
Special Topics in AI

Lecture 9: Communication Efficient FL

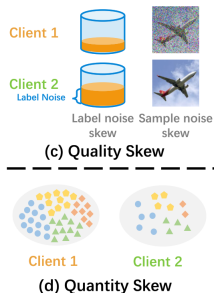
University of Mazandaran

Fall 1405

Recap: Statistical Heterogeneity in FL

Refers to the case where data available at each client come from different underlying distributions

- ▶ label skew
- ▶ feature skew
- ▶ quality skew
- ▶ quantity skew



Recap: Effect of Statistical Heterogeneity

- ▶ Model bias
- ▶ Slower convergence
- ▶ Reduced model accuracy
- ▶ Instability in training
- ▶ Communication inefficiency

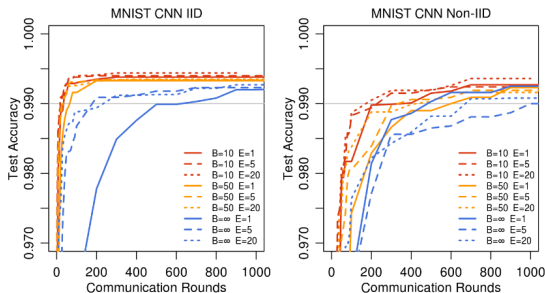


Figure 2: Test set accuracy vs. communication rounds for the MNIST CNN (IID and then pathological non-IID) with $C = 0.1$ and optimized η .

Recap: Effect of Statistical Heterogeneity

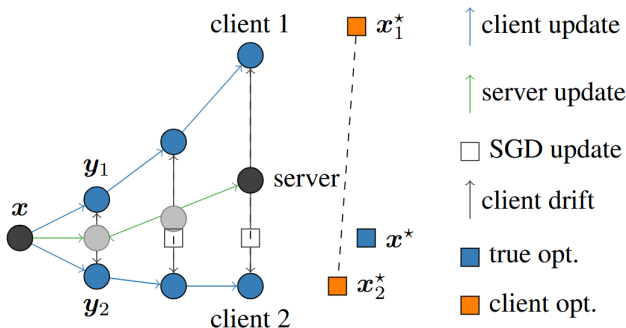
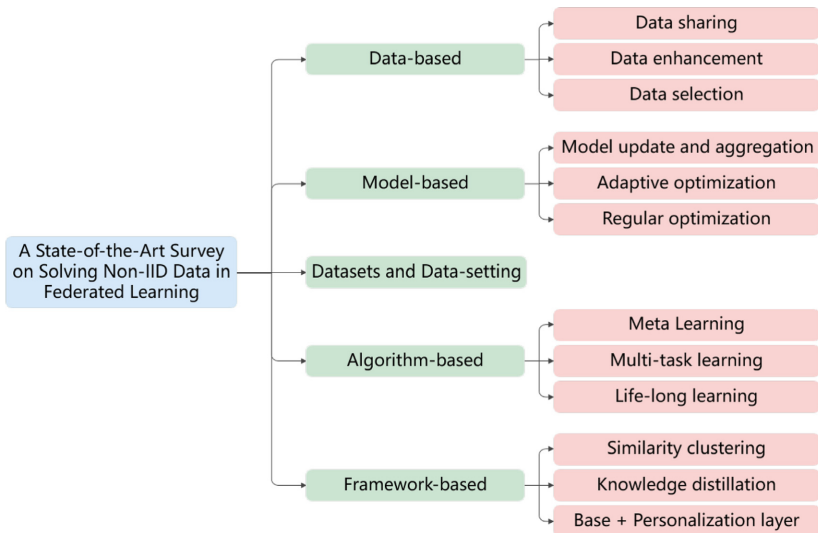


Figure 1. Client-drift in FEDAVG is illustrated for 2 clients with 3 local steps ($N = 2, K = 3$). The local updates y_i (in blue) move towards the individual client optima x_i^* (orange square). The server updates (in red) move towards $\frac{1}{N} \sum_i x_i^*$ instead of to the true optimum x^* (black square).

Karimireddy et al. "Scaffold: Stochastic controlled averaging for federated learning." In International Conference on Machine Learning (ICML), pp. 5132-5143, 2020

Approaches



Ma et al. "A state-of-the-art survey on solving non-iid data in federated learning." *Future Generation Computer Systems*, v. 135, pp. 244-258, 2022.

Communication Overhead in FL

	VGG11 @CIFAR10 Acc. = 0.84
FedAvg (n=25)	3572.7 MB / 3572.7 MB
FedAvg (n=100)	1606.3 MB / 1606.3 MB

Sattler et al. "Robust and Communication-Efficient
Federated Learning From Non-IID Data", 2019

Communication Overhead in FL

	VGG11 @CIFAR10 Acc. = 0.84
FedAvg (n=25)	3572.7 MB / 3572.7 MB
FedAvg (n=100)	1606.3 MB / 1606.3 MB

Sattler et al. "Robust and Communication-Efficient Federated Learning From Non-IID Data", 2019

Our experiments reveals that training a ResNet-152 model for 100 rounds using a decentralized CIFAR-100 dataset over 10 participants require more than 26 TB of data exchange over the network to reach acceptable accuracy.

