Deep Neural Network Fusion via Graph Matching with Applications to Model Ensemble and Federated Learning

Chang Liu [1] Chenfei Lou [1] Runzhong Wang [1] Alan Yuhan Xi [2] Li Shen [3] Junchi Yan [1 4]

Abstract

Model fusion without accessing training data in machine learning has attracted increasing interest due to the practical resource-saving and data privacy issues. During the training process, the neural weights of each model can be randomly permuted, and we have to align the channels of each layer before fusing them. Regrading the channels as nodes and weights as edges, aligning the channels to maximize weight similarity is a challenging NP-hard assignment problem. Due to its quadratic assignment nature, we formulate the model fusion problem as a graph matching task, considering the second-order similarity of model weights instead of previous work merely formulating model fusion as a linear assignment problem. For the rising problem scale and multi-model consistency issues, we propose an efficient graduated assignment-based model fusion method, dubbed GAMF, which iteratively updates the matchings in a consistency-maintaining manner. We apply GAMF to tackle the compact model ensemble task and federated learning task on MNIST, CIFAR-10, CIFAR-100, and Tiny-Imagenet. The performance shows the efficacy of our GAMF compared to state-of-the-art baselines.

1. Introduction

If we have two or more independently trained neural networks, how should we utilize them with the best accuracy? (Utans, 1996) propose the model fusion problem, which aims at fusing several neural networks into a single network without accessing the training data. Compared to the traditional prediction-based model ensemble, the advantage of fusing multiple networks into one is to save memory and inference time, as the prediction ensemble needs to maintain all individual models. Also, it can be applied to privacy-intensive federated learning (FL), where how to efficiently aggregate all locally-trained models remains open.

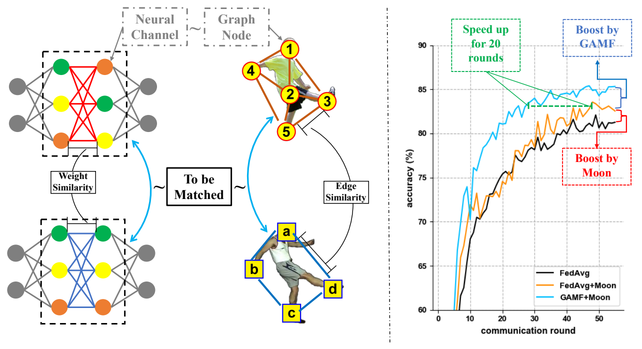


Figure 1: **Left**: Connection between model fusion and graph matching; **Right**: For federated learning, the performance boost and convergence speed up of GAMF on CIFAR-10.

A vanilla strategy is to simply average the weights, assuming that the channels of different networks do not need any alignment. However, due to the randomness and orthogonal invariance of deep neural networks, the channels from different networks are always randomly permuted. The previous study (Singh & Jaggi, 2020) has demonstrated the harmfulness if the channels are not aligned in model fusion because the effective components of the network will be interfered with and canceled by each other.

Recent works (Singh & Jaggi, 2020; Wang et al., 2020a) raise the awareness of the importance of alignment in model fusion. However, they simplify it into a linear assignment problem, which ignore the rich edge-wise information between channels. Graph matching (GM) (Yan et al., 2020; Loiola et al., 2007), which aims at matching nodes to nodes among graphs exploiting the structural information in graphs, appears to be the natural tool for model fusion since the network channels can be regarded as nodes and the weights connecting channels as edges (see Fig. 1).

One step further, multi-model fusion is a more challenging yet frequently encountered scenario where more than two networks needing to be jointly aligned. Existing solution for multiple models are relatively heuristic, e.g. OTFu-

[1]Department of Computer Science and Engineering, and MoE Key Lab of AI, Shanghai Jiao Tong University [2]University of Wisconsin Madison [3]JD Explore Academy [4]Shanghai AI Laboratory. Correspondence to: Junchi Yan <yanjunchi@sjtu.edu.cn>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

sion (Singh & Jaggi, 2020) simply merges all models sequentially, FedMA (Wang et al., 2020a) sequentially selects an anchor at each aggregation step, FedSpa (Huang et al., 2022b) and DisPFL (Dai et al., 2022) fuse the multiple local models in a low-dimensional subspace by extracting the submodels with sparse training (Liu et al., 2021b; 2022), and Wang et al. (2022) adopt distributional robust optimization approach to fuse multiple models. In contrast, multi-graph matching (MGM) algorithms (Yan et al., 2016a; Jiang et al., 2021; Wang et al., 2020b; Leonardos et al., 2017) are developed in the sense of *cycle consistency*, which ensure that the matching of two graphs should not be violated by the matchings involving any third graph. Hence, MGM algorithms can ensure a global alignment of multiple models.

Despite the appealing properties discussed above, existing graph matching methods (Gold & Rangarajan, 1996; Cho et al., 2010; Jiang et al., 2021; Zhou & Torre, 2016; Wang et al., 2020b), including the state-of-the-art commercial solver GUROBI (Optimization, 2020), can not be readily applied in neural network model fusion, due to the memory burden introduced by the $O((d_\Sigma)^4)$-sized affinity tensor, especially for fusing multiple models. Here d_Σ is the sum of channels of neural networks, which can be up to several thousand for modern neural networks. Fortunately, we will show that the affinity tensor contains certain sparse patterns, which leaves the space for more cost-effective exploitation via our graduated assignment technique inspired from the classic work (Gold & Rangarajan, 1996).

In this paper, we resort to a graph matching formulation for network model fusion method: **G**raduated **A**ssignment **M**odel **F**usion (**GAMF**). For scalability, we develop a new graduated assignment algorithm under a memory-efficient slice-and-scan procedure, enabling the feasibility of fusing deep neural networks via graph matching. For multi-model fusion, we propose a multi-model version of GAMF by developing a cycle-consistent MGM algorithm inspired by (Wang & et al, 2020; Solé-Ribalta & Serratosa, 2013).

GAMF is validated under two important applications: compact model ensemble and federated learning. In the first case, we follow the settings in OTFusion and GAMF outperforms OTFusion on both MNIST (LeCun, 1998) and CIFAR-10 (Krizhevsky et al., 2009). In federated learning, we conduct the experiments on the popular open-source framework FedML (He et al., 2020b). Compared with state-of-the-art FL algorithms, GAMF converges faster on CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), and Tiny-Imagenet[1].

The contributions of this paper are four-fold:

1) Unlike previous work using linear assignment problem (LAP) for the emerging paradigm of model fusion without accessing training data, we manage to solve it by graph matching with edge information. To our best knowledge, we are the first to solve model fusion beyond LAP.

2) We propose a scalable graduated assignment method named GAMF under the GM framework, with the design of an iterative memory-efficient slice-and-scan procedure, which is based on our key observation of the intrinsic sparsity characteristic of the problem because the channel matching shall be restricted in the same layer.

3) We consider the cycle consistency property in multi-model fusion to improve the proposed GAMF, which is not studied in model fusion literature before.

4) Experiments show the effectiveness of GAMF in both compact model ensemble and federated learning. Our source code is available at: `https://github.com/Thinklab-SJTU/GAMF`.

