Only Send What You Need: Learning to Communicate Efficiently in Federated Multilingual Machine Translation

Yun-Wei Chu
Purdue University
West Lafayette, Indiana, USA
chu198@purdue.edu

Dong-Jun Han
Purdue University
West Lafayette, Indiana, USA
han762@purdue.edu

Christopher G. Brinton
Purdue University
West Lafayette, Indiana, USA
cgb@purdue.edu

ABSTRACT

Federated learning (FL) is a promising approach for solving multilingual tasks, potentially enabling clients with their own language-specific data to collaboratively construct a high-quality neural machine translation (NMT) model. However, communication constraints in practical network systems present challenges for exchanging large-scale NMT engines between FL parties. In this paper, we propose a meta-learning-based adaptive parameter selection methodology, **MetaSend**, that improves the communication efficiency of model transmissions from clients during FL-based multilingual NMT training. Our approach learns a dynamic threshold for filtering parameters prior to transmission without compromising the NMT model quality, based on the tensor deviations of clients between different FL rounds. Through experiments on two NMT datasets with different language distributions, we demonstrate that MetaSend obtains substantial improvements over baselines in translation quality in the presence of a limited communication budget.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**.

KEYWORDS

Federated Learning, Machine Translation

ACM Reference Format:
Yun-Wei Chu, Dong-Jun Han, and Christopher G. Brinton. 2024. Only Send What You Need: Learning to Communicate Efficiently in Federated Multilingual Machine Translation. In *Companion Proceedings of the ACM Web Conference 2024 (WWW '24 Companion), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3589335.3651931

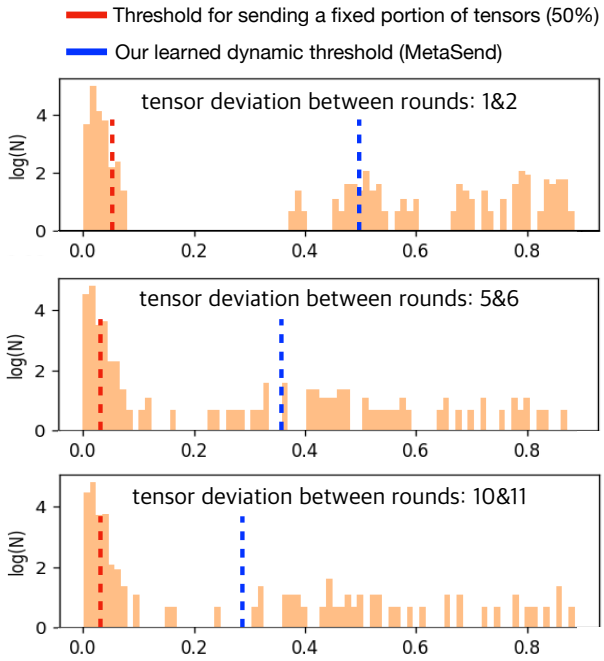


Figure 1: Sample histograms of the difference (absolute-value norms) between tensors of NMT engines computed for clients across consecutive communication rounds in FL training. The traditional method (red thresholds) fails to accurately capture the boundary between clusters during sending, while our MetaSend (blue thresholds) provides a dynamic threshold that adapts to the varying distribution across FL rounds.

1 INTRODUCTION

Federated Learning (FL) has emerged as a popular distributed machine learning paradigm. FL enables collaborative model training among a set of clients via periodic aggregations of local models by a server [9, 15]. The FL property of keeping data local to clients has important privacy advantages that have made it attractive for many learning applications.

Natural language processing (NLP) is one domain standing to benefit from FL since user-generated text may contain sensitive information. Among the applications of FL in NLP, relatively few works have considered *multilingual* NLP and the impact of different languages on FL [14]. In recent years, neural machine translation (NMT) has shown substantial progress in this domain with the advent of large-scale language models such as BERT [2], GPT [22], and their extensions. NMT has a further natural alignment with FL given its setting of non-IID local data distributions [32]: each client (user) typically has a specific language direction they are interested in for translation, which their local dataset will be skewed towards, motivating them to collaborate with each other via FL to construct the general NMT model. This alignment between NMT and FL is particularly significant in the context of the web. With the vast array of languages and diverse translation needs encountered on the web, an NMT model trained via FL can better adapt to the unique language preferences and translation requirements of individual users or communities. By leveraging FL, users can collectively contribute their language-specific data to train a shared NMT

The Web Conference '24, May 13–17, 2024, Singapore

ACM ISBN 979-8-4007-0172-6/24/05
https://doi.org/10.1145/3589335.3651931

model, which, in turn, can provide more contextually appropriate translations for a wide range of content encountered on the web.

On the other hand, resource utilization is often a concern in deploying large-scale NMT models due to demands imposed on computational and memory resources [4, 6]. While FL will distribute the processing load, every client must exchange its model parameters with a central server during the FL communication phases. Communication efficiency is a known bottleneck in traditional FL applications [15] and becomes an even more critical challenge with large-scale NMT models.

In this paper, we are interested in optimizing multilingual NMT performance over an FL system with a limited communication budget. A premise for our work is that exchanging complete NMT engines in FL might not be necessary, similar to the argument in Passban et al. [19]. We can develop some intuition around this through a small FL experiment using the well-known `FedAVG` algorithm [15]. In Figure 1, we perform FL on the UN Corpus dataset (see Section 4 for details) distributed across three clients (each containing one language translation direction), and plot the differences in NMT model tensors between a few consecutive training rounds for one of the clients. These differences are computed and visualized tensor-by-tensor, indicating the deviation of each tensor. We observe that the majority of deviation in the NMT model tensors cluster within small norms, while a small subset of tensors exhibit significant deviations. This observation is consistent across all clients, datasets, and data distribution combinations considered in this study.

Recently, Passban et al. [19] proposed a strategy of focusing on either highly fluctuating or less active tensors during FL communication to reduce computation load. Their approach involves sending only a fixed portion of NMT parameters – namely, by sending either the top 50% or the bottom 50% of tensors based on their deviation, which is computed using the previous round's engine. The result of this is illustrated by the red threshold for the cases in Figure 1. However, sending a fixed portion of the parameters does not account for the fact that the deviation distribution will likely vary dynamically across rounds (as also observed in Figure 1). As a result, this approach is not sensitive to NMT quality, potentially resulting in the transmission of either too many parameters, i.e., extra communication burden without any significant change in translation quality, or too few parameters, i.e., leading to an undesirable model that negatively impacts translation quality.

