IEEE ICASSP 2020 Tutorial on
Distributed and Efficient Deep Learning

Outline of tutorial

1. Introduction (WS)
2. Model Compression & Efficient Deep Learning (WS)
3. (Virtual) Coffee Break
4. Distributed & Federated Learning (FS)
Part I: Introduction

Wojciech Samek & Felix Sattler
Deep Learning "Revolution"

Ingredients for the success
1. Large volumes of data
2. Large (Deep) Models
3. Large Computing Power
Complexity of DNN is Growing

![Graph showing the complexity growth of various DNN models](image)

- Inception-v3
- Inception-v4
- ResNet-50
- ResNet-101
- ResNet-152
- ResNet-34
- VGG-16
- VGG-19
- GoogLeNet
- ENet
- BN-NIN
- BN-AlexNet
- AlexNet

Operational complexity is measured in billions of operations (G-Ops) and accuracy is displayed as Top-1 accuracy [%].
Complexity of DNN is Growing

We need to design efficient architectures and techniques to reduce the model size.
Large Computational Resources Needed

**Common carbon footprint benchmarks**

<table>
<thead>
<tr>
<th>Activity</th>
<th>CO2 Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roundtrip flight b/w NY and SF (1 passenger)</td>
<td>1,984</td>
</tr>
<tr>
<td>Human life (avg. 1 year)</td>
<td>11,023</td>
</tr>
<tr>
<td>American life (avg. 1 year)</td>
<td>36,156</td>
</tr>
<tr>
<td>US car including fuel (avg. 1 lifetime)</td>
<td>126,000</td>
</tr>
<tr>
<td>Transformer (213M parameters) w/ neural architecture search</td>
<td>626,155</td>
</tr>
</tbody>
</table>

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

We need techniques to reduce the computational complexity at *training time*
Processing at the “Edge”

Wireless communications, vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2I) communications are central to autonomous vehicles. These communications require significant processing power to handle the data generated by the vehicle’s sensors, such as cameras and radar. Each sensor generates approximately 6 gigabytes of data every 30 seconds.

Self-driving car prototypes use approximately 2,500 Watts of computing power. This generates wasted heat and some prototypes need water-cooling!

[slide from V. Sze]
Processing at the “Edge”

On-device deep learning

Distributed Data & Privacy

Latency & bandwidth constraints
We need techniques to reduce the computational complexity at *inference time* (i.e., storage, memory, energy, runtime).
We need techniques to robustly train our models on *distributed data* in a *privacy-preserving* manner.
Large Interest in Academia & Industry
Large Interest in Academia & Industry

PUBLICATIONS:
Federated Learning for Mobile Keyboard Prediction

Tencent’s WeBank applying “federated learning” in AI
China’s first mobile bank, Tencent’s WeBank, partnering with a H.K. startup to access sources of data.

How Apple personalizes Siri without hoovering up your data
The tech giant is using privacy-preserving machine learning to improve its voice assistant while keeping your data on your phone.

by Karen Hao
Dec 18, 2020
Large Interest in Academia & Industry

Apple packed an AI chip into the iPhone X

Intel, Qualcomm, Google, and NVIDIA Race to Develop AI Chips and Platforms

source: https://github.com/basicmi/Al-Chip
Large Interest in Academia & Industry

Standard on "Compression of Neural Networks for Multimedia Content Description and Analysis"
IEEE ICASSP 2020 Tutorial on Distributed and Efficient Deep Learning

Part II: Model Compression & Efficient Deep Learning

Wojciech Samek & Felix Sattler
This part will discuss how to reduce the complexity of DNNs by model compression and efficient representation.

1. Background: Pruning, Quantization & Encoding
2. Finding Efficient Architectures
3. Compression Techniques
4. Efficient Neural Network Representation
Background: Pruning, Quantization & Encoding
Source Coding

Source coding is a subfield of information theory that studies the properties of so-called codes.

The primary task of a source codec is to represent a signal with the minimum number of (binary) symbols without exceeding an "acceptable level of distortion", which is determined by the application.
Source Coding

Goal: Minimize the rate-distortion objective:

$$C^* = \arg \min_C \mathbb{E}_{P(w)} \left[ D(w, q) + \lambda L_C(b) \right]$$

where $b = (B \circ Q)(w)$ and $q = (Q^{-1} \circ Q)(w)$. 
Lossless Coding

\[ q = (Q^{-1} \circ Q)(w) = w \quad \forall w \]

The rate-distortion objective simplifies into finding a binarizer \( B^* \) that maximally compresses the input samples.

Information theory already makes concrete statements regarding the minimum information contained in a probability source.