[1] https://www.kaggle.com/c/tiny-imageNet

2. Related Works

Model Fusion in this paper, specifically refers to merging two pretrained networks into a single network without accessing to training data, which can be more effective than traditional model ensembles (Breiman, 1996; Wolpert, 1992; Schapire, 1999) that average the predictions of each network. (Leontev et al., 2020) formulate model fusion as linear assignment and solve it approximately. However, the assumptions in these methods are very strict, e.g., the original models have to share a part of the training history (Smith & Gashler, 2017; Utans, 1996) or rely on SGD for periodic averaging (Malinovskiy et al., 2020). Regrettably, they can not ensure overhead vanilla averaging (Leontev et al., 2020). Note that there are a large number of works concerning the fusion of model computing and results during the inference stage from multi-model (Du et al., 2017), while the fusion in our paper refers to the fusion of trained model parameters.

The most related work to ours is OTFusion (Singh & Jaggi, 2020), which notices the weights alignment nature inside model fusion. It formulates model fusion as a linear assignment problem and solves it via Wasserstein barycenters, which is the first model fusion work for improving vanilla averaging, however, it degenerates the problem by ignoring the second-order similarity of weights. In contrast to OTFusion, GAMF solves a quadratic model fusion problem.

Federated Learning is a paradigm that allows local clients to collaboratively train a shared global model. In its standard pipeline, each local client train the local model with their own datasets, and the global server gathers all local models and merge them into a shared global model. Recently researchers (Kairouz et al., 2019; Wang et al., 2021a) have given insights into a wide range of methods for FL performance improvements (Fraboni et al., 2021; Chen et al., 2020; He et al., 2020a; Wu & Gong, 2020; Dinh et al., 2020;

Deng et al., 2020; Peterson et al., 2019; Liu et al., 2021a; Acar et al., 2020; He et al., 2021a;b; Huang et al., 2022a; Yu et al., 2021). The efforts on improving the performance of federated learning can be summarized into two categories: (i) improving the local optimizer, e.g. FedAvg (McMahan et al., 2017), FedProx (Li et al., 2018), SCAFFOLD (Karimireddy et al., 2020), and Moon (Li et al., 2021) (ii) boosting the model aggregation in server, e.g., FedFTG (Zhang et al., 2022), FedMA (Wang et al., 2020a) and its predecessor PFNM (Yurochkin et al., 2019). Moon is the state-of-the-art federated learning algorithm that adds a contrastive loss in the training epochs of local clients. We emphasize that all these local training type methods are orthogonal to our GAMF, which can serve as a strong plugin to enhance their performance. On the other hand, FedMA uses a similar assignment formulation as OTFusion and proposes an iterative method, which is the most relative federated learning algorithm to ours. Our GAMF falls in line with FedMA in that we all focus on the alignment of different local models.

Graph Matching aims to find node correspondence by considering both node features and edge attributes, which is known NP-hard in its general form (Loiola et al., 2007). By regarding neural channel and parameter weights as node and edge attributes, we can formulate the model fusion problem to a graph matching task that aims to find the best correspond of model parameters. Classic methods mainly resort to different optimization heuristics ranging from random walk (Cho et al., 2010), spectral matching (Leordeanu & Hebert, 2005), path-following algorithm (Zhou & Torre, 2016), graduated assignment (Gold & Rangarajan, 1996), to SDP relaxation (Schellewald & Schnörr, 2005) etc. In recent years, deep graph matching has become an emerging paradigm (Wang et al., 2019; 2021b; Yu et al., 2020; Rolínek et al., 2020; Liu et al., 2020). Readers are referred to the surveys for a detailed review of traditional graph matching (Yan et al., 2016b) and deep GM (Yan et al., 2020).

However, standard graph matching often deals with general images with dozens of key points (Yan et al., 2020), while the scalability of model fusion is at least thousands of neural channels in modern networks. In our attempt, even the commercial solvers can not handle such a large scale efficiently, e.g. GUROBI (Optimization, 2020).

3. Formulation and Methodology

We first formulate the model fusion problem as graph matching, then we devise our graduated assignment approach to tackle fusing two and multiple models .

3.1. Graph Matching Formulation of Model Fusion

During network training, due to the randomness in stochastic gradient descent and the difference of training sets for different models, the permutation of the channels may be shuffled among different models, which calls for channel alignment for model fusion as studied in (Smith & Gashler, 2017; Leontev et al., 2020; Malinovskiy et al., 2020) with early work dating back to (Utans, 1996). In this paper, we show that graph matching is a natural formulation for model fusion. For simplicity, we discuss the GM formulation regarding with fusing two fully-connected networks with two hidden layers without bias[2] (see Fig. 2a):

$$\mathbf{x}_1 = \delta(\mathbf{x}_0\mathbf{W}_1); \mathbf{x}_2 = \delta(\mathbf{x}_1\mathbf{W}_2); \mathbf{x}_3 = \delta(\mathbf{x}_2\mathbf{W}_3), \quad (1)$$

where $\mathbf{x}_0$ is the input, $\mathbf{x}_3$ is the output, each contains n data points. $\mathbf{x}_i \in \mathbb{R}^{n\times d_i}$ represents the data after i-th layer with dimension d_i and $\mathbf{W}_i \in \mathbb{R}^{d_{i-1}\times d_i}$ is the weight matrix of the neural network. δ represents the activation (e.g. ReLU).

In the following, we distinguish different models by superscript with brackets. Model fusion is to find a reasonable permutation (i.e. shuffle) of the channels, which is equivalent to permuting $\mathbf{W}_1^{(1)}, \mathbf{W}_2^{(1)}, \mathbf{W}_3^{(1)}$ to fit $\mathbf{W}_1^{(2)}, \mathbf{W}_2^{(2)}, \mathbf{W}_3^{(2)}$ and can be formulated and handled by graph matching. When fusing two fully-connected networks with 2 hidden layers, we have the following graph matching formulation (the other elements of $\mathbf{P}$ are 0 since cross-layer matchings are meaningless, therefore the structure of $\mathbf{P}$ is very sparse):

$$\max_{\mathbf{P}} \sum_{i=0}^{d_\Sigma-1}\sum_{j=0}^{d_\Sigma-1}\sum_{a=0}^{d_\Sigma-1}\sum_{b=0}^{d_\Sigma-1} \mathbf{P}_{[i,j]}\mathbf{K}_{[i,j,a,b]}\mathbf{P}_{[a,b]} \quad (2)$$

$$s.t.\ \mathbf{P}_0 = \mathbf{I}; \mathbf{P}_3 = \mathbf{I}; \forall j \sum_{i=0}^{d_1-1}\mathbf{P}_{1[i,j]} = 1, \forall i \sum_{j=0}^{d_1-1}\mathbf{P}_{1[i,j]} = 1;$$

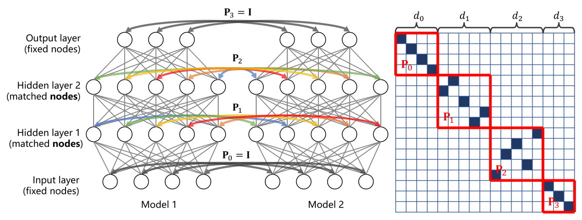
$$\forall j \sum_{i=0}^{d_2-1}\mathbf{P}_{2[i,j]} = 1, \forall i \sum_{j=0}^{d_2-1}\mathbf{P}_{2[i,j]} = 1.$$

where we denote the indices start from 0, $d_\Sigma = d_0+d_1+d_2+d_3$, and the definition of $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ are from Fig. 2b. Note that for Eq. (2): $\mathbf{P}$ encodes the permutations of two hidden layers, and the channels of input/output layers need not to be permuted (see Fig. 2a). The constraints ensure one-to-one mapping between the channels within the same layer. In graph matching formulation, $\mathbf{K} \in \mathbb{R}^{d_\Sigma\times d_\Sigma\times d_\Sigma\times d_\Sigma}$ is a 4-dimensional affinity tensor whose element $\mathbf{K}_{[i,j,a,b]}$ measures the affinity between the edges (i,a) and (j,b), which is the similarity between the elements of weight matrices in the model fusion problem. We adopt the Gaussian kernel as the similarity measure which is widely applied by GM methods (Yan et al., 2016a; Cho et al., 2010):

$$\mathbf{K}_{[i,j,a,b]} = \exp\left(-\frac{||\mathbf{e}_{i,a} - \mathbf{e}_{j,b}||_F^2}{\sigma}\right), \quad (3)$$

where $\mathbf{e}_{i,a}$ denotes the network's weight corresponding to edge (i,a), and σ is a configurable hyperparameter.

