Efficient Federated Learning for Modern NLP

Dongqi Cai¹, Yaozong Wu¹, Shangguang Wang¹, Felix Xiaozhu Lin², Mengwei Xu¹

¹ Beiyou Shenzhen Institute
² University of Virginia
How to understand the meaning of a word?

Natural Language Processing (NLP)
How to understand the meaning of a word?
Natural Language Processing (NLP)

What sparks modern NLP?
Attention-based Transformer
How to understand the meaning of a word?  
Natural Language Processing (NLP)

What sparks modern NLP?  
Attention-based Transformer

How to preserve the privacy of training data?  
Federated Learning
Transformer Models

- BERT, DistilBERT, BART, GPT, etc

Pre-training

Public, centralized

Federated Fine-tuning

- Fine-tuning
  - Text Classification

- Fine-tuning
  - Seq. Tagging

- Fine-tuning
  - Text Generation

Private, geo-distributed

Mobile devices

Deployment

Cloud

Cloud & Clients

Clients
FedNLP: focus of this work

- Pre-training
- Fine-tuning
- Fine-tuning
- Fine-tuning
- Fine-tuning
- Fine-tuning
- Fine-tuning
- Deployment

- BERT, DistilBERT, BART, GPT, etc

- Text Classification
- Seq. Tagging
- Text Generation

- Public, centralized
- Private, geo-distributed

- Cloud
- Cloud&Clients
- Clients
Is FedNLP practical on today's mobile platforms?
Observation 1: Transformer-based NLP models are highly costly.

Observation 2: FedNLP task is extremely slow.

Observation 3: Network transmission dominates the training delay on high-end devices.

Observation 4: Existing techniques are inadequate for FedNLP.
• Tiny adapters (less than 1M for each) are inserted to pre-trained Transformers.
• Only adapters are updated during training, most of Transformer parameters are freezing.
Key Building Block: Pluggable Adapters

- Tiny adapters (less than 1M for each) are inserted to pre-trained Transformers.
- Only adapters are updated during training, most of Transformer parameters are freezing.

Table 1: Computation and communication cost of inserting adapters into each transformer block (width=32) and full model tuning. Batch size: 4. Device: Jetson TX2.
Key Building Block: Pluggable Adapters

- Tiny adapters (less than 1M for each) are inserted to pre-trained Transformers.
- Only adapters are updated during training, most of Transformer parameters are freezing.

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Training Time</th>
<th>Updated Paras.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>Full Fine-tuning</td>
<td>1.86 sec</td>
<td>110.01 x 10^6</td>
</tr>
<tr>
<td></td>
<td>Adapter</td>
<td>1.14 sec</td>
<td>0.61 x 10^6</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>Full Fine-tuning</td>
<td>0.91 sec</td>
<td>67 x 10^6</td>
</tr>
<tr>
<td></td>
<td>Adapter</td>
<td>0.56 sec</td>
<td>0.32 x 10^6</td>
</tr>
</tbody>
</table>

Table 1: Computation and communication cost of inserting adapters into each transformer block (width=32) and full model tuning. Batch size: 4. Device: Jetson TX2.
Challenge: Large Adapter Configuration Space

Different adapter configurations (depth, width) result in a variety of convergence delays, up to $4.7 \times$ gap.
Challenge: Large Adapter Configuration Space

Different adapter configurations (depth, width) result in a variety of convergence delays, up to $4.7 \times$ gap.
Challenge: Large Adapter Configuration Space

Different adapter configurations (depth, width) result in a variety of convergence delays, up to $4.7 \times$ gap.
Challenge: No Silver Bullet Configuration

• The optimal configuration can be switched across FL rounds.

Across different target accuracy and target FedNLP tasks, the optimal adapter configuration (depth, width) varies. Model: BERT; device: Jetson TX2.
Challenge: No Silver Bullet Configuration

- The optimal configuration can be switched across FL rounds.
- Configuration varies across many factors: targeted accuracy, targeted NLP tasks and client resources.

Across different target accuracy and target FedNLP tasks, the optimal adapter configuration (depth, width) varies. Model: BERT; device: Jetson TX2.

<table>
<thead>
<tr>
<th>Model</th>
<th>Datasets</th>
<th>Optimal adapter configuration (depth, width)</th>
<th>99%</th>
<th>95%</th>
<th>90%</th>
<th>80%</th>
<th>70%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>20news</td>
<td>(2,64)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>agnews</td>
<td>(3,16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>semeval</td>
<td>(10,8)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ontonotes</td>
<td>(12, 32)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The optimal adapter configuration (i.e., best time to-accuracy) for different target accuracy (ratio to the full convergence accuracy) and different datasets.
Design: Online Configurator

- **Progressive training**: curriculum upgrading adapter configuration.
Design: Online Configurator

- **Progressive training**: curriculum upgrading adapter configuration.

When and how to upgrade the configuration?
Design: Online Configurator

- **Progressive training**: curriculum upgrading adapter configuration.
- **Sideline trails**: identifying timing and direction to upgrade configuration.

When and how to upgrade the configuration?
Design: Online Configurator

- **Progressive training**: curriculum upgrading adapter configuration.
- **Sideline trails**: identifying timing and direction to upgrade configuration.