The minimum information required to fully represent a sample \( w \) that has probability \( P(w) \) is of \(-P \log_2 P(w)\) bits (Shannon).
Lossless Coding

Consequently, the entropy 
\[ H_P(W) = \sum_{w \in W} -P(w) \log_2 P(w) \]
states the minimum average number of bits required to represent any element 
\[ w \in W \subset \mathbb{R}^n. \]

**Fundamental theorem of lossless coding**

\[ H_P(W) \leq \bar{L}_C(W), \quad \forall C \]

where \( \bar{L}_C(W) = \sum_{w \in W} P(w)L_C(w) \) is the average code-length that any code \( C \) assigns to each element \( w \in W \).
Lossless Coding

We know of the existence of codes that are able to reach the average code-length, up to only 1 bit of redundancy.

\[ \exists C : H_P(W) \leq \bar{L}_C(W) < H_P(W) + 1 \]

Moreover, we even know how to build them.

Example: Huffman coding
Lossless Coding: Huffman Codes

\[ P(7) = 0.29 \]
\[ P(6) = 0.28 \]
\[ P(5) = 0.16 \]
\[ P(4) = 0.14 \]
\[ P(3) = 0.07 \]
\[ P(2) = 0.03 \]
\[ P(1) = 0.02 \]
\[ P(0) = 0.01 \]

\[ \begin{array}{c}
P(0) = 0.03 \\
\text{1} \quad P = 0.57 \\
\text{0} \quad P = 0.43 \\
\text{1} \quad P = 0.27 \\
\text{0} \quad P = 0.06 \\
\text{1} \quad P = 0.06 \\
\text{0} \quad P = 0.03 \\
\end{array} \]

\[ \begin{array}{c}
\text{11} \\
\text{10} \\
\text{01} \\
\text{001} \\
\text{0001} \\
\text{00001} \\
\text{000001} \\
\text{000000} \\
\end{array} \]
Lossless Coding: Huffman Codes

However, Huffman codes can be very inefficient in practice since the Huffman-tree grows very quickly for large input dimensions $n$.

**Scalar Huffman codes**

\[ H_P(W) \leq \bar{L}_{SH}(W) < H_P(W) + n \]

**Arithmetic codes**

Produces only up to two bits more than the minimum possible code length of an $n$-long random process.
Lossless Coding: Arithmetic Coding

Arithmetic coding consist of expressing a particular sequence of samples of an n-long random process as a so-called coding interval.

![Diagram of arithmetic coding process]

100 (index '4')

encode

'10111'
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.
Lossless Coding: Universal Codes

We implicitly assumed that the decoder knows the joint probability distribution of the input source.

This is not the case in many real world scenarios. Hence, in such cases one usually relies on so called *universal codes*.

They basically apply the following principle:
1) start with a general, prior probability model.
2) update the model upon seeing data.
3) encode the input samples with regards to the updated probability model.

\[
\tilde{L}_C(W) \geq H_{P,P_{\text{Dec}}}(W) = H_P(W) + D_{KL}(P \parallel P_{\text{Dec}})
\]
Lossy Coding

Finding the optimal code is in most cases NP-hard

\[ C^* = \arg \min_C \mathbb{E}_{P(w)} \left[ D(w, q) + \lambda L_C(b) \right] \]

**Idea:** Fix the binarization map \( B \) by selecting a particular (universal) lossless code. Then just need to find a scalar quantizer

\[ (Q, Q^{-1})^* = \arg \min_{(Q, Q^{-1})} \mathbb{E}_{P(w_j)} \left[ D(w_j, q_j) + \lambda L_Q(b_j) \right] \]

(we ignore vector quantizers here, which measure the distortion in the respective vector space by grouping a sequence of input samples together)
Lossy Coding: Scalar Lloyd algorithm

It approximates the average code-length of the quantized samples with the entropy of their empirical probability mass distribution (EPMD).

\[ L_C(b_j) = - \log_2 P_{\text{EPMD}}(q_j) \]

k-means clustering:
1) assign to closest cluster centers (= quantize)
2) update cluster centers (= update reconstruction values)
Lossy Coding: Uniform quantization

\[ C^* = \arg \min_C \mathbb{E}_{P(w)} [D(w, q) + \lambda L_C(b)] \]

Only minimize the distortion and ignore the rate term.

\[ \lambda = 0 \]

Only one step size parameter:

\[ q_k = \Delta I_k \]
Lossy Coding: CABAC-based RD-quantization

\[ C^* = \arg \min_C \mathbb{E}_{P(w)} [D(w, q) + \lambda L_C(b)] \]

Select Context Adaptive Binary Arithmetic Coder (CABAC) as our universal code.

Then we can trivially minimize by sequentially quantizing the input samples.
Source Coding vs. NN Coding

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

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

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

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

In NN coding, things are more complicated:
- complex distortion term (non-linear accumulation of errors)
- no clear structure in NN weights (e.g. in video high correlation between frames and neighboring pixels)
- more flexibility (e.g. fine-tuning, sparsification, structural changes)

In the extreme case, the whole training process can be regarded as NN coding (MDL principle)

\[
W^* = \min_W \left[ -\log_2 p(Y|X, W) + \alpha L_W(W) \right]
\]

[encoding prediction error] [encoding model] [Wiedemann et al. 2018]
IEEE ICASSP 2020 Tutorial on Distributed and Efficient Deep Learning