Contributions. Motivated by this, our objective is to explore *dynamic* thresholding techniques for multilingual federated NMT. The central challenge involved is how to adaptively determine a threshold that selectively filters out parameters from transmission until we expect that the translation quality will start to be compromised. To address this, our methodology, **`MetaSend`**, incorporates a meta-learning approach that generates a dynamic sending threshold adapting to the varying deviation distribution across training rounds. The result is depicted by the blue thresholds for the cases in Figure 1. In doing so, `MetaSend` considers translation quality and communication efficiency as important objectives in multilingual NMT training. In developing `MetaSend`, we make three major contributions:

- We conduct the first research on the communication efficiency of FL in multilingual NMT, and study the relationship between translation quality and the volume of transmitted parameters in multilingual NMT engines.
- We propose a novel meta-learning-based adaptive parameter selection method to partition the tensors at a client into those which have vs. have not evolved significantly since the last FL transmission, balancing between translation quality and communication burden.
- Through extensive experiments on two benchmark NMT datasets with different language distributions, we demonstrate that our `MetaSend` methodology consistently enhances translation quality compared to baselines while operating within a fixed FL communication budget.

2 RELATED WORK

2.1 Efficient NLP

Previous research has explored efficiency enhancements for large NLP models from a computational perspective, i.e., achieving comparable results with fewer resources [31]. Some studies have focused on the data side, e.g., showing how smart downsampling of available datasets can result in equal or improved performance compared to using the entire dataset [11, 34]. On the other hand, efforts to enhance efficiency through model designs include questioning the necessity of full attention heads in large language models and demonstrating that removing certain attention heads does not significantly impact test performance [10, 23, 30]. Compared to these works, motivated by the recent demand for FL in NLP, we specifically focus on communication efficiency in federated multilingual NMT and design a strategy that selectively transmits only the essential parameters of NMT engines for learning.

2.2 FL in NLP

Recent research has begun exploring FL methods for NLP applications requiring privacy preservation [5, 13, 21, 29]. During the FL communication phase, large NLP models are exchanged, introducing a significant communication cost associated with model updates. To address this, Melas-Kyriazi and Wang [16] proposed a gradient compression methodology for language modeling, while Ro et al. [25] leveraged similarities between smaller and larger models in cross-device FL. The parameter selection method proposed in Passban et al. [19] targeting mixed-domain NMT is most relevant to our work. As discussed in Section 1, they analyze how much deviation each client has from its previous round and send a fixed-portion of model's tensors during FL communication. However, sending a fixed amount of tensors without considering the varying deviation distribution can lead to lower translation quality by potentially removing meaningful parameters. To handle these challenges, we design a meta-learning-based parameter selection method that sends the client's parameters with dynamic thresholds to capture the tradeoff between communication efficiency and translation quality.

3 METASEND FOR FEDERATED NMT

3.1 Problem Setup: Federated NMT

When training the NMT model over multiple clients, we follow the general cross-silo FL setting introduced by Li et al. [12]. Algorithm 1

Algorithm 1 Cross-Silo Federated Learning

1: Server $\mathcal{S}$, Client C_k, total number of clients K
2: **for** Each round $r = 1, 2, ..., R$ **do**
3: Each client **Receive**($\mathcal{S}^{r-1}$)
4: $C_k^r \leftarrow$ **Local Iterations**
5: **Send**(C_k^r) to server
6: $\mathcal{S}^r \leftarrow$ **Aggregation**($C_1^r, C_2^r, ..., C_K^r$)
7: **end for**

summarizes the overall training procedure. Each client first runs stochastic gradient descent (SGD) on its local data and then Sends the learned NMT model to the server. The server then executes a global aggregation after receiving all the trained models. We assume the standard FedAVG algorithm [15] as the aggregation method. Using FedAVG as opposed to other aggregation procedures allow us to focus on the communication efficiency of NMT in the FL scenario.

The FedAVG aggregation is defined as:

$$W_s^r = \sum_{k=1}^{K} \frac{n_k}{n} W_k^r, \tag{1}$$

where K is the total number of clients, n_k is the number of samples in the k-th client's dataset, n is the total number of all training data points, and W_s^r and W_k^r are the model parameters at the r-th communication round for server $\mathcal{S}^r$ and the k-th client C_k^r, respectively. The system has finished one FL communication round once the server has completed the aggregation. For the next round, the clients will Receive the server's weights for initialization. The overall process is repeated for $r = 1, 2, ..., R$ FL communication rounds.

3.2 Overview of MetaSend

During FL communication, a large amount of NMT model weights have to be uploaded to the server during the Send action (in Algorithm 1) for aggregation. This communication can be quite costly for a large NMT model, which is a key bottleneck for FL. To tackle this challenge, we propose MetaSend, which adapts the NMT tensors sent based on a customized sending threshold for each communication round. The key idea of MetaSend is to build Model-Agnostic Meta-Learning (MAML) [27] into the FL rounds to balance communication efficiency and translation quality.

Figure 2 and Algorithm 2 summarize the overall procedure. In each round r, after completing the local iterations using their local training data, each client C_k^r will retain its learned model weight W_k^r and training loss L_k^r. In each round r, MetaSend operates according to the following steps: (i) After every client has finished training their local models, the training losses $L_1^r, L_2^r, ..., L_K^r$ of all K clients are inputted into our MAML module, which is implemented as a multi-layer perceptron (MLP) network. The MAML module serves as a server-side component that leverages the clients' losses to learn a threshold, which is subsequently shared with all clients. The purpose of this module is to generate a customized threshold θ^r based on the extracted losses (line 5 in Algorithm 2), which should consider the anticipated impact on learning performance. (ii) Based on the threshold θ^r, each client C_k^r selects which model tensors to send based on a deviation comparison with its previous version C_k^{r-1} (line 6 in Algorithm 2). (iii) After receiving the transmissions, the server $\mathcal{S}^r$ executes the aggregation by taking the resulting models' weights from all clients (line 8 in Algorithm 2). (iv) Subsequently, the MAML module is updated through meta-learning (line 9 in Algorithm 2), taking into account the translation quality of the global model at the server $\mathcal{S}^r$.

There are two key challenges in designing MetaSend: (a) How to design the sending criterion for NMT models in Step (ii)? (Answered in Section 3.3); (b) How can the MAML module effectively learn to produce a customized threshold for each FL round in Step (iv)? (Answered in Section 3.4)

3.3 MetaSend: Customized Sending and Aggregation

In this subsection, we answer the first question mentioned above. Our intuition is that the extent of model parameter deviation relative to the original norm provides an indication of whether information is worth sending. Our observation in Section 1 shows that the tensors of the NMT model responsible for learning exhibit a clustered pattern in the deviation distribution.