[source-link](#)

Communication Overhead in FL

	VGG11 @CIFAR10 Acc. = 0.84
FedAvg (n=25)	3572.7 MB / 3572.7 MB
FedAvg (n=100)	1606.3 MB / 1606.3 MB

Sattler et al. "Robust and Communication-Efficient Federated Learning From Non-IID Data", 2019

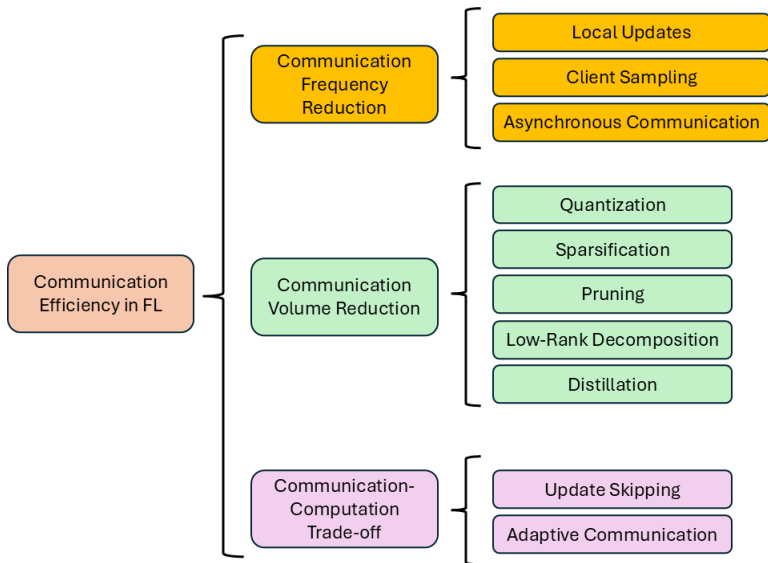
Our experiments reveals that training a ResNet-152 model for 100 rounds using a decentralized CIFAR-100 dataset over 10 participants require more than 26 TB of data exchange over the network to reach acceptable accuracy.

[source-link](#)

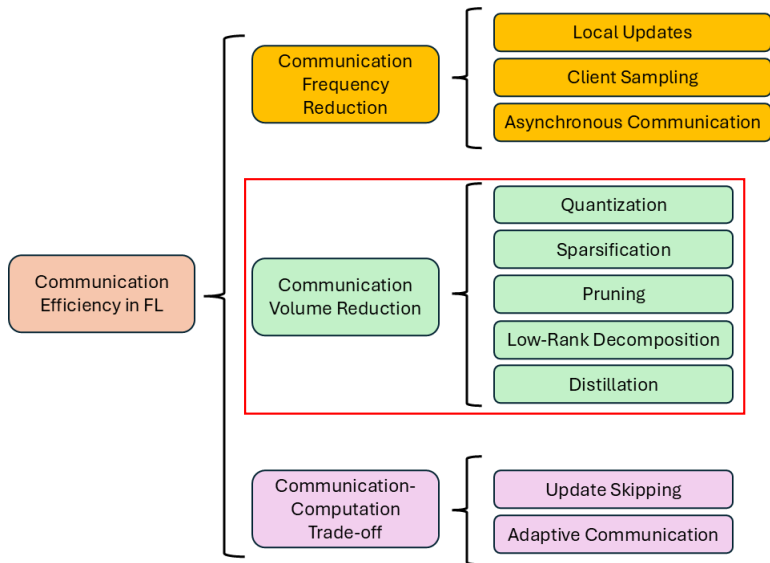
In a keyboard auto-completion for android devices application, training a small ML model ($\sim 1.3M$ parameters) with 6,500 clients over 2,000 rounds, requires sending a total bandwidth of $\sim 61TB$ to and from the PS.

[source-link](#)

Overview of Communication Efficient FL



Overview of Communication Efficient FL



Quantization

Reducing the number of bits required to represent parameters (for computation and storing) by converting them from FP to INT

Most commonly used data types in deep learning are:

- ▶ **FP32** (a.k.a. full precision): 32-bits to represent a number
 - ▶ 1 bit for sign, 8 bits for exponent, 23 bits for significand
 - ▶ very accurate, huge load!
- ▶ **FP16** (a.k.a. half precision): 16-bits to represent a number
 - ▶ 1 bit for sign, 5 bits for exponent, 10 bits for significand
 - ▶ memory-efficient compared to FP32, numerical instability (impacts model accuracy)
- ▶ **BF16** (a.k.a. half precision): 16-bits to represent a number
 - ▶ 1 bit for sign, 8 bits for exponent, 7 bits for significand
 - ▶ decreases the risk of underflow and overflow compared to FP16

Assume a 150M parameter DL model:

$$\mathbf{32bit} : \frac{32}{8} \times 150M \approx 0.6\text{GB}$$

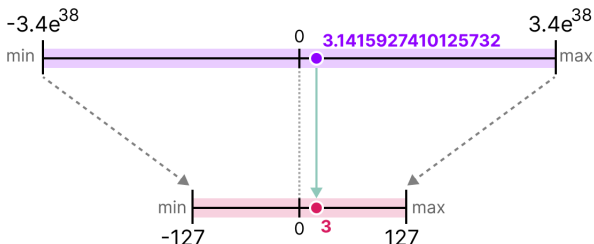
$$\mathbf{16bit} : \frac{16}{8} \times 150M \approx .3\text{GB}$$