[2]Our formulation can naturally generalize to networks with more layers or bias, and convolutional neural networks.


P₃ = I
Output layer (fixed nodes)
P₂
Hidden layer 2 (matched nodes)
P₁
Hidden layer 1 (matched nodes)
P₀ = I
Input layer (fixed nodes)
Model 1
Model 2
d₀
d₁
d₂
d₃
P₀
P₁
P₂
P₃

(a) Model fusion as graph matching. (b) $\mathbf{P}$'s structure.

Figure 2: **Left:** Illustration of the graph matching formulation of fusing two networks with 2 hidden layers. The channels of the hidden layers are regarded as nodes to be matched and the permutations of input/output layers are fixed. The weights connecting different channels of different layers are regarded as edges. **Right:** The permutation (sparse) matrix $\mathbf{P}$ can be decomposed as $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$.

Though the formulation of model fusion illustrated in Fig. 2a is the basic fully connected layers, it has been proved in previous work (Singh & Jaggi, 2020; Wang et al., 2020a) that several commonly used layers including convolutional layers can also be fused in such formulation.

3.2. Intrinsic Sparsity of the Model Fusion Task

The problem of Eq. (2) is known (Yan et al., 2020) NP-hard and a special case of Lawler's Quadratic Assignment Problem (Lawler, 1963) whose memory cost can be $O((d_\Sigma)^4)$ in general cases, where d_Σ is the sum of channels of all layers. For deep networks with thousands of channels considered in model fusion, it seems intractable, and in fact we empirically find that the commercial solver GUROBI (Optimization, 2020) cannot handle such a memory burden.

Fortunately, it is worth noting that though the scales of the matrix $\mathbf{K}$ and $\mathbf{P}$ are very large, the components of the two matrices are relatively sparse. As shown in Fig. 2b, it is important to realize for network model fusion, components other than $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ are 0 since the cross-layer matchings are meaningless. Suppose the number of layers is l, the optimal upper bound of cost is $O(\Sigma(\frac{d_\Sigma}{l})^4) = O(\frac{1}{l^3} * (d_\Sigma)^4)$, which is at one-thousandth of the original $O((d_\Sigma)^4)$ cost for a 10-layer network. Therefore, we aim to explore the sparse structure for efficiently solving the problem.

3.3. Graduated Assignment for Model Fusion

Based on the above analysis to the inherent sparsity of the problem, we resort to the classic idea of graduated assignment (Gold & Rangarajan, 1996), which repeatedly optimizes the first-order Taylor series of Eq. (2) and then maps it to the (relaxed) feasible space with gradually tightened constraints. Our proposed algorithm, namely **G**raduated **A**ssignment for **M**odel **F**usion (**GAMF**), exploits the underlying sparse structure of $\mathbf{K}$ and $\mathbf{P}$, thus resolves the computational and memory burden, and can scale up to networks like VGG11 in experiments.

Graduated Assignment Model Fusion. Given a feasible permutation matrix $\mathbf{P}^0$, we denote the objective in Eq. (2) as J. Inspired by (Gold & Rangarajan, 1996), it can be rewritten by its Taylor series:

$$\begin{aligned} J = &\sum_{i=0}^{d_\Sigma-1}\sum_{j=0}^{d_\Sigma-1}\sum_{a=0}^{d_\Sigma-1}\sum_{b=0}^{d_\Sigma-1} \mathbf{P}^0_{[i,j]}\mathbf{K}_{[i,j,a,b]}\mathbf{P}^0_{[a,b]} + \\ &\sum_{i=0}^{d_\Sigma-1}\sum_{j=0}^{d_\Sigma-1} \mathbf{R}_{[i,j]}(\mathbf{P}_{[i,j]} - \mathbf{P}^0_{[i,j]}) + ... \end{aligned} \tag{4}$$

and $\mathbf{R}_{[i,j]} = \frac{\partial J}{\partial \mathbf{P}}\big|_{\mathbf{P}=\mathbf{P}^0} = \sum_{a=0}^{d_\Sigma-1}\sum_{b=0}^{d_\Sigma-1}\mathbf{K}_{[i,j,a,b]}\mathbf{P}^0_{[a,b]}$. By only considering the zero and first order Taylor series of Eq. (2), it equals to maximizing the following objective because all the other elements are constants:

$$\max_{\mathbf{P}} \sum_{i=0}^{d_\Sigma-1}\sum_{j=0}^{d_\Sigma-1} \mathbf{R}_{[i,j]}\mathbf{P}_{[i,j]} \ s.t. \text{ constraints in Eq. (2).} \tag{5}$$

The above constrained optimization problem is a linear assignment problem, which can be solved to optimal by projecting the real-valued square matrix $\mathbf{R}$ to a 0/1 permutation matrix by Hungarian algorithm (Kuhn, 1955) in polynomial time. Moreover, a relaxed projection can be achieved by Sinkhorn algorithm, which firstly normalizes the regularization factor τ: $\mathbf{S} = \exp(\mathbf{R}/\tau)$, and then performs row- and column-wise normalization alternatively:

$$\mathbf{D}_r = \text{diag}(\mathbf{S}\mathbf{1}), \mathbf{S} = \mathbf{D}_r^{-1}\mathbf{S}; \mathbf{D}_c = \text{diag}(\mathbf{S}^\top\mathbf{1}), \mathbf{S} = \mathbf{S}\mathbf{D}_c^{-1},$$

where $\text{diag}(\cdot)$ means building a diagonal matrix from a vector, $\mathbf{1}$ is a column vector whose elements are all 1. When the above algorithm converges, $\mathbf{S}$ is a doubly-stochastic matrix whose elements are continuous and all of its rows and columns sum to 1, which is a relaxed version of the permutation matrix. τ controls the gap between the Sinkhorn algorithm and the Hungarian algorithm: given a smaller τ, the result of the Sinkhorn algorithm becomes closer to Hungarian algorithm, at the cost that it takes more iterations to converge. The graduated assignment algorithm works by firstly randomly initializing $\mathbf{P}^0$, computing $\mathbf{R}$, and then computing $\mathbf{S}$ by Sinkhorn algorithm, and repeating these steps. τ of the Sinkhorn algorithm gradually shrinks at each step. This framework is developed and adopted by (Gold & Rangarajan, 1996; Wang & et al, 2020) to efficiently tackle hard combinatorial problems like graph matching.

