Client-Customized Adaptation for Parameter-Efficient Federated Learning

Yeachan Kim[1]*, **Junho Kim**[1]*, **Wing-Lam Mok**[1], **Jun-Hyung Park**[2], **SangKeun Lee**[1,3]
[1]Department of Artificial Intelligence, Korea University, Seoul, South Korea
[2]BK21 FOUR R&E Center for Artificial Intelligence, Korea University, Seoul, South Korea
[3]Department of Computer Science and Engineering, Korea University, Seoul, South Korea
{yeachan,monocrat,wlmokac,irish07,yalphy}@korea.ac.kr

Abstract

Despite the versatility of pre-trained language models (PLMs) across domains, their large memory footprints pose significant challenges in federated learning (FL), where the training model has to be distributed between a server and clients. One potential solution to bypass such constraints might be the use of parameter-efficient fine-tuning (PEFT) in the context of FL. However, we have observed that typical PEFT tends to severely suffer from heterogeneity among clients in FL scenarios, resulting in unstable and slow convergence. In this paper, we propose **C**lient-**C**ustomized **A**daptation (C2A), a novel hypernetwork-based FL framework that generates client-specific adapters by conditioning the client information. With the effectiveness of the hypernetworks in generating customized weights through learning to adopt the different characteristics of inputs, C2A can maximize the utility of shared model parameters while minimizing the divergence caused by client heterogeneity. To verify the efficacy of C2A, we perform extensive evaluations on FL scenarios involving heterogeneity in label and language distributions. Comprehensive evaluation results clearly support the superiority of C2A in terms of both efficiency and effectiveness in FL scenarios[1].

1 Introduction

The advent of large-scale pre-trained language models (PLMs) for natural language processing (NLP) has led to exceptional performance across a broad spectrum of domains. However, the high memory requirements for PLMs impede their applicability to resource-constrained environments. These challenges are particularly evident in federated learning (FL), where model weights are transmitted between the server and clients to preserve data privacy (Konečný et al., 2016; McMahan et al., 2017). While recent FL studies have expanded the application of PLMs in various tasks, such as text classification (Zhu et al., 2020; Qin et al., 2021; Weller et al., 2022), language modeling (Chen et al., 2019), and question answering (Chen et al., 2021), communicating the training model among clients requires huge computational resources and bandwidth, presenting a significant challenge in terms of practicality.

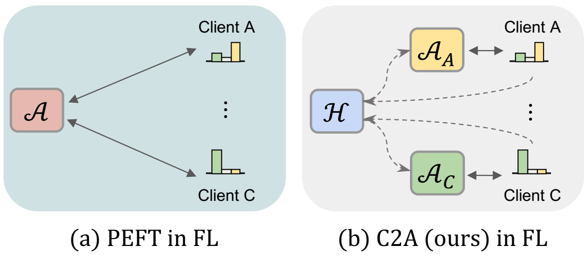


Figure 1: Conceptual illustration of the existing PEFT modules ($\mathcal{A}$) and the client-customized adaptation ($\mathcal{H}$). The proposed method learns to generate the client-customized PEFT modules rather than fitting a single global module to all clients.

Parameter-efficient fine-tuning (PEFT) approach is thereby a promising strategy for reducing communication costs in FL. Through tuning only a small fraction of parameters, such as adapter-based tuning (Houlsby et al., 2019; Hu et al., 2022; Mahabadi et al., 2021a), bias tuning (Zaken et al., 2022), and prompt-tuning (Lester et al., 2021), PEFT approaches significantly enhance the memory efficiency in centralized scenarios. However, the feasibility of PEFT in decentralized scenarios has not been well explored.

Hence, we investigate the applicability of typical PEFT approaches in FL scenarios. Specifically, we measure the performance and *client drifts* (Karimireddy et al., 2020; Li et al., 2021) of PEFT approaches in FL. Our discoveries are as follows: (1) typical PEFT approaches show large performance degradation in FL scenarios as the degree of non-

*These authors contributed equally to this work.
[1]Our code is available at https://github.com/yeachan-kr/c2a

IID increases; (2) these approaches usually suffer from large client drifts in non-IID scenarios, resulting in slow convergence and detrimental model performance. The above observations reveal that adopting PEFT in FL is not trivial, and posing the necessity to address large client drift.

To overcome the identified limitations, we propose a novel hypernetwork-based FL framework, **C**lient-**C**ustomized **A**daptation (C2A), that leverages the information of different data distributions on clients. Our key idea is to generate the adapter parameters tailored to each client via hypernetworks by taking the information of client data distribution, rather than naively fitting a single global adapter to all heterogeneous data distributions (Figure 1). By learning to adopt the different data distributions to generate adapters for each client, C2A enables robust training for various non-IID conditions while sharing knowledge among clients. Moreover, in order to manage the large number of parameters associated with hypernetworks, we introduce factorized hypernetworks, thereby significantly reducing the number of parameters without sacrificing the performance.