INT8 is the most common integer precision

- ▶ Only integer numbers
- ▶ Range: $[-127, 127]$ or $[-128, 127]$

Quantization

Reducing the number of bits required to represent parameters (for computation and storing) by converting them from FP to INT



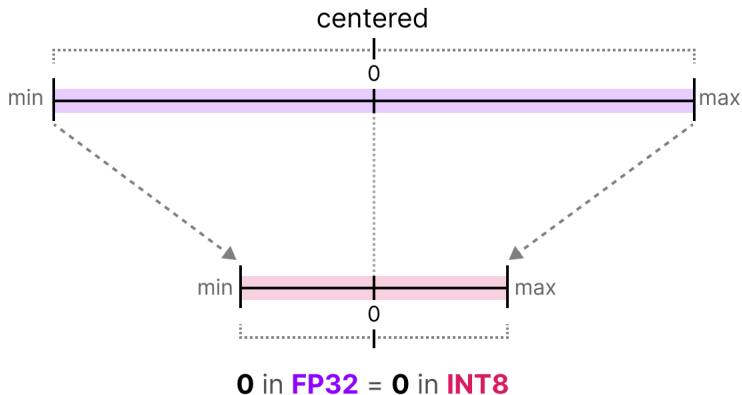
Reducing the number of bits required to represent parameters (for computation and storing) by converting them from FP to INT

- ▶ In practice, not necessary to map the entire FP32 range into INT8
- ▶ Symmetric vs Asymmetric

Symmetric Quantization

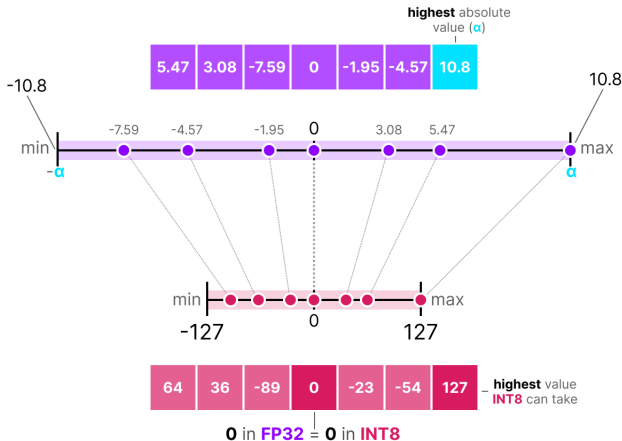
Quantized value for 0 in FP space is exactly zero in the INT8:

A range in the FP domain maps to the INT domain symmetrically around zero



Symmetric Quantization

Given a list of values, the *highest* absolute value is used as the range



Symmetric Quantization

Given a vector of values $\mathbf{x}$, the *highest* absolute value is used as the range

$$x_i^{\text{quant}} = \text{round} \left(\frac{127}{\max |\mathbf{x}|} \cdot x_i \right)$$
$$x_i^{\text{dequant}} = \frac{\max |\mathbf{x}|}{127} x_i^{\text{quant}}$$

Symmetric Quantization

```
import torch

def absmax_quantize(X):
    # Calculate scale
    scale = 127 / torch.max(torch.abs(X))

    # Quantize
    X_quant = (scale * X).round()

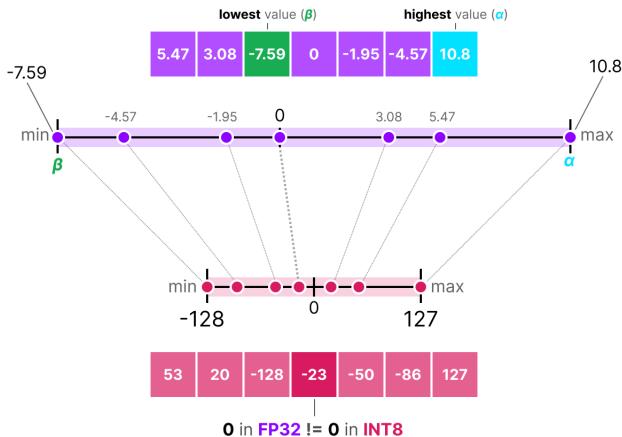
    # Dequantize
    X_dequant = X_quant / scale

    return X_quant.to(torch.int8), X_dequant
```

Asymmetric Quantization

Not symmetric around zero:

The minimum and maximum values from the FP range are mapped to the minimum and maximum of the quantized range.



