Enhancing Efficiency in Decentralized AI: A Dynamic Quantization Paradigm with Outlier Elimination

Juyoung Park, Hanwen Ju, Sushil Pokhrel

University of Waterloo, 2023 Fall STAT940 Project

Abstract

As the era of On-device AI unfolds, integrating AI models trained on edge devices with central server models presents significant challenges. This shift to decentralized AI leverages edge devices' data and computational resources to enhance privacy and reduce latency. To address network cost challenges in this distributed architecture, we have implemented dynamic quantization. This approach efficiently compresses model sizes, optimizing data transmission between edge devices and central servers and balancing performance with network efficiency. Our project demonstrates the integration of these AI realms, using dynamic quantization to enhance network communication, and evaluates the feasibility through dataset preprocessing and model optimization on each device.

Methodology

1. **Quantization mapping function** The quantization mapping function Q transforms a real number x into a quantized value q using scaling parameter s and zero point z:

$$f(x,s,z) = Clip\left(round\left(\frac{x}{s}\right)+z\right)$$

- s is the scaling parameter, used to adjust the range of the real values

$$s = \frac{R_{max} - R_{min}}{Q_{max} - Q_{min}}$$

- z is the zero point, representing the offset for the quantized values

$$z = Q_{max} - \frac{R_{max}}{s}$$

2. **Local Model Update (Client-Side)**

- Let $w_{t,i}$ represent the parameters of the model on the i^{th} client device at training round t.
- The local update on each client can be expressed as:

$$w_{t+1,i} = w_{t,i} - \eta \nabla L(w_{t,i}, D_i)$$

where η is the learning rate, $\nabla L(w_{t,i}, D_i)$ is the gradient of the loss function L with respect to the model parameters $w_{t,i}$ on the client's dataset D_i.
- Exclude model weights with high loss:

$$w_{t+1,i} = \begin{cases} w_{t+1,i} & \text{if } L(w_{t+1,i}, D_i) \le \text{threshold} \\ \text{null} & \text{otherwise} \end{cases}$$

where threshold is a predefined loss threshold.

3. **Aggregation of Model Parameters (Server-Side)**

- After training, each client sends its updated model parameters $w_{t+1,i}$ back to the server, excluding those with high loss.
- The server then averages these parameters to update the global model. The updated global model W_{t+1} can be computed as:

$$W_{t+1} = \frac{1}{N_{valid}} \sum_{i=1}^{N} w_{t+1,i}$$

where N_{valid} is the number of client devices with valid updates (i.e., updates not excluded due to high loss).