We carefully design the experimental setting to verify the efficacy of C2A on realistic FL scenarios, considering on both label and language heterogeneous. The experimental results show clearly that C2A can be robust to the heterogeneity of clients, thereby leading to the state-of-the-art results on diverse non-IID setups. In addition, our framework shows a significant enhancement in training efficiency across a range of downstream tasks. Finally, we demonstrate that our C2A successfully mitigates the large client drifts among local clients in non-IID scenarios. A summary of our main contributions is as follows:

- We investigate the effectiveness of PEFT among various FL scenarios. To the best of our knowledge, our work is one of the few researches for adapting PEFT in FL.
- We propose **C**lient-**C**ustomized **A**daptation (C2A), a novel hypernetwork-based framework that strengthens the robustness of adapter concerning FL heterogeneity.
- We demonstrate that C2A works quite well on various non-IID scenarios while preserving the benefits of efficiency in PEFT.

2 PEFT in FL Scenario

2.1 Background of FL

The goal of federated learning is to collaboratively train a single global model without sharing any private data between clients. To this end, FL proceeds through the communication of training models between clients and the server in a round-by-round manner. For each round, the server first distributes a single global model θ to a set of sampled clients, participating clients then perform local optimization on their own data. Upon the completion of the optimization, the server again aggregates all locally-trained models to update the global model. Formally, let the dataset of the i-th client be $\mathcal{D}_i$, the above process for updating the global model can be formulated as follows:

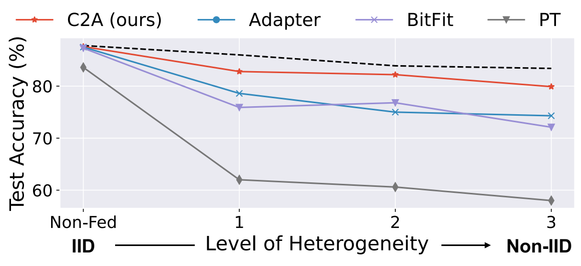
$$\widetilde{\theta} = \sum_{i=1}^{K} \alpha_i \cdot \mathcal{L}(\mathcal{D}_i; \theta), \qquad (1)$$

where $\mathcal{L}(\mathcal{D}_i; \theta)$ is the function that returns the trained model based on the given dataset and the initial model, K is the number of participating clients, and α_i is the contributing factor of the client i to build a global model, which is typically determined by the dataset size of each client, i.e., $\alpha_i = \frac{|\mathcal{D}_i|}{\sum_i |\mathcal{D}_i|}$. While there are various aggregation methods, we focus on FedAvg due to its wide applicability in the FL community (Karimireddy et al., 2020; Li et al., 2021; Luo et al., 2021).

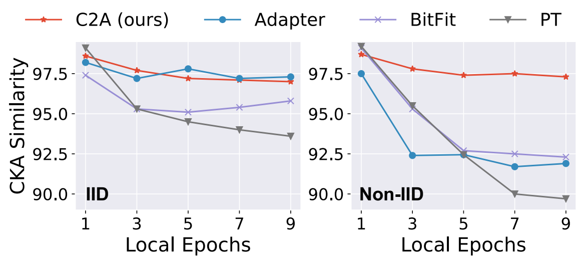
However, utilizing cumbersome PLMs for the communication process of FL poses two challenges. Firstly, the function $\mathcal{L}(\cdot)$ requires high computing resources due to the large number of trainable parameters associated with PLMs. Secondly, in the aggregation step (i.e., weighted summation), significant network bandwidth is required to transmit and receive the models. Therefore, it is crucial to find an optimal solution that can mitigate these constraints, providing a more efficient and less resource-intensive mechanism for FL with PLMs.

2.2 Impact of Heterogeneity on PEFT

To verify the applicability of PEFT in federated context, we conduct a preliminary investigation in which only small components (e.g., adapters, prompt embeddings, biases) are fine-tuned on local data and subsequently shared between clients. The experimental configuration comprises 100 clients engaged in the task of multilingual news classifica-



(a) Different level of heterogeneity



(b) Drift from the global model

Figure 2: Sensitivity analysis of PEFT methods in federated context in terms of data heterogeneity and divergence from the global model. PT indicates prompt-tuning (Lester et al., 2021), and the dotted line indicates full fine-tuning method.

tion[2] (Liang et al., 2020).

We first examine the robustness of PEFT on heterogeneous data distribution between clients, which is common in real-world scenarios. We report the test accuracy of the global model with respect to the increasing heterogeneity[3]. The overall results are depicted in Figure 2(a). In the non-federated scenario (i.e., IID), the existing PEFT methods manage to achieve strong performances comparable to that of the full fine-tuning. However, as the level of heterogeneity increases, the performances of the PEFT methods significantly lag behind that of the full fine-tuning. This verifies that PEFT methods exhibit greater susceptibility to heterogeneity than full fine-tuning.

To gain a deeper understanding of the susceptibility, we further analyze the local optimization of the PEFT methods. Specifically, we measure the CKA similarity (Kornblith et al., 2019) of the logits between the training model and the global model on the IID and non-IID setups. Figure 2(b) shows the results. Comparing between IID and non-IID setups, all PEFT methods noticeably deviate from the global model on non-IID. This indicates that the model gradually converges to the client optima while drifting apart from the global model's optima, which are believed to be more generalized (Li et al., 2021). This observation aligns with prior results (Luo et al., 2021), and we suspect that such deviation attributes the slow and unstable convergence.

