Banff International Research Station
for Mathematical Innovation and Discovery

Communication-Efficient Federated Learning: Challenges, Techniques, and Insights

Stefano Rini

陽明交大
NYCU

CA friends:

Yangyi Liu
Sadaf Salehkalaibar
Jun Chen

TW friends:

Zhong-Jing Chen,
Eduin Hernandez
Jerry Huang

Disclaimer

- Due some miscommunication between me and the organizers I was not able to give the talk in person
- I recorded the video asynchronous here: please enjoy!

Communication-Efficient Federated Learning

Communication-Efficient Federated Learning

Learning refers to a machine learning task: you can think of regression, classification, clustering, dimensionality reduction, structured prediction. This task can be implemented through DL or some other ML algorithm. The setting is quite general

Federated – Dictionary definition: “set up as a single centralized unit within which each state or division keeps some internal autonomy”. IE: One server, many remote users

Communication-Efficient: communication is limited in some sense: transmission rates, packet loss, connectivity, delays…you name it

J. Koneˇcn`y, H. B. McMahan, F. X. Yu, P. Richt´arik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” arXiv preprint arXiv:1610.05492, 2016.
T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” arXiv preprint arXiv:1908.07873, 2019.

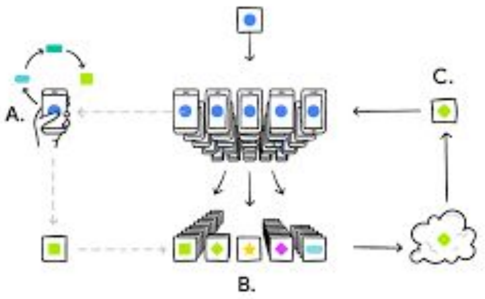
Federated learning

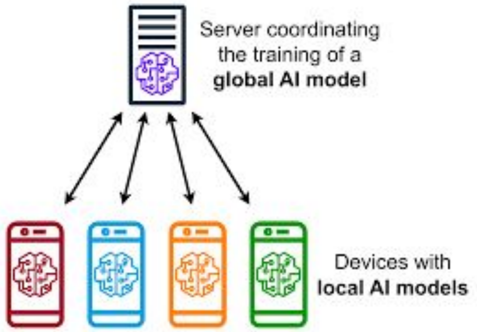
Introduced in the Google AI blog in 2017

The idea is simple

- Remote devices have a local dataset
- Each of the remote device trains a global AI model over this local data
- The model updates are transmitted to the parameter server (PS)
- The PS aggregates the model updates
- Model is redistributed to all clients

...repeat until convergence




Server coordinating the training of a **global AI model**
Devices with **local AI models**

Federated learning

Everything is perfect, right?

- ➔ Many advantages
  - ◆ Avoid data centralization
    - Robust
    - Scalable
    - Private
    - Anonymous
- ➔ Some disadvantages
  - ◆ Communication is onerous
  - ◆ Requires remote users to have uniform computational capabilities

A great unfulfilled promise – for now…

- ➔ Very few applications - for now
  - ◆ Train a speech model on user data without collecting the user data
    - Amazon Alexa
  - ◆ Train a keyboard with autocorrect features without collecting the user data
    - Google keyboard

Machine learning settings

Let us introduce some notation quickly

- We have a **dataset**

$$\mathcal{D} = \{\mathbf{d}_k\}_{k \in [|\mathcal{D}|]}$$

- We have a **loss function**

$$\mathcal{L}(\mathbf{w}) = \frac{1}{[|\mathcal{D}|]} \sum_{k \in [|\mathcal{D}|]} \mathcal{L}(\mathbf{w}, \mathbf{d}_k)$$

- We have a **model** that is updated through the learning process until we reach the optimal solution

$$\mathbf{w}_0 \longrightarrow \mathbf{w}_t \longrightarrow \mathbf{w}^*$$

Machine learning settings

Q: How does learning happen?

A: In many ways

… but the more general setting that we have found so far is also the simplest:

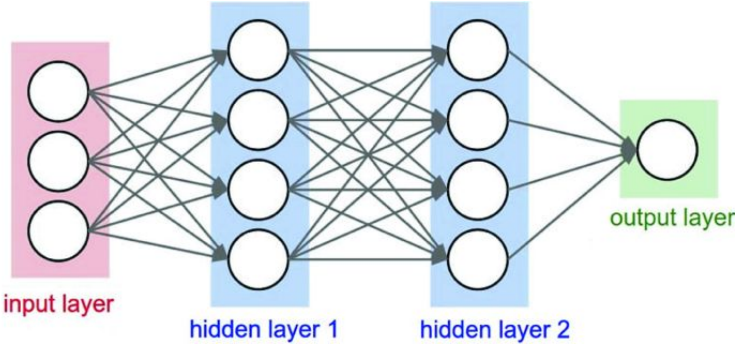
- Stochastic gradient descent (SGD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\mu}{[|\mathcal{D}|]} \sum_{k \in [|\mathcal{D}|]} \frac{\partial \mathcal{L}(\mathbf{w}, \mathbf{d}_k)}{\partial \mathbf{w}}$$

Deep Learning

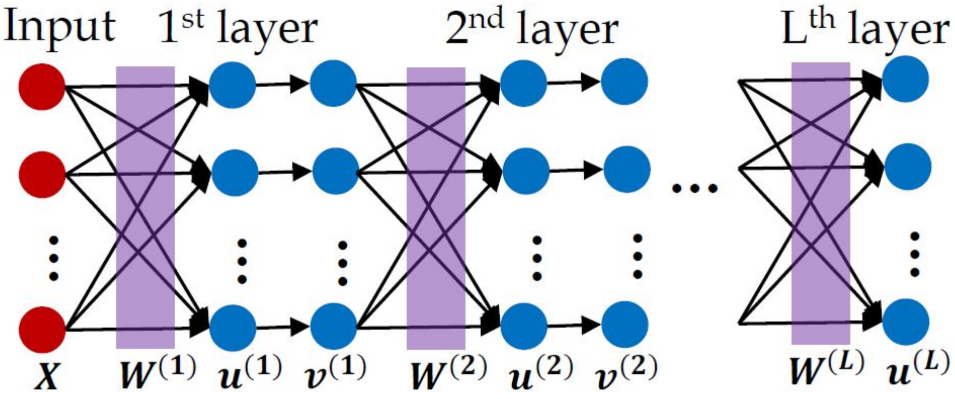
Deep neural networks are a variation of the above setting
Nothing more, nothing less

- SGD can be performed very efficiently
- The function implemented by DNNs are highly non-linear
- DNNs naturally show robustness
- Robustness comes at the cost of high redundancy.