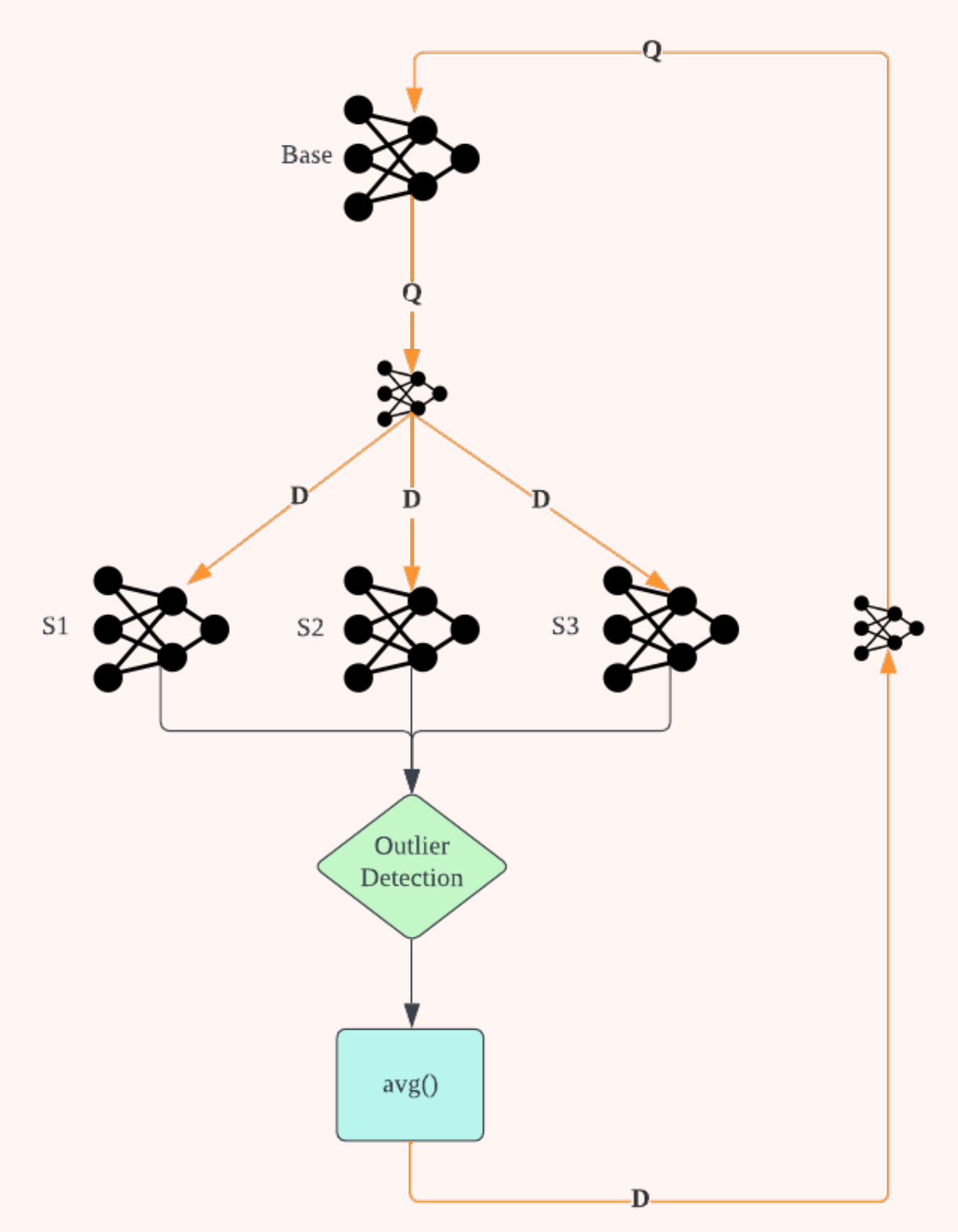

Figure 1. Flowchart of our proposed method, Q represents Quantization and D represents DeQuantization