NN Coding

is developing a standard on "Compression of Neural Networks for Multimedia Content Description and Analysis"
Finding Efficient Architectures
Pruning Techniques
**Pruning Techniques**

**Network Pruning:**

1. Given a pre-trained model in the target domain
2. Define a pruning criterion
3. Repeatedly prune the network as follows:
   i. For each layer,
      a. For each element (weight/filter), evaluate the importance according to the pruning criterion (compute magnitudes)
      b. *optional:* Globally scale the magnitudes with regularization (e.g. $l_p$-norm)
   ii. Sort the magnitudes for all the layers throughout the network
   iii. Prune the least important elements and their inputs and outputs
   iv. *optional:* Further fine-tune to compensate performance degradation
4. Stop pruning if the model is reduced to a desired amount of model size or performance
Pruning Techniques: Different Criteria

**Taylor expansion**: Leverage a second-order Taylor expansion based on the Hessian matrix of the loss function to select parameters for deletion (optimal brain damage and optimal brain surgeon)

**Gradient**: A sparsified back propagation approach for neural network training using the magnitude of the gradient to find essential and non-essential features.

**Weight**: Prune the weights whose magnitude is below a certain threshold and to subsequently fine-tune with a l1-norm regularization.

**XAI-based**: LRP decomposes a classification decision into contributions called "relevances" of each network element to the overall classification score.
Layer-wise Relevance Propagation is a general approach to explain predictions of AI.

\[ \sum_i R_i = f(x) \]

(see for more information)

Layer-wise Relevance Propagation is a general approach to explain predictions of AI.

[Bach et al., PLOS ONE, 2015]
Results with Fine-Tuning

Pruning with fine-tuning.  

[Yeom et al., arXiv, 2019]
Results without Fine-Tuning

[Yeom et al., arXiv, 2019]
Distilling & Neural Architecture Search

[source: https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764]

[source: Elsken et al. 2019]
Compression Techniques
From Source Coding to NN Coding

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

\[y' \sim P(y'|x, w) \quad y'' \sim P(y''|x, q)\]
From Source Coding to NN Coding

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

Use KL-divergence as distortion measure

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

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

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

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

If the output distributions do not differ too much, we can approximate KL with the Fisher Information Matrix (FIM).
From Source Coding to NN Coding

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

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

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

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

Approximate FIM by only its diagonal elements
DeepCABAC: Weighted RD-based Quantization + CABAC

DeepCABAC-v1

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

$q_k = \Delta I_k$.

[Wiedemann et al. 2019, ODML-CDNNR]

best paper award

https://github.com/fraunhoferhhi/DeepCABAC
DeepCABAC: Uniform Quantization + CABAC

DeepCABAC-v3

$$F_j = 1 \quad \forall j \quad \lambda = 0$$

[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
Properties of CABAC

The first n+2 bits allow to adapt to any type of shape around 0 since they are encoded with 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>
</tbody>
</table>

**VGG16 553.4MB → 8.7MB at an acc. 69.43%**

<table>
<thead>
<tr>
<th></th>
<th>Org. Acc. Top1 [%]</th>
<th>Os_size [MB]</th>
<th>DeepCABAC (acc. [%])</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet50</td>
<td>102.2MB→ 4.85MB</td>
<td>99.22</td>
<td>0.72 (99.16)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Small-VGG16 (7.57)</th>
<th>91.35</th>
<th>60.01</th>
<th>1.6 (91.00)</th>
</tr>
</thead>
<tbody>
<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
Some Results

![Graph showing top-1 accuracy vs. compression ratio for ResNet50 with different compression methods.](image-url)
Some Results

![Graph showing the compression ratio and top-1 accuracy for MobileNetV2 with different quantization methods.]
Efficient Neural Network Representation
Efficient Representation

**Goal**: Find a representation for the neural network, which is:

1) efficient with regard to storage / memory
2) efficient with regard to inference complexity
3) efficient with regard to energy consumption
Fixed-Point Neural Nets

Reduce Precision

32-bit float: 10100101000000001010000000100

8-bit fixed: 01100110

Binary: 0

Advantages

- Arithmetic with lower bit-depth is faster
- From 32-bits to 8-bits, we get (almost) 4x reduction in memory
- Lower bit-widths we can squeeze more data into caches/registers
- Floating point arithmetic not always be supported on microcontrollers on some ultra low-power embedded devices
Efficient Representation Format

**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

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

Storage requirements: 60 entries

dense format
Matrix Formats: Dense Format

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

Storage requirements: 60 entries

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

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

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: Compressed Entropy Row Format

Compressed neural networks have *weight sharing* property.