[2]Further details for the experiment in Section 4.3.

[3]The increasing heterogeneity implies the corresponding degree of skewness deterioration towards certain classes

3 C2A: Client-Customized Adaptation

In this section, we elaborate on the proposed framework in detail. The core strategy is to generate customized PEFT modules tailored to each client to mitigate the negative impact of heterogeneity among clients. To achieve this, we first derive latent vectors to represent the data distribution of each client (Section 3.2). The resulting embeddings are then conditioned on the hypernetworks so as to generate parameters of the PEFT modules tailored to each client (Section 3.3). Regarding on the large number of parameters induced from hypernetworks, we effectively factorize the weights of the hypernetworks (Section 3.4).

3.1 Adapter Architecture

We start with defining the structure of the PEFT modules to be generated. While lots of different modules have been proposed, we focus on Adapter (Houlsby et al., 2019), given its versatility across domains, such as vision-and-image (Sung et al., 2022) and audio (Hou et al., 2021), as well as its demonstrated efficacy in performing given tasks. The adapter consists of down- and up-projection functions that are interleaved between self-attention layers and feed-forward layers within every block of the PLMs. The adapting process can be formulated as:

$$\mathcal{A}^l(x) = \mathbf{U}^l \text{GeLU}(\mathbf{D}^l x) + x \tag{2}$$

where $\mathbf{D}^l \in \mathbb{R}^{r \times d}$ and $\mathbf{U}^l \in \mathbb{R}^{d \times r}$ are the weights for the down- and up-projection in the l-th layer of PLMs, respectively, d is the hidden dimension of PLMs, and r is the bottleneck dimension.

3.2 Construction of Client Embeddings

To represent the characteristics of the clients, we consider two different types of information: 1) label embeddings and 2) context embeddings.

Label Embeddings The label embedding plays a role in conveying the explicit information of class distribution on each client. Since mini-batches are generally sampled by uniform distribution, the label distributions on mini-batches can sufficiently

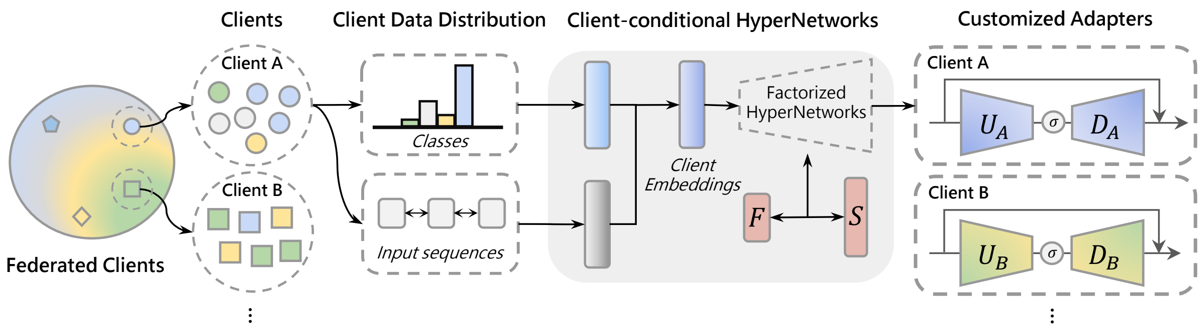


Figure 3: Overview of the proposed framework, denoted as **C**lient-**C**ustomized **A**daptation (C2A). To perform customized adaptation, C2A takes into account the client information as a form of label and context. Based on the client embeddings, the factorized hypernetworks generate adapters that are specialized for each client.

represent the data distributions of clients. Thus we construct label embeddings from the label distributions of the mini-batches. Let the mini-batches of the client i be $\mathcal{B} \subset \mathcal{D}_i$, the label embeddings can be derived as follows:

$$L(\mathcal{B}) = \mathbf{W}_L \text{avg}([y_1; ...; y_{|B|}]) + b_L, \qquad (3)$$

where y_i is a one-hot label vector for the instance x_i, [;] denotes the concatenating function, avg$(\cdot)$ denotes the average pooling within mini-batches, $\mathbf{W}_L \in \mathbb{R}^{C\times t}$ and $b_L \in \mathbb{R}^t$ are the linear transformation weights and biases for the number of classes C and t is the dimensionality of input embeddings. It is important to note that, since the labels for test data are not accessible, we opt for a uniform distribution for the inference phase to generate adapters that are not biased toward dominant classes.

Context Embeddings Considering the contextual information in data can also provide an enhanced understanding of each client by taking a more comprehensive viewpoint (e.g., languages, text styles). Specifically, the contextual information is extracted from every layer, so as to generate layer-specialized adapters. Inspired by the sentence embeddings (Li et al., 2020), context embeddings are extracted by averaging word vectors over the lengths with ℓ_2 normalization. Let the resulting vectors of the sample x_j from the l-th layer of PLMs be $f^l(x_j)$, the context embeddings of the l-th layer are derived as follows:

$$F^l(\mathcal{B}) = \mathbf{W}_F \text{max}([f^l(x_1); ...; f^l(x_{|\mathcal{B}|})]) + b_F, \qquad (4)$$

where max$(\cdot)$ denotes the max-pooling across the batch, and $\mathbf{W}_F \in \mathbb{R}^{d\times t}$ and $b_F \in \mathbb{R}^t$ are the linear transformation weights and biases, respectively.