Communication-Efficient Federated Learning

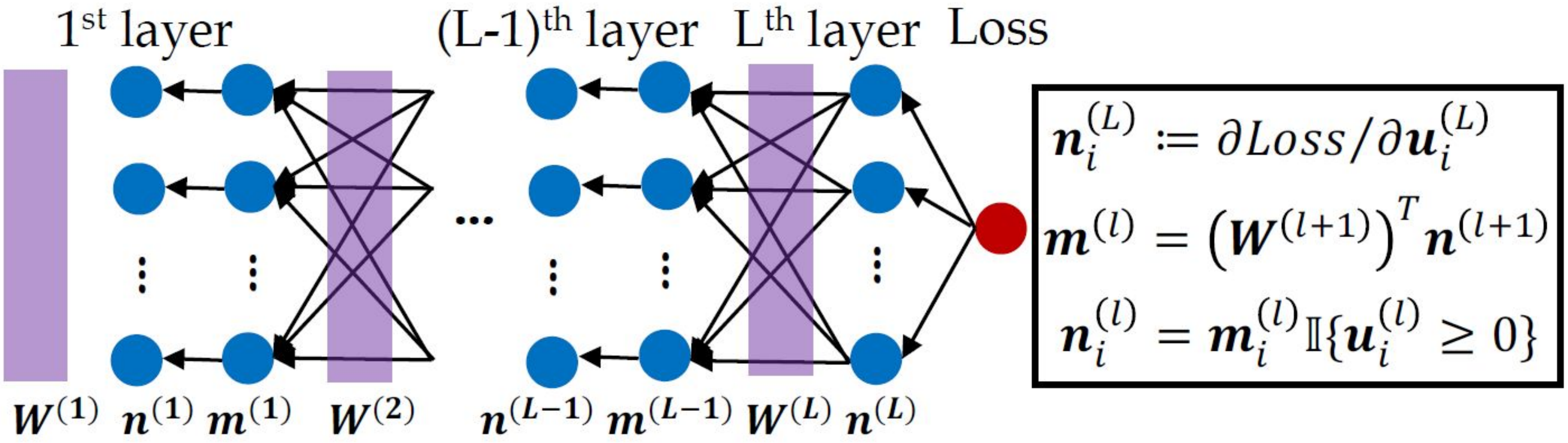
Forward propagation



$$
\boxed{
\begin{aligned}
\boldsymbol{u}^{(0)} &:= \boldsymbol{X} \\
\boldsymbol{u}^{(l)} &= \boldsymbol{W}^{(l)}\boldsymbol{v}^{(l-1)} \\
\boldsymbol{v}_i^{(l)} &= \mathrm{ReLU}\left(\boldsymbol{u}_i^{(l)}\right) \\
&= \boldsymbol{u}_i^{(l)}\mathbb{I}\{\boldsymbol{u}_i^{(l)} \geq 0\}
\end{aligned}
}
$$

Communication-Efficient Federated Learning

Backward propagation


1st layer
(L-1)th layer
Lth layer
Loss
$W^{(1)}$ $n^{(1)}$ $m^{(1)}$ $W^{(2)}$
$n^{(L-1)}$ $m^{(L-1)}$ $W^{(L)}$ $n^{(L)}$

$$\boldsymbol{n}_i^{(L)} := \partial Loss / \partial \boldsymbol{u}_i^{(L)}$$

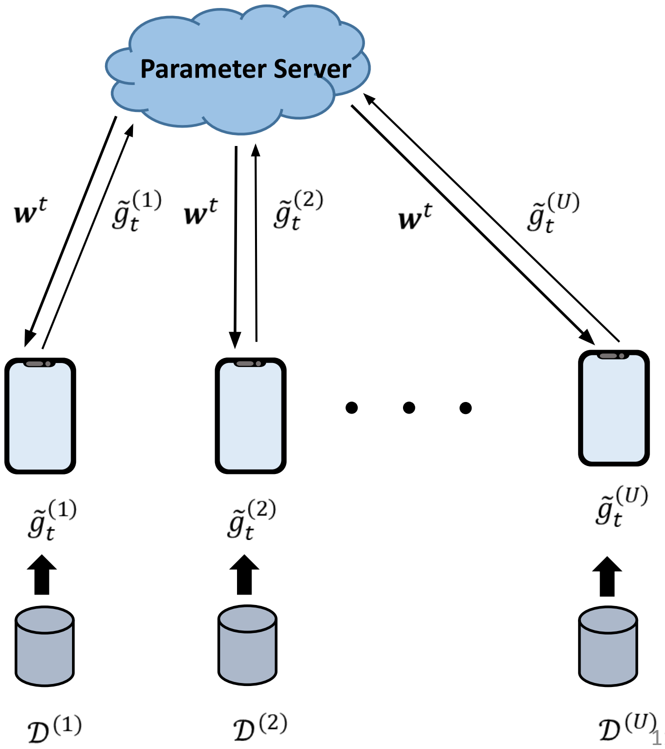
$$\boldsymbol{m}^{(l)} = \left(\boldsymbol{W}^{(l+1)}\right)^T \boldsymbol{n}^{(l+1)}$$

$$\boldsymbol{n}_i^{(l)} = \boldsymbol{m}_i^{(l)} \mathbb{I}\{\boldsymbol{u}_i^{(l)} \geq 0\}$$

Federated learning

Ok, now back to our federated learning scenario:

- Many users, one central parameter server
- Data is remains at the remote users
- Computation is performed at the remote users
- Updates are sent to the PS
- PS averages updates and te model is sent



Federated Learning

Some more details

- N remote users
- The dataset of user n is

$$\mathcal{D}_n = \{\mathbf{d}_{nk}\}_{k \in [|\mathcal{D}_n|]}$$

- Users compute their local gradient over the local dataset, let's call it g_t

$$\mathbf{g}_{nt} = \frac{1}{[|\mathcal{D}_n|]} \sum_{k \in [|\mathcal{D}_n|]} \frac{\partial \mathcal{L}(\mathbf{w}_t, \mathbf{d}_{nk})}{\partial \mathbf{w}_t}$$

- The server aggregates the gradients

$$\mathbf{g}_t = \frac{1}{N} \sum_{n \in [N]} \mathbf{g}_{nt}.$$

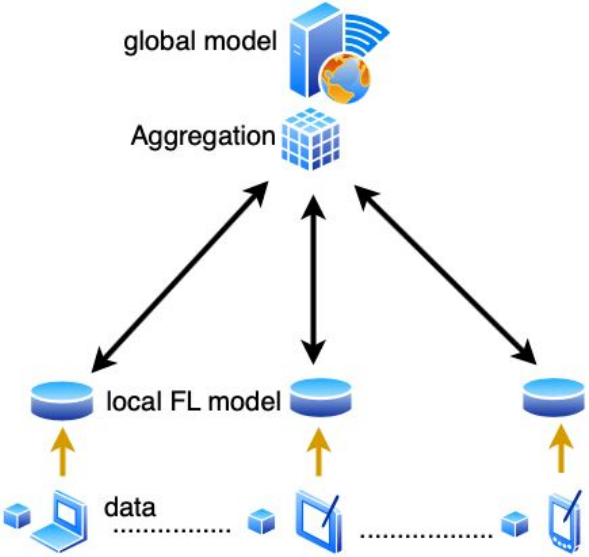
Deng, Yuyang, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. "Distributionally robust federated averaging." *Advances in neural information processing systems* 33 (2020): 15111-15122.

Communication-Efficient Federated Learning

Let us assume *a more realistic scenario*

➔ The communication between each remote user and the PS is limited
  - ◆ R scalars
  - ◆ R bits
  - ◆ In expectation
  - ◆ Deterministic

➔ The communication between the PS and the remote users is unbounded
  - ◆ The PS is not limited it transmission capabilities



N. Shlezinger, S. Rini, and Y. C. Eldar, “The communication-aware clustered federated learning problem,” in 2020 IEEE International Symposium on Information Theory (ISIT). IEEE, 2020, pp. 2610–2615.
D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, “QSGD: Communication-efficient sgd via gradient quantization and encoding,” Advances in Neural Information Processing Systems, vol. 30, 2017.
J. Wangni, J. Wang, J. Liu, and T. Zhang, “Gradient sparsification for communication-efficient distributed optimization,” Advances in Neural Information Processing Systems, vol. 31, 2018.