Asymmetric Quantization

$$\text{scale} = \frac{255}{\max(\mathbf{x}) - \min(\mathbf{x})}$$

$$\text{zeropoint} = -\text{round}(\text{scale} \times \min(\mathbf{x})) - 128$$

$$x_i^{\text{quant}} = \text{round}(\text{scale} \times x_i + \text{zeropoint})$$

$$x_i^{\text{dequant}} = \frac{x_i^{\text{quant}} - \text{zeropoint}}{\text{scale}}$$

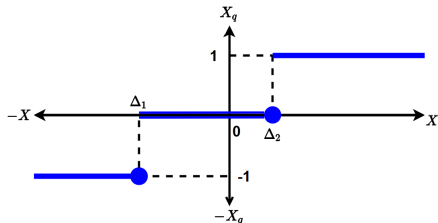
Asymmetric Quantization

```
def zeropoint_quantize(X):  
    # Calculate value range (denominator)  
    x_range = torch.max(X) - torch.min(X)  
    x_range = 1 if x_range == 0 else x_range  
  
    # Calculate scale  
    scale = 255 / x_range  
  
    # Shift by zero-point  
    zeropoint = (-scale * torch.min(X) - 128).round()  
  
    # Scale and round the inputs  
    X_quant = torch.clip((X * scale + zeropoint).round(), -128, 127)  
  
    # Dequantize  
    X_dequant = (X_quant - zeropoint) / scale  
  
    return X_quant.to(torch.int8), X_dequant
```

Ternary Quantization

A mathematical transform that converts a continuous real-valued variable into a discrete variable with three states

- ▶ Weights and/or activations are constrained to the ternary symbols $\{-1, 0, +1\}$
- ▶ Claim to provide near full-precision performance and high model compression
- ▶ You can come-up with new clever approaches



Sparse Ternary Compression (STC)¹

1 **input:** flattened tensor $T \in \mathbb{R}^n$, sparsity p
2 **output:** sparse ternary tensor $T^* \in \{-\mu, 0, \mu\}^n$
3 $\cdot$ $k \leftarrow \max(np, 1)$
4 $\cdot$ $v \leftarrow \text{top}_k(|T|)$
5 $\cdot$ $\text{mask} \leftarrow (|T| \geq v) \in \{0, 1\}^n$
6 $\cdot$ $T^{\text{masked}} \leftarrow \text{mask} \odot T$
7 $\cdot$ $\mu \leftarrow \frac{1}{k} \sum_{i=1}^n |T_i^{\text{masked}}|$
8 **return** $T^* \leftarrow \mu \times \text{sign}(T^{\text{masked}})$

¹ Sattler et al., "Robust and communication efficient federated learning from non-iid data", IEEE Trans. on Neural Networks and Learning Systems, 2019

Sparse Ternary Compression (STC)

		-3	-3	1.8	1.5
	1.1	.8	-.5	1.1	-1.0
1.5	-1.3	2.7	-.4	1.7	-.6
3.1	-.6	2.3	1.4	1.6	2.6
-1.8	.5	.9	-3.2	.4	2.3
1.0	2.5	.3	1.1	-3.2	
.7	-.1	1.8	-.7		

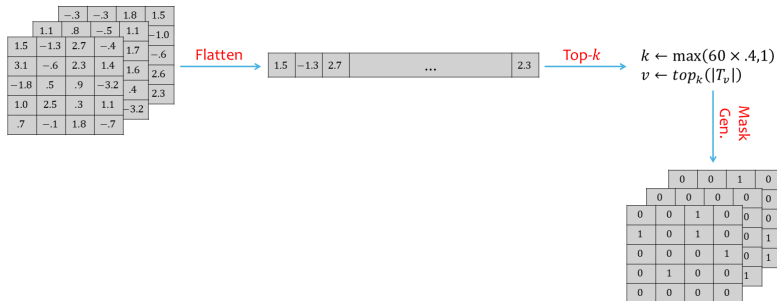
Sparse Ternary Compression (STC)



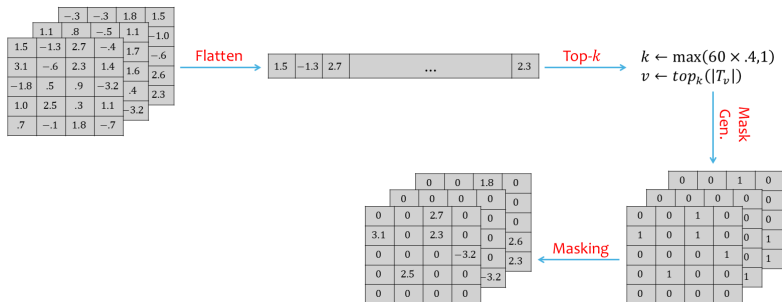
Sparse Ternary Compression (STC)



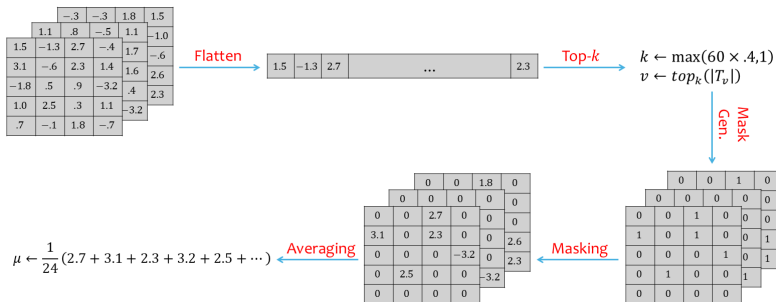
Sparse Ternary Compression (STC)



Sparse Ternary Compression (STC)



Sparse Ternary Compression (STC)



A mathematical transform that maps entries of weights and activation tensors of a NN to $\{-1, 1\}$