**Trick:**

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

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

**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]
Storage Efficiency

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

\[\log_2 K\]

Spike-and-Slab distributions

Dense

CER/CSER

Sparse

\[H = -\log_2 p_0\]

[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]
Conclusion

Efficiency in training and efficiency in inference.
Many different techniques to compress NN.
Different options (e.g. fine-tuning, structural changes, NAS).
Hardware co-design is crucial.
MPEG standardization.
References

Neural Network Compression


References

Efficient Deep Learning

http://dx.doi.org/10.1109/TNNLS.2019.2910073


References

**Federated Learning**


http://dx.doi.org/10.1109/TNNLS.2019.2944481

References

Federated Learning

http://dx.doi.org/10.1109/ICASSP40776.2020.9054676

http://dx.doi.org/10.1109/IJCNN.2019.8852172
Slides and Papers available at

![Diagram of servers and clients](image-url)

www.federated-ml.org
Part III: Distributed & Federated Learning

Wojciech Samek & Felix Sattler
Distributed Learning
Traditional Centralized ("Cloud") ML

→ Data is gathered Centrally

Problems

- Privacy
- Ownership (→ who owns the data?)
- Security (→ single point of failure)
- Efficiency (→ need to move data around)
Distributed (/ “Embedded”) ML

→ Data never leaves the local Devices

→ Instead model Updates are communicated

Problems

- Privacy
- Ownership (→ who owns the data?)
- Security (→ single point of failure)
- Efficiency (→ need to move data around)

Solved
Distributed ( ⁄ “Embedded”) ML: Settings

- Federated Learning
- Peer-to-Peer Learning
- Distributed Training
- On-Device Inference

Detailed Comparison: Sattler, Wiegand, Samek. "Trends and Advancements in Deep Neural Network Communication."
Federated Learning

“Federated Learning is a machine learning setting where \textbf{multiple entities collaborate} in solving a learning problem, \textbf{without directly exchanging data}. The \textbf{Federated training process} is \textbf{coordinated by a central server}.”

Kairouz, Peter, et al. "Advances and open problems in federated learning."
Federated Learning

Nvidia uses federated learning to create medical imaging AI

Federated learning technique predicts hospital stay and patient mortality

Federated Learning for Mobile Keyboard Prediction

How Apple personalizes Siri without hoovering up your data

Tencent’s WeBank applying “federated learning” in A.I.

Artificial Intelligence / Machine Learning

IEEE ICASSP 2020 Tutorial on Distributed and Efficient Deep Learning
Federated Learning

![Diagram showing Federated Learning process with SGD updates and data exchange between devices and server.]

IEEE ICASSP 2020 Tutorial on Distributed and Efficient Deep Learning
Federated Learning

Server

averaging

Data

Data

Data
Federated Learning

Server

Data

Client Devices

Data

Data

Data

Data

Fraunhofer Heinrich Hertz Institute

IEEE ICASSP 2020 Tutorial on Distributed and Efficient Deep Learning
Federated Learning - Settings

Cross Device
- Large Number of Clients
- Only fraction of Clients available at any given time
- Few data points per Client
- Limited computational resources

Cross Silo
- Small number of Clients
- Clients are always available
- Large local data sets
- Strong computational resources
Federated Learning - Challenges

Challenges in Federated Learning

- Communication
- Convergence
- Privacy
- Robustness
- Personalization
- Heterogeneity
Federated Learning - Communication

Download

Expensive communication!

Upload

Expensive communication!
Federated Learning - Communication

Total Communication = [#Communication Rounds] x [#Parameters] x [Avg. Codeword length]

Case Study: VGG16 on ImageNet
- Number of Iterations until Convergence: 900,000
- Number of Parameters: 138,000,000
- Bits per Parameter: 32

→ Total Communication = 496.8 Terabyte (Upload+Download)
Federated Learning - Compression Methods

Total Communication = \[#\text{Communication Rounds}\] \times \[#\text{Parameters}\] \times \text{[Avg. Codeword length]}\n
Compression Methods

- Communication Delay
- Lossy Compression: Unbiased
- Lossy Compression: Biased
- Efficient Encoding
Communication Delay

**Distributed SGD:**
For \( t=1,\ldots,\) [Communication Rounds]:
For \( i=1,\ldots,\) [Participating Clients]:

**Client does:**
\[
g_i \leftarrow \nabla_{\theta} l(\theta_t, D_i^b)
\]

**Server does:**
\[
\theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_i g_i
\]

**Federated Averaging:**
For \( t=1,\ldots,\) [Communication Rounds]:
For \( i=1,\ldots,\) [Participating Clients]:

**Client does:**
\[
\theta_i = \text{SGD}_K(\theta_t, D_i)
\]

**Server does:**
\[
\theta_{t+1} = \frac{1}{M} \sum_i \Delta\theta_i
\]
Communication Delay

**Advantages:**

- Simple
- Reduces Communication *Frequency* (advantageous in on-device FL)
- Reduces both Upstream and Downstream communication
- Easy to integrate with Privacy mechanisms

**Disadvantages:**


Communication Delay

Convergence Analysis for Convex Objectives:

\[ E[F(x_T) - F(x^*)] \in O\left(\frac{HM}{T} + \frac{\sigma}{\sqrt{TKM}}\right) \]

- \( M \) – Clients Participating per Round
- \( T \) – Total communication Rounds
- \( K \) – Local Iterations per Round
- \( L \) – Lipschitz Parameter of the Loss Function
- \( \sigma \) – Bound on the Variance of the Stochastic Gradients