Communication-Efficient Federated Learning

Finally, we have a communication problem in our hands: fit the gradients in the prescribed rate constraints

This is a rate-distortion problem, as long as:

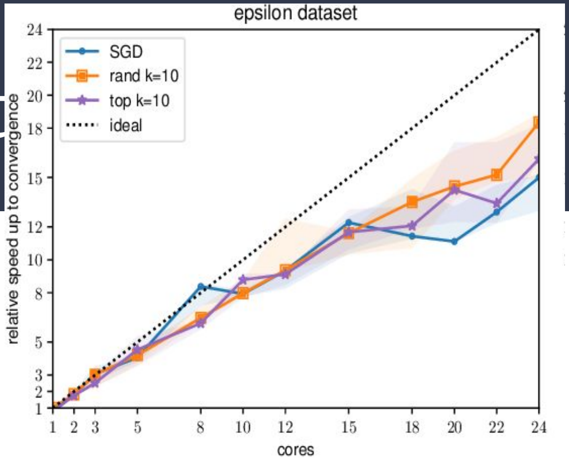
- **Source:** treat the gradient across iterations as a random process
- **Distortion:** find how a distortion in the gradient reconstruction affects the learning accuracy

Call this problem **gradient compression**

Apply some classic techniques to chase the information theoretical optimal performance:

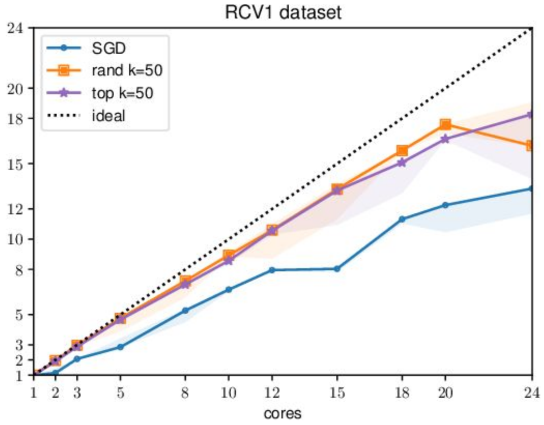
- ➔ Quantization
- ➔ Lossless source coding
- ➔ Variable length coding
- ➔ Universal compression
- ➔ …

Communication-Efficient Federat



Before we begin, what's the **simplest** strategy we can use?
Does this give indication that the system is **robust** to gradient compression?

- More than that: we know that the system enjoys a bit of additional noise
  - Practitioners have figure out that sparsification really helps
    - **Rand_k**
      Keep k random gradient entries, set the rest to zero
    - **Top_k**
      Keep the k largest gradient entries, set the rest to zero



D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," arXiv preprint arXiv:1809.10505, 2018.
S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in Advances in Neural Information Processing Systems 31, 2018, pp. 4448–4459.

Communication-Efficient Federated Learning

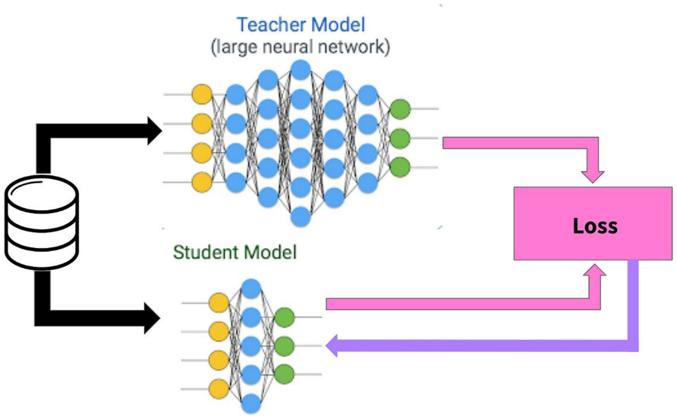
Now we can formulate an “engineering” assumption

Assumption:
Gradient entries are independent and identically distributed in each layer and across each iteration

- ➔ Now we have to discuss which distribution better fits our observations…
  - ◆ But how can we setup a number of learning instances in which we can gather enough statistical significance?
    - We can take some classical learning problem and networks
      - ○ What else?

Teacher/student network

- **Teacher network:** is a FIXED network which we initialize randomly
- Apply standard Gaussian iid inputs and produce an output
- **Student network**: use the output of the teacher network to train this network to produce the same set of inputs under L2 loss



This is the simplest learning problem we can “produce” in large quantity

Goldt, Sebastian, et al. "Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup." Advances in neural information processing systems 32 (2019).

Gradient distribution modelling

Back to our problem: modelling the gradient entry distribution

In the literature:

- **1 parameter**
  - Gaussian
  - Laplacian
- **2 parameters**
  - Double-Weibull

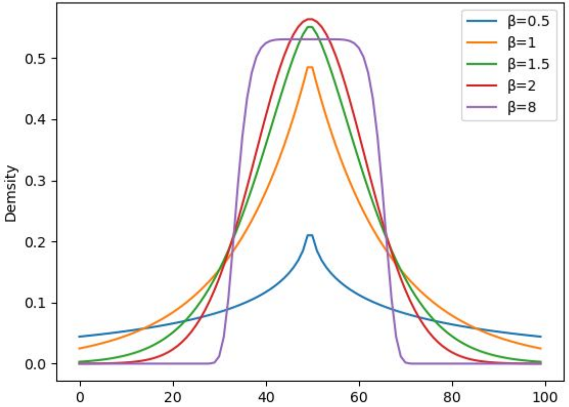
More generally, sub-Weibull distribution $\mathbb{P}(|X| \geq x) \leq \exp\left(-x^{1/\theta}/K\right)$ *for all* $x \geq 0.$

F. Fu, Y. Hu, Y. He, J. Jiang, Y. Shao, C. Zhang, and B. Cui, "Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript," in International Conference on Machine Learning. PMLR, 2020, pp. 3304–3314.
B. Isik, A. No, and T. Weissman, "Successive pruning for model compression via rate distortion theory," arXiv preprint arXiv:2102.08329, 2021.

Gradient distribution modelling

Our proposed distribution:

- Generalized Gaussian(GenNorm)



$\underline{\beta = 1 \rightarrow Laplace\ distribution}$
$\underline{\beta = 2 \rightarrow Normal\ distribution}$

$$GenNorm(x, \mu, \alpha, \beta) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\{-(\frac{|x-\mu|}{\alpha})^{\beta}\}$$

$$Mean = \mu$$

$$Variance = \frac{\alpha^2\Gamma(3/\beta)}{\Gamma(1/\beta)}$$

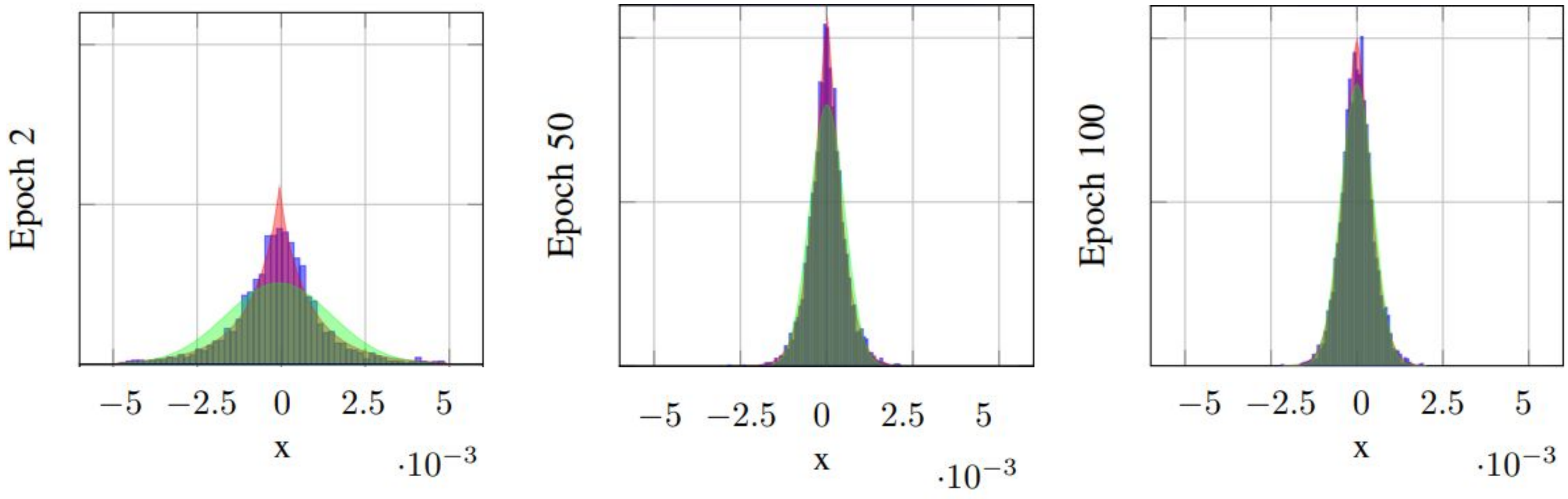
$$Kurtosis = \frac{\Gamma(5/\beta)\Gamma(1/\beta)}{\Gamma(3/\beta)^2}$$