Compared with the clients in the previous round, MetaSend will first compute the deviation (dev) for each tensor, with dev defined as:

$$dev = \frac{||W_k^r(\ell) - W_k^{r-1}(\ell)||}{||W_k^{r-1}(\ell)||}, \tag{2}$$

where $\ell \in \mathcal{L}$ denotes a particular tensor of the model, and $||\cdot||$ is the absolute-value norm that measures the difference between clients' weights in different rounds. Based on dev and the learned threshold θ^r (line 6 in Algorithm 2), MetaSend may select each tensor to be sent based on one of two criteria: whether its dev is greater (g) or less (l) than the threshold θ^r. Each of these has potential advantages: deviations above the threshold (g) will promote sending tensors that have experienced the largest changes, which could potentially be an informative or noisy update, while deviations below the threshold (l) will encourage more gradual tensor refinements that are not susceptible to sudden large fluctuations. As a result, MetaSend generates two sending methods, namely $\texttt{MetaSend}_g$ and $\texttt{MetaSend}_l$:

$$\begin{cases} \texttt{MetaSend}_g : W_k'^r(\ell) = \{W_k^r(\ell); dev \geq \theta^r\}, \\ \texttt{MetaSend}_l : W_k'^r(\ell) = \{W_k^r(\ell); dev < \theta^r\}, \end{cases} \tag{3}$$

where $W_k'^r$ represents the selected model's weights for the k-th client in round r.

Given the resulting weights $W_k'^r$ of every client, the server then executes aggregation via FedAVG (line 8 in Algorithm 2). Formally, Equation 1 is:

$$W_s^r = \sum_{k=1}^{K} \frac{n_k}{n} W'^r_k. \tag{4}$$

3.4 MetaSend: MAML Module Update

To address the second question, we aim for our MAML module to generate an adaptive sending threshold based on translation quality. Figure 3 shows the learning process and the optimization flow of this module. ϕ^r represents the hyperparameter set of our

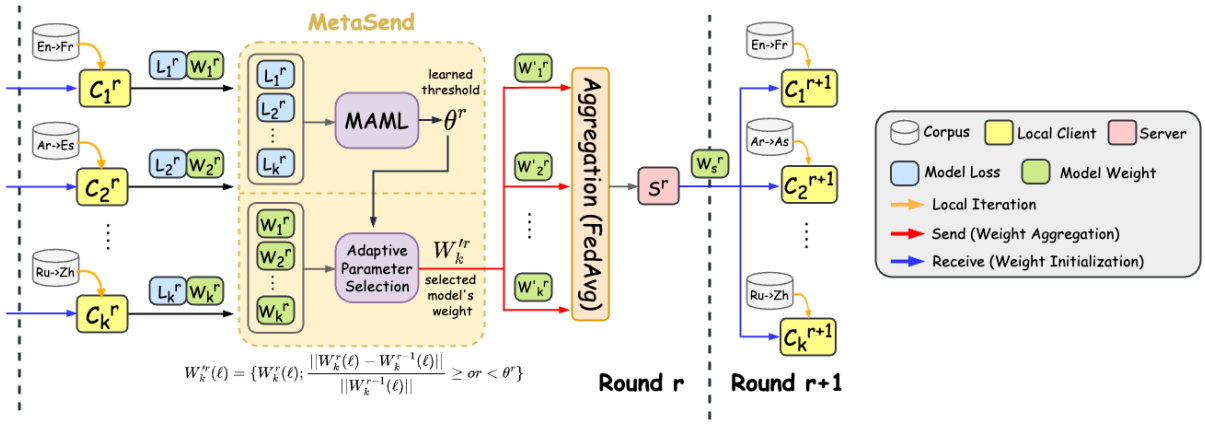


Figure 2: Overview of MetaSend for federated NMT. MetaSend enables clients to adaptively select important parameters of NMT models based on a learned threshold for each communication round. Each client sends only a subset of model tensors to the server for aggregation, enhancing efficiency within a limited communication budget.

Algorithm 2 FL with MetaSend

Model Parameter: W_s for server $\mathcal{S}$, W_k for each client C_k , ϕ for MAML module (MLP)
for Each round $r = 1, 2, ..., R$ **do**
  Initialize all K clients by W_s^{r-1}
  $W_k^r, L_{train}(W_k^r) \leftarrow$ **Local Iterations**
  $\theta^r \leftarrow$ **MLP**$(L_{train}(W_1^r), ..., L_{train}(W_K^r))$
  ${W'}_k^r \leftarrow$ **MetaSend**(W_k^r, θ^r)
  Send resulting local model ${W'}_k^r$ to server
  $W_s^r \leftarrow$ **Aggregation**$({W'}_1^r, ..., {W'}_K^r)$
  $\phi^r \leftarrow$ **MetaUpdate**$(L_{val}(W_s^r))$
end for

MAML module in communication round r, and $\theta^r(\phi^r)$ is the generated threshold from the module. After the parameter selection process and aggregation (Equations 3 and 4), the parameters of the k-th resulting client model and the global model can be expressed as $W_k'^r(\phi^r)$ and $W_s^r(\phi^r)$, respectively. To assess the quality of the global model $W_s^r(\phi^r)$, we randomly select b batches of samples from the validation dataset and evaluate the global model using these samples. Subsequently, we employ the validation loss $L_{val}(W_s^r(\phi^r))$ as the optimization objective for the MAML module, which encourages the module to adapt in the direction of superior translation quality. Thus, our MAML module update can be written as:

$$\phi^{r+1} = \phi^r - \beta \cdot \nabla_\phi L_{val}(W_s^r(\phi^r)), \tag{5}$$

where β is the learning rate for the meta update. By optimizing the MAML module with consideration of the translation quality of the global NMT model, our MAML module can generate a customized threshold θ^r for each round that considers both the deviation distribution and the translation quality. We will see in Section 5 how this process of learning what parameters to send results in substantial translation quality and communication efficiency improvements.

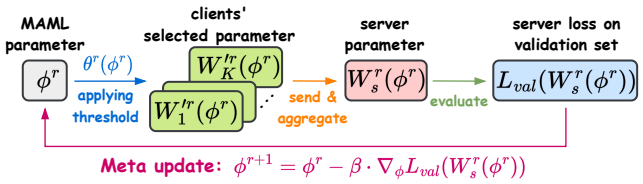


Figure 3: Optimization of our MAML module in an FL setup. The module aims to adapt the sending threshold based on NMT model quality.