Efficient Adaption to Model Fusion. By analyzing $\mathbf{K}$, $\mathbf{P}$ in Fig. 2, these two matrices contain sparse structures because only the nodes and edges from the same layer are effective when computing the objective score in Eq. (2). To save memory, we introduce the memory-efficient graduated assignment algorithm for model fusion as follows. For the example in Fig. 2, the weights of two neural networks

Algorithm 1: Graduated Assignment for Model Fusion (of Two Neural Networks)

```
Input: weights $\{\mathbf{W}_i^{(1)}\}, \{\mathbf{W}_i^{(2)}\}$; initial annealing $\tau_0$;
       descent factor $\gamma$; minimum $\tau_{min}$; Gaussian kernel $\sigma$.
Randomly initialize $\{\mathbf{P}_i\}$; projector $\leftarrow$ Sinkhorn; $\tau \leftarrow \tau_0$;
while True do
    while $\{\mathbf{P}_i\}$ not converged do
        $\forall i = 1, 2, ... :$
        $\mathbf{R}_{i[a,b]} =$
          $\sum_j \exp\left(-\frac{\left|(\mathbf{P}_{i-1}^\top \mathbf{W}_i^{(1)})_{[j,a]} - \mathbf{W}_{i[j,b]}^{(2)}\right|^2}{\sigma}\right) +$
          $\sum_j \exp\left(-\frac{\left|(\mathbf{W}_{i+1}^{(1)} \mathbf{P}_{i+1})_{[a,j]} - \mathbf{W}_{i+1[b,j]}^{(2)}\right|^2}{\sigma}\right)$;
        $\mathbf{P}_i = \text{projector}(\mathbf{R}_i, \tau)$;
    # graduated assignment control
    if projector == Sinkhorn AND $\tau \geq \tau_{min}$ then
        $\tau \leftarrow \tau \times \gamma$;
    else if projector == Sinkhorn AND $\tau < \tau_{min}$ then
        projector $\leftarrow$ Hungarian;
    else
        break;
Output: The set of permutation matrices $\{\mathbf{P}_i\}$.
```

are denoted as $\mathbf{W}_1^{(1)}, \mathbf{W}_2^{(1)}, \mathbf{W}_3^{(1)}$ and $\mathbf{W}_1^{(2)}, \mathbf{W}_2^{(2)}, \mathbf{W}_3^{(2)}$, respectively. $\mathbf{R}_i$ at layer i can be obtained via fusing the information from layer $i-1$ and $i+1$. It can be viewed as replacing the original computation of $\mathbf{R}$ by a slide-and-scan procedure which is more memory efficient. Our GAMF for two models is summarized in Alg. 1. In theory, the space complexity of GAMF is only $O(c * l * d_{max}^2)$ since we use an iterative slice-and-scan procedure, where c is the client amount, l is the number of layers, and d_{max} is the maximum number of channels across all the layers. It makes GAMF a memory-efficient method that has no risk of exceeding memory or accidental crashing.

Fusing Multiple Models by Multi-Graph Matching. The multi-graph matching problem (Yan et al., 2016a; Jiang et al., 2021; Shi et al., 2016) is a natural extension from the classic two-graph matching problem where more than two graphs need to be jointly handled, by enforcing the multi-graph regularization namely *cycle consistency*. The cycle consistency property ensures that the permutation results do not violate any two pairs of graphs, which is appealing to the model fusion problem. In this paper, we extend Alg. 1 with GAMGM (Wang & et al, 2020) for fusing multiple models: $\mathbf{P}_i$ in Alg. 1 is replaced by $\mathbf{U}_i^{(k)}$, where $\mathbf{U}_i^{(1)}\mathbf{U}_i^{(2)\top}$ denotes the permutation of layer i for model 1 and model 2. The information from multiple graphs is also fused in the same manner under the special condition of two graphs. The model fusion algorithm for multiple models is in Alg. 2. Additionally, the weights of different models are set equally in multiple model fusion, since we assume that the importance of each model is the same.

Algorithm 2: Graduated Assignment for Model Fusion (of Multiple Neural Networks)

```
Input: weight matrices $\{\mathbf{W}_i^{(k)}\}$; initial annealing $\tau_0$; descent
       factor $\gamma$; minimum $\tau_{min}$; Gaussian kernel $\sigma$.
Random initial $\{\mathbf{U}_i^{(k)}\}$; projector $\leftarrow$ Sinkhorn; $\tau \leftarrow \tau_0$;
while True do
    while $\{\mathbf{U}_i^{(k)}\}$ not converged do
        $\forall i = 1, 2, ...; \forall k = 1, 2, ... :$
        $\mathbf{R}_{i[a,b]}^{(k)} =$
          $\sum_{k' \neq k} \Bigg[ \sum_j \exp\left(-\frac{\left|(\mathbf{U}_{i-1}^{(k')\top} \mathbf{W}_i^{(k')})_{[j,a]} - (\mathbf{U}_{i-1}^{(k)\top} \mathbf{W}_i^{(k)})_{[j,b]}\right|^2}{\sigma}\right) +$
          $\sum_j \exp\left(-\frac{\left|(\mathbf{U}_{i+1}^{(k')\top} \mathbf{W}_{i+1}^{(k')})_{[a,j]} - (\mathbf{U}_{i+1}^{(k)\top} \mathbf{W}_{i+1}^{(k)})_{[b,j]}\right|^2}{\sigma}\right) \Bigg]$;
        $\mathbf{U}_i^{(k)} = \text{projector}(\mathbf{R}_i^{(k)}, \tau)$;
    # graduated assignment control
    if projector == Sinkhorn AND $\tau \geq \tau_{min}$ then
        $\tau \leftarrow \tau \times \gamma$;
    else if projector == Sinkhorn AND $\tau < \tau_{min}$ then
        projector $\leftarrow$ Hungarian;
    else
        break;
Output: The set of permutation matrices $\{\mathbf{U}_i^{(k)}\}$.
```

Connections and Differences with OTFusion. Comparing our method with OTFusion (Singh & Jaggi, 2020), a similar step is also encountered in our pipeline when computing the first-order Taylor series of the objective score. However, the major drawback of OTFusion is that the second-order information is underutilized, and OTFusion can be viewed as a special and simple case of our algorithm by taking a single iteration. In contrast, we are capable of handling the second-order information in graph matching by optimizing the quadratic objective score in Eq. (2) by a multi-step pipeline. Meanwhile, we introduce the multi-graph matching algorithms that can be readily integrated into the common scenario of fusing multiple neural networks. Moreover, we successfully apply our model fusion method GAMF in federated learning, which is not considered in (Singh & Jaggi, 2020). More importantly, we discover that OTFusion fails in the federated learning experiments in Section 4.3.

Overall, our contributions beyond existing work are three folds: First, we non-trivially generalize the model fusion problem from 1st-order to 2nd-order, formulate it as a GM problem, and achieve performance gain. Second, we adapt GAMF to the emerging FL field and show GAMF can also handle FL, but OTFusion cannot work well. Third, our work extends graph matching to model fusion area beyond classic image matching, which is a different problem, and especially, the problem scale raises from tens of key points to thousands of points. To solve the scalability issue, we design GAMF with a novel three-layer slicing-window scheme.

4. Experiments

4.1. Protocols

Datasets We strictly follow the model fusion (Singh & Jaggi, 2020; Li et al., 2021; Wang et al., 2020a) literature, by using the same image classification datasets: MNIST, CIFAR-10, CIFAR-100, and Tiny-Imagenet. We adopt the data (image) augmentation settings in OTFusion. For the detailed dataset and augmentation settings, refer to the open-source repository of OTFusion[4].

We consider two data partition settings for different local models/clients: 1) Homogeneous data partition denotes that each client acquires data from an IID, and ends up including a nearly equal proportion of the data from each class. 2) Heterogeneous data partition denotes each client acquires data from a non-IID. Following the settings in (Wang et al., 2020a; Li et al., 2021), we sample $p_k \sim Dir_N(\beta)$ and allocate a $p_{k,j}$ proportion of the instances of class k to client j, where $Dir(\beta)$ denotes the Dirichlet distribution with a concentration parameter $\beta = 0.5$ by default.