IID-Assumption:

\[ D_i \sim \varphi(x, y) \]

Statistical Heterogeneity

Convergence speed drastically decreases with increasing heterogeneity in the data

→ This effect aggravates if the number of participating clients ("reporting fraction") is low

Hsu, Tzu-Ming Harry, Hang Qi, and Matthew Brown. "Measuring the effects of non-identical data distribution for federated visual classification."
## Communication Delay

<table>
<thead>
<tr>
<th>Method</th>
<th>Non-IID</th>
<th>Other assumptions</th>
<th>Variant</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lian et al. [266]</td>
<td>BCGV</td>
<td>BLGV</td>
<td>Dec; AC; 1step</td>
<td>$O(1/T) + O(1/\sqrt{NT})$</td>
</tr>
<tr>
<td>PD-SGD [265]</td>
<td>BCGV</td>
<td>BLGV</td>
<td>Dec; AC</td>
<td>$O(N/T) + O(1/\sqrt{NT})$</td>
</tr>
<tr>
<td>MATCHA [401]</td>
<td>BCGV</td>
<td>BLGV</td>
<td>Dec</td>
<td>$O(1/\sqrt{TKM}) + O(M/KT)$</td>
</tr>
<tr>
<td>Khaled et al. [232]</td>
<td>BOGV</td>
<td>CVX</td>
<td>AC; LBG</td>
<td>$O(N/T) + O(1/\sqrt{NT})$</td>
</tr>
<tr>
<td>Li et al. [264]</td>
<td>BOBD</td>
<td>SCVX; BLGV; BLGN</td>
<td>-</td>
<td>$O(K/T)$</td>
</tr>
<tr>
<td>FedProx [261]</td>
<td>BGV</td>
<td>BNCVX</td>
<td>Prox</td>
<td>$O(1/\sqrt{T})$</td>
</tr>
<tr>
<td>SCAFFOLD [227]</td>
<td>-</td>
<td>SCVX; BLGV</td>
<td>VR</td>
<td>$O(1/TKM) + O(e^{-T})$</td>
</tr>
</tbody>
</table>

Kairouz, Peter, et al. "Advances and open problems in federated learning."
Communication Delay

**Advantages:**

- Simple
- Reduces Communication *Frequency* (more practical in on-device FL)
- Reduces Upstream + Downstream communication
- Easy to integrate with Privacy mechanisms

**Disadvantages:**

- Bad performance on non-iid data
- Low sample efficiency
Federated Learning - Compression Methods

Total Communication = [#Communication Rounds] x [#Parameters] x [Avg. Codeword length]

Compression Methods

- Communication Delay
- Lossy Compression: Unbiased
- Lossy Compression: Biased
- Efficient Encoding
Update Compression

Distributed SGD:
For $t=1,\ldots,[\text{Communication Rounds}]$:
For $i=1,\ldots,[\text{Participating Clients}]$:

Client does:
\[ g_i \leftarrow \nabla_{\theta} l(\theta_t, D^b_i) \]

Server does:
\[ \theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_i g_i \]

Distributed SGD with Compression:
For $t=1,\ldots,[\text{Communication Rounds}]$:
For $i=1,\ldots,[\text{Participating Clients}]$:

Client does:
\[ g_i \leftarrow \nabla_{\theta} l(\theta_t, D^b_i) \]
\[ \tilde{g}_i \leftarrow \text{comp}(g_i) \]

Server does:
\[ \theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_i \tilde{g}_i \]
Update Compression

--- Unbiased -----

----- Biased ------
Unbiased Compression

Definition: A compression operator \( \text{comp} : \mathbb{R}^d \rightarrow \mathbb{R}^d \) is called unbiased iff,

\[
E[\text{comp}(x)] = x \quad \forall x \in \mathbb{R}^d
\]

Pros:

- "Straight forward" Convergence Analysis (Stochastic Gradients with increased variance)
- Variance reduction (uncorrelated noise)

Strongly convex bound:

\[
E[f(x_T) - f^*] \leq \mathcal{O}\left(\frac{\sigma^2}{\mu^2 T}\right)
\]

\[
\sigma^2\left[\frac{1}{n} \sum_{i=1}^{n} \text{comp}(x_i)\right] = \frac{1}{n^2} \sum_{i=1}^{n} \sigma^2[\text{comp}(x_i)]
\]
Unbiased Compression

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Test accuracy</th>
<th>Data/epoch</th>
<th>Time per batch</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>94.3%</td>
<td>1023 MB</td>
<td>312 ms +0%</td>
</tr>
<tr>
<td>Atomo</td>
<td>92.6%</td>
<td>113 MB</td>
<td>948 ms +204%</td>
</tr>
<tr>
<td>Signum</td>
<td>93.6%</td>
<td>32 MB</td>
<td>301 ms -3%</td>
</tr>
<tr>
<td>Rank 2</td>
<td>94.4%</td>
<td>8 MB</td>
<td>239 ms -23%</td>
</tr>
</tbody>
</table>

- **Pros**: “Straight forward” Convergence Analysis (Stochastic Gradients, increased variance)
- **Cons**: Variance blow-up leads to poor empirical performance
Biased Compression

**Definition:** A compression operator \( \text{comp} : \mathbb{R}^d \rightarrow \mathbb{R}^d \) is called **biased** iff,

\[
E[\text{comp}(x)] \neq x \quad \forall x \in \mathbb{R}^d
\]

Biased compression methods do not necessarily converge!