4 EXPERIMENTAL DESIGN

4.1 Datasets and Client Partitioning

We utilize two NMT datasets: MTNT [17] and UN Corpus [36]. The MTNT dataset comprises English to French (En → Fr) and English to Japanese (En → Ja) translations, while the UNMT dataset includes three official UN language directions: English to French (En → Fr), Arabic to Spanish (Ar → Es), and Russian to Chinese (Ru → Zh). Three training settings are considered: (i) centralized training without FL, (ii) FL with IID data, where the data for each client is sampled randomly from all data, and (iii) FL with non-IID data, where each client only sees data for one language direction. See Appendix A for more details on the datasets.

4.2 Base Model and Evaluation Metrics

Following the multilingual FL experimental settings in Weller et al. [32], we use the M2M-100 model [3] to conduct machine translation. The M2M-100 model is a sequence-to-sequence model with 418M parameters, and it can translate between any pair of 100 languages. We measure the quality of translation by using sacreBLEU [20] and COMET [24]. SacreBLEU is a commonly used metric for evaluating NMT quality, while COMET is a more advanced metric that shows some degree of correlation with human judgement. This is particularly relevant as some NLP studies [7, 8, 26, 28, 35] have highlighted the limitations of traditional BLEU-based metrics in capturing human preferences accurately. See Appendix A for more details on these translation metrics.

Training	Method	MTNT			UNMT			
		En → Fr	En → Ja	Avg	En → Fr	Ar → Es	Ru → Zh	Avg
Centralized	w/ pre-trained	33.4	21.7	27.6	44.5	44.5	45.0	44.7
IID FL	PMFL	32.3	17.9	25.1	43.7	39.2	42.1	41.7
Non-IID FL	PMFL	20.6	13.9	17.2	30.1	30.8	29.4	30.1
IID FL	RandSend	29.7	11.4	20.6	35.2	33.7	36.9	35.3
	DP_g	31.1	14.6	22.9	37.9	38.5	38.7	38.4
	DP_l	31.9	15.5	23.7	38.4	38.7	39.2	38.8
	$MetaSend_g$ (ours)	32.0	16.8	24.4	42.0	39.1	**41.9**	41.0
	$MetaSend_l$ (ours)	**32.7**	**17.3**	**25.0**	**42.3**	**39.3**	41.6	**41.1**
Non-IID FL	RandSend	16.1	10.0	13.1	22.6	24.3	16.8	21.2
	DP_g	19.0	12.6	15.8	29.8	28.3	26.5	28.2
	DP_l	18.3	13.1	15.7	30.5	29.9	26.1	28.8
	$MetaSend_g$ (ours)	19.7	**13.9**	16.8	31.2	30.1	**29.7**	30.3
	$MetaSend_l$ (ours)	**20.1**	13.6	**16.9**	**32.8**	**31.3**	**29.7**	**31.3**

Table 1: SacreBLEU scores obtained with centralized and FL (IID and Non-IID) methods for various strategies on the MTNT and UNMT datasets. The bold scores indicate that `MetaSend` outperforms other methods in all cases.

Besides translation quality, we consider two metrics for FL efficiency: tensor saving and processing time. Tensor saving is defined as the ratio of tensors that are not exchanged between the server and clients during the `Sending` step in Algorithm 1 (or line 7 in Algorithm 2). For efficiency evaluation, we will report the average tensor savings and the exact processing time over all training rounds.

4.3 FL Training and MAML Module

We build our FL experiments using the Flower framework [1] for training and evaluation. For centralized experiments, we train models for 50 epochs and discuss the effect of pre-trained knowledge for NMT. For every FL experiment, we train each method for 25 communication rounds (epochs) and initialize the clients using a pre-trained M2M-100 model from Hugging Face's transformers library [33]. As a reference, we also conduct FL experiments by initializing the clients' model with random weights; the corresponding results can be found in Section 5.3.4.

For our MAML module, we use a multi-layer perceptron (MLP) network with one hidden layer containing 100 neurons as the default setting. The ablation study in Section 5.3 presents the results of `MetaSend` considering different numbers of neurons in the MAML module. To randomly sample a small portion of the validation set for the MAML update, we use 16 batches (i.e., $b = 16$). In Section 5.3, we also present an ablation study on b to see the effect of MAML optimization for NMT quality. See Appendix B for detailed hyperparameters and compute settings.

4.4 Baselines

We use several competitive baseline approaches and parameter selection strategies to evaluate federated NMT. **PMFL** [32] is the basic FL framework that uses a pre-trained model for federated NMT without any decision-making mechanism. **DP_g** and **DP_l** are the recent methods from [19] that select which tensors to send by comparing the norm difference between the previous and current client models. Their thresholding mechanism sorts based on the norm difference and either send the top 50% (DP_g) or bottom 50% (DP_l) of tensors during the aggregation process. We also include the results from a random configuration, **RandSend**, which randomly sends 50% of the tensors during FL aggregation.

5 EXPERIMENTAL RESULTS

5.1 Translation Performance Evaluation

Table 1 and 2 present the SacreBLEU and COMET results of the translation task for both datasets. In the first section, we observe that the centralized method outperforms PMFL methods, as we would expect; on the other hand, it compromises data privacy by not preserving individual client data confidentiality. Further, the performance decrease of PMFL from IID to non-IID FL training reveals the challenges in the practical NMT scenario of clients having only single language directions.

By randomly sending the model parameters, RandSend achieves the lowest performance among all methods for both IID and non-IID FL. The significant performance improvements of `MetaSend` over DP show the advantage of modeling an adaptive sending threshold based on the norm difference distribution. Moreover, our MAML-learned threshold learns what to send during communication to better optimize the NMT task. Note that this threshold is dynamic and can adapt to different norm difference distributions in each round. Specifically, our `MetaSend` method achieves average sacreBLEU improvements of 3.9 and 3.4 points over DP on IID and non-IID data, respectively. Among all the parameter selection methods, $\texttt{MetaSend}_l$ achieves the highest scores in both sacreBLEU and COMET metrics. It demonstrates comparable translation quality to PMFL, indicating its ability to preserve communication resources without compromising translation quality. Translation examples generated by our method are provided in Appendix C, where it is evident that our method shows better alignment with the ground truth regarding sentiment and accurate word usage.