- Histogram (upper layer of ResNet50V2):

🟩 Gradient

🟥 GenNorm pdf

🟩 Norm pdf



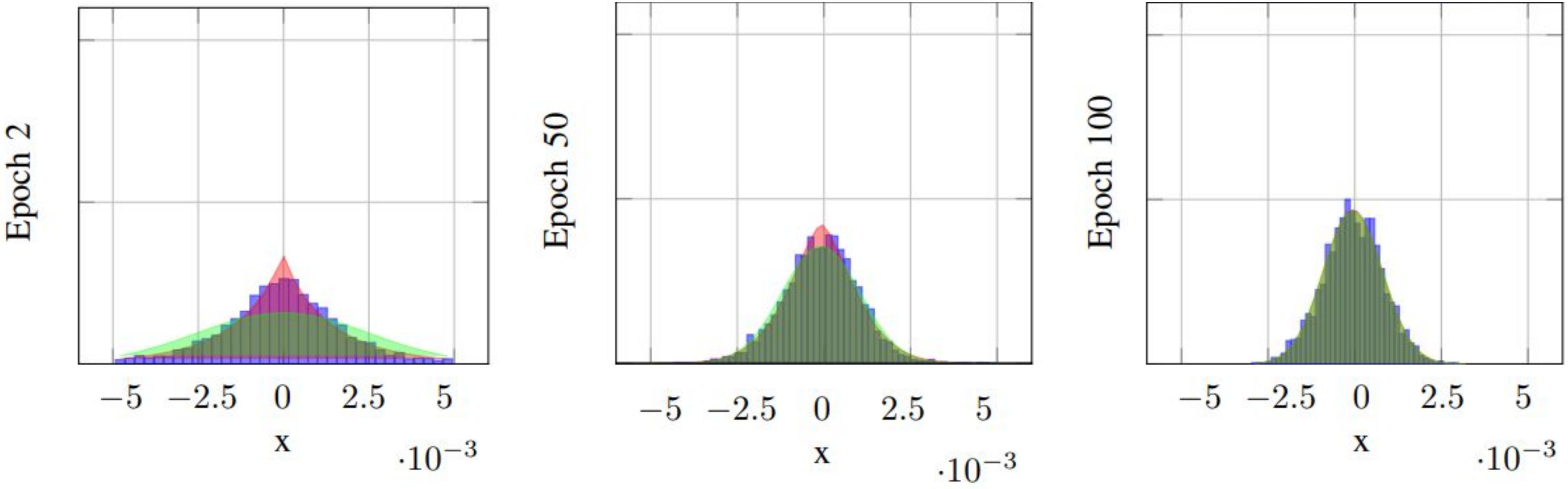
Gradient distribution modelling

- Histogram (upper layer of NASNetMobile):

🟩 Gradient

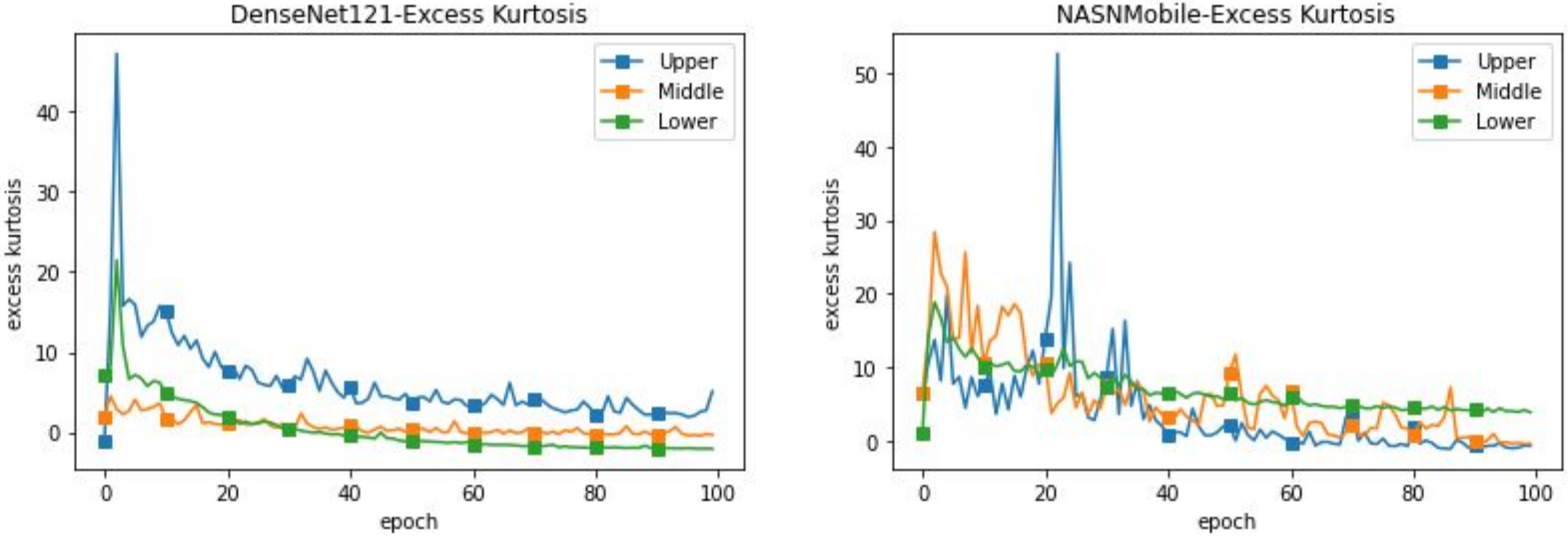
🟥 GenNorm pdf

🟩 Norm pdf



Gradient distribution modelling

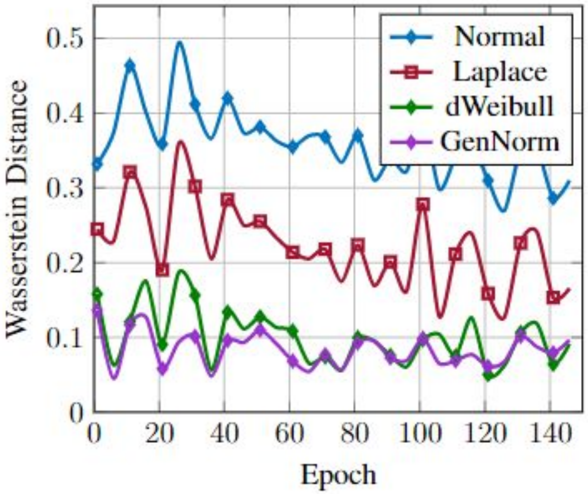
- Excess kurtosis:



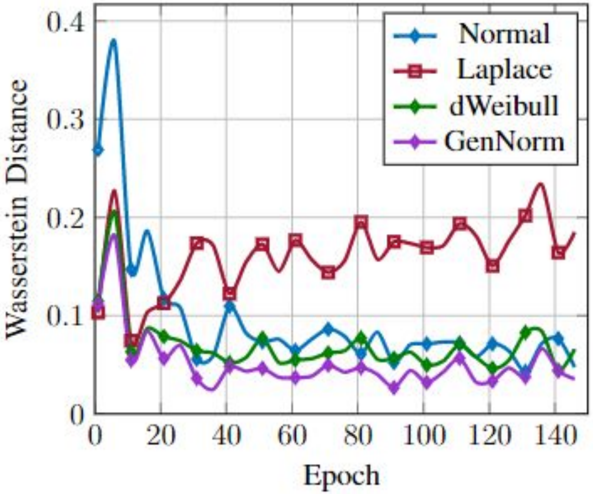
Gradient distribution modelling

- Wasserstein distance of order 2:

$$W_2(X,Y) = (\int_0^1 |F_X^{-1}(z) - F_Y^{-1}(z)|^2 dz)^{1/2}$$



DenseNet121(Upper layer)



NASNetMobile(Upper layer)

Gradient distribution modelling

Gradient modelling

Q: How about independence? Can we verify the independence assumption through some statistical test?

