Distributed Fine-tuning of Vision Transformer Models

Muhammad I. Qureshi
Dept. of Electrical and Computer Engineering
Tufts University
Medford, USA
muhammad.qureshi@tufts.edu

Usman A. Khan
Dept. of Electrical and Computer Engineering
Tufts University
Medford, USA
khan@ece.tufts.edu

***Abstract*—In this paper, we propose a framework for fine-tuning vision transformer models to learn attention in distributed settings, where the computational nodes communicate through a peer-to-peer network. The nodes are not allowed to share their private training dataset, but they can share some model parameters with the neighboring nodes. We discuss how the proposed framework helps each node acquire global understanding and attention using only its local dataset. We address the problem of parameter-efficient fine-tuning of large transformer models for downstream learning tasks and demonstrate that our proposed framework enables each computational node to achieve performance comparable to that of a single central device with access to the entire training dataset. We present the fine-tuning results for `ViT`, `DeiT`, and `Swin`-transformer models on a variety of datasets, and we also show their attention maps to provide insights into the distributed learning process of transformers.**

***Index Terms*—distributed fine-tuning, vision transformers, gradient-tracking, attention maps, peer-to-peer networks.**

I. Introduction

Machine learning techniques have recently gained considerable attention for their effectiveness in solving emerging problems in computer vision and natural language processing [1]–[4]. Their rapid advancement is driven by (a) deeper insight into learning methods, (b) powerful computational resources, and (c) easier access to large datasets. In particular, vision transformers have excelled in image understanding tasks [5]–[8], yet training these models is extremely resource-intensive because it involves billions of parameters and requires huge datasets. As a result, training from scratch is often infeasible, making the use of pre-trained models followed by fine-tuning on specific tasks more advantageous, as demonstrated by their success across various applications [9].

Another important consideration in many practical applications is data acquisition, where heterogeneous data is acquired by geographically distributed sensors, making it impractical to consolidate on a single server (computational node). If these distributed nodes are trained on local datasets, they often struggle to generalize well. For example, one node may have images of cats and dogs, while another has images of cars and ships, making it difficult for either to classify all four classes. Moreover, these local datasets cannot be used during pre-training because (i) they often contain private information, and (ii) new data is continually collected, making it unavailable at that stage. Since training an entire model from scratch is practically infeasible across distributed nodes, leveraging existing pre-trained models and fine-tuning them for new downstream tasks becomes crucial.

In this paper, we propose a distributed training framework for parameter-efficient fine-tuning of vision transformers in scenarios where heterogeneous data is distributed across a peer-to-peer network of nodes. The nodes are allowed to communicate with their neighbors, but due to privacy constraints, they are restricted from sharing private data samples. The main contributions of this paper are as follows: (i) we propose a privacy-aware distributed fine-tuning method, `P2P-FT`, which leverages weight-mixing and gradient-tracking strategies. Each node shares a subset of its model parameters (which require fine-tuning), along with the local gradient vectors (evaluated with respect to the fine-tuning parameters), with its neighboring nodes. The proposed fusion strategy enables the estimation of the global gradient and the computation of updated model parameters; (ii) the proposed framework is applicable to any vision transformer, as it encourages the model to learn attention globally and directs the model parameters toward the global solution. In this paper, we provide numerical experiments on **`ViT`**, **`DeiT`**, and **`Swin`** transformer models to highlight the performance of `P2P-FT` and compare the results with local fine-tuning; (iii) we analyze the attention maps generated by `P2P-FT` and compare them with the results generated by locally fine-tuned and centralized models, highlighting the ability to demonstrate improved feature representation learning for previously unseen images; and (iv) we show that `P2P-FT` effectively handles heterogeneous data distributions, eliminating bias introduced by non-uniform data distributions across computational nodes.

II. Preliminaries and Related Work

In this section, we first establish the preliminaries of the fine-tuning problem for vision transformer models. Subsequently, we introduce the distributed peer-to-peer learning framework. Finally, we review related work and highlight the limitations inherent in the existing literature.

This work has been supported by NSF under award PIRE-2230630. Usman A. Khan holds concurrent appointments as a Professor at Tufts University and as an Amazon Scholar with Amazon Robotics. This paper describes work performed at Tufts University and is not associated with Amazon.