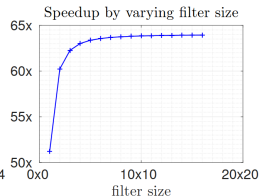
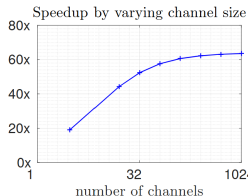
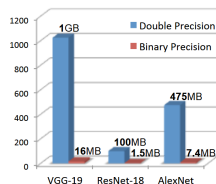
- ▶ A pure binary neural network usually fails on large-scale dataset
- ▶ XNOR-net, a well-known BNN, incorporates a real-coefficient

Sparse Binary Compression (SBC)²

```
1 input: tensor  $\Delta W$ , sparsity  $p$ 
2 output: sparse tensor  $\Delta W^*$ 
3 •  $\text{val}^+ \leftarrow \text{top}_{p\%}(\Delta W)$ ;  $\text{val}^- \leftarrow \text{top}_{p\%}(-\Delta W)$ 
4 •  $\mu^+ \leftarrow \text{mean}(\text{val}^+)$ ;  $\mu^- \leftarrow \text{mean}(\text{val}^-)$ 
5 if  $\mu^+ \geq \mu^-$  then
6 |   return  $\Delta W^* \leftarrow \mu^+(W \geq \min(\text{val}^+))$ 
7 else
8 |   return  $\Delta W^* \leftarrow -\mu^-(W \leq -\min(\text{val}^-))$ 
9 end
```

²Sattler et al., "Sparse binary compression: towards distributed deep learning with minimal communication", International Joint Conference on Neural Networks (IJCNN), 2019

Performance of Binary Networks³



	ResNet-18		GoogLenet	
Network Variations	top-1	top-5	top-1	top-5
Binary-Weight-Network	60.8	83.0	65.5	86.1
Full-Precision-Network	69.3	89.2	71.3	90.0

³ Rastegari et al. "Xnor-net: Imagenet classification using binary convolutional neural networks." In European Conference on Computer Vision (ECCV), pp. 525-542, 2016.

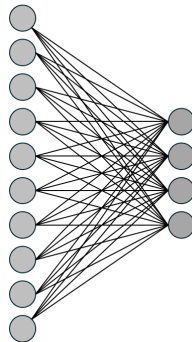
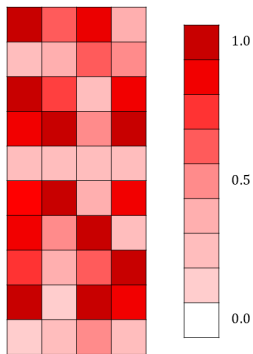
Zeroing out unimportant elements of parameter tensors of NNs.
The resulting model is

- ▶ More compressible using entropy coding
 - ▶ symbol distribution is highly skewed
 - ▶ assign shorter bit-lengths to more frequent symbols
- ▶ Faster during inference (if structured and highly sparse)
 - ▶ multiplying by zero is redundant and can be skipped
- ▶ Accurate

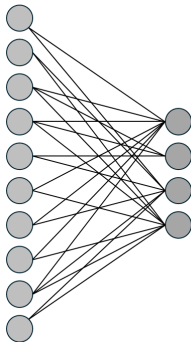
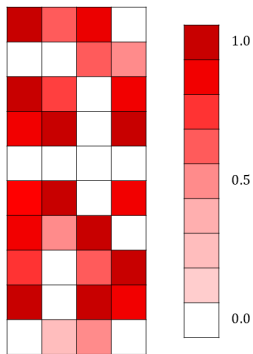
Sparsification: Categories

- ▶ Unstructured sparsity: individual weights are set to zero without any constraints on their positions
 - ▶ fine-grained sparsity
 - ▶ hard for hardware to exploit for speedup
- ▶ Structured sparsity: an entire group of weights (neurons, channels, filters, rows, or block) are zeroed out
 - ▶ coarse-grained sparsity
 - ▶ Maintains structured weight matrices
 - ▶ Easily supported by deep learning libraries (e.g., PyTorch, TensorFlow)
 - ▶ hardware friendly and faster inference

Sparsification: Overview



Sparsification: Overview



Top- K Sparsification

Idea: only communication/send the top- k largest weights/gradients

- ▶ Flatten weight/gradient tensor
- ▶ Select top- k entries by magnitude
- ▶ Transmit indices + values of the top- k

Advantages

- ▶ Dramatically reduces communication cost
- ▶ The most informative updates are preserved

Limitations

- ▶ Adds overhead of index transmission
- ▶ Needs error feedback mechanisms to avoid accuracy drop

Sparsity-Inducing Regularization

Idea: Encourage weights to shrink to zero by adding an L1 penalty term to the loss function

$$\min_{\mathbf{W}} \sum_{i=1}^n \mathcal{L}(\mathbf{W}) + \lambda \|\mathbf{W}\|_1$$

Advantages

- ▶ Simple to implement (just a regularization term)
- ▶ Produces natural unstructured sparsity

Limitations

- ▶ Zeros are scattered irregularly $\rightarrow$ not hardware-friendly
- ▶ Sparsity level depends on regularization strength λ

Structured Sparsification: Filter-level Sparsification

Idea: Instead of sparsifying individual weights, zero-out an entire filter (or feature map) from convolutional layers.

Key motivation:

- ▶ Reduces both communication *and* computation costs.
- ▶ Better hardware compatibility compared to unstructured sparsity.