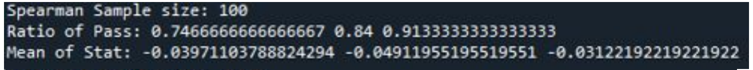
A: yes and no…it all comes down to the p-value…

Still, we can try two tests:

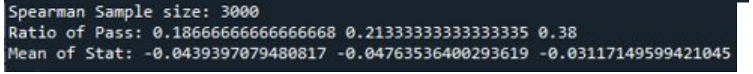
➔ Sperman rank correlation coefficient:
It assesses how well the relationship between two variables can be described using a monotonic function

➔ Kolmogorov-Smirnov test:
is a nonparametric goodness-of-fit test and is used to determine whether two groups of samples come from the same distribution

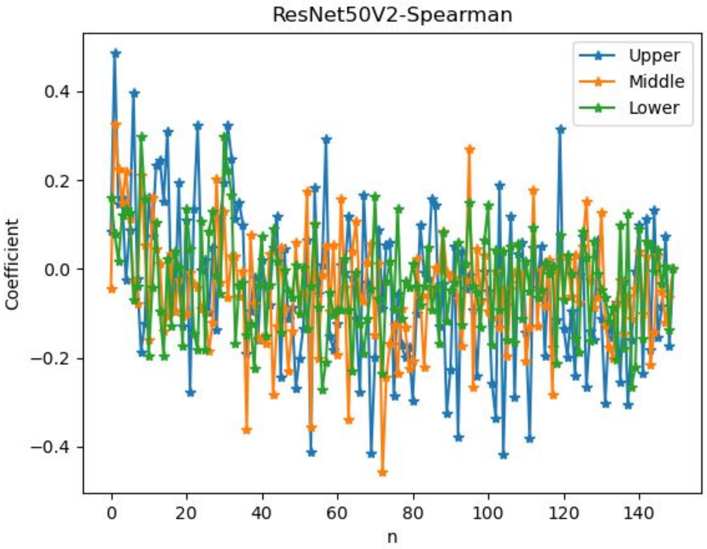
- Spearman’s Rank Correlation(full bits)
  - Tests whether two samples have a monotonic relationship.
  - The ratio of epoch passing test $\rho = \frac{\sum_i^N 1_{x>0.05}(p_i)}{N}$

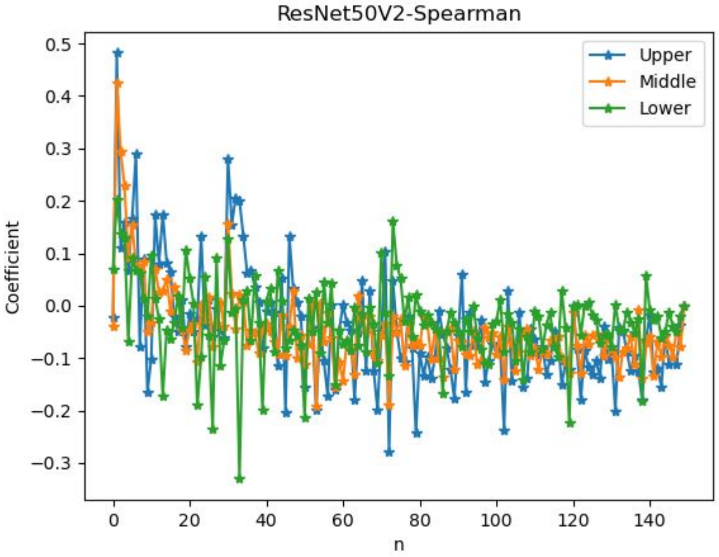


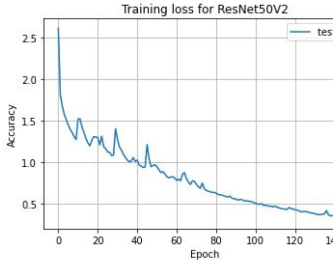
```
Spearman Sample size: 100
Ratio of Pass: 0.7466666666666667 0.84 0.9133333333333333
Mean of Stat: -0.03971103788824294 -0.04911955195519551 -0.03122192219221922
```



```
Spearman Sample size: 3000
Ratio of Pass: 0.18666666666666668 0.21333333333333335 0.38
Mean of Stat: -0.0439397079480817 -0.04763536400293619 -0.03117149599421045
```

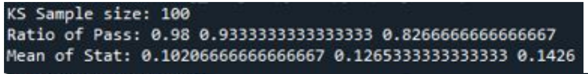




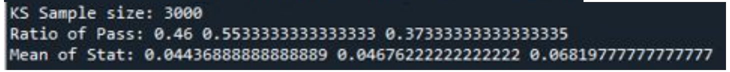


Gradient independence test

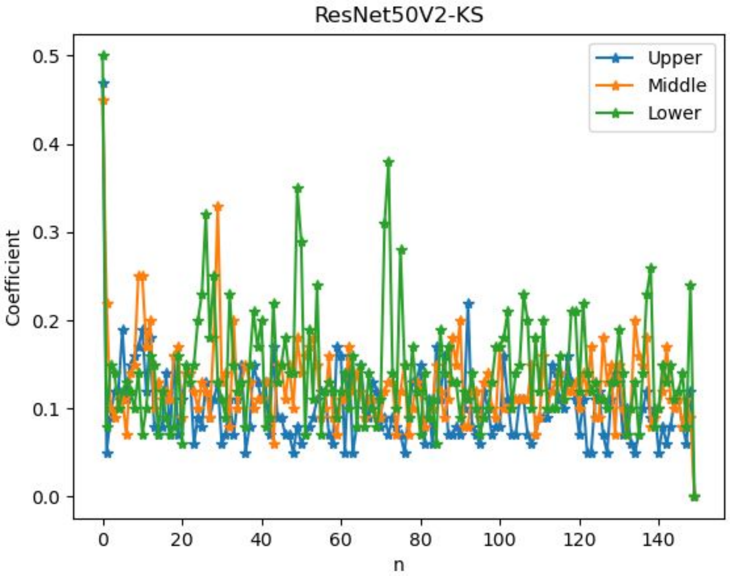
- Kolmogorov-Smirnov Test (full bits)
    - KS statistic: 0.5 -> not similar, 0.1-> similar

