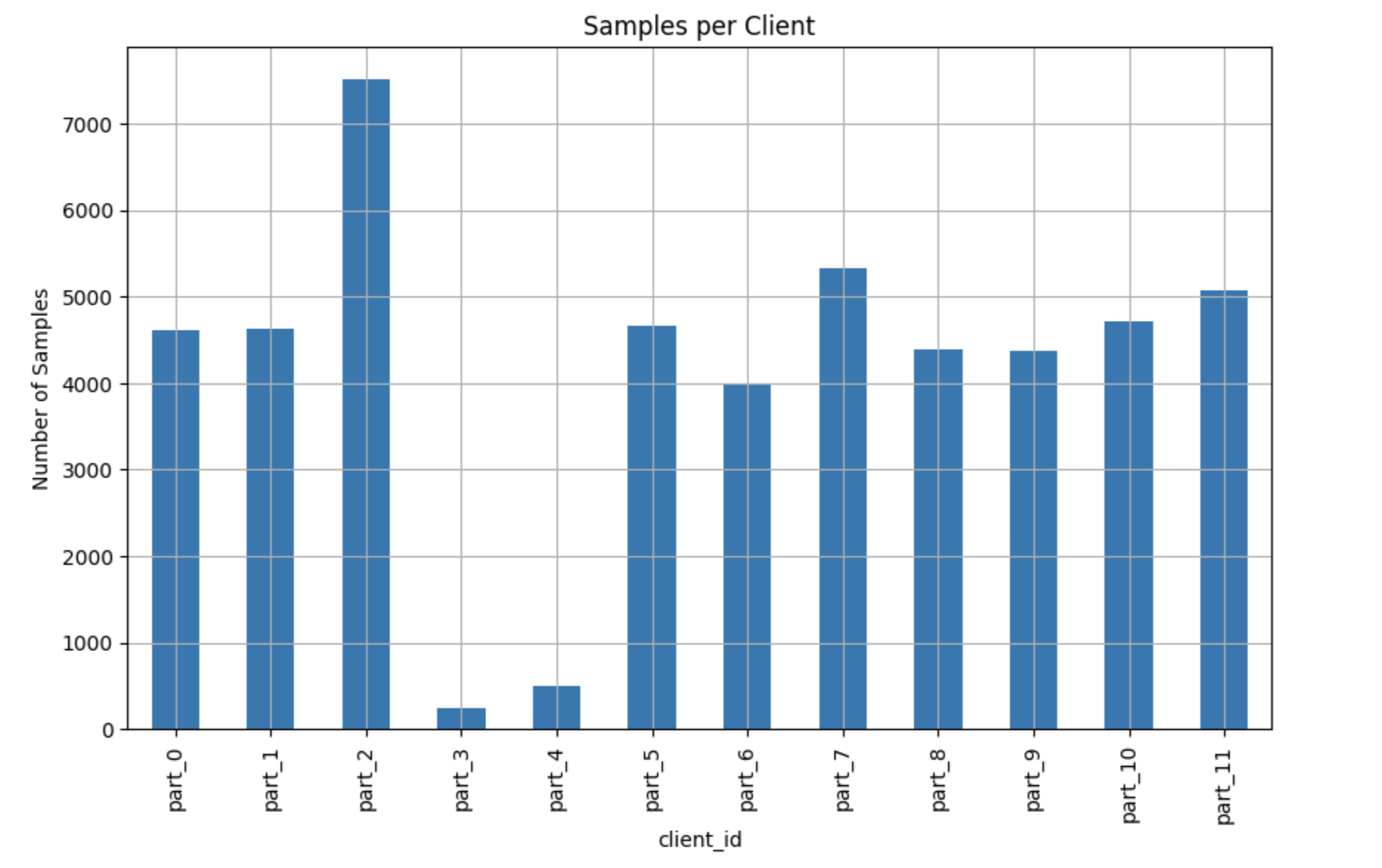
# Technical Report: Federated Learning Client Selection