Approach:

- ▶ Compute an importance score for each filter (e.g., ℓ_2 -norm of its weights).
- ▶ Retain top- k filters per layer and set others to zero.
- ▶ Communicate only active filters to the server.

Example:

$$S_j = \|W_j\|_2, \quad \text{keep top-}k \text{ filters with largest } S_j$$

Structured Sparsification: Layer-wise Sparsification

Goal: Selectively drop entire layers or compress their outputs to reduce communication.

Two common strategies:

- ▶ **Layer Skipping:** Skip gradient updates for less sensitive layers (e.g., early convolutional layers).
- ▶ **Layer-wise Compression:** Transmit gradients only for a subset of layers per round.

Example: Periodic Layer Update

$$g_l^{(t)} = \begin{cases} \nabla_l^{(t)} & \text{if } t \bmod f(l) = 0 \\ 0 & \text{otherwise} \end{cases}$$

where $f(l)$ controls update frequency for layer l .

Pruning

Definition: Removing an entire unit, e.g. filter or layer, that contribute little to the output.

- ▶ Considered as a form of structured sparsification
- ▶ Originally proposed for model compression, but also applicable for reducing communication cost in FL

Workflow:

- ▶ Train the full model locally.
- ▶ Compute importance of weights or neurons (e.g., via magnitude or sensitivity analysis).
- ▶ Remove (prune) the least important components.
- ▶ Fine-tune the remaining model.

Filter Pruning by ℓ_1 -norm

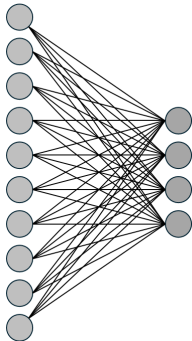
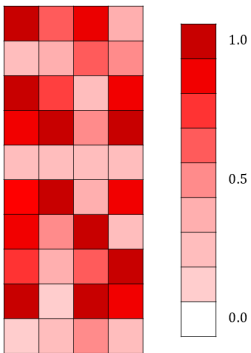
Prune filters j with $\|W_j\|_1 < \tau$

where τ is a threshold controlling sparsity level.

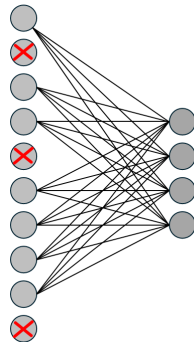
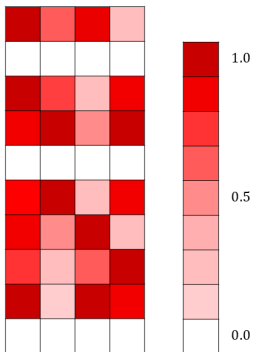
Effects

- ▶ Reduces uplink communication from clients to server.
- ▶ Lowers inference latency on resource-limited devices.
- ▶ May require retraining to recover accuracy.

Pruning



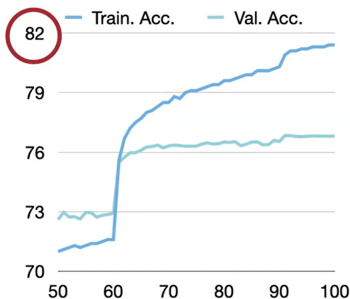
Pruning



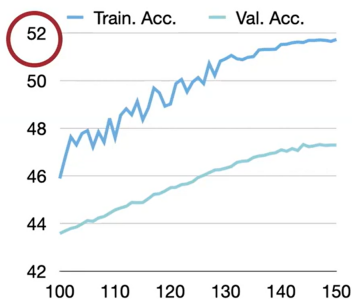
Knowledge Distillation

Small models underfit large datasets

ResNet50 on ImageNet

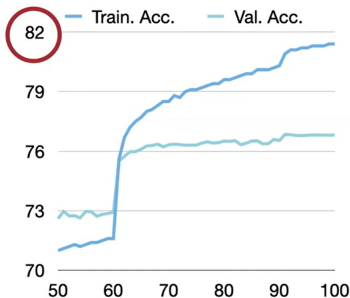


MobileNetv2-Tiny on ImageNet

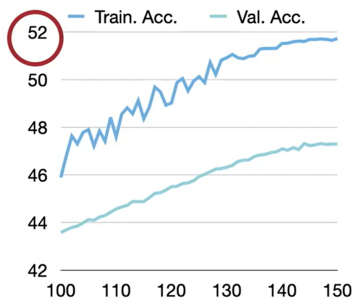


Small models underfit large datasets

ResNet50 on ImageNet



MobileNetv2-Tiny on ImageNet



can we improve the training of tiny models with the help of large models?