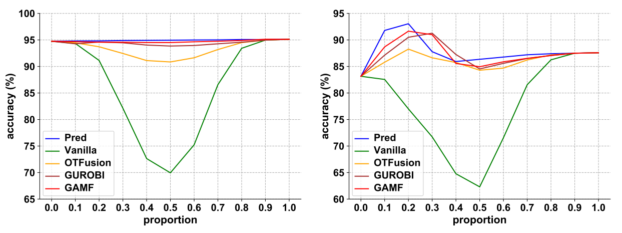
Backbone Models and Training Settings We use three different network architectures. For MNIST, we choose a small convolution neural network (denoted by Convnet[3]); We use the same hyperparameters but remove the dropout layer. In addition, we try the classic LeNet (LeCun et al., 1998). For CIFAR-10/100 and Tiny-ImageNet, we adopt the popular VGG11 (Simonyan & Zisserman, 2014), with the same modifications as in OTFusion (Singh & Jaggi, 2020).

1) Compact ensemble. The ensemble experiments basically follow the protocol of the open-source OTFusion. For the Convnet in MNIST, we use the Adam optimizer with the learning rate of 0.0005 and train the network for 20 epochs with the batch size of 64. For the LeNet and VGG11 in the CIFAR-10, we set the learning rate as 0.001, batch size as 256, and train for 100 epochs. For the finetune process, we set 100 epochs for the homogeneous experiments and 20 epochs for the heterogeneous experiments.

2) Federated learning. In the federated learning experiments, we use the open-source federated learning framework FedML (He et al., 2020b) , which is a research-oriented federated learning library and benchmark for a fair comparison. By default, we set the learning rate as 0.0005, the batch size as 256, the local epochs as 10, and the communication rounds as 55. We test the performance of the federated learning algorithms with 5 and 10 local clients, respectively.

Implementation Details We use Pytorch (Paszke et al., 2019) to implement our proposed GAMF, and the hyperparameters are: initialization $\tau = 0.05$, max iteration rounds 200, $\gamma = 0.9$, and $\tau_{min} = 0.005$ for the Sinkhorn algorithm.

[3]https://keras.io/examples/vision/mnist_convnet/



(a) Homogeneous (b) Heterogeneous

Figure 3: Comparison of ensemble methods in MNIST. "pred" denotes the direct prediction ensemble. Horizontal axis denotes the proportion of the second model in fusion.

We empirically set $\sigma = 2$. We stop the GAMF algorithm once the 2-norm of the total change of permutation matrix is sufficiently small, i.e., $\sum_{\forall k} \sum_{\forall i} \left|\left|\mathbf{U}_i^{(k)}\right|\right|_F < 0.002$.

We conduct our experiments on the open-source GitHub repositories OTFusion[4] and FedML[5], which leads to a fair comparison. Experiments run on AMD Ryzen Threadripper 3970X 32-Core Processor and 3 GTX 3090 GPUs. Our experiments can be easily reproduced based on the open-source code by providing the configurations.

Compared Methods **FedAvg (Vanilla)** (McMahan et al., 2017). This method direct averages parameters of the local models, which is a simple but efficient algorithm. It is called Vanilla in the compact model ensemble experiments.

OTFusion (Singh & Jaggi, 2020). As aforementioned, it formulates the model fusion problem as a linear assignment problem and utilizes the Wasserstein barycenter to solve it.

FedMA (Wang et al., 2020a). This method uses a similar problem formulation as OTFusion, then it matches and averages weights of local models in a layer-wise manner. In the implementation of FedMA, we remove some tricks for fairness, including that FedMA requires the data distribution information over classes and needs triple local epochs for the last layer training. Moreover, FedMA requires the number of communication rounds is integer multiplied by the number of layers in the neural networks (11 for VGG11). Therefore, we run 55 communication rounds in experiments.

Moon (Li et al., 2021). It is the state-of-the-art FL method which adapts the contrastive learning framework in the local training process. Since Moon focuses on the local training while GAMF focuses on the channel alignment, GAMF and Moon can be complementary to each other.

GUROBI (Optimization, 2020). As we formulate the model fusion as a graph matching problem, we also try to solve the problem by the popular commercial solver GUROBI in compact ensemble experiments.

[4]https://github.com/sidak/OTFusion
[5]https://github.com/FedML-AI/FedML

Table 1: Results with the effect of finetuning the fused LeNet models on CIFAR-10. "individual models": performance of each individual model; "Pred": prediction ensemble with N times cost than others to maintain all individual models.

	Data Partition	# $(=N)$ of Models	Individual Models [Acc(%) of each model, ...]	Pred ($N\times$ cost)	Vanilla ($1\times$ cost)	OTFusion ($1\times$ cost)	GAMF ($1\times$ cost)
One-shot	Homogeneous	2	[61.32, 62.64]	67.28	16.85	39.04	**49.79**
Finetune			[61.46, 62.94]	–	62.53	63.67	**65.37**
One-shot	Heterogeneous	2	[58.81, 60.70]	67.31	17.52	32.00	**47.91**
Finetune			[63.44, 63.79]	–	58.73	62.29	**64.15**
One-shot	Homogeneous	4	[61.32, 62.64, 63.03, 61.58]	68.97	13.21	14.13	**33.51**
Finetune			[62.02, 61.28, 62.34, 61.55]	–	64.59	64.90	**66.35**
One-shot	Heterogeneous	4	[56.94, 54.15, 57.55, 59.00]	67.81	12.43	27.10	**41.25**
Finetune			[63.58, 61.72, 62.98, 63.79]	–	59.1	63.63	**64.33**

Table 2: Results for finetuning the fused VGG11 models on CIFAR-10. The settings are the same with Table 1.

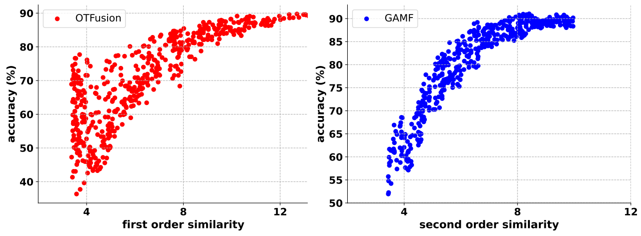
	Data Partition	# $(=N)$ of Models	Individual Models [Acc(%) of each model, ...]	Pred ($N\times$ cost)	Vanilla ($1\times$ cost)	OTFusion ($1\times$ cost)	GAMF ($1\times$ cost)
One-shot	Homogeneous	2	[90.31, 90.50]	91.34	17.01	85.98	**87.02**
Finetune			[90.29, 90.53]	–	90.41	90.68	**90.75**
One-shot	Heterogeneous	2	[69.29, 71.89]	75.46	9.84	9.87	**36.73**
Finetune			[71.37, 75.96]	–	60.34	62.08	**79.40**
One-shot	Homogeneous	4	[90.31, 90.50, 90.47, 90.56]	91.91	9.99	**73.56**	73.42
Finetune			[90.29, 90.53, 90.45, 90.55]	–	69.33	**90.89**	90.87
One-shot	Heterogeneous	4	[73.88, 70.73, 72.50, 71.53]	79.87	9.24	9.99	**12.35**
Finetune			[76.76, 75.96, 77.25, 75.24]	–	43.63	48.21	**50.54**

4.2. Compact Model Ensemble Experiments

In this experiment, we aim at testing the performance of the model fusion to generate a compact ensemble model. The ensemble model is a combination of all local models, and the traditional approach is to average predictions (output from the last layer) of them. The prediction ensemble can always reach better performance, but the cost is to maintain the parameters of all local models. Following OTFusion (Singh & Jaggi, 2020), we hope to find a more efficient and compact way of the model ensemble, that is to maintain only one single model instead of maintaining all. We conduct the compact model ensemble experiments in MNIST and CIFAR-10, with the same settings as OTFusion.