→ Can be turned into convergent methods via error accumulation.

Karimireddy, et al. "Error feedback fixes signsgd and other gradient compression schemes."
Stich, Cordonnier, Jaggi. "Sparsified SGD with memory."
Error Accumulation

Distributed SGD:
For $t=1,\ldots,[\text{Communication Rounds}]$:
For $i=1,\ldots,[\text{Participating Clients}]$:

Client does:
$$g_i \leftarrow \nabla_{\theta} l(\theta_t, D_i^b)$$

Server does:
$$\theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_i g_i$$

Distributed SGD with Error Accumulation:

For $t=1,\ldots,[\text{Communication Rounds}]$:
For $i=1,\ldots,[\text{Participating Clients}]$:

Client does:
$$R_i \leftarrow R_i + \nabla_{\theta} l(\theta_t, D_i^b)$$
$$\tilde{g}_i \leftarrow \text{comp}(R_i)$$
$$R_i \leftarrow R_i - g_i$$

Server does:
$$\theta_{t+1} = \theta_t - \eta \frac{1}{M} \sum_i \tilde{g}_i$$
Error Accumulation

For a parameter $\alpha \in [0, 1)$, a $\alpha$-contraction operator is a (possibly randomized) operator $\text{comp} : \mathbb{R}^d \to \mathbb{R}^d$ that satisfies the contraction property

$$E\|x - \text{comp}(x)\|^2 \leq \alpha \|x\|^2, \quad \forall x \in \mathbb{R}^d$$

Theorem (Stich et al.): For any contraction operator, compressed SGD with Error Accumulation for large $T$ achieves convergence rate on $\mu$-strongly convex objective functions with asymptotic rate:

$$E[f(x_T) - f(x^*)] \leq \mathcal{O}\left(\frac{G^2}{\mu T}\right)$$

Independent of alpha!
### Federated Learning - Recap Compression

<table>
<thead>
<tr>
<th></th>
<th>Unbiased</th>
<th>Biased</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Methods</strong></td>
<td>TernGrad, QSGD, Atomo</td>
<td>Gradient Dropping, Deep Gradient Compression, signSGD, PowerSGD,</td>
</tr>
<tr>
<td><strong>Convergence</strong></td>
<td>Bounded Variance Assumption</td>
<td>k-contraction Framework (Stich et al. 2018)</td>
</tr>
<tr>
<td><strong>Proofs</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Combining Methods: Sparse Binary Compression

Sattler, et al. "Sparse binary compression: Towards distributed deep learning with minimal communication." 2019 International Joint Conference on Neural Networks (IJCNN).
### Sparse Binary Compression

<table>
<thead>
<tr>
<th>Compression Method</th>
<th>Method</th>
<th>Baseline</th>
<th>DGC</th>
<th>Fed. Avg.</th>
<th>SBC (1)</th>
<th>SBC (2)</th>
<th>SBC (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet5-Caffe @MNIST</td>
<td>Accuracy</td>
<td>0.9946</td>
<td>0.994</td>
<td>0.994</td>
<td>0.994</td>
<td>0.994</td>
<td>0.991</td>
</tr>
<tr>
<td>@MNIST</td>
<td>Compression</td>
<td>×1</td>
<td>×718</td>
<td>×500</td>
<td>×2071</td>
<td>×3166</td>
<td>×24935</td>
</tr>
<tr>
<td>ResNet18 @CIFAR10</td>
<td>Accuracy</td>
<td>0.946</td>
<td>0.9383</td>
<td>0.9279</td>
<td>0.9422</td>
<td>0.9435</td>
<td>0.9219</td>
</tr>
<tr>
<td>@CIFAR10</td>
<td>Compression</td>
<td>×1</td>
<td>×768</td>
<td>×1000</td>
<td>×2369</td>
<td>×3491</td>
<td>×31664</td>
</tr>
<tr>
<td>ResNet34 @CIFAR100</td>
<td>Accuracy</td>
<td>0.773</td>
<td>0.767</td>
<td>0.7316</td>
<td>0.767</td>
<td>0.7655</td>
<td>0.701</td>
</tr>
<tr>
<td>@CIFAR100</td>
<td>Compression</td>
<td>×1</td>
<td>×718</td>
<td>×1000</td>
<td>×2370</td>
<td>×3166</td>
<td>×31664</td>
</tr>
<tr>
<td>ResNet50 @ImageNet</td>
<td>Accuracy</td>
<td>0.737</td>
<td>0.739</td>
<td>0.724</td>
<td>0.735</td>
<td>0.737</td>
<td>0.728</td>
</tr>
<tr>
<td>@ImageNet</td>
<td>Compression</td>
<td>×1</td>
<td>×601</td>
<td>×1000</td>
<td>×2569</td>
<td>×3531</td>
<td>×37208</td>
</tr>
<tr>
<td>WordLSTM @PTB</td>
<td>Perplexity</td>
<td>76.02</td>
<td>75.98</td>
<td>76.37</td>
<td>77.73</td>
<td>78.19</td>
<td>77.57</td>
</tr>
<tr>
<td>@PTB</td>
<td>Compression</td>
<td>×1</td>
<td>×719</td>
<td>×1000</td>
<td>×2371</td>
<td>×3165</td>
<td>×31658</td>
</tr>
<tr>
<td>WordLSTM* @WIKI</td>
<td>Perplexity</td>
<td>101.5</td>
<td>102.318</td>
<td>131.51</td>
<td>103.95</td>
<td>103.95</td>
<td>104.62</td>
</tr>
<tr>
<td>@WIKI</td>
<td>Compression</td>
<td>×1</td>
<td>×719</td>
<td>×1000</td>
<td>×2371</td>
<td>×3165</td>
<td>×31657</td>
</tr>
</tbody>
</table>
Sparse Binary Compression for non-iid Data