5.2 Communication Efficiency Evaluation

In Figure 4, we present the average sacreBLEU score and tensor savings for each method across 25 communication rounds on the MTNT (a and b) and UNMT (c and d) datasets. The RandSend and

Training	Method	MTNT			UNMT			
		En → Fr	En → Ja	Avg	En → Fr	Ar → Es	Ru → Zh	Avg
Centralized	w/ pre-trained	0.778	0.759	0.769	0.875	0.863	0.855	0.864
IID FL	PMFL	0.758	0.734	0.746	0.853	0.839	0.830	0.841
Non-IID FL	PMFL	0.666	0.659	0.663	0.737	0.742	0.709	0.729
IID FL	RandSend	0.726	0.715	0.721	0.811	0.773	0.811	0.798
	DP_g	0.729	0.721	0.725	0.825	0.801	0.823	0.816
	DP_l	0.737	0.730	0.734	0.830	0.805	0.829	0.821
	$MetaSend_g$ (ours)	**0.755**	0.733	0.744	0.846	**0.829**	**0.833**	**0.836**
	$MetaSend_l$ (ours)	**0.755**	**0.736**	**0.746**	**0.849**	0.827	0.831	**0.836**
Non-IID FL	RandSend	0.639	0.623	0.631	0.651	0.663	0.659	0.658
	DP_g	0.651	0.650	0.651	0.709	0.720	0.684	0.704
	DP_l	0.659	0.648	0.6534	0.713	0.729	0.692	0.711
	$MetaSend_g$ (ours)	**0.668**	0.653	0.661	0.744	0.739	0.713	0.732
	$MetaSend_l$ (ours)	0.666	**0.659**	**0.663**	**0.749**	**0.744**	**0.721**	**0.738**

Table 2: COMET scores obtained with centralized and different FL methods on MTNT and UNMT datasets.

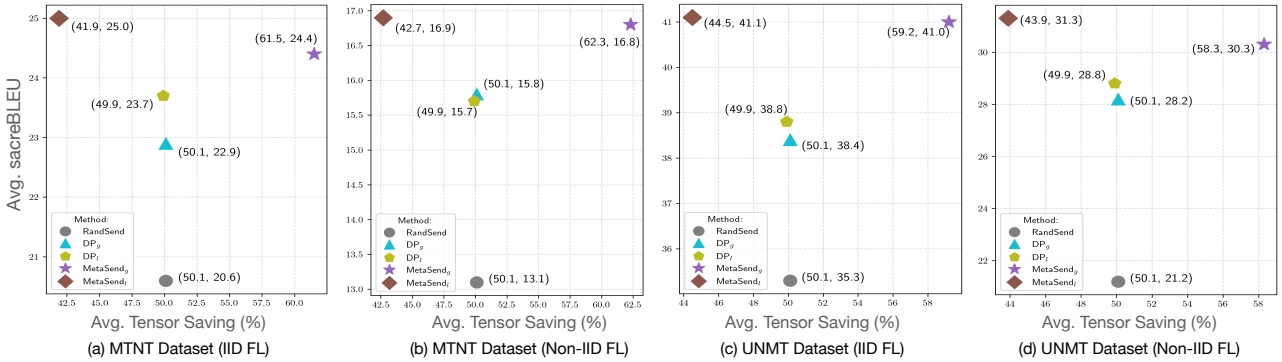


Figure 4: The average sacreBLEU score and the amount of tensor savings for each method. We see that $\texttt{MetaSend}_g$ and $\texttt{MetaSend}_l$ exhibit different tradeoffs between tensor savings and translation quality. Across all settings, $\texttt{MetaSend}_l$ consistently demonstrates improvement over baselines in terms of translation performance and resource savings.

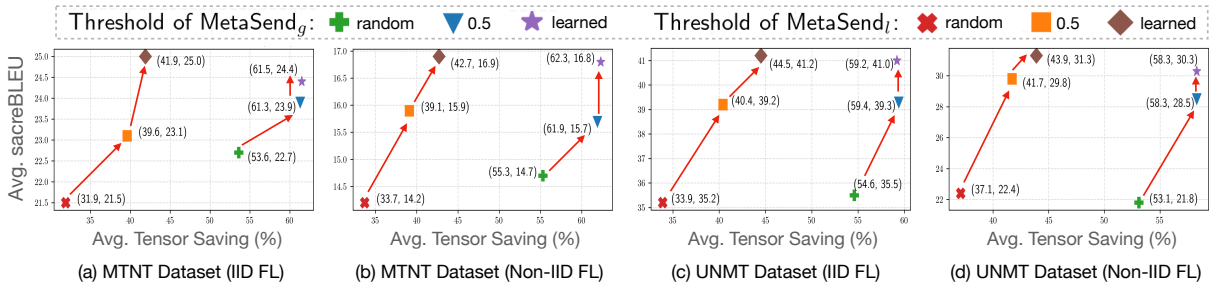


Figure 5: Average sacreBLEU score and tensor savings of $\texttt{MetaSend}$ with different sending thresholds.

DP methods can save around 50% of tensors during FL communication due to their designs. By sending the tensors based on a specific learned threshold, $\texttt{MetaSend}$ methods obtain substantial improvements in translation quality compared to DP methods, while also obtaining varying degrees of tensor savings. Among our two $\texttt{MetaSend}$ methods, $\texttt{MetaSend}_l$ outperforms in translation quality, indicating that sending the majority of tensors for update ensures significant performance improvement. On the other hand, $\texttt{MetaSend}_g$ demonstrates higher tensor savings, with an average of 10.3% more tensors saved compared to DP methods.

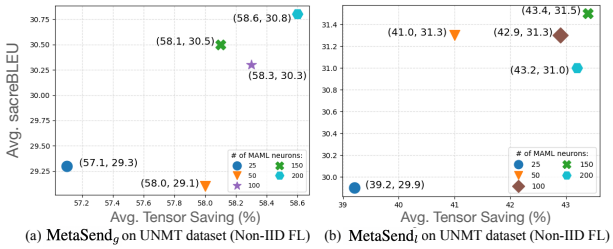
In addition to evaluating tensor savings, Table 3 provides the specific training time for each method. During local iterations, all methods require a similar amount of time to process the entire local training dataset and update all parameters in the model. Although PMFL does not spend time on the parameter selection process, it consumes the most time in client communication and aggregation

Method	Local Training	Compare Layers & Select	MAML Module	Send Clients	Aggregation	Total Time (w/o Local Training)
PMFL	✓	–	–	32.712	18.224	50.936
DP_g	✓	5.492	–	22.359	11.195	39.046
DP_l	✓	5.510	–	23.107	12.371	40.988
$MetaSend_g$	✓	3.547	0.878	21.395	10.795	**36.615**
$MetaSend_l$	✓	3.485	0.893	23.519	12.914	40.811

Table 3: The average training time (in seconds) spent over 25 training rounds for each method on our machine.