Distilling the Knowledge in a Neural Network

Geoffrey Hinton^{*†}

Google Inc.
Mountain View

geoffhinton@google.com

Oriol Vinyals[†]

Google Inc.
Mountain View

vinyals@google.com

Jeff Dean

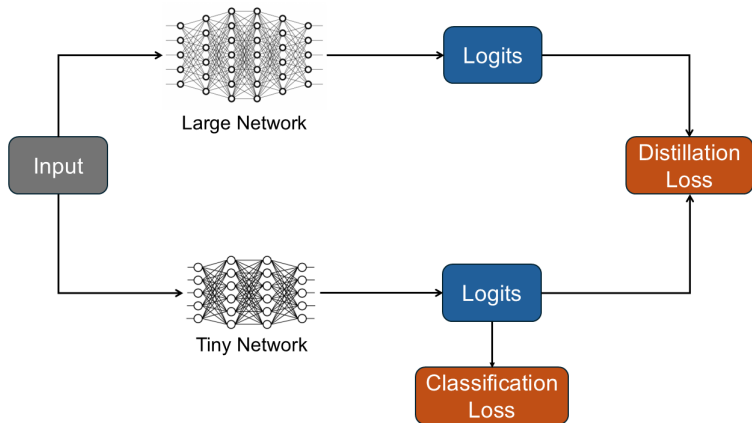
Google Inc.
Mountain View

jeff@google.com

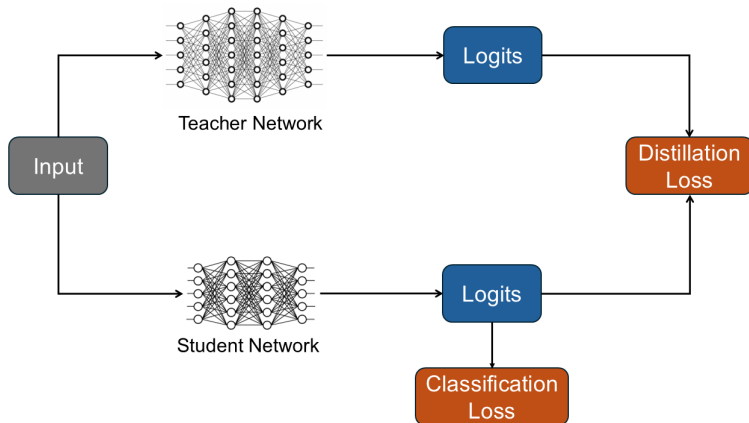
Abstract

A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions [3]. Unfortunately, making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural nets. Caruana and his collaborators [1] have shown that it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy and we develop this approach further using a different compression technique. We achieve some surprising results on MNIST and we show that we can significantly improve the acoustic model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

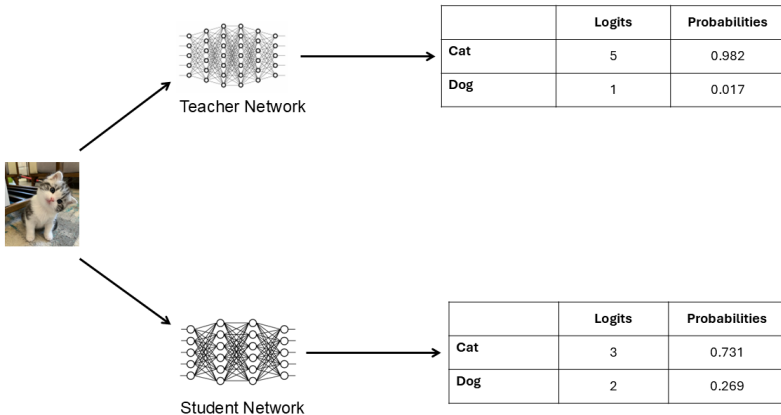
Overview



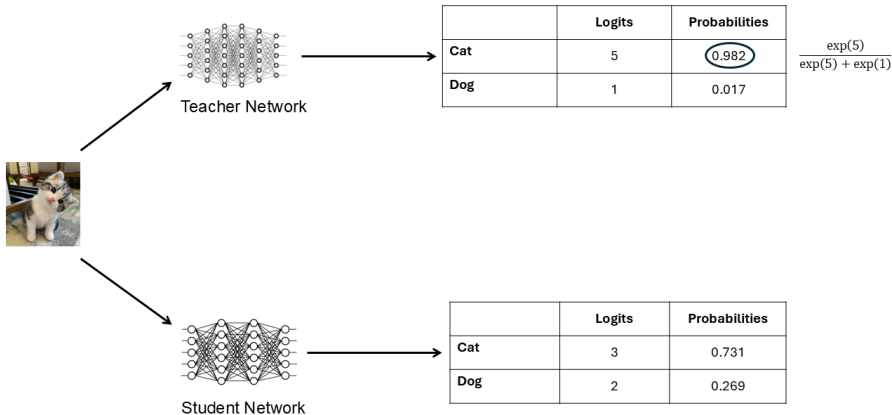
Larger network is called “Teacher” and smaller one is called “Student”