Algorithm

Algorithm 1 The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
    Server Executes
1:  initialize $w_0$
2:  for each round $t = 1, 2, \ldots$ do
3:      $m \leftarrow \max(C \cdot K, 1)$
4:      $S_t \leftarrow$ (random set of $m$ clients)
5:      for each client $k \in S_t$ in parallel do
6:          $w^k_{t+1} \leftarrow$ ClientUpdate$(k, w_t)$
7:      end for
8:      $m_t \leftarrow \sum_{k \in S_t} n_k$
9:      $w_{t+1} \leftarrow \frac{1}{m_t} \sum_{k \in S_t} n_k w^k_{t+1}$
10: end for
    ClientUpdate$k, w$
11: $B \leftarrow$ (split $P_k$ into batches of size $B$)
12: for each local epoch $i$ from 1 to $E$ do
13:     for batch $b \in B$ do
14:         $w \leftarrow w - \eta \nabla \ell(w; b)$
15:     end for
16: end for
17: return $w$ to server =0
```

Experiment

To validate our proposed method, we designed 3 different experiments to simulate 3 different scenarios. We choose CIFAR-10 dataset which consists of 60000 32x32 colour images.

Base Model
conv1 - conv2 - conv3 - conv4 - pool - dropout - fc1 - fc2 - fc3

Experiment 1 This experiment simulates the biased local data scenario.