Experiments on MNIST. We try to merge two simple networks with the homogeneous and heterogeneous data partition, as shown in Fig. 3. We show the results with different ensemble proportions, as the proportion of the second model raises from 0.0 to 1.0. The model fusion is completed in one-shot without any further finetune. As shown in Fig. 3, we can see that in the homogeneous data partition settings, the performance of the ensemble is not significant, since the two models are too similar and well-trained and both GAMF and GUROBI are close to the prediction ensemble. In the heterogeneous data partition, GAMF reaches promising performance, and even outperforms the prediction ensemble.

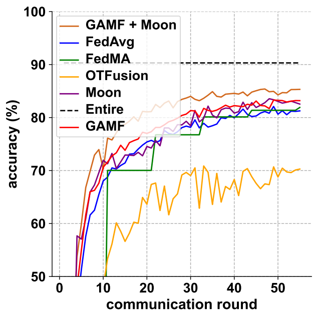
Experiments on CIFAR-10. We test the performance of model fusion in a standard image classification dataset with modern neural networks. We choose CIFAR-10 and LeNet/VGG11 networks in these experiments. For each backbone neural network, we test the model fusion performance with 2/4 models and homogeneous/heterogeneous data partition. Please note that we do not use the baseline GUROBI from now on, since LeNet and VGG11 are too large for GUROBI to handle. The results are shown in Table 1 and Table 2. Similar to (Singh & Jaggi, 2020), We report the one-shot and finetune results for the fused model.



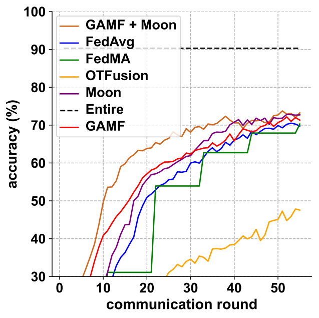
(a) OTFusion (b) GAMF

Figure 4: Similarity-accuracy scatter of OTFusion (1st-order) and GAMF (2nd-order) on CIFAR-10 with VGG11.

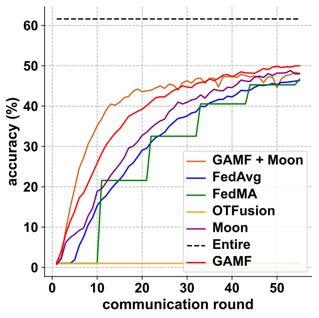
In the LeNet experiments, GAMF outperforms OTFusion and Vanilla in all settings, especially in multi-model settings. Although the one-shot performances of all methods are poor, GAMF can still reach about 70% accuracy of the original



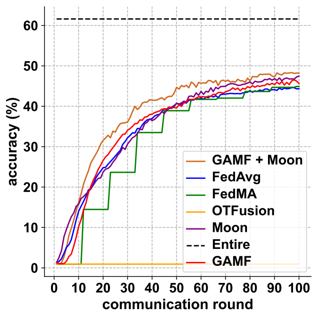
(a) CIFAR-10; 5 clients



(b) CIFAR-10; 10 clients



(c) CIFAR-100; 5 clients



(d) CIFAR-100; 10 clients

Figure 5: Convergence rates in four FL scenarios: training VGG11 on CIFAR-10/CIFAR-100 with 5/10 clients in total.

Table 3: The top-1 accuracy of the compared model fusion methods on CIFAR-10, CIFAR-100, and Tiny-Imagenet.

	CIFAR-10		CIFAR-100		Tiny-Imagenet	
	5 clients	10 clients	5 clients	10 clients	5 clients	10 clients
FedAvg	81.01% ± 0.31%	69.99% ± 0.40%	45.94% ± 0.32%	44.42% ± 0.13%	22.87% ± 0.11%	17.41% ± 0.13%
OTFusion	69.83% ± 0.55%	46.40% ± 1.01%	1.00% ± 0.00%	1.00% ± 0.00%	0.50% ± 0.00%	0.50% ± 0.00%
FedMA	81.46% ± 0.20%	70.29% ± 0.69%	47.50% ± 0.52%	44.95% ± 0.19%	23.19% ± 0.16%	17.28% ± 0.20%
Moon	82.78% ± 0.57%	72.42% ± 0.45%	48.24% ± 0.28%	46.99% ± 0.28%	23.49% ± 0.10%	19.01% ± 0.15%
GAMF (ours)	82.82% ± 0.58%	72.39% ± 0.54%	**49.80%** ± 0.25%	45.99% ± 0.41%	23.96% ± 0.12%	20.42% ± 0.13%
GAMF + Moon	**84.92%** ± 0.39%	**73.43%** ± 0.59%	48.72% ± 0.78%	**48.24%** ± 0.39%	**24.61%** ± 0.11%	**21.51%** ± 0.15%

models. The finetuned performance of GAMF is better than the original models but slightly worse than the prediction ensemble. It is common that the fused model performs worse than the prediction ensemble (Singh & Jaggi, 2020). The fused model of GAMF is slightly worse in terms of accuracy, but it enjoys 2× or 4× efficiency.

In the VGG11 experiments, OTFusion performs in two extremes. OTFusion performs well in the homogeneous data partition and even better than GAMF in the multi-model settings, but it fails to converge in the heterogeneous data partition. Our GAMF performs more stably except for the last row. The settings of 4 models with heterogeneous data partition seem too hard for the model fusion methods since all methods do not work well. The finetune in this settings is more like re-training the networks from the scratch.

Comparison of first-order and second-order similarity. In the design of GAMF, we consider the second-order similarity instead of existing work only caring the first-order similarity, such as OTFusion and FedMA. To show the effectiveness of second-order similarity, we conduct extra compact ensemble experiments on CIFAR-10 with VGG11. We randomly disturb the solved solution by OTFusion and GAMF, and draw the scatter plots of similarity-accuracy in Fig. 4. We can see that the curve of GAMF is more smooth and correlated than that of OTFusion. Therefore, we believe that second-order similarity is better than first-order similarity in the sense of effective model fusion.

4.3. Federated Learning Experiments

We use the framework FedML (He et al., 2020b) and conduct the experiments on CIFAR-10/100, and Tiny-Imagenet. Except for baselines, we report the accuracy of training with the entire data as "Entire", which can be the upper bound.

Experiments on the CIFAR-10. The results are shown in Fig. 5a and 5b for 5 clients and 10 clients respectively. GAMF and Moon reach similar final accuracy, but GAMF converges faster than Moon. It is reasonable that GAMF does not surpass Moon, since Moon and GAMF focus on the different parts of federated learning to improve. Moon aims at improving the local training while GAMF focuses on the model fusion in communication.

For the competitive methods OTFusion and FedMA that focus on the same part as GAMF, GAMF can outperform them nearly all the time. People may notice that the curve of FedMA is stair-like, which is because FedMA update only the parameter of one layer in each communication round, which needs 11 rounds to update all 11 layers in VGG11. If looking only at the lift points of the FedMA curve, the performance of FedMA is over FedAvg. It seems that OTFusion is not suitable for federated learning since

Table 4: Comparison of the time consumption (in hours).