---

(a) Clients (w/ adapter)  
(b) Clients (w/o adapter)
Further optimization: Activation Cache

[w/o cache]

Input

Transformer

Transformer

Transformer

Transformer

Transformer

Transformer

Output

Layer freezing

Adapter tuning

Loss

Label
Further optimization: Activation Cache

**Diagram:**
- **Input:** Text
- **Transformers:** Different stages of the model
- **Intermediate activations:** Between transformers
- **Output:** Loss and label processing
- **Adapter tuning** and **Layer freezing** options

**Legend:**
- **w/o cache** (without cache usage)
Further optimization: Activation Cache

An unique opportunity: Most of the Transformer parameters are freezing.
Further optimization: Activation Cache

An unique opportunity: Most of the Transformer parameters are freezing.
Evaluation: Setup

- **Implementation**
  - FedNLP\[^1\]
  - AdapterHub\[^2\]

- **Setups**
  - 3 devices
  - 2 models (BERT & DistilBERT)
  - 4 datasets

- **Baselines**
  1. Vanilla Fine-Tuning (FT)
  2. FineTuning-Quantized (FTQ)
  3. LayerFreeze-Oracle (LF\textsubscript{oracle})
  4. LayerFreeze-Quantized-Oracle (LFQ\textsubscript{oracle})

<table>
<thead>
<tr>
<th>Device</th>
<th>Processor</th>
<th>Per-batch Latency (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jetson TX2 [1]</td>
<td>256-core NVIDIA Pascal™ GPU.</td>
<td>0.88</td>
</tr>
<tr>
<td>Jetson Nano [2]</td>
<td>128-core NVIDIA CUDA® GPU.</td>
<td>1.89</td>
</tr>
<tr>
<td>RPI 4B [3]</td>
<td>Broadcom BCM2711B0 quad-core A72 64-bit @ 1.5GHz CPU.</td>
<td>18.27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th># of Clients</th>
<th>Labels</th>
<th>Non-IID</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>20NEWS [44]</td>
<td>100</td>
<td>20</td>
<td>/</td>
<td>18.8k</td>
</tr>
<tr>
<td>TC</td>
<td>AGNEWS [92]</td>
<td>1,000</td>
<td>4</td>
<td>a=10</td>
<td>127.6k</td>
</tr>
<tr>
<td>TC</td>
<td>SEMEVAL [31]</td>
<td>100</td>
<td>19</td>
<td>a=100</td>
<td>10.7k</td>
</tr>
<tr>
<td>ST</td>
<td>ONTONOTES [60]</td>
<td>600</td>
<td>37</td>
<td>a=10</td>
<td>5.5k</td>
</tr>
</tbody>
</table>

Evaluation: End-to-end Performance

• Our system reduces model convergence delays significantly.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>20NEWS</th>
<th>AGNEWS</th>
<th>SEMEVAL</th>
<th>ONTONOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>99%</td>
<td>95%</td>
<td>90%</td>
<td>99%</td>
</tr>
<tr>
<td>Relative Accuracy</td>
<td>99%</td>
<td>95%</td>
<td>90%</td>
<td>99%</td>
</tr>
<tr>
<td>FT</td>
<td>44.0</td>
<td>23.4</td>
<td>13.1</td>
<td>124.3</td>
</tr>
<tr>
<td>FTQ</td>
<td>12.7</td>
<td>6.8</td>
<td>3.8</td>
<td>32.0</td>
</tr>
<tr>
<td>LF_oracle</td>
<td>18.5</td>
<td>8.1</td>
<td>4.3</td>
<td>74.0</td>
</tr>
<tr>
<td>LFQ_oracle</td>
<td>5.2</td>
<td>2.5</td>
<td>1.1</td>
<td>16.8</td>
</tr>
<tr>
<td>AdaFL</td>
<td>1.3</td>
<td>0.4</td>
<td>0.1</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Table 1: Elapsed training time taken to reach different relative target accuracy. NLP model: BERT-base. Unit: Hour.
Evaluation: System Scalability

- Our system outperforms baselines in network environments.
- It outperforms baselines on various client hardware.

![Graph](image)

Fig. 1: Model convergence delays under different network bandwidths. Training targets 99% relative target accuracy.

![Graph](image)

Fig. 2: Model convergence delays with a variety of client hardware. ‘Heterogenous’ means a mixture of heterogeneous hardware capacity.
Evaluation: Key design

- Our key designs contribute to the results significantly.

Fig. 1: Model convergence delays with and without our system’s key designs, showing their significance.
Evaluation: System Cost

Our system is resource-efficient.
- It saves up to $220.7 \times$ network traffic. (Fig. 1)
- It reduces up to $32.2 \times$ energy consumption. (Fig. 2)
- It nontrivially reduces the memory usage. (Fig. 3)

Fig. 1: Network traffic (downlink and uplink) of all 15 client devices.

Fig. 2: Per-client average energy consumption, normalized to that of ours.

(a) DistilBERT
(b) BERT

Fig. 3: Peak memory usage of a client device.
Conclusion

• Our system is a federated learning framework for fast NLP model fine-tuning.

• It uses adapter as the only trainable module in NLP model to reduce the training cost.

• To identify the optimal adapter configuration on the fly, it integrates a progressive training paradigm and trail-and-error profiling technique.

• It can reduce FedNLP’s model convergence delay to no more than several hours, which is up to $155\times$ faster compared to vanilla FedNLP and $48\times$ faster compared to strong baselines.
Conclusion

• Our system is a federated learning framework for fast NLP model fine-tuning.

• It uses adapter as the only trainable module in NLP model to reduce the training cost.

• To identify the optimal adapter configuration on the fly, it integrates a progressive training paradigm and trail-and-error profiling technique.

• It can reduce FedNLP’s model convergence delay to no more than several hours, which is up to $155\times$ faster compared to vanilla FedNLP and $48\times$ faster compared to strong baselines.

Thanks for listening!

<Efficient Federated Learning for Modern NLP>

Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, Mengwei Xu