Client Embeddings The comprehensive client embeddings $\mathcal{I}^l_\mathcal{B}$ are constructed by summing up two types of embeddings. Additionally, we add layer-index embeddings into the client embeddings of each layer, further encouraging the generator to encode more diverse layer-wise information (Van Aken et al., 2019; de Vries et al., 2020).

3.3 Client-conditional HyperNetworks

Based on the client embeddings, we tailor adapters to each heterogeneous client. Drawing inspiration from the concept of hypernetworks (Ha et al., 2017) that generates parameters based on given input embeddings, we introduce the *client*-conditional hypernetworks, which generate adapter parameters by taking the client embeddings $\mathcal{I}^l_\mathcal{B}$ as inputs. Formally, the parameters of the adapters (i.e., U^l, D^l) are generated by following the function of hypernetworks:

$$(\mathbf{U}^l_\mathcal{B}, \mathbf{D}^l_\mathcal{B}) := h(\mathcal{I}_\mathcal{B}) = (\mathbf{W}_U, \mathbf{W}_D)\mathcal{I}^l_\mathcal{B}, \qquad (5)$$

where $\mathcal{I}$ is the input embeddings with dimensionality t, $W^l_D \in \mathbb{R}^{(r\times d)\times t}$, $W^l_U \in \mathbb{R}^{(d\times r)\times t}$ are the weights for the hypernetworks. Note that the hypernetworks are shared between different layers with the layer-specific information that are encoded to the input embeddings.

3.4 Factorization of HyperNetworks

While customized adapters can be generated from the aforementioned hypernetworks, hypernetworks typically comprise a relatively large number of parameters. We thus factorize the proposed hypernetworks into two smaller weights. Moreover, the resultant matrices from the factorized components are ℓ_2 normalized, such that the generated parameters are not biased towards any of the local majority classes in the client's data distribution (Zhong et al., 2021). Formally, the up-projection weights in Eq. (5) are reconstructed by two factorized components as follows:

$$\mathbf{U}_B^l = \mathbf{W}_U \mathcal{I}_B = \sigma(\mathbf{F}_U \mathbf{S}_U)\mathcal{I}_B \tag{6}$$

where $\mathbf{F}_U \in \mathbb{R}^{d\times s}$ and $\mathbf{S}_U \in \mathbb{R}^{s\times(r\times t)}$ indicate the factorized components from W_U with latent factor s, $\sigma(\cdot)$ denotes the Frobenius normalization.

For factorization, the latent factor s plays a crucial role in determining the complexity and expressivity of the resulting adapters. To allow for a larger dimensionality of latent factors, the two projection weights are tied similarly as if the tied auto-encoder (Alain and Bengio, 2014), i.e., $D_B^l = {U_B^l}^\top$. This strategy enables to halve the memory requirements without compromising the task accuracy.

3.5 Aggregation Phase for C2A

Upon the completion of the training phase on each client data, the respective trained models are transmitted back to the centralized server to update the global model (Eq. (1)). Considering that the training models for C2A are hypernetworks, each client sends the parameters associated with the hypernetworks and the layer-index embeddings to the server in order to update the global hypernetworks.

4 Evaluation

In this section, we evaluate the efficacy of our C2A on two realistic FL scenarios: 1) heterogeneity in label distributions, and 2) heterogeneity in both label and language distributions.

4.1 Datasets

To simulate the two challenging scenarios, we mainly consider two text classification datasets, 20Newsgroup (Lang, 1995) and XGLUE-NC (Liang et al., 2020), which have recently served as benchmarks for evaluating FL for NLP (Lin et al., 2022; Weller et al., 2022).

20Newsgroup The dataset comprises 18,000 news posts that pertain to 20 distinct topics. Given its larger categorical space (i.e., 20 labels) than the typical sentiment analysis datasets, it is favored to the verification for the important factor of the label distribution heterogeneity scenarios.

XGLUE-NC The dataset includes 10,000 posts written in multiple languages that pertain to 10 news categories. This diversity in languages adds an extra layer of complexity to the FL. The dataset comprises five languages: English, Spanish, French, German, and Russian. Furthermore, due to the varying categorical distribution between languages (e.g., the English dataset is skewed towards `Sports`, while the French dataset is skewed toward `News`), the distribution shifts among clients are naturally introduced to the dataset.

4.2 Non-IID Client Partitioning

Building upon the two datasets, we adopt two non-IID partitioning strategies to inject heterogeneity into the label and language distributions.

Label Distribution. Following the benchmark setup (Lin et al., 2022), we apply Dirichlet distribution Dir(β) to the datasets in reorganizing the data into the non-IID label distribution circumstance. The value β controls the degree of non-IID, the smaller the β, the more likely the clients in holding examples from only one class. Thus, we eventually construct a FL dataset respecting the label heterogeneity scenarios.

Language Distribution. Following the language setup in (Weller et al., 2022), we randomly divide clients into five distinct groups, with each group being exclusively dedicated to a specific language. Subsequently, we split the dataset of each language in the same manner with the strategy of non-IID label distribution, which is more challenging and not even being explored in previous works.

