**Federated Learning:** machine learning setting where the goal is to train a high-quality centralized model with training data distributed over a large number of clients (e.g. phones), each with unreliable and relatively slow network connections.

A prototypical round consists of:
1. Select some clients, each downloads current model
2. Each client updates the model, based on local data
3. The updates are uploaded back to server
4. Server aggregates the updates (e.g. by summing), and forms an improved global model

\[ W_{t+1} = W_t + \eta H_t, \quad H_t = \sum_i H_t^i \]

**Model state at time \( t \)**

<table>
<thead>
<tr>
<th>Update by client ( i ), uploaded to server</th>
<th>Improved global model</th>
</tr>
</thead>
<tbody>
<tr>
<td>(downloaded by clients)</td>
<td>Aggregated updates</td>
</tr>
</tbody>
</table>

Reasons why Step 3 can be a practical bottleneck:
- Asymmetric internet connections – slower upload
- Additional cryptographic protocols for privacy reasons – expansion in \# of bits communicated

**Goal:**
Reduce the size of updates \( H_t^i \) uploaded to server, in bits, without sacrificing (much) of the performance.

---

**Structured Updates:** Enforce client update \( H_t^i \) to be of pre-specified structure.

- **Low Rank:**
  \[ A_t^i B_t^i \]

\( A_t^i \) can be compressed as a random seed

- **Random Mask:**
  Enforce the update \( H_t^i \) to be a sparse matrix, with a pre-defined random sparsity pattern, communicating only its non-zero values.

**Sketched Updates:** Train update \( H_t^i \) without constraints, encode in a (lossy) compressed form and send to server, using one or more of the following tools combined.

- **Random Mask:** Randomly subsample and scale the update on a per-element basis.
- **Binary Quantization:** Consider \( h_{\text{max}}, h_{\text{min}} \) to be the largest and smallest elements of \( H_t^i \).
  We quantize every element \( h \) of \( H_t^i \) as follows
  \[ \tilde{h} = \begin{cases} h_{\text{max}}, & \text{with probability } \frac{h - h_{\text{min}}}{h_{\text{max}} - h_{\text{min}}} \\ h_{\text{min}}, & \text{with probability } \frac{h_{\text{max}} - h}{h_{\text{max}} - h_{\text{min}}} \end{cases} \]

Can be verified this yields an unbiased estimate.

- **Random Structured Rotation:** Generate \( \mathcal{O}(d) \) and apply \( \mathcal{O}(d \log d) \) randomized structured rotation as preprocessing, apply inverse rotation on server. Randomness compressed in random seed.

---

**Experiments:** [1]
- CIFAR data
- Partitioned (randomly) to 100 clients, each with 500 data points
- TF tutorial model
- Base optimizer [2]
- Model Averaging

Major improvement in terms of total MB uploaded vs convergence speed

Quantization with masking can train model while in total communicating much less than the size of original data.

---

**Structured Updates:**

- **Low Rank:**
  \[ A_t^i B_t^i \]

\( A_t^i \) can be compressed as a random seed

- **Random Mask:**
  Enforce the update \( H_t^i \) to be a sparse matrix, with a pre-defined random sparsity pattern, communicating only its non-zero values.

**Sketched Updates:** Train update \( H_t^i \) without constraints, encode in a (lossy) compressed form and send to server, using one or more of the following tools combined.

- **Random Mask:** Randomly subsample and scale the update on a per-element basis.
- **Binary Quantization:** Consider \( h_{\text{max}}, h_{\text{min}} \) to be the largest and smallest elements of \( H_t^i \).
  We quantize every element \( h \) of \( H_t^i \) as follows
  \[ \tilde{h} = \begin{cases} h_{\text{max}}, & \text{with probability } \frac{h - h_{\text{min}}}{h_{\text{max}} - h_{\text{min}}} \\ h_{\text{min}}, & \text{with probability } \frac{h_{\text{max}} - h}{h_{\text{max}} - h_{\text{min}}} \end{cases} \]

Can be verified this yields an unbiased estimate.

- **Random Structured Rotation:** Generate \( \mathcal{O}(d) \) and apply \( \mathcal{O}(d \log d) \) randomized structured rotation as preprocessing, apply inverse rotation on server. Randomness compressed in random seed.

---

**Experiments:** [1]
- CIFAR data
- Partitioned (randomly) to 100 clients, each with 500 data points
- TF tutorial model
- Base optimizer [2]
- Model Averaging

Major improvement in terms of total MB uploaded vs convergence speed

Quantization with masking can train model while in total communicating much less than the size of original data.