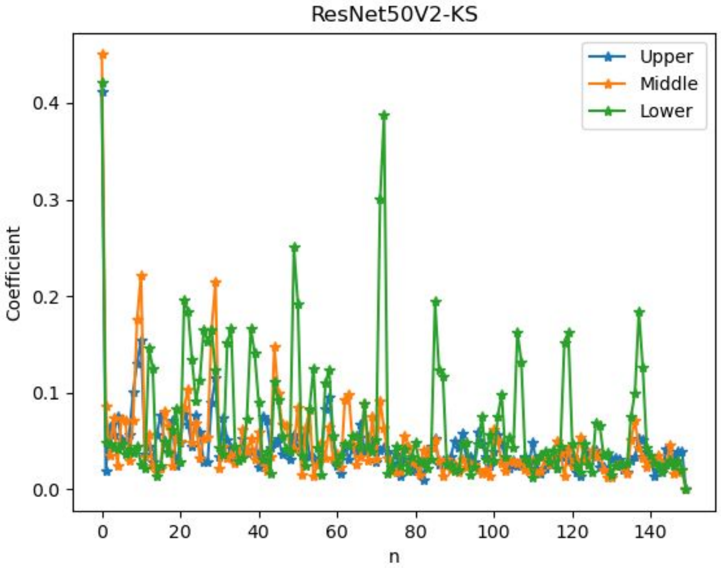


```
KS Sample size: 100
Ratio of Pass: 0.98 0.9333333333333333 0.8266666666666667
Mean of Stat: 0.10206666666666667 0.12653333333333333 0.1426
```



```
KS Sample size: 3000
Ratio of Pass: 0.46 0.5533333333333333 0.37333333333333335
Mean of Stat: 0.04436888888888889 0.04676222222222222 0.06819777777777777
```





Gradient independence test

Gradient modelling

Are we satisfied with this modelling? Yes and no…

- **YES:** We have a statistically significant set of simulations that validates the gen-norm assumption.
  - We can use the happily assume (as engineers) that gradients can be treated as gen-norm.

- **NO**
  We have no theoretical justification for the parameter evolution, in time and depth…
  - Even for the case of teacher/student network, this could be very insightful
  - Let alone new architectures, RNN, diffusion,...

Distortion modelling

Next the tricky part:

In source coding we need to describe the source and how to measure the distance between original samples and reconstructed samples

- What is the question we want to answer
  If we perturb the weights of epsilon, how much will the accuracy be perturbed?
  - Different learning problems
  - Different network sizes
  - Different input distributions

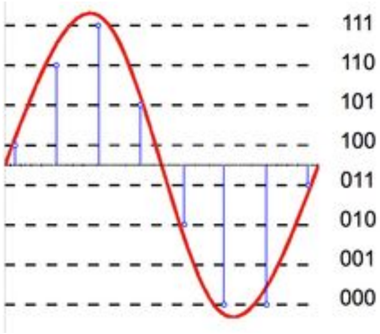
- How can we hope to obtain a general formulation?
  - From an engineering perspective, we are happy to have an approximate measure that we can adapt to the problem at hand
  - The teacher/student network always provides a nice setting to study

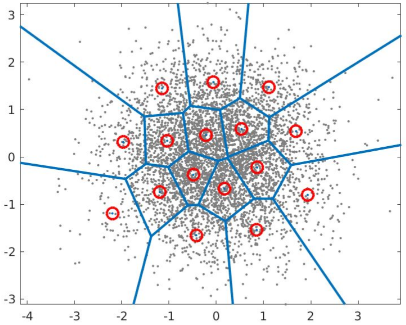
Distortion modelling

REMINDER:

Why do we need to know this relationship?

We want to build a quantizer: convert continuous values (i.e. fp16 numbers) to a set of discrete representations





This is a problem with a long history: for now let's just focus on one dimensional quantizers, so that lossless compression might be needed after quantization

Distortion modelling

M-magnitude weighted L2 distortion

a family of distortion measures with M being a tunable hyper-parameter: M

- IDEA:
  - Have a measure that bridges
    - Sparsification
    - L_2 loss

  - We want to skew the distortion to be larger for larger gradient values, even if the error is the same

$$d_{M-L_2}(\mathbf{g}, \widehat{\mathbf{g}}) = \frac{1}{d} \sum_{j \in [d]} |g_j|^M ||g_j - \hat{g}_j||_2$$

Now we can put things together

We call this scheme M22

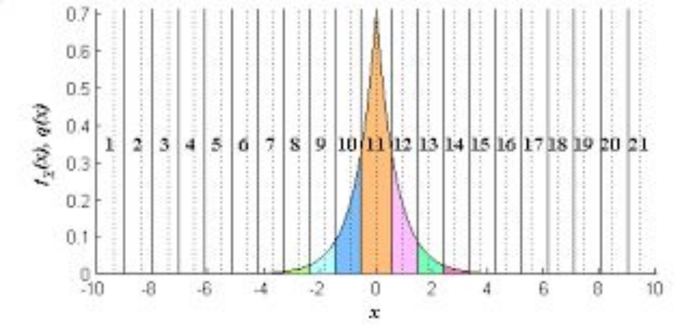
Recap

- ➔ Gradient iid Gen. norm
  - ◆ Fit the distribution at each layer and each iteration
- ➔ Optimize the hyperparameter M to minimize the loss of accuracy from compression
- ➔ Design the quantizer by minimizing the M-L2 distortion using the Max-Lloyd quantizer
- ➔ Compress each gradient, possibly apply universal lossless compression

Loss modelling

For the one-dimensional case, can design something like this

- Minimize the expected distortion over the sample probability



We can apply these two steps iteratively

Step 1

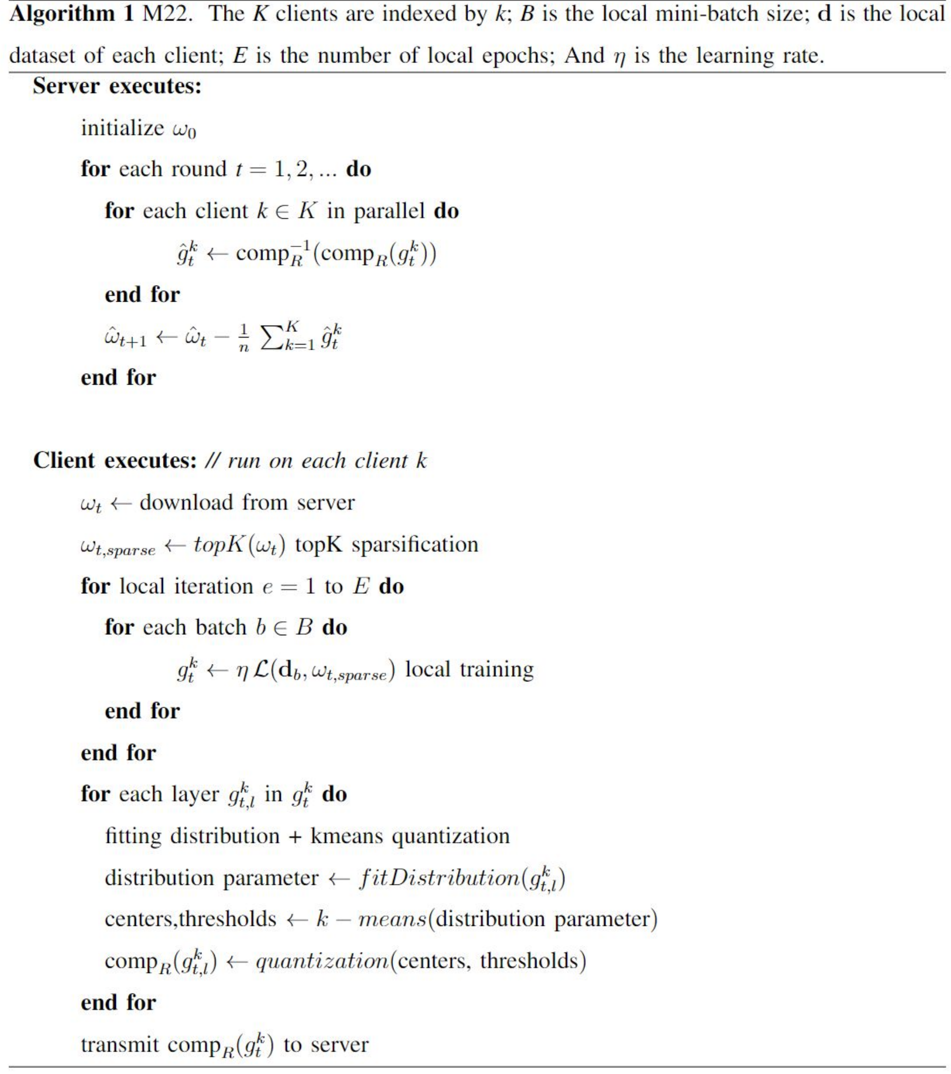
$$c_{k+1}(i+1) = \frac{\int_{t_k(i)}^{t_k(i+1)} g^{M+1}\,\text{pdf}\,(g)dg}{\int_{t(i)}^{t(i+1)} g^{M}\,\text{pdf}\,(g)dg}$$