Operation	Time (sec)
Forward MLP (size:100) & Ouput θ^r	7.275×10^{-4}
Meta Evaluation (16 batches)	8.754×10^{-1}
Meta Optimization	1.137×10^{-3}

Table 4: Detailed time spent within our MAML module.



(a) $MetaSend_g$ on UNMT dataset (Non-IID FL) (b) $MetaSend_l$ on UNMT dataset (Non-IID FL)

Figure 6: Average sacreBLEU scores and tensor savings for MetaSend with varying numbers of neurons in the MAML module on the UNMT dataset in Non-IID FL.

due to the necessity of transmitting and aggregating every tensor in the model. Both DP and MetaSend require computation time for calculating deviations between a client's current tensors and its tensors from the previous round. However, DP carries out its operation by selecting either the top or bottom 50% of tensors, which occurs after all deviations have been calculated and sorted. In contrast, MetaSend immediately decides whether to send a tensor based on the learned threshold after calculating a single deviation and without any sorting calculation. As a result, MetaSend requires less time for parameter selection compared to DP.

Both $MetaSend_l$ and $MetaSend_g$ involve additional computation for the MAML module, and they spend a similar amount of time on this module as it is independent of the operator. The breakdown of the time spent on the MAML module for MetaSend methods is provided in Table 4. We observe that the most time-consuming configuration is the meta-evaluation, which requires forward-passing a few batches to the global model and obtaining the validation loss. Therefore, we conduct an ablation study in Section 5.3 to examine the impact of the number of sampled batches. In sum, MetaSend significantly enhances translation quality while achieving greater resource savings compared to both PMFL and DP methods.

5.3 Ablation Studies

5.3.1 Effectiveness of learned threshold. To isolate the impact of the sending threshold, we compare MetaSend with different thresholds, including our learned threshold, a fixed threshold ($\theta^r = 0.5$), and a random threshold selected from 0 to 1. The red arrows in Figure 5 show the improvements in MetaSend when using different thresholds within a single operator (l or g). Compared with other thresholds, our learned threshold significantly increases both translation quality and the amount of tensor savings. By comparing Figures 5 and 4, MetaSend with a fixed threshold sometimes outperforms DP methods, suggesting that sending based on the deviation distribution should be considered instead of sending approximately half of the model's tensors.

5.3.2 Parameters used in MAML module. To explore the impact of the number of neurons on performance, we keep the learning parameters consistent while varying the number of neurons in the hidden layer of the MAML module. In Figure 6, we see that using more neurons in the MAML module generally leads to improved results in terms of sacreBLEU score and tensor savings. The performance gain from using more neurons is intuitive since it provides additional degrees of freedom for learning optimization. However, it is important to note that using more neurons also incurs higher resource requirements during system construction.

5.3.3 Meta evaluation for MAML module. To examine the influence of the number of batches used to optimize our MAML module, Figure 7 shows the performance of our method with different numbers of batches used for optimization. We see that increasing the number of samples used for MAML optimization generally results in improved translation quality and efficiency. Naturally, using more batches will increase the exact time spent on our MAML module. However, the time taken by clients to send parameters for aggregation may be more critical than this optimization time, as the optimization process is performed only once in each round.

5.3.4 The use of pre-trained model. Tables 5 shows the sacreBLEU scores of each method without utilizing pre-trained knowledge for the M2M-100 model in each client. Specifically, each method was trained from scratch, utilizing randomly initialized weights. Although the results of $MetaSend_l$ do not exhibit complete superiority over $MetaSend_g$ as discussed in Section 5.1, our MetaSend methods display higher average scores overall in most cases compared with baseline approaches. However, the overall results are relatively poor and unstable compared to the results shown in Table 1 when pre-training is utilized. This discrepancy can be attributed to the fact that large neural MT systems typically demand a substantial amount of data and prior knowledge to perform effectively.

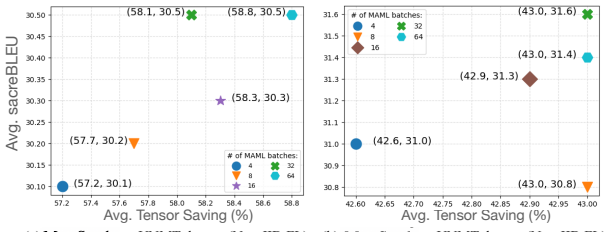
5.3.5 Insufficient data sample. To mirror the limited data scenario of each client in practical FL, we performed experiments by randomly sampling a small portion of data from MTNT and UNMT

Training	Method	MTNT			UNMT			
		En → Fr	En → Ja	Avg	En → Fr	Ar → Es	Ru → Zh	Avg
Centralized	w/o pre-trained	18.1	11.0	14.6	27.9	25.5	27.2	26.9
Non-IID FL	PMFL	14.9	7.0	10.9	18.9	16.6	18.0	17.8
Non-IID FL	RandSend	11.4	5.5	8.5	13.6	10.3	12.0	11.9
	DP_g	14.6	6.6	10.6	16.3	14.9	15.0	15.4
	DP_l	14.9	6.8	10.9	16.6	15.4	15.3	15.8
	$MetaSend_g$ (ours)	14.9	6.9	10.9	**18.5**	16.6	**16.3**	17.1
	$MetaSend_l$ (ours)	**15.1**	**7.3**	**11.2**	17.7	**18.2**	16.0	**17.3**

Table 5: SacreBleu scores obtained with each method without using pre-trained weights as initialization. We observe that our method shows superiority over baselines even with limited prior knowledge.

Training	Method	MTNT			UNMT			
		En → Fr	En → Ja	Avg	En → Fr	Ar → Es	Ru → Zh	Avg
Centralized	w/o pre-trained	10.7	6.1	8.4	13.9	11.8	12.5	12.7
Centralized	w/ pre-trained	28.4	14.3	21.4	34.6	33.8	34.8	34.4
Non-IID FL	PMFL	16.6	8.8	12.7	19.4	19.6	17.7	18.9
Non-IID FL	RandSend	14.9	7.4	11.1	18.2	17.4	16.4	17.3
	DP_g	15.3	**7.9**	11.6	18.4	19.5	16.1	18.0
	DP_l	15.0	7.3	11.2	18.9	19.0	16.4	18.1
	$MetaSend_g$ (ours)	15.8	**7.9**	**11.9**	**19.5**	19.5	17.5	18.8
	$MetaSend_l$ (ours)	**16.0**	7.8	**11.9**	19.4	**19.7**	**17.7**	**18.9**

Table 6: SacreBleu scores obtained with each method with the reduced number of training samples. We observe that our method shows improved translation quality over baselines even with limited data resources in each client.