A. Fine-tuning Transformers

Now, we formally describe the training process. We denote the model parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ and the pre-training dataset $\mathcal{D}_{pre}$. We can mathematically represent the pre-training problem as: $\min_\theta \left\{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{pre}} L(\mathbf{x};\boldsymbol{\theta})\right\}$, where $L(\mathbf{x};\boldsymbol{\theta})$ is the function that evaluates the loss based on the training data $\mathbf{x}$ sampled from the pre-training dataset $\mathcal{D}_{pre}$. The objective is to minimize the loss by learning the optimal model parameters $\boldsymbol{\theta}^*$, after which we proceed to fine-tune the vision transformers for downstream tasks. This involves taking the pre-trained model and modifying it by replacing the MLP head to match the number of outputs required for our task. Subsequently, the model undergoes training using the task-specific dataset. In this process, a subset of pre-trained parameters $\boldsymbol{\theta}' \subset \boldsymbol{\theta}$ are typically kept constant, while training the remaining parameters $\widehat{\boldsymbol{\theta}}$ to minimize the modified loss $\widehat{L}$:

$$\min_{\widehat{\boldsymbol{\theta}}} \left\{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{tune}} \widehat{L}(\mathbf{x};\widehat{\boldsymbol{\theta}},\boldsymbol{\theta}')\right\},$$

where the training dataset $\mathcal{D}_{tune}$ is specific to the new task. As an example, we can consider the use of a pre-trained "**timm/vit_small_patch16_224**" model (from **timm** library) for classifying the CIFAR-10 dataset. This model has been pre-trained on the ImageNet dataset. To fine-tune **timm/vit_small_patch16_224** for CIFAR-10, we replace the MLP head with one of size 10, corresponding to the number of classes in the dataset. Subsequently, the transformer is trained on this downstream task using the CIFAR-10 dataset. This approach significantly accelerates the training process, as the pre-trained model already encodes valuable feature representations learned from ImageNet, which are leveraged to improve the model's understanding of the CIFAR-10 examples.

B. Distributed Learning Framework

The distributed learning methods consider the problem to be divided over a network of n nodes. Each node possesses its local loss function L_i and some private data $\mathcal{D}_i$. The global problem is to minimize the average of all local cost functions, i.e., for $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_n\}$,

$$\min_{\boldsymbol{\theta}} \left\{ L(\boldsymbol{\theta}, \mathcal{D}) := \frac{1}{n}\sum_{i=1}^{n} L_i(\boldsymbol{\theta}, \mathcal{D}_i) \right\}.$$

An intuitive solution considers the federated learning framework, where the gradients are evaluated at the client nodes and then sent to the server node. The server aggregates these gradients and uses them to update its model parameters. These updated parameters are then transmitted back to the client nodes, allowing them to update their local models using these parameters. The limitation of this approach is the dependence on the reliability of the server and the constraints imposed by network connectivity. In this setup, all clients are required to maintain a bi-directional connection with the server, which can result in significant communication bandwidth demands. Additionally, the server represents a single point of failure within this architecture. In case of a server outage or an attack, all clients are adversely impacted. Hence, there is a demand for methods designed to operate within a fully distributed peer-to-peer network topology. Under the assumption of bi-directional communication, some decentralized federated learning methods are proposed [10], [11]. These methods attempt to comprehend the global dataset using a weight-mixing methodology and a variant of gradient descent. However, this approach is not desirable in large-scale networks due to the high cost associated with bi-directional communication links.