	CIFAR-10		CIFAR-100		Tiny-Imagenet	
	5 clients	10 clients	5 clients	10 clients	5 clients	10 clients
OTFusion	3.83	11.96	4.11	14.71	10.67	36.85
GAMF	5.28	18.05	6.58	20.03	18.56	59.58

Table 5: The top-1 accuracy (%) of compared methods on the CIFAR-10 dataset.

client #	FedAvg	Moon	GAMF	GAMF+Moon
10	69.99 ± 0.40	72.42 ± 0.45	72.39 ± 0.54	$\mathbf{73.43 \pm 0.59}$
16	68.27 ± 0.46	71.69 ± 0.28	71.94 ± 0.55	$\mathbf{72.07 \pm 0.47}$
20	65.70 ± 0.50	69.88 ± 0.19	$\mathbf{71.37 \pm 0.36}$	71.21 ± 0.31

it is under FedAvg. It proves the generalize ability of our method GAMF since it can work in both compact model ensemble and federated learning experiments.

Experiments on CIFAR-100. Fig. 5c and 5d show the results on CIFAR-100 with 5 clients and 10 clients respectively. In the 5 client settings, we can see that GAMF can outperform all baselines in the whole training process. We admit the final accuracy is not as good as we expect, and we consider it is because VGG11 is hard for federated learning on CIFAR-100. Compared to FedMA, GAMF can boost the performance of model fusion to about 3%.

In the 10 clients experiment, we add the number of communication rounds from 55 to 100. As Fig. 5d shows, the final performance of Moon is slightly over our GAMF, but it is still within an acceptable range. The performance of GAMF is over Moon in the middle stage of the training process, which means the convergence speed of GAMF is relatively remarkable. Though under Moon, GAMF outperforms the other baselines FedMA, FedAvg, and OTFusion. Regrettably, OTFusion fails to converge on CIFAR-100.

Trade off between time and accuracy. Compared to OTFusion, our proposed GAMF considers the second-order similarities. The second-order similarities can better align the channels, but has a clear drawback is the cost of extra time. Therefore, we record the time consumption of OTFusion and GAMF in Table 4 to intuitively see the extra time used by GAMF due to the second-order similarities. We can see that the time consumption of GAMF is only about 50% higher than OTFusion. We consider this extent of extra time is acceptable since the performance boost of GAMF is way more than 50% compared to OTFusion.

Combination of Moon and GAMF. Moon is the state-of-the-art FL method that focuses on improving the local training of each client, unlike GAMF focusing on the model aggregation. We find the performance of Moon is similar to our GAMF in some settings. It suggests that both Moon and GAMF are useful for FL. The relation between Moon and GAMF is complementary instead of competitive due to their different focuses. Therefore, we conduct additional experiments for combining GAMF with Moon. We show the performance of the combined method with the same experiment settings. The results are shown in Table 3 and Fig. 5. It turns out the combination of GAMF and Moon does reach a better performance compared to the single GAMF and Moon. It also shows that model fusion is an important factor in FL since the performance boosted by GAMF is close or even better than that boosted by Moon.

Experiments on larger scale. In the aforementioned experiments, we test the performance of GAMF on CIFAR-10, CIFAR-100, and Tiny-Imagenet with 5/10 clients. Here, we conduct extra experiments with more clients to see the scalability of GAMF. As shown in Table 5, we compare FedAvg, Moon, GAMF, and GAMF+Moon with 10/16/20 clients on CIFAR-10, which are the common number of clients in the experiments of federated learning papers. We can see that our proposed GAMF can still outperform the baselines under more client scenarios. These experiments prove the scalability of our proposed GAMF.

5. Conclusion

In this paper, we manage to formulate model fusion as the graph matching problem. For scalability and consistency, we propose a graduated assignment based method named GAMF for fusing two or multiple models, which captures and suits the inherent sparsity of the fusion problem. On the tasks of compact model ensemble and FL, GAMF outperforms peer methods. Admittedly, even with our efficient algorithm design, GAMF has extra time complexity compared to OTFusion because of the higher complexity of graph matching. However, a better fused model may be worth such an overhead, and in FL the bottleneck is usually the communication costs, which can be significantly cut by our method. In the future, we will explore model fusion with different sized models and more clients, to improve the scalability of GAMF. Moreover, we plan to improve multi-model fusion by multi-graph matching from traditional approaches (Jiang et al., 2020) to learning-based ones (Rolínek et al., 2020), which is an emerging area in the intersection of combinatorial optimization and machine learning.

Acknowledgments

This work was partly supported by National Key Research and Development Program of China (2020AAA0107600), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and NSFC (72061127003), Science and Technology Innovation 2030 –"Brain Science and Brain-like Research" Major Project (No. 2021ZD0201402 and 2021ZD0201405).

References

Acar, D. A. E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., and Saligrama, V. Federated learning based on dynamic regularization. In *ICLR*, 2020.

Breiman, L. Bagging predictors. *Machine learning*, 24(2): 123–140, 1996.

Chen, W., Horvath, S., and Richtarik, P. Optimal client sampling for federated learning. *arXiv preprint arXiv:2010.13723*, 2020.

Cho, M., Lee, J., and Lee, K. M. Reweighted random walks for graph matching. In *Eur. Conf. Comput. Vis.*, 2010.

Dai, R., Shen, L., He, F., Tian, X., and Tao, D. Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training. In *International Conference on Machine Learning*, 2022.

Deng, Y., Kamani, M. M., and Mahdavi, M. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Dinh, C. T., Tran, N. H., and Nguyen, T. D. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*, 2020.

Du, X., El-Khamy, M., Lee, J., and Davis, L. Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 953–961. IEEE, 2017.

Fraboni, Y., Vidal, R., Kameni, L., and Lorenzi, M. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. *arXiv preprint arXiv:2105.05883*, 2021.

Gold, S. and Rangarajan, A. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

He, C., Annavaram, M., and Avestimehr, S. Group knowledge transfer: Federated learning of large cnns at the edge. *arXiv preprint arXiv:2007.14513*, 2020a.

He, C., Li, S., So, J., Zeng, X., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020b.

He, C., Balasubramanian, K., Ceyani, E., Yang, C., Xie, H., Sun, L., He, L., Yang, L., Yu, P. S., Rong, Y., et al. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145*, 2021a.

He, C., Shah, A. D., Tang, Z., Sivashunmugam, D. F. N., Bhogaraju, K., Shimpi, M., Shen, L., Chu, X., Soltanolkotabi, M., and Avestimehr, S. Fedcv: A federated learning framework for diverse computer vision tasks. *arXiv preprint arXiv:2111.11066*, 2021b.

Huang, T., Lin, W., Shen, L., Li, K., and Zomaya, A. Y. Stochastic client selection for federated learning with volatile clients. *IEEE Internet of Things Journal*, 2022a.

Huang, T., Liu, S., Shen, L., He, F., Lin, W., and Tao, D. Achieving personalized federated learning with sparse local models. *arXiv preprint arXiv:2201.11380*, 2022b.

Jiang, Z., Wang, T., and Yan, J. Unifying offline and online multi-graph matching via finding shortest paths on supergraph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Jiang, Z., Wang, T., and Yan, J. Unifying offline and online multi-graph matching via finding shortest paths on supergraph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3648–3663, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *ICML*, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kuhn, H. W. The hungarian method for the assignment problem. In *Export. Naval Research Logistics Quarterly*, pp. 83–97, 1955.