Step 2

$$t_{k+1}(i+1) = \frac{c_k(i+1) + c_k(i)}{2}$$

M22

Pseudocode



Algorithm 1 M22. The K clients are indexed by k; B is the local mini-batch size; $\mathbf{d}$ is the local dataset of each client; E is the number of local epochs; And η is the learning rate.

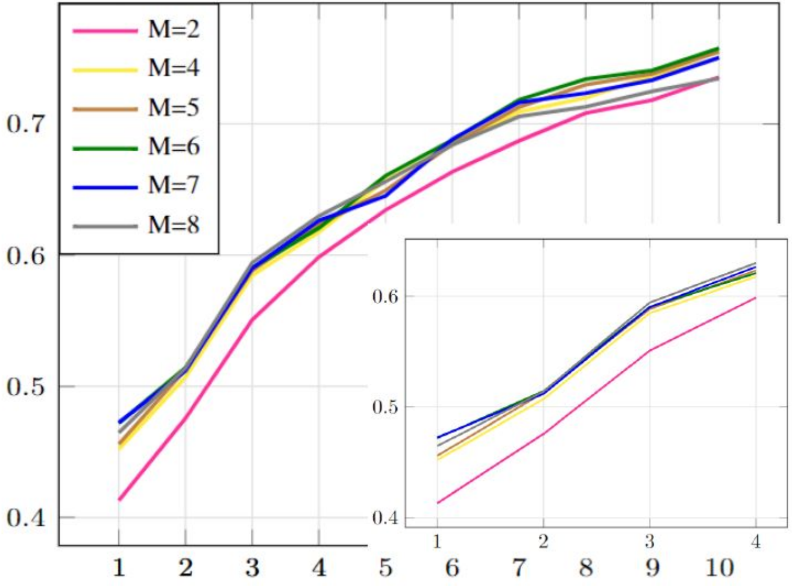
```
Server executes:
    initialize $\omega_0$
    for each round $t = 1, 2, \ldots$ do
      for each client $k \in K$ in parallel do
          $\hat{g}_t^k \leftarrow \text{comp}_R^{-1}(\text{comp}_R(g_t^k))$
      end for
      $\hat{\omega}_{t+1} \leftarrow \hat{\omega}_t - \frac{1}{n} \sum_{k=1}^{K} \hat{g}_t^k$
    end for

Client executes: // run on each client k
    $\omega_t \leftarrow$ download from server
    $\omega_{t,sparse} \leftarrow topK(\omega_t)$ topK sparsification
    for local iteration $e = 1$ to $E$ do
      for each batch $b \in B$ do
          $g_t^k \leftarrow \eta \, \mathcal{L}(\mathbf{d}_b, \omega_{t,sparse})$ local training
      end for
    end for
    for each layer $g_{t,l}^k$ in $g_t^k$ do
      fitting distribution + kmeans quantization
      distribution parameter $\leftarrow fitDistribution(g_{t,l}^k)$
      centers,thresholds $\leftarrow k - means$(distribution parameter)
      $\text{comp}_R(g_{t,l}^k) \leftarrow quantization$(centers, thresholds)
    end for
    transmit $\text{comp}_R(g_t^k)$ to server
```

M22

Optimization of the hyperparameter M

Is this any good?

- ➔ It’s really hard to say, since we building a baseline is really hard
- ➔ We can show is that optimizing M bring benefits in terms of accuracy



Simulation setting

Settings
Cifar10 dataset
Heterogeneous local data
Number of local updates=1
Number of remote users=2
Stochastic Gradient Descent
Adaptive Sparsification

M22

Comparison with other schemes in the literature

Settings
Cifar10 dataset
Heterogeneous local data
Number of local updates=1
Number of remote users=2
Stochastic Gradient Descent
Adaptive Sparsification

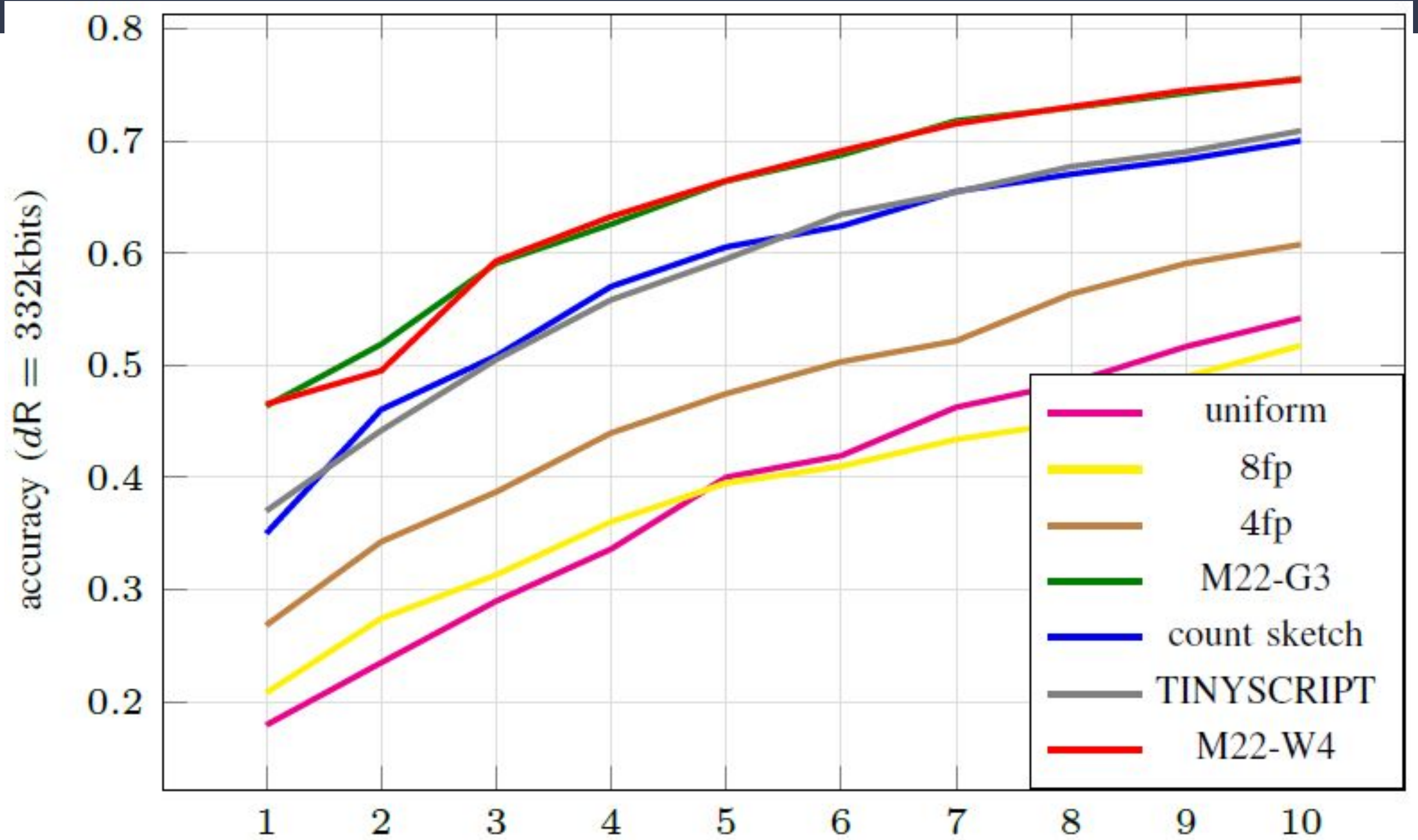
- ➔ Uniform quantization
  Quantize each gradient entry using a uniform quantizer