4.3 Federated Learning Setup

Baselines and Implementations Following the previous work (Lin et al., 2022), we use the uncased version of DistilBERT [4] (Sanh et al., 2019) with 66M parameters. We compare C2A with six strong baselines, which include Adapter (Houlsby et al., 2019), LoRA (Hu et al., 2022), Compacter (Mahabadi et al., 2021a), Prompt-tuning (Lester et al., 2021), BitFit (Zaken et al., 2022), and AdaMix (Yaqing Wang and Gao, 2022), to encompass a broad range of PEFT methods. These modules are optimized by AdamW (Loshchilov and Hutter, 2019) with the searched learning rate ranging from {2e-4, 3e-4, 4e-4, 5e-4}.

Local Optimization and Aggregation We assign 100 clients for each dataset and randomly selected 25% of the clients to join the local optimization in each round. During the local optimization,

[4]In multi-lingual FL scenarios, we adopt the multi-lingual version of DistilBERT with 134M parameters

Table 1: Evaluation results of test accuracy (%) on 20Newsgroup. The best and second best results are highlighted in **boldface** and <u>underlined</u>, respectively.

Methods	Params (%)	Non-Fed	Federated scenario		
			$\beta = 5.0$	$\beta = 1.0$	$\beta = 0.1$
Full Fine-tuning	100%	85.8	77.6	77.2	66.8
Adapter (Houlsby et al., 2019)	0.455%	84.0	69.1	65.5	56.1
LoRA (Hu et al., 2022)	0.111%	<u>84.3</u>	<u>69.5</u>	<u>67.7</u>	<u>56.6</u>
Compacter (Karimi Mahabadi et al., 2021)	0.043%	83.2	65.9	62.8	50.1
Prompt-tuning (Lester et al., 2021)	0.024%	74.2	51.6	46.4	28.2
BitFit (Zaken et al., 2022)	0.078%	82.8	67.1	66.5	55.1
AdaMix (Yaqing Wang and Gao, 2022)	0.559%	**84.7**	68.7	65.3	54.5
C2A (ours.)	0.097%	83.9	**71.6**	**70.4**	**61.0**

Table 2: Evaluation results of test accuracy (%) on XGLUE-NC. The best and second best results are highlighted in **boldface** and <u>underlined</u>, respectively.

Methods	Params (%)	Non-Fed	Federated scenario		
			$\beta = 5.0$	$\beta = 2.0$	$\beta = 0.5$
Full Fine-tuning	100%	87.6	84.5	83.7	80.7
Adapter (Houlsby et al., 2019)	0.225%	87.5	78.6	75.0	74.3
LoRA (Hu et al., 2022)	0.055%	**87.8**	<u>80.4</u>	78.4	74.6
Compacter (Karimi Mahabadi et al., 2021)	0.021%	87.3	75.9	73.4	71.0
Prompt-tuning (Li and Liang, 2021)	0.017%	85.6	61.2	60.6	58.0
BitFit (Zaken et al., 2022)	0.038%	87.3	78.4	76.8	72.1
AdaMix (Yaqing Wang and Gao, 2022)	0.277%	<u>87.6</u>	79.6	<u>79.1</u>	<u>76.6</u>
C2A (ours.)	0.049%	87.4	**82.8**	**82.2**	**80.2**

we use a batch size of 16 and 64 for 20Newsgroup and XGLUE-NC, respectively. Each client performs a single local epoch, and the server aggregates the locally-trained model based on FedAvg (McMahan et al., 2017).

4.4 Main Results

To thoroughly evaluate each baseline on various FL setups, we start from a non-federated setup and progressively increase the level of heterogeneity by manipulating β. The results are shown in Table 1 (20Newsgroup) and Table 2 (XGLUE-NC).

The proposed method, C2A, achieves the state-of-the-art performance for almost all setups. Specifically, despite that AdaMix uses multiple adapters for ensemble, our model improves the respective performance by 3% on both datasets. It is also noteworthy that while most PEFT approaches manage to achieve fair performance in non-FL scenarios, their performances significantly decrease as the degree of heterogeneity increases. In contrast, our C2A shows only marginal performance degradation even for high degree non-IID settings. Moreover, in the multilingual setting, C2A achieves a comparable performance to full fine-tuning. These results indicate that C2A is more resilient to heterogeneity in decentralized scenarios.

5 Further Analysis on C2A

In order to gain a deeper understanding of the benefits of C2A, we perform a series of analytical experiments utilizing XGLUE-NC with a value of $\beta = 0.5$, which represents the most challenging setup within our experimentation.

5.1 Ablation Studies

We conduct ablation studies to explore the contributions brought by each component of C2A. Specifically, we focus on the effect of client embeddings, which are composed of label embedding (LE), context embedding (CE), and factorization. Detailed

Table 3: Ablation studies for C2A. LE and CE represent the label and the context embedding, respectively.

Methods	Params (%)	Accuracy (%)
C2A(ours)	0.049%	**80.2**
Client embedding		
w/o LE	0.049%	78.4
w/o CE	0.049%	78.0
w/o LE,CE	0.049%	77.3
Factorization		
w/o Factorization	0.106%	79.8
w/o Normalization	0.049%	78.8

results are presented in Table 3.