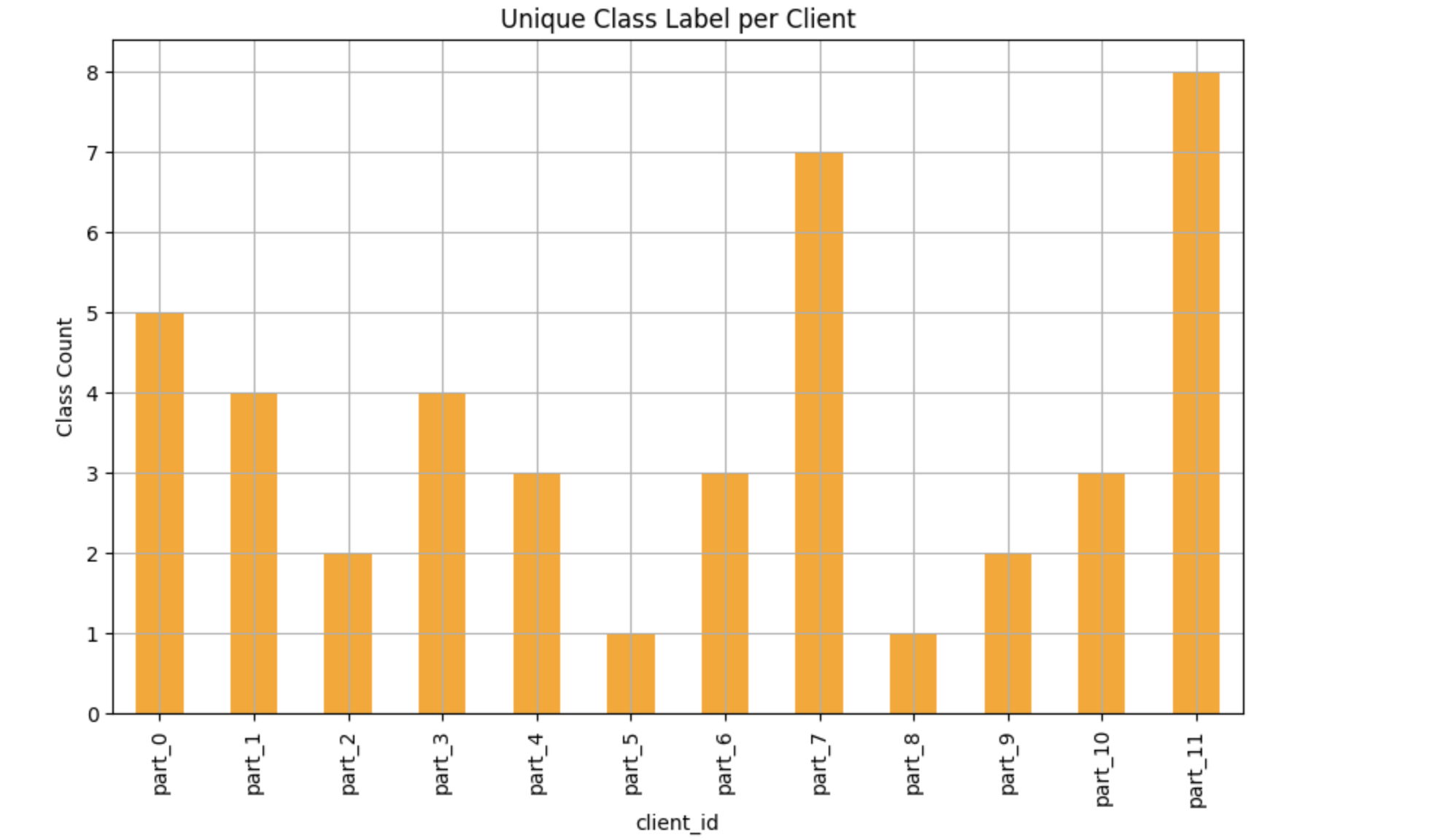
## Section 1: Per-Client Data Analysis

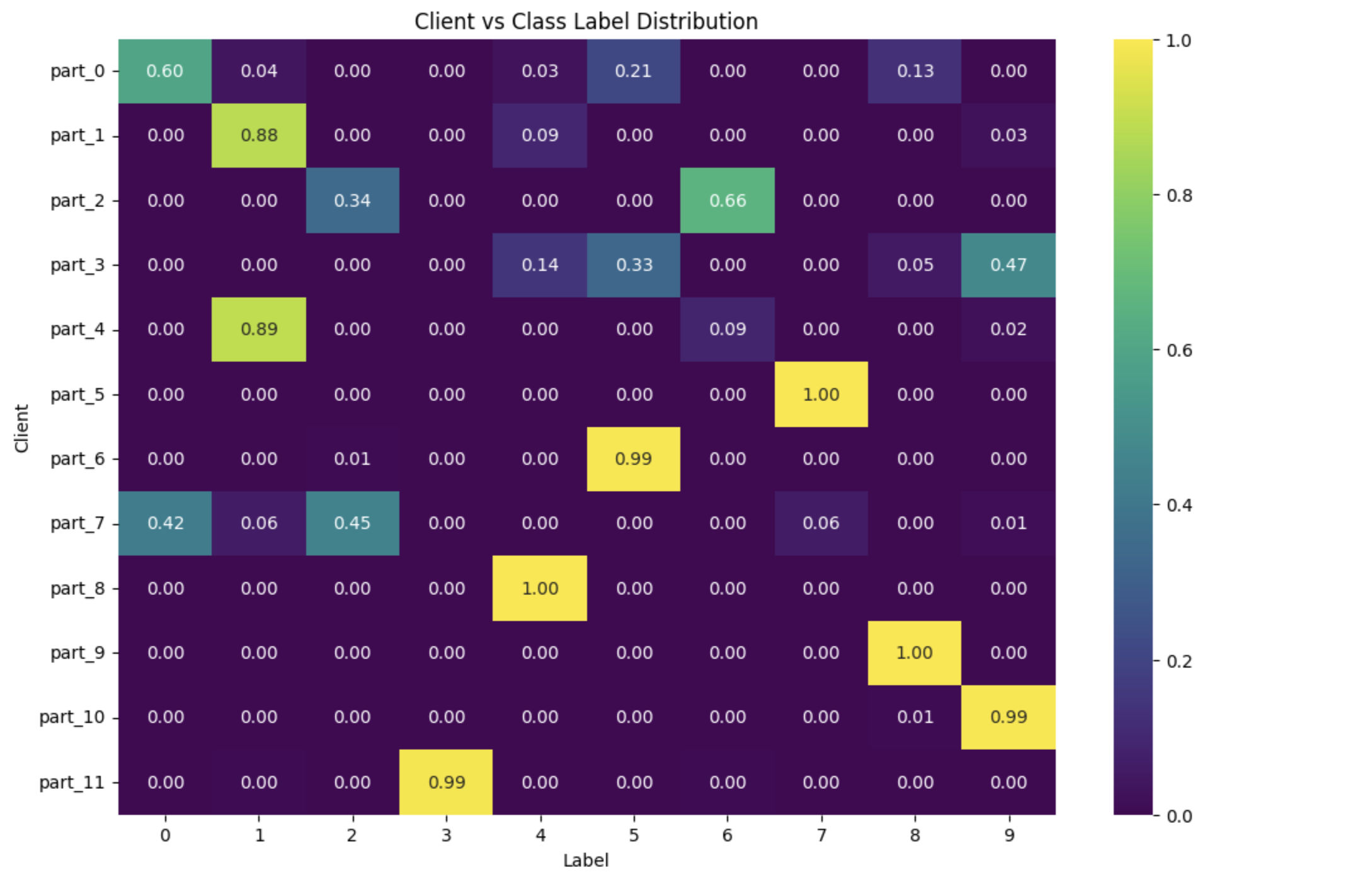
We analyzed a CIFAR-10 dataset partitioned across 12 clients using a Dirichlet distribution to simulate non-IID conditions. Our goals were to understand:

* **Label distribution (entropy):** We computed entropy for each client to measure class diversity.
* **Sample size:** Ranged significantly across clients (from hundreds to thousands).
* **Class imbalance:** Some clients contained data from only 1–3 labels, while others had near-uniform class distribution.

### Findings:

* High heterogeneity in label spread; entropy ranged from ~0.3 to ~2.3.
* Some clients were highly specialized (e.g., only label 5), while others were generalists.
* This variance affects model generalization in FL settings.
* 





## Section 2: Profiling Client Training Time

To evaluate training efficiency and resource impact across batch sizes, we profiled a client's training performance on a higher-capacity machine. Each batch size was run for 100 mini-batches, and the following metrics were collected:

* **Average time per batch**
* **Peak memory usage**
* **Throughput**: defined as batch\_size / avg\_time

### Results:

### 

### Conclusion:

Throughput increases consistently until it **peaks at batch size 1500**, with a metric of **261.27**.

Beyond this, gains taper off or drop due to overhead and memory limits.

**Memory usage exceeds 700 MB** beyond batch size 1500, potentially risky for constrained devices

## Section 4: Smarter Client Selection Strategy (1–2 pages)

### Baseline (Random Selection)

* Ran 20 rounds of FL using 3 randomly selected clients each round.
* Final accuracy: ~0.33
* Fairness (Jain’s Index): ~0.71
* Widely distributed client participation, but slow accuracy improvement.

### Smart Strategy: Entropy / Time

We introduced a new selection heuristic:

score = entropy / minibatch\_time

* **Entropy:** Rewards label diversity
* **Minibatch time:** Rewards faster clients

### Implementation:

* Added entropy calculation to each client
* Overrode client\_selection\_smart in the server
* Logged selections and accuracy over 20 rounds

### Results:

* Final accuracy: ~0.36
* Accuracy grew faster and more smoothly
* Fewer clients participated (only 3 selected repeatedly)
* Fairness dropped (Jain’s Index = 1.00)

### Plots Included:

### Trade-offs:

| Factor | Random | Smart |
| --- | --- | --- |
| Accuracy | Slower | Faster |
| Speed | Lower | Higher |
| Fairness | High | Low |

### Future Work:

* Add penalty for overused clients:

score = (entropy / time) \* (1 - usage\_ratio)

* Introduce a rotation mechanism to ensure wider client inclusion

## Analysis & Conclusion

This study confirms that leveraging client heterogeneity significantly boosts FL performance. By using a smart client selection strategy that prioritizes label diversity and training speed, we:

* Improved accuracy by ~3% over 20 rounds
* Reduced convergence time
* Traded off fairness for efficiency

A hybrid method with both fairness and performance in mind would offer the best real-world solution.

## Appendix

### Client Selection Algorithm

def client\_selection\_smart(self, clients, args: dict) -> list:  
 num\_clients = args["num\_clients\_per\_round"]  
 client\_scores = []  
 for client in clients:  
 e = getattr(client, "entropy", 1.0)  
 t = getattr(client, "minibatch\_time", 1.0)  
 score = e / t  
 client\_scores.append((client.cid, score))  
 top\_clients = sorted(client\_scores, key=lambda x: x[1], reverse=True)[:num\_clients]  
 return [cid for cid, \_ in top\_clients]