(a) $MetaSend_g$ on UNMT dataset (Non-IID FL) (b) $MetaSend_l$ on UNMT dataset (Non-IID FL)

Figure 7: Average sacreBLEU scores and tensor savings for `MetaSend` with different MAML module batch numbers on the UNMT dataset in Non-IID FL.

Meta Evaluation	Time (sec)
4 batches	2.621×10^{-1}
8 batches	5.012×10^{-1}
16 batches (default)	8.754×10^{-1}
32 batches	1.731
64 batches	3.682

Table 7: Time spent for passing different numbers of batch samples to the NMT engine for meta-evaluation.

datasets as training data. Specifically, we randomly selected 20% of samples from each dataset, resulting in 1k and 10k training samples in each language direction for MTNT and UNMT datasets, respectively, while keeping the validation and test sets at the same size. All other hyperparameters, such as batch size for the NMT engine or MAML optimization, neurons in the MAML module, and learning rate, remain the same as in Section 5. Table 6 presents the translation quality of each method when trained with limited data. Even in scenarios with limited resources, it is evident that our `MetaSend` methods consistently outperform other baselines and, in some cases, achieve comparable or even slightly better performance compared to PMFL.

6 CONCLUSION

This work has pioneered an in-depth exploration into the efficiency challenges of federated multilingual NMT, unveiling the intricacies that hinder seamless communication. To address the practical challenges that arise in this setup, we proposed `MetaSend`, which selects tensors for transmission that are most critical to the NMT. By adaptively learning the sending threshold in each FL round based on meta-learning, we saw that `MetaSend` not only improves communication efficiency, but also effectively captures the NMT threshold for sending. Our extensive experiments across two diverse NMT datasets underscored the prowess of `MetaSend`, encompassing a range of scenarios and in-depth ablation studies. Not only did it outperform existing baselines in terms of machine translation quality, but it also showcased a remarkable reduction in communication costs within the federated learning framework. These results solidify the practical applicability and superiority of `MetaSend`.

ACKNOWLEDGEMENTS

This work is partly supported by NSF grants CNS-2146171 and CPS-2313109, as well as an ONR grant N00014-23-C-1016.

REFERENCES

[1] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *ArXiv* abs/2007.14390 (2020).

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[3] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Çelebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2021. Beyond English-Centric Multilingual Machine Translation. *J. Mach. Learn. Res.* 22 (2021), 107:1–107:48.

[4] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Y. Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. 2020. Compressing Large-Scale Transformer-Based Models: A Case Study on BERT. *Transactions of the Association for Computational Linguistics* 9 (2020), 1061–1080.

[5] Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2020. FedNER: Medical Named Entity Recognition with Federated Learning. *arXiv: Computation and Language* (2020).

[6] Manish Gupta and Puneet Agrawal. 2020. Compression of Deep Learning Models for Text: A Survey. *ACM Trans. Knowl. Discov. Data* 16 (2020), 61:1–61:55.

[7] Chi-Yang Hsu, Yun-Wei Chu, Vincent Chen, Kuan-Chieh Lo, Chacha Chen, Ting-Hao 'Kenneth' Huang, and Lun-Wei Ku. 2022. Learning to Rank Visual Stories From Human Ranking Data. In *Annual Meeting of the Association for Computational Linguistics*.

[8] Chi-Yang Hsu, Yun-Wei Chu, Ting-Hao 'Kenneth' Huang, and Lun-Wei Ku. 2021. Plot and Rework: Modeling Storylines for Visual Storytelling. In *Findings*. https://api.semanticscholar.org/CorpusID:234682123

[9] Jakub Konecný, H. B. McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *ArXiv* abs/1610.05492 (2016).

[10] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. In *Conference on Empirical Methods in Natural Language Processing*.

[11] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2021. Deduplicating Training Data Makes Language Models Better. In *Annual Meeting of the Association for Computational Linguistics*.

[12] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1. https://doi.org/10.1109/TKDE.2021.3124599

[13] 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*. Association for Computational Linguistics, Seattle, United States, 157–175. https://doi.org/10.18653/v1/2022.findings-naacl.13

[14] Ming Liu, Stella Ho, Mengqi Wang, Longxiang Gao, Yuan Jin, and Heng Zhang. 2021. Federated Learning Meets Natural Language Processing: A Survey. *ArXiv* abs/2107.12603 (2021).

[15] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[16] Luke Melas-Kyriazi and Franklyn Wang. 2022. Intrinsic Gradient Compression for Scalable and Efficient Federated Learning. *Proceedings of the First Workshop on Federated Learning for Natural Language Processing (FL4NLP 2022)* (2022).

[17] Paul Michel and Graham Neubig. 2018. MTNT: A Testbed for Machine Translation of Noisy Text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 543–553. https://doi.org/10.18653/v1/D18-1050

[18] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *ACL*.

[19] Peyman Passban, Tanya Roosta, Rahul Gupta, Ankit Chadha, and Clement Chung. 2022. Training Mixed-Domain Translation Models via Federated Learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 2576–2586. https://doi.org/10.18653/v1/2022.naacl-main.186

[20] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. *ArXiv* abs/1804.08771 (2018).

[21] Han Qin, Guimin Chen, Yuanhe Tian, and Yan Song. 2021. Improving Federated Learning for Aspect-based Sentiment Analysis via Topic Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 3942–3954. https://doi.org/10.18653/v1/2021.emnlp-main.321

[22] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

[23] Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation. *ArXiv* abs/2002.10260 (2020).

[24] Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A Neural Framework for MT Evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 2685–2702. https://doi.org/10.18653/v1/2020.emnlp-main.213

[25] Jae Ro, Theresa Breiner, Lara McConnaughey, Mingqing Chen, Ananda Suresh, Shankar Kumar, and Rajiv Mathews. 2022. Scaling Language Model Size in Cross-Device Federated Learning. In *Proceedings of the First Workshop on Federated Learning for Natural Language Processing (FL4NLP 2022)*. Association for Computational Linguistics, Dublin, Ireland, 6–20. https://doi.org/10.18653/v1/2022.fl4nlp-1.2

[26] Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning Robust Metrics for Text Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 7881–7892. https://doi.org/10.18653/v1/2020.acl-main.704

[27] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. In *Neural Information Processing Systems*.