Client Embedding. We observe that omitting either of the embeddings does hurt the model performance. Notably, comparing "w/o LE" to "w/o CE", ablating context embedding leads to more significant performance degradation. We suspect this is because that context embedding can provide more discriminating information of each client through implicit representations, such as language types, and text styles. Moreover, removing all the embeddings shows the worst performance, which demonstrates that our C2A with the client embeddings can generate more suitable adapters for each client.

Factorization. To examine the impact of factorization, we first compare it with the C2A results neglecting factorization. Despite using only half the parameters, our model achieves comparable performance as the model without factorization. In addition, we observe that omitting normalization significantly hurts performance. The results demonstrate that our normalization alleviates the performance drop by factorization.

5.2 Local Epochs vs. Communication Rounds

One of the crucial aspects in FL is communication efficiency. A simple way to achieve such efficiency is to reduce communication rounds while increasing local epochs. However, the increased local updates can result in greater susceptibility to client drifts (Li et al., 2021). Thus we examine the trade-off between local epochs and communication rounds, as shown in Figure 4. We compare C2A with three baselines under the same number of model updates (local epochs × communication rounds). We observe that increasing the local epochs leads to worse performance due to the detrimental effect of client drift. Nevertheless, C2A clearly outperforms the other baselines in all settings. This further verifies the potency of C2A in mitigating the negative effects of the drift caused by excessive local updates, and shows that C2A can be efficiently trained with only a few rounds of communication.

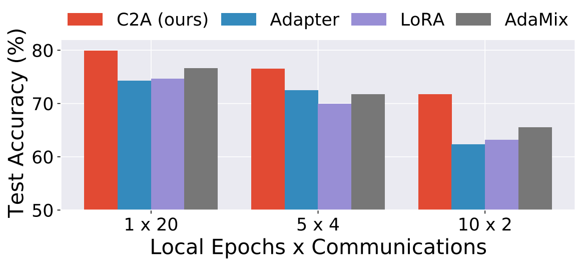


Figure 4: Evaluation results of the test accuracy with different numbers of local epochs.

5.3 Communication Cost for Target Accuracy

In FL scenarios, the communication between clients typically continues until the model attains a target accuracy or the allocated budgets are exhausted. As such, attaining the target accuracy with minimal communication rounds is crucial for reducing the total costs in practical FL. To analyze the baselines through the lens of such communication efficiency, we compare the number of required communications to reach the targeted performance for each baseline. The results are shown in Table 4. Our proposed C2A consistently performs the best over the baselines on all target accuracy. Specifically, C2A reaches the targeted performance approximately two times faster than the vanilla adapter. These results show that C2A engages fewer communication costs with less requirement on the parameters and communication rounds.

5.4 Scalability of C2A

We evaluate whether C2A can be scaled to larger PLMs. To this end, we adopt all PEFT baselines to XLM-RoBERTa with 278M parameters. The results are summarized in Table 5. We observe that our C2A still outperforms the baselines by a large margin. Specifically, our C2A achieves 3.1 points improvement compared with the adapter model. These results indicate that our approach can be well generalized to larger models.

Table 4: The number of communication rounds required to achieve the desired performance. The relative speedup of each baseline is also compared to the vanilla Adapter (Houlsby et al., 2019).

Methods	Communication Rounds	SpeedUp
	Target accuracy = 70%	
Adapter	13	×1.00
LoRA	18	×0.72
Compacter	19	×0.68
Prompt-tuning	46	×0.28
BitFit	18	×0.72
AdaMix	12	×1.10
C2A (ours.)	**7**	×1.86
	Target accuracy = 80%	
Adapter	33	×1.00
LoRA	44	×0.75
Compacter	71	×0.46
Prompt-tuning	100↑	×0.33↓
BitFit	55	×0.60
AdaMix	50	×0.66
C2A (ours.)	**18**	×1.83

Table 5: Evaluation results of test accuracy (%) with XLM-RoBERTa (278M) (Conneau et al., 2019). Best and second best results are highlighted in **boldface** and underlined, respectively.

Methods	Params (%)	Test Accuracy (%)
Full Fine-tuning	100%	85.8
Adapter	0.217%	81.5
LoRA	0.106%	80.7
Prompt-tuning	0.008%	65.8
Compacter	0.021%	77.7
BitFit	0.037%	79.7
AdaMix	0.165%	79.1
C2A (ours.)	0.028%	**84.6**

5.5 Robustness to Client Drifts

In order to showcase the robustness of C2A in non-IID scenarios, we employ CKA similarity to quantify the drift from the global model. Figure 2 shows that C2A is superior to other baselines in effectively reducing client drift. This justifies our hypothesis that creating tailored modules for each client is more effective in non-IID scenarios compared to a one-size-fits-all approach in training a single module for all clients.

