
References

**NN Compression**


References


S Wiedemann, A Marban, KR Müller, W Samek. Entropy-Constrained Training of Deep Neural Networks. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 1-8, 2019

S Wiedemann, KR Müller, W Samek. Compact and Computationally Efficient Representation of Deep Neural Networks. NIPS Workshop on Compact Deep Neural Network Representation with Industrial Applications (CDNNRIA), 1-8, 2018

Federated Learning


References


F Sattler, S Wiedemann, KR Müller, W Samek. *Sparse Binary Compression: Towards Distributed Deep Learning with minimal Communication*. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 1-8, 2019

Standardization


References


References


Thank you for your attention!

www.federated-ml.org

Federated Learning

Peer-to-Peer Learning

Distributed Training

On-Device Inference

Papers

Slides

Software