A well-known approach in the literature on distributed optimization considers a first-order gradient descent method `DGD` [12] to minimize the loss when dealing with the data distributed over a peer-to-peer network of nodes. The nodes are prohibited from sharing private data $\mathcal{D}_i$ but can exchange their local model parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ with neighboring nodes. We define $W = \{w_{i,j}\} \in \mathbb{R}^{n\times n}$ as the weight matrix representing network connectivity, where W is assumed to be doubly stochastic for a strongly connected, weight-balanced graph. For each node i, we consider $\boldsymbol{\theta}_i^k \in \mathbb{R}^p$ as the local parameter estimate vector computed at k-th iteration. For simplicity of notation, we define $L_i(\boldsymbol{\theta}_i^k) := L_i(\boldsymbol{\theta}_i^k, \mathcal{D}_i)$. Then, at each iteration `DGD` computes the following:

$$\boldsymbol{\theta}_i^{k+1} \leftarrow \sum_{j=1}^{n} w_{i.j} \left(\boldsymbol{\theta}_i^k - \alpha \nabla L_i(\boldsymbol{\theta}_i^k)\right), \qquad \forall k > 0, \tag{1}$$

where α is the learning rate. We note that $\boldsymbol{\theta}_i^k - \alpha\nabla L_i(\boldsymbol{\theta}_i^k)$ represents a local gradient descent update, while the aggregation helps attaining consensus over all nodes. In summary, each node tries to learn its local solution while being influenced by the parameters possessed by its neighboring nodes. This approach performs well when the data distributions are homogeneous across all nodes. However, in most practical applications, data heterogeneity leads to a notable difference between the global and the local solution. This discrepancy results in a finite gap between local and global losses, i.e., $\forall i$, $\|\nabla L_i(\boldsymbol{\theta}) - \nabla L(\boldsymbol{\theta})\| \neq 0$, which leads to inexact convergence. To address this problem, a gradient-tracking methodology, `GT-DGD`, was introduced in [13]. This involves computing an additional term, $\boldsymbol{\tau}_i^k$, to estimate the gradient of the global loss function, which can be evaluated as:

$$\boldsymbol{\tau}_i^{k+1} \leftarrow \sum_{j=1}^{n} w_{i.j} \left(\boldsymbol{\tau}_i^k + \nabla L_i(\boldsymbol{\theta}_i^{k+1}) - \nabla L_i(\boldsymbol{\theta}_i^k)\right), \quad \forall k > 0,$$

where $\boldsymbol{\theta}_i^0 \in \mathbb{R}^p$ and $\boldsymbol{\tau}_i^0 = \nabla L_i(\boldsymbol{\theta}_i^0)$. `GT-DGD` replaces the local gradient $\nabla L_i(\boldsymbol{\theta}_i^k)$, as evaluated in (1), with $\boldsymbol{\tau}_i^k$. It can be verified that at each node, $\boldsymbol{\tau}_i \to \nabla L$. Consequently, this strategy eliminates the gap between local and global losses caused by the heterogeneous data distribution. Furthermore, distributed processing enables the network to share the computational demand, leading to faster convergence.

Another limitation of both `DGD` and `GT-DGD` is their deterministic nature, which requires the evaluation of full-batch gradients at every iteration. In machine learning applications, dealing with huge datasets makes this computation infeasible. Particularly in streaming scenarios, where data is acquired in

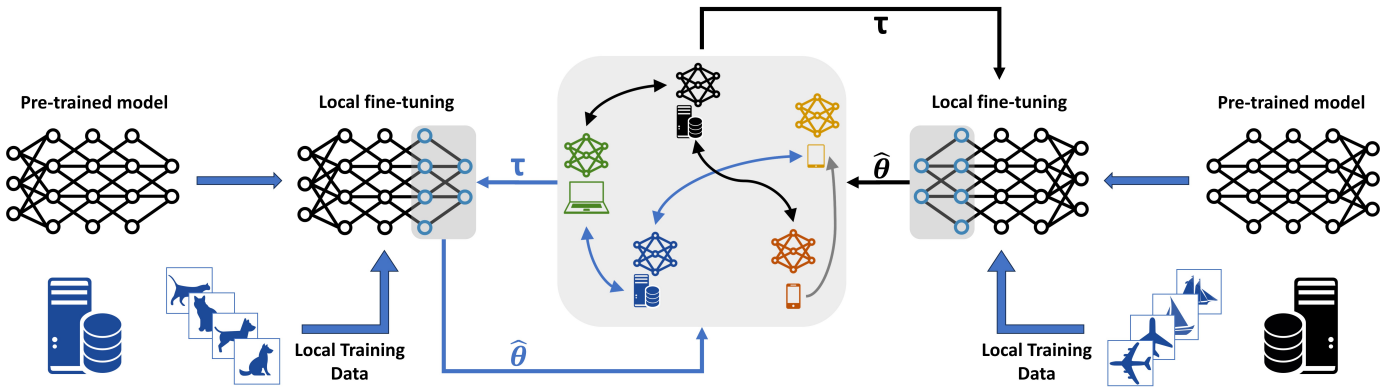

Fig. 1. Proposed framework for distributed fine-tuning of vision transformer models.