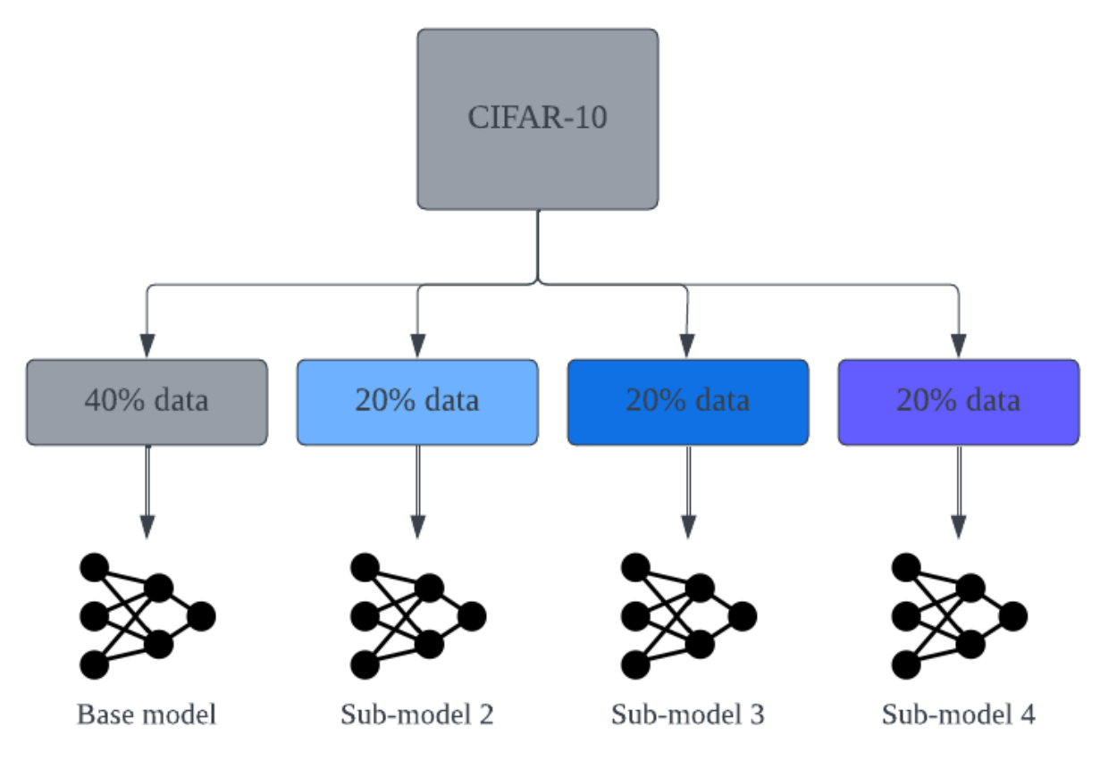

Figure 2. Experiment 1 split CIFAR-10 dataset into 4 partitions (40%, 20%, 20%, 20%), the colored partition represent represents noised data (e.g rotation, ColorJitter)

Experiment 2 This experiment simulates the Non-IID scenario.

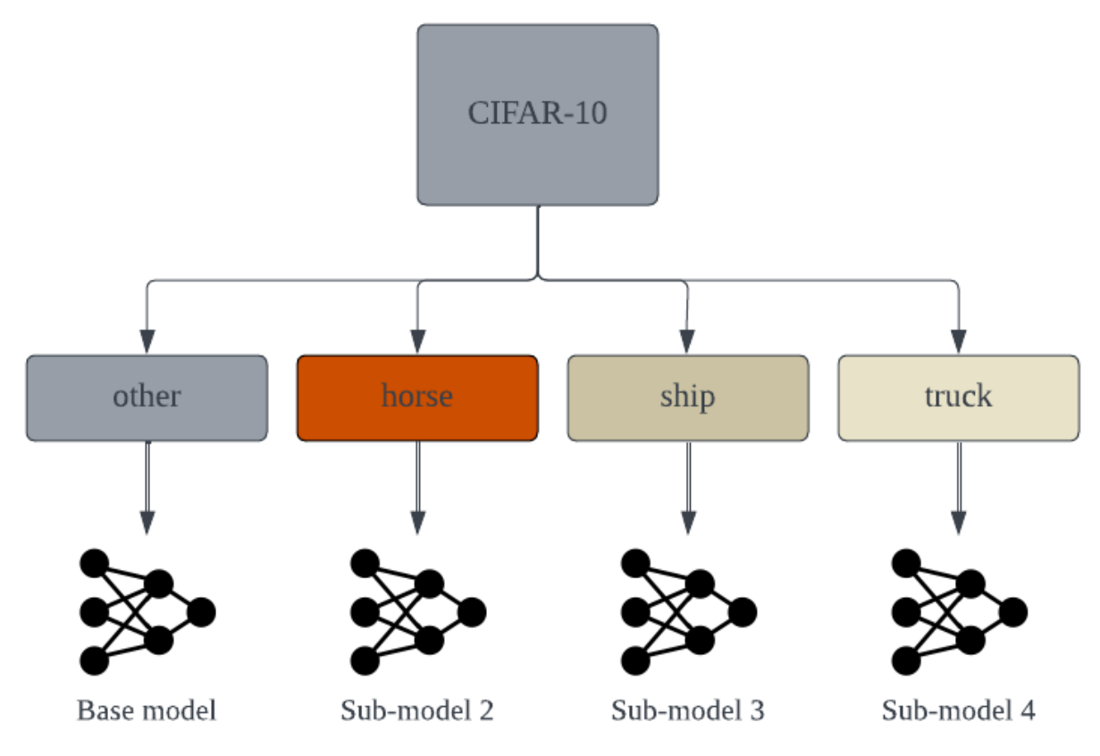

Figure 3. Experiment 2 contains 3 independent datasets for horse, ship, and truck respectively, the remaining data is used for training base model

Experiment 3 This experiment simulates an outlier case.

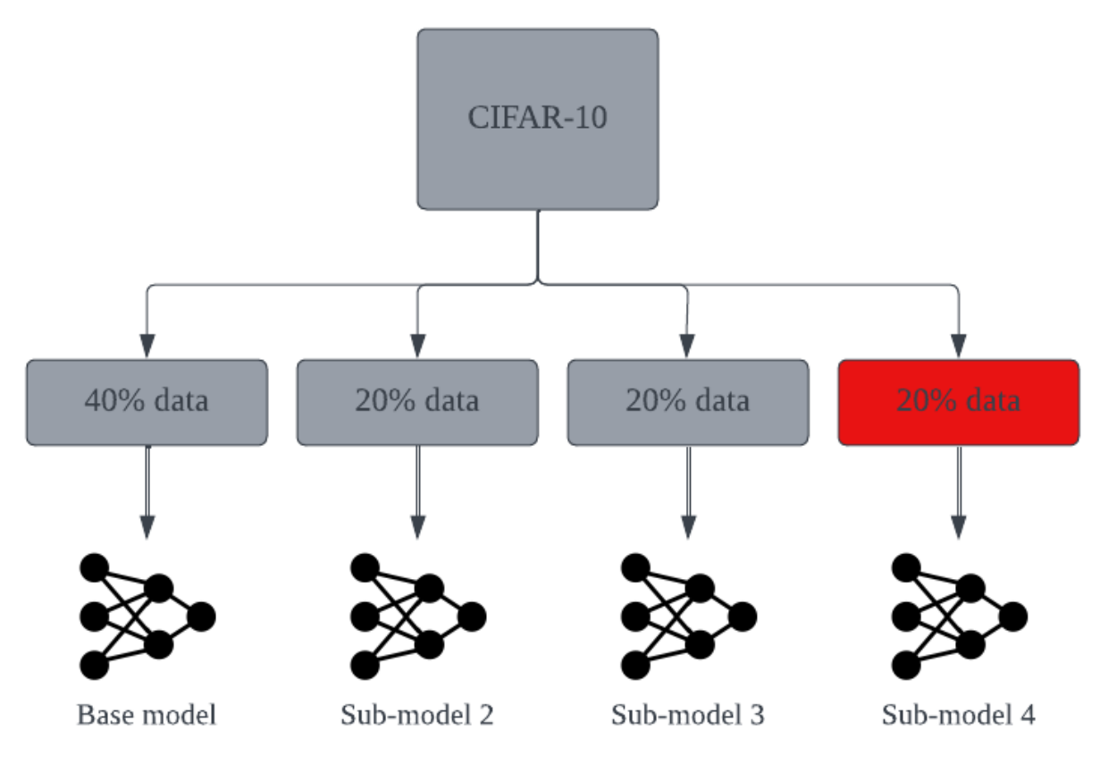

Figure 4. Keep the splitting setting in Experiment 1 (40%, 20%, 20%, 20%). And completely shuffle the labels randomly in one of the 20% partitions to represent the scenario of "bad data".

Numerical Result

One of our noteworthy accomplishments is the successful reduction of the model size by 54.83%.

We first test the accuracy of the model in 3 experiments.

Exp #	Base	Sub 1	Sub 2	Sub 3	Comb 1	Comb 2
Exp 1	0.74	0.80	0.73	0.75	0.80	-
Exp 2	0.30	0.43	0.38	0.41	0.40	-
Exp 3	0.76	0.78	0.76	0.10	0.75	0.81