- ➔ 8fp/4fp
  Sparsification + floating point quantization
  Sparsification level is chosen to meet the rate constraints

- ➔ Countsketch
  Low dimensional compression using a +1/-1 projection matrix

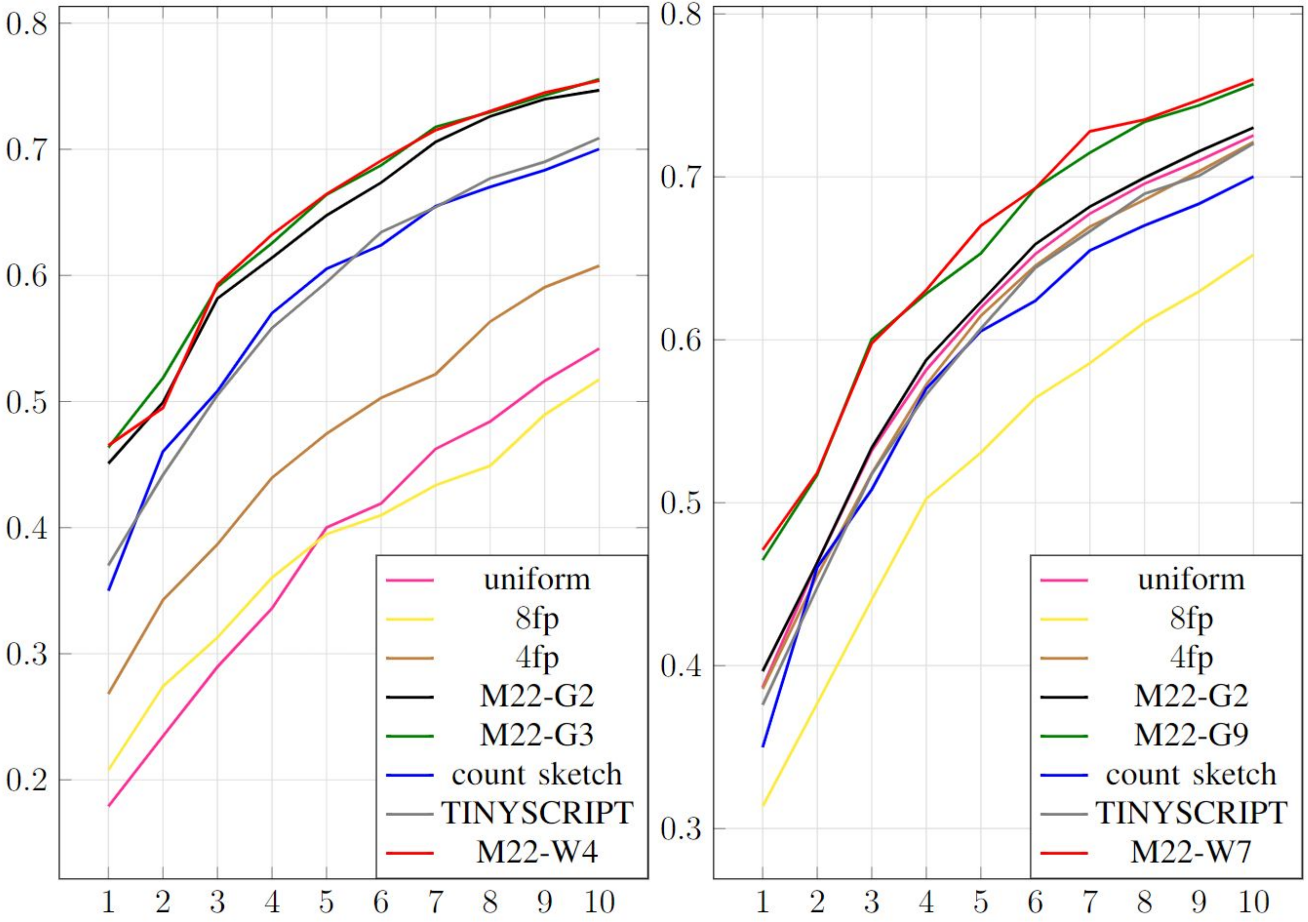
- ➔ TINYSCRIPT
  Quantize using double Weibull modelling + L2 distortions

M22


0.8
0.7
0.6
0.5
0.4
0.3
0.2
accuracy (dR = 332kbits)
1
2
3
4
5
6
7
8
9
10
uniform
8fp
4fp
M22-G3
count sketch
TINYSCRIPT
M22-W4

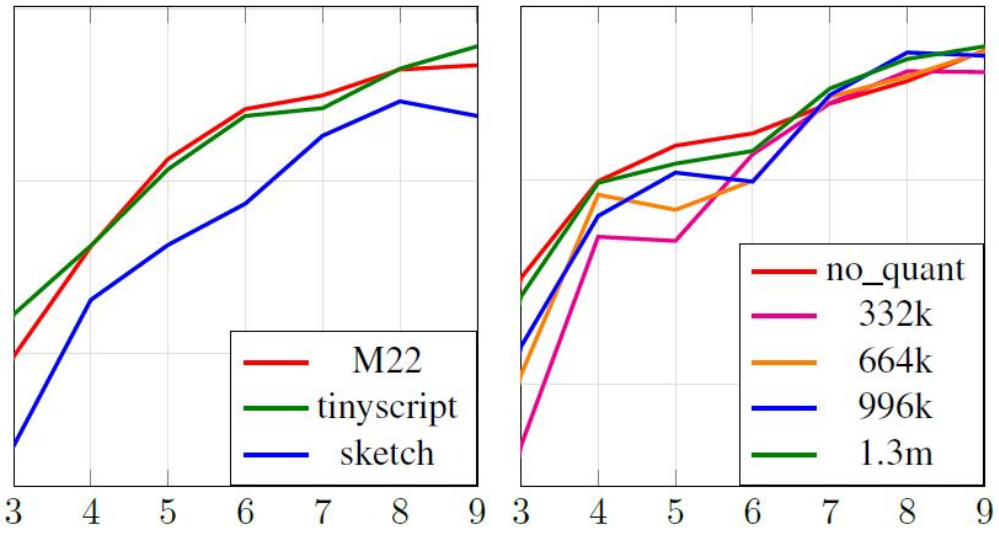
R = 332k (left)

R = 996k (right)



M22

Generalizing to other Models



ResNet18

VGG16

Beyond M22

- Many things we didn't account for
  - Lossless compression
  - Per-layer rate allocation
  - Optimization of the learning rate & quantization
  - Adaptive transmission rate
  - ….

- Are these variations interesting?
  - IMO only if you are willing to build a true application

Papers to reference

Liu, Yangyi, et al. "M22: A Communication-Efficient Algorithm for Federated Learning Inspired by Rate-Distortion." arXiv preprint arXiv:2301.09269 (2023).

Chen, Zhong-Jing, et al. "Convert, compress, correct: Three steps toward communication-efficient DNN training." *arXiv preprint arXiv:2203.09044* (2022).

Thank you for your attention!

Feel free to email me for questions!

rini.stefano@gmail.com