real-time, models cannot access the entire dataset to compute the gradient. These situations demand the use of stochastic mini-batch distributed gradient descent methods. Furthermore, it is noteworthy that DGD and GT-DGD are primarily designed for weight-balanced directed networks and cannot be directly applied to generic directed networks [14]–[16]. In the following section, we present our proposed framework, which utilizes distributed gradient descent with gradient-tracking for fine-tuning vision transformer models.

III. Proposed Framework

In this section, we describe a distributed learning framework for fine-tuning transformer model parameters, designed to learn attention over distributed heterogeneous datasets while preserving data privacy. Our proposed method uses efficient weight-mixing and gradient-tracking strategies, enabling us to achieve performance comparable to the centralized fine-tuning setup. We define $\boldsymbol{\theta} \in \mathbb{R}^p$ as the set of all pre-trained model parameters out of which $\boldsymbol{\theta}' \in \mathbb{R}^{p-q}$ are kept constant and $\widehat{\boldsymbol{\theta}} \in \mathbb{R}^q$ are to be fine-tuned. Each node possesses a local dataset $\mathcal{D}_i \in \mathcal{D} := \{\mathcal{D}_1, \cdots, \mathcal{D}_n\}$ and a local loss function $L_i(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}', \mathcal{D}_i)$. We note that $\boldsymbol{\theta}'$ remains fixed throughout the training process. To simplify the notation, we define $L_i(\widehat{\boldsymbol{\theta}}) := L_i(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}', \mathcal{D}_i)$ and the global loss $L(\widehat{\boldsymbol{\theta}}) := \frac{1}{n}\sum_{i=1}^n L_i(\widehat{\boldsymbol{\theta}})$. The goal is to minimize the global loss, which can be expressed as follows:

$$\mathbf{P}: \qquad \min_{\widehat{\boldsymbol{\theta}}} \left\{ L(\widehat{\boldsymbol{\theta}}) := \frac{1}{n}\sum_{i=1}^n L_i(\widehat{\boldsymbol{\theta}}) \right\}.$$

To this aim, we propose a distributed stochastic gradient descent method that leverages weight-mixing and gradient-tracking strategies to update only the fine-tuning parameters using the estimate of the global gradient. We consider n nodes communicating over a strongly connected network, with $W \in \mathbb{R}^{n\times n}$ representing the connectivity matrix. In general, this matrix may not be weight-balanced. To address the asymmetry, we define two mixing matrices $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$, where $a_{i,j} = w_{i,j}/\sum_{j=1}^n w_{i,j}$ and $b_{i,j} = w_{i,j}/\sum_{i=1}^n w_{i,j}$ for all $i, j = \{1, 2, \cdots n\}$. This normalization ensures that A is row stochastic, while B is column stochastic. Furthermore, for each node i, we define $\widehat{\boldsymbol{\theta}}_i^k \in \mathbb{R}^q$ as the estimate of the fine-tuning parameters evaluated at k-th iteration. Figure 1 illustrates the evolution of local models using P2P-FT method. Depending on the type of device, each node may differ in computational capabilities and have different local data classes and sizes. In each iteration, every node shares its local fine-tuning parameters with its neighbors and obtains their local gradient-tracking terms. Subsequently, each node aggregates these gradient estimates and computes a weighted sum to update the local models.