Table 1. Test accuracy of across different models for in experiment. Base represents the base model, Sub 1-3 represent 3 sub-models, Comb 1 is the average of 3 sub-models, Comb 2 is also the average but ignored outlier sub-model

And the training accuracy trend for every epoch for 3 different experiments.

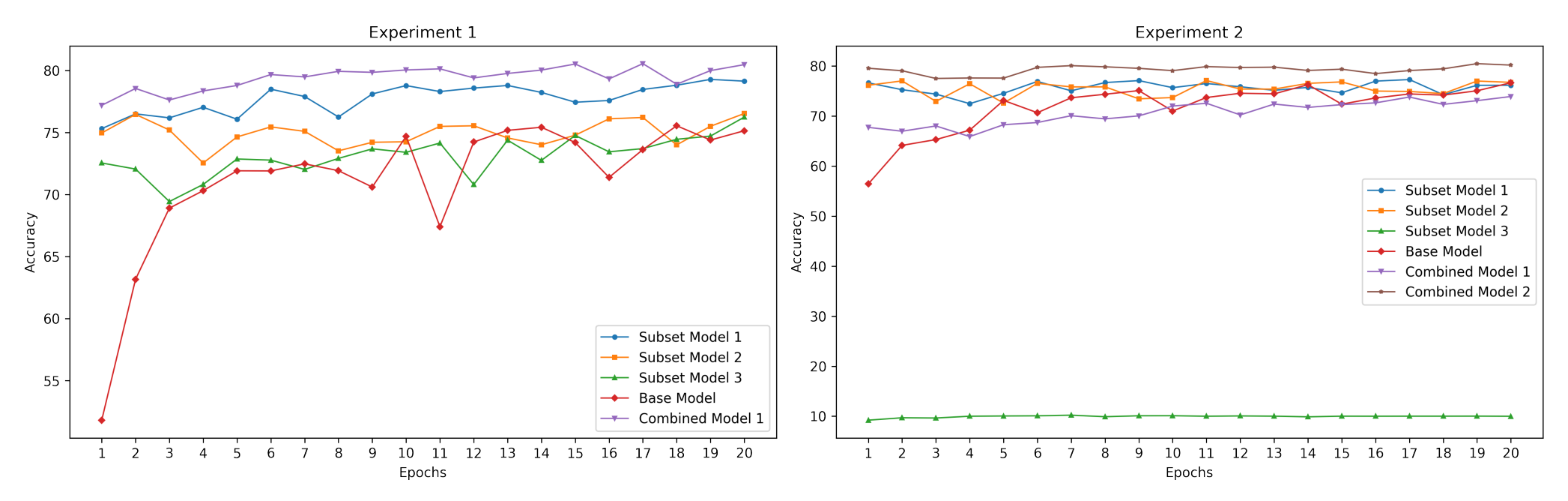

Figure 5. Validation accuracy trend for each epoch in 3 experiments

We compared the averaged weight difference for each layer of the model in experiment 3. We observe that the parameters of the sub-model which trained "bad data" vary from others.

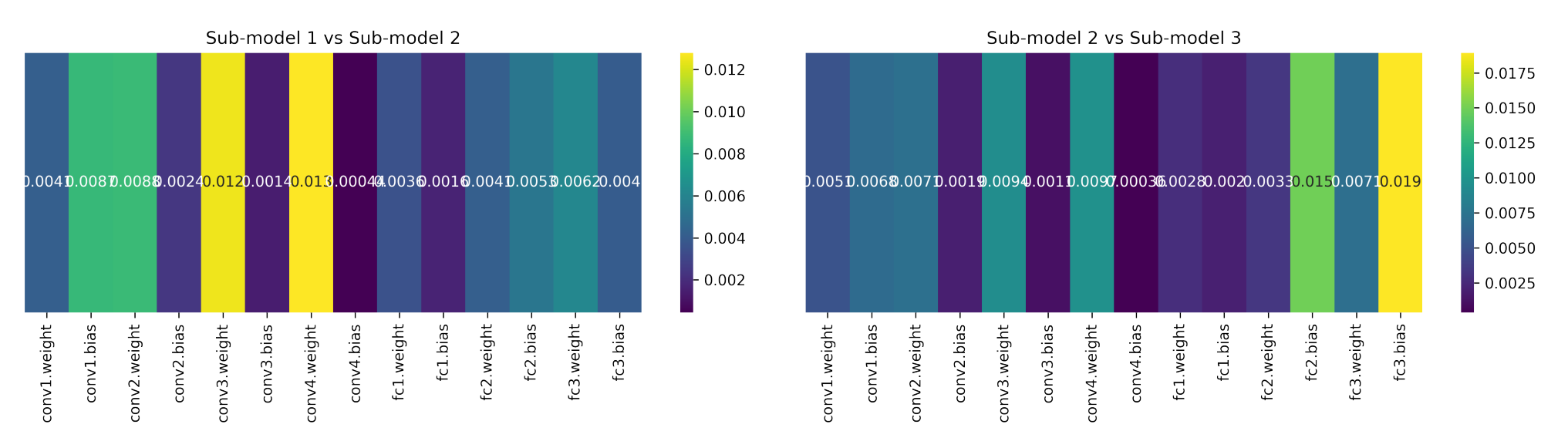

Figure 6. Averaged Weight difference between each sub-model

Based on this observation, we tried dimensional reduction techniques such as t-SNE and PCA, but neither performed well.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas.
Communication-efficient learning of deep networks from decentralized data.
In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Claude E. Shannon.
A mathematical theory of communication.
Bell System Technical Journal, 27(3):379–423, 1948.