VGG11* @ CIFAR
IID Data

VGG11* @ CIFAR
NON-IID Data (2)

VGG11* @ CIFAR
NON-IID Data (1)
Federated Learning - Combining Methods

Sattler, Wiedemann, Müller, Samek. "Robust and communication-efficient federated learning from non-iid data." IEEE TNNLS (2019).
Efficient Encoding - DeepCABAC

- CABAC best encoder for quantized parameter tensors
- Plug & Play
- Can be used as a final lossless compression stage for all compression methods that we have presented

Wiedemann et al. "DeepCABAC: A Universal Compression Algorithm for Deep Neural Networks."
Federated Learning - Challenges

Challenges in Federated Learning

- Privacy
- Robustness
- Personalization
- Heterogeneity
- Communication
- Convergence

- ✔
- ✔
- ✔
Hitaj, Briland, Giuseppe Ateniese, and Fernando Perez-Cruz. "Deep models under the GAN: information leakage from collaborative deep learning."
Federated Learning - Privacy
Federated Learning - Privacy

Privacy Protection Mechanisms:

- Secure Multi-Party Computation
- Homomorphic Encryption
- Trusted Execution Environments
- **Differential Privacy**

Dwork, Cynthia, and Aaron Roth. "The algorithmic foundations of differential privacy."
Differential Privacy

A mechanism $A : \mathcal{P}(\mathcal{X}) \rightarrow \mathcal{Y}$ is called differentially private with parameter $\varepsilon$ iff,

$$\sup_{S \subseteq \mathcal{Y}} \log \left( \frac{P[A(D') \in S]}{P[A(D) \in S]} \right) \leq \varepsilon$$

for any two data sets $D$ and $D'$ which differ in only one element.
Differential Privacy – Mechanisms

Global Sensitivity: \[
S(f) = \max_{\text{dist}(D, D')=1} |f(D) - f(D')|
\]

The Laplace mechanism \[ A(D) = f(D) + Z \text{ with } Z \sim \frac{S(f)}{\varepsilon} \text{Lap}(0, 1) \] is \( \varepsilon \)-differentially private.
Differential Privacy - Post Processing

Data released via a $(\varepsilon, \delta)$-private mechanism is $(\varepsilon, \delta)$-private under arbitrary post processing!
Differential Privacy - Basic Composition

After applying $R$ algorithms with $(\varepsilon_i, \delta_i) : i = 1, \ldots, R$ to the data

the total privacy loss is:

$\left( \sum_{i=1}^{R} \varepsilon_i, \sum_{i=1}^{R} \delta_i \right)$

→ privacy loss is additive!

This is a worst-case analysis – better bounds can be found using more elaborated accounting mechanisms (e.g. moments accountant)
Privacy and Communication

We need methods which are both communication-efficient and privacy-preserving!

Differential Privacy:

→ adds artificial noise to the parameter updates to obfuscate them

Compression Methods:

→ add quantization noise to the parameter updates to reduce the bitwidth

→ combine the two approaches!

Federated Learning - Challenges

Challenges in Federated Learning

- Privacy ✔
- Communication ✔
- Convergence ✔
- Robustness
- Personalization
- Heterogeneity ( )
Federated Learning - Meta- and Multi Task-Learning

Federated Learning Environments are characterized by a high degree of statistical heterogeneity of the client data.

→ In many situations, learning one single central model is suboptimal or even undesirable.
Federated Learning - Meta- and Multi Task-Learning

Client data:
\[ p_i(x, y), \quad i = 1, .., n \]

\[ p_i(y|x) \text{ shared} \quad \implies \text{one model can be learned} \]

\[ p_i(y|x) \text{ varies} \quad \implies \text{no single model can fit the data of all clients} \]

\[ p_i(x) \text{ shared} \quad \implies \text{IID data} \]

\[ p_i(x) \text{ and/or } p_i(y) \text{ varies} \quad \implies \text{non-IID data} \]
Federated Learning - Meta- and Multi Task-Learning

I like ice cream.