6 Related Work

6.1 Parameter-efficient Fine-tuning

Recent works on PEFT can be categorized into two lines of work: (1) tuning a subset of the existing parameters within the PLMs, including head fine-tuning (Lee et al., 2019), and bias tuning (Zaken et al., 2022), (2) tuning with a small amount of additional trainable parameters, such as adapters (Houlsby et al., 2019; Mahabadi et al., 2021a; Yaqing Wang and Gao, 2022), prefix-tuning (Li and Liang, 2021), prompt-tuning (Lester et al., 2021), and low-rank adaption (Hu et al., 2022). Previous studies showed that PEFT achieves comparable performance compared to fine-tuning using only a small set of parameters. Given the advances brought by previous studies focused on centralized datasets, attention towards decentralized scenarios in FL remains under-explored. Yet, we discover that current PEFT approaches suffer from client drifts on non-IID setup, resulting in serious performance degradation in FL. Different from previous studies, we focus on improving the robustness of PEFT in decentralized scenarios by generating client-customized adapters.

6.2 Federated Learning for NLP

While much attention for FL has been focused on the field of computer vision, recent efforts have been done in applying FL to NLP tasks. For example, FedNLP (Lin et al., 2022) introduced benchmarks for evaluating FL methods and performed systematic analysis in the context of PLMs. Weller et al. (2022) examined FL in multilingual scenarios, where each client uses different languages. Similarly, several works attempted to extend the setting toward diverse tasks. For example, Chen et al. (2021) adopted FL for question answering, and Qin et al. (2021) proposed an aspect-based sentiment analysis method to enhance the performance under the restriction of data isolation. However, to the best of our knowledge, none of the prior works has been done on tackling the training complexity of FL on PLMs, which is directly related to the practicality.

6.3 Hypernetworks in PEFT

Prior studies have demonstrated that utilizing hypernetwork (Ha et al., 2017) is conducive to more efficient fine-tuning for PLMs in centralized scenarios. For instance, Hyperformer (Mahabadi et al., 2021b) and HyperPrompt (He et al., 2022) generated task-specific parameters by incorporating task-specific and layer-specific information on multi-task learning. Moreover, for multi-lingual learning, Hyper-X (Üstün et al., 2022) learned about the task and language-specific embeddings for generating adapters. While most previous works have been conducted for improving the efficiency of PEFT by utilizing the hypernetwork, they only focused on multi-task or multi-lingual situations. Instead, our work mitigates the client drifts issue of PEFT in federated scenarios by incorporating the data distributions of each client.

7 Conclusion

In this paper, we have observed significant performance degradation for typical PEFT approaches in decentralized scenarios. By carefully designed analysis, we have also shown that typical PEFT suffers from large client drifts, resulting in slow convergence and performance degradation. To address these issues, we have proposed C2A, a novel hypernetwork-based FL framework, which generates client-customized adapters by incorporating the data distribution of each client. Our experimental results show that C2A achieves state-of-the-art results in various decentralized scenarios. Moreover, we have verified that C2A successfully mitigates the large client drift problem among local clients in FL scenarios.

8 Limitations

While we show that C2A successfully improves the effectiveness and efficiency of PEFT in FL, we have mainly focused on improving the effectiveness of the vanilla adapter. However, it is an open question whether our framework can improve other PEFT approaches, such as prompt tuning(Lester et al., 2021), and LoRA (Hu et al., 2022). Although we didn't analyze whether our framework can generate parameters for alternative PEFT, one recent approach reveals that hypernetworks can generate parameters for various types of PEFT in multi-task learning (He et al., 2022; Üstün et al., 2022). Likewise, as C2A generates parameters with hypernetwork, we believe that C2A is highly expected to improve the performance of any alternative PEFT modules.

Ethics Statement

This study covers work that utilizes PLMs, which have a wide variety of positive applications, such as the application to summarization, or language understanding. At the same time, there are a number of ethical concerns with PLMs in general, including concerns regarding the generation of biased or discriminative text (Bordia and Bowman, 2019), the leakage of private information from training data (Carlini et al., 2021), and the environmental impact of training or tuning them (Strubell et al., 2019).

Our framework attempts to train PLMs with minimal changes made to their pre-existing parameters in FL scenarios. Our work is believed to bring some insights into the two ethical dimensions: privacy and environment. First, with respect to private information leakage, although our work has not addressed address the privacy issue in the pre-train process, our FL framework can mitigate the data privacy issues in the fine-tuning stages. In addition, with respect to environmental impact, our work may obviate the need for full fine-tuning, which may also significantly reduce the cost in terms of memory or deployed servers.

Acknowledgment

This work was supported by the Basic Research Program through the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2021R1A2C3010430) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00079, Artificial Intelligence Graduate School Program (Korea University)).

References

Guillaume Alain and Yoshua Bengio. 2014. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593.

Shikha Bordia and Samuel R. Bowman. 2019. Identifying and reducing gender bias in word-level language models. In *Proc. the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 7934–7949.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. In *USENIX Security Symposium*, pages 2633–2650.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2021. Fedmatch: Federated learning over heterogeneous question answering data.

In *Proc. the ACM Conference on Information and Knowledge Management (CIKM)*, pages 181–190.

Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. 2019. Federated learning of n-gram language models. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, pages 121–130.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Wietse de Vries, Andreas van Cranenburgh, and Malvina Nissim. 2020. What's so special about bert's layers? a closer look at the nlp pipeline in monolingual and multilingual models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1273–1282.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *Proc. the International Conference on Learning Representations (ICLR)*.