Algorithm 1 formally describes the P2P-FT method. The estimate of the fine-tuning parameters $\widehat{\boldsymbol{\theta}}_i^k$ is initialized (partially) randomly. The updates can be divided into two main steps: (i) computing the gradient-tracking term $\boldsymbol{\tau}_i^k$ to estimate the global gradient direction (line 7); and (ii) updating the local model parameters $\widehat{\boldsymbol{\theta}}_i^k$ by performing a gradient descent step and aggregating model parameters from neighboring nodes (line 4). We note that the model parameters for the MLP head are always initialized randomly. In contrast, the other parameters belonging to the layers to be fine-tuned can be

Algorithm 1 P2P-FT at each node i

Require: $\widehat{\boldsymbol{\theta}}_i^0 \in \mathbb{R}^q, \alpha > 0, \mathcal{D}_i, \{a_{ij}\}, \{b_{ij}\}$

1: Select the parameters of the last few layers to be fine-tuned $\widehat{\boldsymbol{\theta}}_i$ and freeze the rest of the parameters $\boldsymbol{\theta}_i'$.
2: Sample a mini-batch from $\mathcal{D}_i$ and evaluate the gradient $\boldsymbol{\tau}_i^0 := \nabla L_i(\widehat{\boldsymbol{\theta}}_i^0)$
3: **for** $k = 0, 1, 2, \cdots$ **do**
4: $\quad \widehat{\boldsymbol{\theta}}_i^{k+1} \leftarrow \sum_{j=1}^n a_{i,j}\left(\widehat{\boldsymbol{\theta}}_j^k - \alpha\boldsymbol{\tau}_j^k\right)$
5: $\quad$ Sample a mini-batch from $\mathcal{D}_i$ and evaluate $\nabla L_i(\widehat{\boldsymbol{\theta}}_i^{k+1})$
6: $\quad \mathbf{g}_i^{k+1} \leftarrow \nabla L_i(\widehat{\boldsymbol{\theta}}_i^{k+1})$
7: $\quad \boldsymbol{\tau}_i^{k+1} \leftarrow \sum_{j=1}^n b_{ij}\left(\boldsymbol{\tau}_j^k + \mathbf{g}_j^{k+1} - \mathbf{g}_j^k\right)$
8: **end for**
9: **return** $\widehat{\boldsymbol{\theta}}_i^k$ ▷ The fine-tuned parameters are $\widehat{\boldsymbol{\theta}}_i^k$

Model	Datasets	Local-FT				P2P-FT			
		Node 1	Node 2	Node 3	Node 4	Node 1	Node 2	Node 3	Node 4
ViT	**Pets**	13.80	43.10	23.88	13.13	**87.21**	**87.28**	**87.21**	**87.21**
	Flowers	13.36	15.65	44.84	25.98	**99.80**	**99.80**	**99.80**	**99.80**
	CIFAR-10	20.07	19.88	29.78	29.94	**97.44**	**97.45**	**97.44**	**97.44**
	CIFAR-100	9.86	9.81	49.28	24.07	**87.40**	**87.40**	**87.44**	**87.40**
DeiT	**Pets**	13.26	42.96	24.22	13.06	**86.40**	**86.40**	**86.40**	**86.40**
	Flowers	13.36	15.58	44.63	25.78	**92.81**	**92.98**	**92.87**	**92.84**
	CIFAR-10	19.88	19.63	29.45	29.70	**95.15**	**95.12**	**95.10**	**95.11**
	CIFAR-100	9.70	9.34	45.96	23.20	**79.05**	**79.04**	**79.14**	**79.03**
Swin	**Pets**	13.13	43.37	24.15	12.79	**86.81**	**86.74**	**86.60**	**86.67**
	Flowers	13.36	15.65	44.90	25.98	**99.83**	**99.83**	**99.83**	**99.83**
	CIFAR-10	19.98	19.72	29.72	29.88	**97.59**	**97.65**	**97.55**	**97.59**
	CIFAR-100	9.86	9.86	48.79	24.05	**87.04**	**86.99**	**87.03**	**87.02**

TABLE I
ACCURACY COMPARISON OF **ViT**, **DeiT**, AND **Swin**-TRANSFORMER MODELS AFTER FINE-TUNING FOR 100 EPOCHS.

initialized with the same values as those obtained from the pre-trained model. This typically leads to faster convergence as the model leverages the knowledge acquired by the pre-trained parameters. Since transformer models consist of billions of parameters, we train only a subset, ensuring efficient training while achieving excellent performance by effectively leveraging pre-trained knowledge (as discussed in the next section).