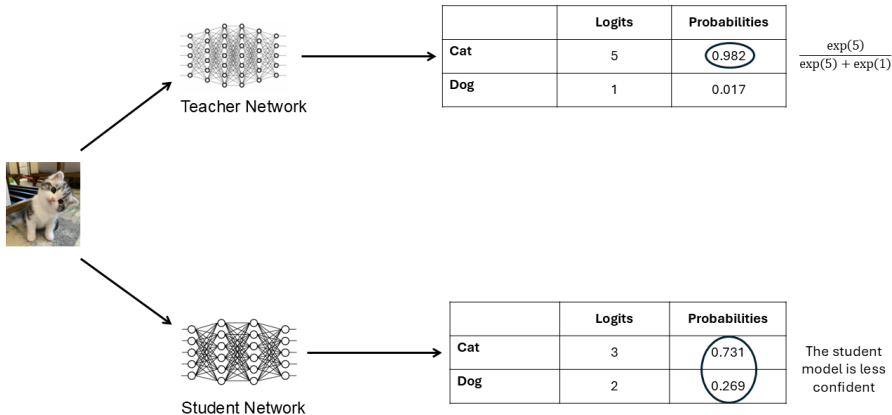
Intuition



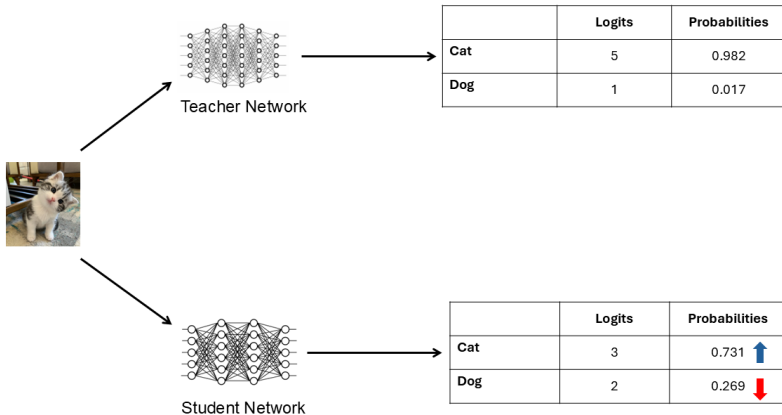
Intuition



Intuition



Intuition



Formal Definition

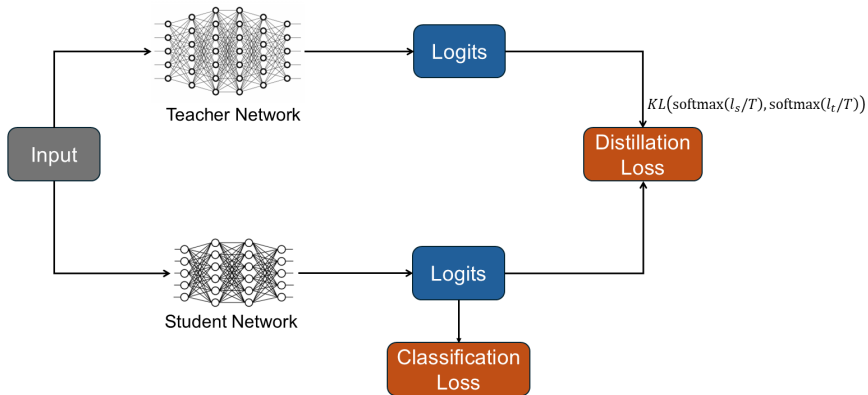
Neural networks typically use softmax to generate logits z_i to class probabilities

$$p(z_i, T) = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

where, $i, j = 0, 1, \dots, C - 1$ are class indices when there is in total C classes. T is normally set to 1

Given a large (teacher) model, the goal of KD is to train a small (student) model whose class probabilities align with the class probabilities of the teacher network.

Formal Definition

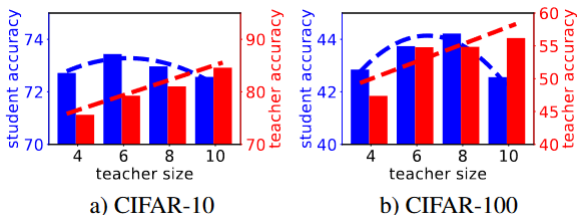


Student's loss function will be:

$$\mathcal{L}_{student} = (1 - \lambda) \mathcal{L}_{classification} + \lambda \mathcal{L}_{KD}$$

What Is The Best Student Size?

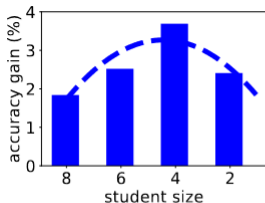
Q1: Given a fixed student network, what is the best teacher size?



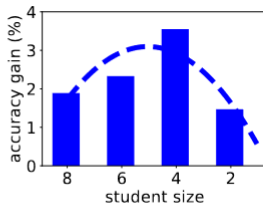
Mirzadeh et al. "Improved KD via Teacher Assistant", 2020

What Is The Best Student Size?

Q2: Given a fixed teacher network, what is the best student size?



a) CIFAR-10



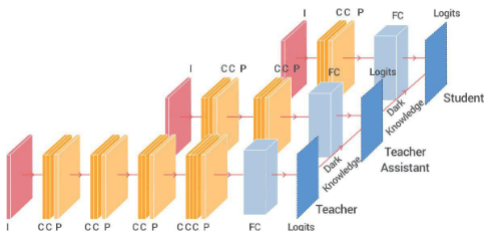
b) CIFAR-100

Mirzadeh et al. "Improved KD via Teacher Assistant", 2020

- ▶ By increasing teacher's size, it's performance increases and provides better supervision for the student
- ▶ The teacher is becoming so complex that the student does not have sufficient capacity to mimic teacher's behavior

Solution: Teacher Assistant!

- ▶ TA lies in between teacher and student in terms of size
- ▶ TA is distilled from teacher
- ▶ TA plays the role of a teacher for the student



Mirzadeh et al. "Improved KD via Teacher Assistant." Proceedings of the AAAI Conference on Artificial Intelligence, pp. 5191-5198, 2020