Yun He, Huaixiu Steven Zheng, Yi Tay, Jai Prakash Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, YaGuang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. 2022. Hyperprompt: Prompt-based task-conditioning of transformers. In *Proc. the International Conference on Machine Learning (ICML)*, pages 7934–7949.

Wenxin Hou, Han Zhu, Yidong Wang, Jindong Wang, Tao Qin, Renjun Xu, and Takahiro Shinozaki. 2021. Exploiting adapters for cross-lingual low-resource speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pages 317–329.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proc. the International Conference on Machine Learning (ICML)*, pages 2790–2799.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proc. the International Conference on Learning Representations (ICLR)*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, pages 1022–1035.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *Proc. the International Conference on Machine Learning (ICML)*, pages 5132–5143.

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. 2019. Similarity of neural network representations revisited. In *Proc. the International Conference on Machine Learning (ICML)*, pages 3519–3529.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proc. the International Conference on Machine Learning (ICML)*, pages 331–339.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *CoRR*, abs/1911.03090.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3045–3059.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130.

Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proc. the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10713–10722.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proc. the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4582–4597.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. XGLUE: A new benchmark datasetfor cross-lingual pre-training, understanding and generation. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018.

Bill Yuchen Lin, Chaoyang He, Zihang Ze, Hulin Wang, Yufen Hua, Christophe Dupuy, Rahul Gupta, Mahdi Soltanolkotabi, Xiang Ren, and Salman Avestimehr. 2022. Fednlp: Benchmarking federated learning methods for natural language processing tasks. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 157–175.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proc. the International Conference on Learning Representations (ICLR)*.

Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. 2021. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. pages 5972–5984.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021a. Compacter: Efficient low-rank hypercomplex adapter layers. In *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, pages 1022–1035.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021b. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proc. the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 565–576.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTAT)*, pages 1273–1282.

Han Qin, Guimin Chen, Yuanhe Tian, and Yan Song. 2021. Improving federated learning for aspect-based sentiment analysis via topic memories. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3942–3954.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proc. the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7934–7949.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proc. the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5217–5227.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, Gertjan van Noord, and Sebastian Ruder. 2022. Hyper-x: A unified hypernetwork for multi-task multilingual transfer. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7934–7949.

Betty Van Aken, Benjamin Winter, Alexander Löser, and Felix A Gers. 2019. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proc. the ACM Conference on Information and Knowledge Management (CIKM)*, pages 1823–1832.

Orion Weller, Marc Marone, Vladimir Braverman, Dawn J. Lawrie, and Benjamin Van Durme. 2022. Pretrained models for multilingual federated learning. In *Proc. the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1413–1421.

Subhabrata Mukherjee Xiaodong Liu Jing Gao Ahmed Hassan Awadallah Yaqing Wang, Sahaj Agarwal and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proc. the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5744–5760.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proc. the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–9.

Zhisheng Zhong, Jiequan Cui, Shu Liu, and Jiaya Jia. 2021. Improving calibration for long-tailed recognition. In *Proc. the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16489–16498.

Xinghua Zhu, Jianzong Wang, Zhenhou Hong, and Jing Xiao. 2020. Empirical studies of institutional federated learning for natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 625–634.

Supplementary Appendix

A Impact of Structure

We analyze the effect with varied dimensions of the client embeddings and factorization in C2A. The detailed results are presented in Figure 5.

Effect of dimensions for client embeddings. To investigate the effect of dimensions for client embeddings, we investigate the number of dimensions in C2A ranging from 1,4,8, and 32, during training. The results are shown in Figure 5(a). We observe that using a larger dimension of embeddings for adapters improves the training efficiency. Specifically, the model using eight dimensions shows the best performance. Thereby, we adopt a client embedding size of 8 in all our models.

Effect of dimensions for factorization. Figure 5(b) represents the impact of latent dimensions for adapters in C2A. The dimension of factorization size 64 appears to be the best. Based on these results, we use an embedding size of 64 in all our models.

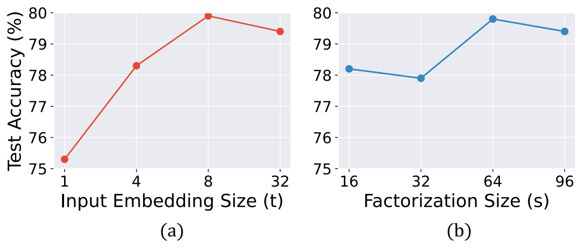


Figure 5: Evaluation results of test accuracy on NC dataset with the different number of dimensions for client embedding and factorization.

B Implementation details for C2A

We implement C2A in Pytorch using four RTX 3090 GPUs for experiments with detailed hyperparameter configurations as follows. We set the dimensionality of latent factors to $s = 64$ and client embeddings size of eight in all our models. Besides, for the low-rank dimension, we use a dimension of 16. We report the average results for all models of four random fine-tunings.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
8

☑ A2. Did you discuss any potential risks of your work?
9

☑ A3. Do the abstract and introduction summarize the paper's main claims?
above the first section

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Not applicable. Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Not applicable. Left blank.

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.