[28] Koustuv Sinha, Prasanna Parthasarathi, Jasmine Wang, Ryan Lowe, William L. Hamilton, and Joelle Pineau. 2020. Learning an Unreferenced Metric for Online Dialogue Evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2430–2441. https://doi.org/10.18653/v1/2020.acl-main.220

[29] Dianbo Sui, Yubo Chen, Kang Liu, and Jun Zhao. 2021. Distantly Supervised Relation Extraction in Federated Settings. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 569–583. https://doi.org/10.18653/v1/2021.findings-emnlp.52

[30] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2020. Synthesizer: Rethinking Self-Attention in Transformer Models. In *International Conference on Machine Learning*.

[31] Marcos Vinícius Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro Henrique Martins, André F. T. Martins, Peter Milder, Colin Raffel, Edwin Simpson, Noam Slonim, Niranjan Balasubramanian, Leon Derczynski, and Roy Schwartz. 2022. Efficient Methods for Natural Language Processing: A Survey. *ArXiv* abs/2209.00099 (2022).

[32] Orion Weller, Marc Marone, Vladimir Braverman, Dawn Lawrie, and Benjamin Van Durme. 2022. Pretrained Models for Multilingual Federated Learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 1413–1421. https://doi.org/10.18653/v1/2022.naacl-main.101

[33] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

[34] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. *ArXiv* abs/2205.01068 (2022).

[35] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SkeHuCVFDr

[36] Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The United Nations Parallel Corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), Portorož, Slovenia, 3530–3534. https://aclanthology.org/L16-1561

DP	**Output (Zh):** 至于最近,联合国千年宣言把团结列为国际关系至关重要的基本价值观念。 As of late, the United Nations Millennium Declaration listed solidarity as a fundamental value of fundamental importance in international relations.
MetaSend	**Output (Zh):** 最近,联合国千年宣言把团结列为国际关系至关重要的基本价值。 Recently, the United Nations Millennium Declaration listed solidarity as a fundamental value of fundamental importance in international relations.
Ground Truth	**Output (Zh):** 最近,《联合国千年宣言》将团结作为对国际关系至关重要的基本价值之一。 Recently, the United Nations Millennium Declaration identified solidarity as one of the fundamental values essential to international relations.

Figure 8: Translation examples (Ru → Zh) of DP$_l$, MetaSend$_l$, and ground truth. Our method aligns better with ground truth, and DP$_l$ generates redundant tokens ("至于" and "观").

DP	**Output (Zh):** 2007年预算为这些办事处编列经费,但延误了这些办事处的开设时间。 The 2007 budget included funds for these offices, but there was a delay in their opening.
MetaSend	**Output (Zh):** 2007年预算已为这些办事处编列经费,但延迟了这些办事处的开设。 The 2007 budget included funds for these offices but delayed their opening.
Ground Truth	**Output (Zh):** 2007年预算已经编列了这些办事处的经费,但延迟了开设时间。 Funding for these offices was included in the 2007 budget, but the opening was delayed.

Figure 9: Translation examples (Ru → Zh) of DP$_l$, MetaSend$_l$, and ground truth. Our method generates the same sentiment-meaning word ("延迟") as ground truth, while DP$_l$ generates similar but different sentiment-meaning words ("延误").

A DETAILS FOR DATASETS AND EVALUATION METRICS

This paper considers two widely used NMT datasets: MTNT [17] and UN Corpus [36]. The Machine Translation of Noisy (MTNT) dataset [17] was gathered from user comments on Reddit discussion threads. The dataset contains two language directions: English to French (En → Fr) and English to Japanese (En → Ja). The dataset contains 5,605 instances in each direction for training and approximately 1k each for validation and test set. The UN Corpus [36] consists of manually translated UN documents over the years 1990 to 2014, and we consider three official UN language directions: English to French (En → Fr), Arabic to Spanish (Ar → Es), and Russian to Chinese (Ru → Zh). The dataset contains 80k instances in each direction for training and approximately 10k each for validation and test set.

The evaluation metric sacreBLEU is widely used in the machine translation community and is built upon BLEU [18]. Our study uses the standard sacreBLEU settings, including nrefs:1, mixed case, eff:no, tok:13a, smooth:exp, and version 2.0.0. For, Japanese (Ja) and Chinese (Zh), we use their respective tokenizers to ensure accurate evaluation. We utilize the default COMET model as suggested by the authors, which employs a reference-based regression approach and is developed based on XLM-R. This model has been trained on direct assessments from WMT17 to WMT20 and assigns scores ranging from 0 to 1, where 1 indicates a perfect translation. The COMET metric has been found to exhibit segment-level correlation with human evaluations and has demonstrated its potential to distinguish between high-performing systems more effectively.

B HYPERPARAMETERS AND COMPUTE SETTINGS

For all the experiments, the MT engine (M2M-100 model) is optimized using the Adam optimizer. We search the learning rates from [1e-2, 5e-3, 1e-3, 5e-4, 1e-4] and select 5e-3 as the optimal learning rate for the MT engine. The batch size for the MT engine is set to 2 for both client training and updating the MAML module. The MAML module is optimized via Adam optimizer with a learning rate 1e-3, which is also searched from [1e-2, 5e-3, 1e-3, 5e-4, 1e-4].

We run all experiments on a 3-GPU cluster of Tesla V100 GPUs, with each GPU having 32GB of memory. Centralized experiments will be conducted on one of our 3 Tesla V100 GPUs, while FL experiments will utilize K GPUs, where K is the total number of clients. Each centralized experiment ran for approximately four hours and two days for the NTMT and UNMT datasets, respectively, when run on a single GPU. For FL experiments on the NTMT dataset, each simulation was completed in about 2 hours by distributing clients' data on 2 GPUs. For FL experiments on the UNMT dataset, each simulation was run for approximately 14 hours by distributing clients' data on 3 GPUs.

C CASE STUDY

This section provides some translation examples to Section 5.1. Figure 8 illustrates that our method generates translations that closely align with the ground truth by utilizing similar words. In contrast, the baseline method produces translations with redundant tokens, leading to potential confusion within the sentence. Figure 9 shows that our method employs the same words as the ground truth, conveying a neutral sentiment. In comparison, the baseline method generates similar words but with a negative sentiment.