IV. EXPERIMENTAL EVALUATION

In this section, we consider the distributed fine-tuning of three vision transformer architectures: **ViT**, **DeiT**, and **Swin** transformer [5]–[7]. Each of these models consists of multiple attention blocks. Our approach involves freezing the model parameters for the majority of these blocks while training only a limited number of selected layers. We evaluate classification accuracy for P2P-FT and compare it with locally trained models (Local-FT) across multiple datasets. We perform all experiments using a HPC cluster with a fixed learning rate of 10^{-3} for fair comparison.

A. Network

We now describe the distributed training setup. We consider a peer-to-peer network of $n = 4$ nodes communicating over a strongly connected directed graph (for example, see Figure 1). Each node possesses a pre-trained model and a local training dataset. These nodes exchange fine-tuning model parameters and gradients computed during backpropagation but are prohibited from sharing their private datasets.

B. Datasets

We fine-tune vision transformer models and evaluate their classification accuracy using four datasets: (i) Oxford-Pets, (ii) Oxford-Flowers, (iii) CIFAR-10, and (iv) CIFAR-100. Each dataset consists of colored images categorized into multiple classes. The Oxford-Pets dataset includes 37 categories of dogs and cats, with approximately 200 images per class. The Oxford-Flowers dataset comprises 102 flower categories, with class sizes ranging from 40 to 258 images. The CIFAR-10 dataset consists of 60,000 images across 10 categories, while CIFAR-100 contains 100 classes, each with 600 images. All datasets are divided into training and testing sets, with fine-tuning performed on the training set and accuracy evaluated on the test set.

Datasets	Node 1	Node 2	Node 3	Node 4
Pets	0-4	5-19	20-29	30-36
Flowers	0-19	20-39	40-79	80-101
CIFAR-10	0-1	2-3	4-6	7-10
CIFAR-100	0-9	10-19	20-74	75-99

TABLE II
DISTRIBUTION OF CLASSES ACROSS DIFFERENT NODES.

Due to space limitations, we focus on the most challenging scenario where, in the distributed fine-tuning setup across a peer-to-peer network, nodes are assigned non-overlapping classes, i.e., Node 1 is never fine-tuned on the classes possessed by any other node. The class distributions across nodes are explicitly described in Table II. We note that this data heterogeneity cannot be tackled without using gradient-tracking as suggested by the theoretical results found in [17]. Therefore, in our experimental evaluations, we refrain from comparing the results of P2P-FT with such methods.

C. Transformer Architectures and Performance Results

We now describe the architectures of **ViT**, **DeiT**, and **Swin** transformer models used in our distributed fine-tuning setup. The **ViT** model, implemented using **timm/vit_small_patch16_224**, utilizes a convolutional layer to generate patch embeddings, which are then passed through a sequence of 12 blocks comprising normalization, attention, and MLP layers. The **DeiT** model, based on **timm/deit_small_patch16_224**, follows a similar architecture but incorporates data-efficient training mechanisms. In contrast, the **Swin** transformer, implemented using **timm/swin_small_patch4_window7_224.ms_in22k**, consists of four stages, each containing several blocks, and performs self-attention within local windows. For all models, we modify the classification head to match the number of classes in the global dataset and freeze all blocks except the last one (or the last block of the fourth stage in **Swin** transformer). The unfrozen weights are then updated using the method outlined in Algorithm 1. Each node possesses a non-overlapping private fine-tuning dataset consisting of images from distinct classes, while the testing dataset includes samples from all classes.