I like Beyoncé.

I like cats.

A linear classifier can correctly separate the data of every single client, but not simultaneously for all clients.
→ Clustered Federated Learning groups the client population into clusters with jointly trainable data distributions and trains a separate model for every cluster.

How to identify the clusters?
Clustered Federated Learning

How to identify the clusters?

→ via the model updates!
At every stationary solution of the Federated Learning objective, the angle between the parameter updates of the different clients is highly indicative of their distribution similarity!
Clustered Federated Learning - Algorithm

1.) Run Federated Learning until convergence to a stationary solution

\[ \theta^* \leftarrow \text{FederatedLearning}(\theta, c) \]

2.) Compute the **pairwise cosine similarity** between the latest parameter updates from all clients

\[ \alpha_{i,j} \leftarrow \frac{\langle \nabla r_i(\theta^*), \nabla r_j(\theta^*) \rangle}{\|\nabla r_i(\theta^*)\| \|\nabla r_j(\theta^*)\|} \]

3.) If there exists a client whose local empirical risk is not sufficiently minimized by the federated learning solution ...

\[
\text{if } \max_{i \in c} \|\nabla r_i(\theta^*)\| > \varepsilon \text{ then }
\]

4.) ... then **bi-partition** the client population into two groups of **minimal pairwise similarity**

\[ c_1, c_2 \leftarrow \text{arg min}_{c_1 \cup c_2 = c} \left( \max_{i \in c_1, j \in c_2} \alpha_{i,j} \right) \]

5.) Repeat everything for the two groups, starting from 1.)
Clustered Federated Learning - Clustering Guarantees

Let \( D_i \sim \varphi_{I(i)} \)

\[
    r_i(\theta) := \frac{1}{|D_i|} \sum_{(x,y) \in D_i} l(f_\theta(x), y)
\]

\[
    R_i(\theta) := \int l(f_\theta(x), y) d\varphi_{I(i)}(x, y)
\]

\[
    F(\theta) := \sum_{i=1}^{m} \frac{|D_i|}{|D|} r_i(\theta)
\]

and \( \theta^* \) s.t. \( \nabla_\theta F(\theta^*) = 0 \)

Then the proposed mechanism will correctly separate the clients if

\[
    \max_{i=1, \ldots, m} \gamma_i < \sin\left(\frac{\pi}{4(k-1)}\right)
\]

with

\[
    \gamma_i := \frac{\|\nabla_\theta R_{I(i)}(\theta^*) - \nabla_\theta r_i(\theta^*)\|}{\|\nabla_\theta R_{I(i)}(\theta^*)\|}
\]

and \( k \) being the number of clusters.
Federated Learning - Clustered Federated Learning

Correct Clustering with $P \approx 1$

Correct Clustering with $P \approx 0$

Correct Clustering
Clustered Federated Learning

Measure cluster quality via: $g(\alpha) := \min_{i,j} \alpha_{i,j} - \min_{I(i)=I(j)} \left( \max_{c_1 \cup c_2 = c} \alpha_{i,j} \right)$

Then: $g(\alpha) > 0 \Rightarrow "correct\ clustering"$
Clustered Federated Learning

Few communication rounds are sufficient in order to obtain a correct clustering.

Work also with weight updates:

- Few data points are sufficient to obtain correct clustering.

![Graphs showing the effect of communication rounds and data points on clustering accuracy.](image)

IEEE ICASSP 2020 Tutorial on Distributed and Efficient Deep Learning

Fraunhofer Heinrich Hertz Institute

BIFOLD
Clustered Federated Learning

Federated Learning

1st Split

2nd Split

3rd Split
Clustered Federated Learning

1) FL has converged to a stationary solution

2) After the 1st split Accuracy drastically increases for the group of clients that was separated out

3) After the third round of splitting $g(a)$ has reduced to below zero for all remaining clusters

Sattler, Müller, Samek. "Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints."
Sattler, Müller, Samek. “On the Byzantine Robustness of Clustered Federated Learning” (ICASSP 2020)
Federated Learning - Challenges

Challenges in Federated Learning

- Privacy
- Communication
- Convergence
- Robustness
- Personalization
- Heterogeneity

✔️ ✔️ ✔️ ✔️ ✔️
References

Neural Network Compression

http://dx.doi.org/10.1109/JSTSP.2020.2969554

https://arxiv.org/abs/1912.08881


http://dx.doi.org/10.1109/IJCNN.2019.8852119
References

Efficient Deep Learning


References

Federated Learning


References

Federated Learning

http://dx.doi.org/10.1109/ICASSP40776.2020.9054676

http://dx.doi.org/10.1109/IJCNN.2019.8852172
Slides and Papers available at

www.federated-ml.org