Lawler, E. L. The quadratic assignment problem. *Management Science*, 1963.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Leonardos, S., Zhou, X., and Daniilidis, K. Distributed consistent data association via permutation synchronization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2645–2652, 2017.

Leontev, M. I., Islenteva, V., and Sukhov, S. V. Non-iterative knowledge fusion in deep convolutional neural networks. *Neural Processing Letters*, 51(1):1–22, 2020.

Leordeanu, M. and Hebert, M. A spectral technique for correspondence problems using pairwise constraints. *ICCV*, 2005.

Li, Q., He, B., and Song, D. Model-contrastive federated learning. In *CVPR*, pp. 10713–10722, 2021.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Liu, C., Wang, R., Jiang, Z., Yan, J., Huang, L., and Lu, P. Revocable deep reinforcement learning with affinity regularization for outlier-robust graph matching. *arXiv preprint arXiv:2012.08950*, 2020.

Liu, Q., Chen, C., Qin, J., Dou, Q., and Heng, P.-A. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *CVPR*, 2021a.

Liu, S., Chen, T., Chen, X., Atashgahi, Z., Yin, L., Kou, H., Shen, L., Pechenizkiy, M., Wang, Z., and Mocanu, D. C. Sparse training via boosting pruning plasticity with neuroregeneration. *Advances in Neural Information Processing Systems*, 34, 2021b.

Liu, S., Chen, T., Chen, X., Shen, L., Mocanu, D. C., Wang, Z., and Pechenizkiy, M. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. *arXiv preprint arXiv:2202.02643*, 2022.

Loiola, E. M., de Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P., and Querido, T. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 2007.

Malinovskiy, G., Kovalev, D., Gasanov, E., Condat, L., and Richtarik, P. From local sgd to local fixed-point methods for federated learning. In *ICML*, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Optimization, G. Gurobi optimizer reference manual. http://www.gurobi.com, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

Peterson, D., Kanani, P., and Marathe, V. J. Private federated learning with domain adaptation. *arXiv preprint arXiv:1912.06733*, 2019.

Rolínek, M., Swoboda, P., Zietlow, D., Paulus, A., Musil, V., and Martius, G. Deep graph matching via blackbox differentiation of combinatorial solvers. In *ECCV*, 2020.

Schapire, R. E. A brief introduction to boosting. In *Ijcai*, volume 99, pp. 1401–1406. Citeseer, 1999.

Schellewald, C. and Schnörr, C. Probabilistic subgraph matching based on convex relaxation. In *International Workshop on Energy Minimization Methods in CVPR*, 2005.

Shi, X., Ling, H., Hu, W., Xing, J., and Zhang, Y. Tensor power iteration for multi-graph matching. In *CVPR*, 2016.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Singh, S. P. and Jaggi, M. Model fusion via optimal transport. *NeurIPS*, 2020.

Smith, J. and Gashler, M. An investigation of how neural networks learn from the experiences of peers through periodic weight averaging. In *ICMLA*. IEEE, 2017.

Solé-Ribalta, A. and Serratosa, F. Graduated assignment algorithm for multiple graph matching based on a common labeling. *IJPRAI*, 2013.

Utans, J. Weight averaging for neural networks and local resampling schemes. In *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models*, 1996.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020a.

Wang, J., Charles, Z., Xu, Z., Joshi, G., McMahan, H. B., Al-Shedivat, M., Andrew, G., Avestimehr, S., Daly, K., Data, D., et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021a.

Wang, R. and et al. Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning. In *NeurIPS*, 2020.

Wang, R., Yan, J., and Yang, X. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, 2019.

Wang, R., Yan, J., and Yang, X. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021b.

Wang, T., Jiang, Z., and Yan, J. Clustering-aware multiple graph matching via decayed pairwise matching composition. *AAAI*, 2020b.

Wang, Z., Wang, X., Shen, L., Suo, Q., Song, K., Yu, D., Shen, Y., and Gao, M. Meta-learning without data via wasserstein distributionally-robust model fusion. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

Wolpert, D. H. Stacked generalization. *Neural networks*, 5 (2):241–259, 1992.

Wu, G. and Gong, S. Decentralised learning from independent multi-domain labels for person re-identification. *arXiv preprint arXiv:2006.04150*, 2020.

Yan, J., Cho, M., Zha, H., Yang, X., and Chu, S. Multi-graph matching via affinity optimization with graduated consistency regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016a.

Yan, J., Yin, X.-C., Lin, W., Deng, C., Zha, H., and Yang, X. A short survey of recent advances in graph matching. In *ICMR*, 2016b.

Yan, J., Yang, S., and Hancock, E. Learning graph matching and related combinatorial optimization problems. In *IJCAI*, 2020.

Yu, F., Zhang, W., Qin, Z., Xu, Z., Wang, D., Liu, C., Tian, Z., and Chen, X. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2066–2074, 2021.

Yu, T., Wang, R., Yan, J., and Li, B. Learning deep graph matching with channel-independent embedding and hungarian attention. In *ICLR*, 2020.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In *ICML*, 2019.

Zhang, L., Shen, L., Ding, L., Tao, D., and Duan, L.-Y. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. *arXiv preprint arXiv:2203.09249*, 2022.

Zhou, F. and Torre, F. D. L. Factorized graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:1774–1789, 2016.

In the appendix, we mainly supplemented the federated learning experiments that were not detailed discussed in the submitted paper due to the page limit.

A. Environment

We choose FedML [6] (He et al., 2020b) as the environment in our federated learning experiments. FedML is an open research library and benchmark that facilitates the development of new federated learning algorithms and fair performance comparisons. We use the code of FedML as the framework and implement our GAMF on it.

B. Baseline Implementation

- **FedAvg** (McMahan et al., 2017). We use the standard implementation of FedAvg in FedML, which simply averages the weights from all local clients by their amount of data. In our experiments, we assume that only the amount of data of each local client is known by the global server.
- **OTFusion** (Singh & Jaggi, 2020). We use the open-source code provided by the authors in GitHub [7]. Since OTFusion does not include the federated learning experiments in their paper, we modify their code as an extra module for model aggregation, juxtaposed with this FedAvg. That is, the code for modified OTFusion is the same as the FedAvg, except for the aggregation of local clients.
- **FedMA** (Wang et al., 2020a). We read the code from the open-source repository [8] of FedMA, and find the pipeline of FedMA is a little different from the standard pipeline of federated learning. Instead of updating the whole model in one communication round, FedMA only updates one layer of the model sequentially. Therefore, FedMA needs the number of layers communication rounds for updating the whole model. Besides, we remove some unfair tricks that are used in FedMA: 1) FedMA needs the data distribution accurate to each category of each client, but we only allow the total data number of each client known. 2) Due to the importance of the last layer (output layer), FedMA requires to triple the local epochs in the communication rounds, while we keep the local epochs unchanged all the time.
- **Moon** (Li et al., 2021). The code of Moon [9] is very neat and easy to understand, so the implementation of Moon is nothing special to mention. Here we want to introduce the combination of Moon and our proposed GAMF. As we mentioned in the paper, Moon focuses on improving the local training process while GAMF focuses on improving the model aggregation part. The original aggregation part of Moon is FedAvg, and we simply replace FedAvg with our GAMF. We understand the simple combination may not always be the optimal solution, and we think how to effectively combine Moon and GAMF is still worth studying and leaving for future work.

[6] https://github.com/FedML-AI/FedML
[7] https://github.com/sidak/OTFusion
[8] https://github.com/IBM/FedMA
[9] https://github.com/QinbinLi/MOON