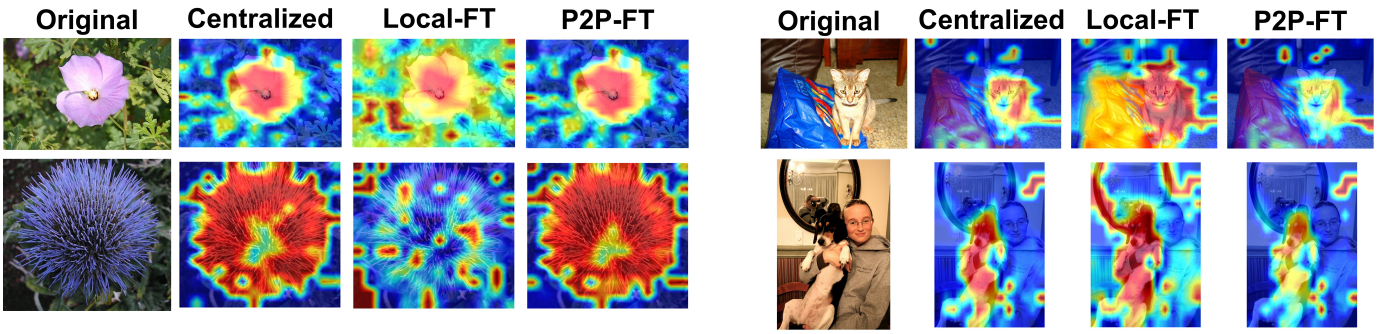

Fig. 2. Visualizing attention maps for test samples belonging to unseen classes.

Table I highlights the accuracy results after fine-tuning all models for 100 epochs. It can be observed that fine-tuning `Local-FT` on local data results in poor accuracy, as the nodes focus on learning features of their private datasets. However, `P2P-FT` uses weight-mixing and gradient-sharing strategies to achieve significantly higher accuracy, similar to centralized training setups.

Figure 2 presents attention maps overlaid on test images from unseen classes, highlighting the differences between `Local-FT` and `P2P-FT`. It can be observed that `Local-FT` struggles to focus accurately on target objects, often misplacing attention or distributing it across multiple regions. In contrast, `P2P-FT` generates attention maps that closely resemble those obtained from centralized fine-tuning, even though the corresponding classes are absent from the local fine-tuning dataset. This is because `P2P-FT` enables each node to learn feature representations for unseen images through weight-mixing and gradient-tracking strategies.

V. Conclusion

Training large transformer models is not feasible in many applications. Fine-tuning the pre-trained models usually guarantees the best performance. In numerous practical scenarios, heterogeneous data is distributed across a network of nodes, and accumulating all data at a central location is not possible. When the nodes fine-tune their models using only their local datasets, they struggle to generalize effectively due to the bias caused by heterogeneous data distribution. The small size of local datasets and the incomplete representation of all classes impact the performance of vision transformers. We propose a privacy-aware framework for the distributed fine-tuning of vision transformer models. The proposed method `P2P-FT` uses weight-mixing and gradient-sharing strategies to eliminate bias and learn global attention weights, leading to remarkable performance even for locally unseen classes. We illustrate the performance of `P2P-FT` for fine-tuning distributed **`ViT`**, **`DeiT`**, and **`Swin`** transformer models.

References

[1] K. O'Shea and R. Nash, "An introduction to convolutional neural networks.," *CoRR*, vol. abs/1511.08458, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[6] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers amp; distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357, PMLR, 18–24 Jul 2021.

[7] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.

[8] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 87–110, 2023.

[9] R. Bommasani and et al., "On the opportunities and risks of foundation models," *CoRR*, vol. abs/2108.07258, 2021.

[10] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Communications Surveys amp; Tutorials*, 2023.

[11] L. Yuan, L. Sun, P. S. Yu, and Z. Wang, "Decentralized federated learning: A survey and perspective," 2023.

[12] S. Kar, J. M. F. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," *IEEE Transactions on Information Theory*, vol. 58, no. 6, pp. 3575–3605, 2012.

[13] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[14] A. Nedić and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3936–3947, 2016.

[15] M. Assran, N. Loizou, N. Ballas, and M. G. Rabbat, "Stochastic gradient-push for distributed deep learning," in *36th International Conference on Machine Learning*, vol. 97, pp. 344–353, Jun. 2019.

[16] R. Xin, S. Pu, A. Nedić, and U. A. Khan, "A general framework for decentralized optimization with first-order methods," *Proceedings of the IEEE*, vol. 108, pp. 1869–1889, 2020.

[17] R. Xin, U. A. Khan, and S. Kar, "An improved convergence analysis for decentralized online stochastic non-convex optimization," *IEEE Transactions on Signal Processing*, Feb. 2021.