

# FedGradNorm: Personalized Federated Gradient-Normalized Multi-Task Learning

Matin Mortaheb    Cemil Vahapoglu    Sennur Ulukus  
Department of Electrical and Computer Engineering  
University of Maryland, College Park, MD 20742  
mortaheb@umd.edu    cemilnv@umd.edu    ulukus@umd.edu

**Abstract**—Multi-task learning (MTL) is a novel framework to learn several tasks simultaneously with a single shared network where each task has its distinct personalized header network for fine-tuning. MTL can be implemented in federated learning settings as well, in which tasks are distributed across clients. In federated settings, the statistical heterogeneity due to different task complexities and data heterogeneity due to non-iid nature of local datasets can both degrade the learning performance of the system. In addition, tasks can negatively affect each other’s learning performance due to negative transference effects. To cope with these challenges, we propose *FedGradNorm* which uses a dynamic-weighting method to normalize gradient norms in order to balance learning speeds among different tasks. *FedGradNorm* improves the overall learning performance in a personalized federated learning setting. We provide convergence analysis for *FedGradNorm* by showing that it has an exponential convergence rate. We also conduct experiments on multi-task facial landmark (MTFL) and wireless communication system dataset (RadCom-Dynamic). The experimental results show that our framework can achieve faster training performance compared to equal-weighting strategy. In addition to improving training speed, *FedGradNorm* also compensates for the imbalanced datasets among clients.

## I. INTRODUCTION

Multi-task learning (MTL) is a learning paradigm that aims to learn multiple related tasks simultaneously by learning a shared representation for all tasks [1], [2]. MTL is motivated by the idea that different data used for related tasks can have a common representation [3]. In MTL, all tasks share the pre-layers of a network, and task-specific layers are stacked on top of the shared base to output task-specific predictions. The loss of the overall model is expressed as a weighted sum of individual task losses multiplied by their corresponding loss weights. Using the synergy among multiple tasks, MTL offers data efficiency, robust regularization through a shared representation, better overall system performance, and fast learning exploiting the auxiliary information [4]. MTL is particularly suitable for distributed learning settings, as it may not be feasible to expect a single centralized unit to have data and labels relevant to multiple fundamentally different tasks. Thus, in this paper, we consider MTL in a federated learning setting, enhanced with personalization.

Federated learning (FL) is a distributed learning framework where many clients train a shared model under the orchestration of a centralized server while keeping the training data decentralized and private. In personalized federated learning (PFL), see Fig. 1, clients have different tasks: while the pa-

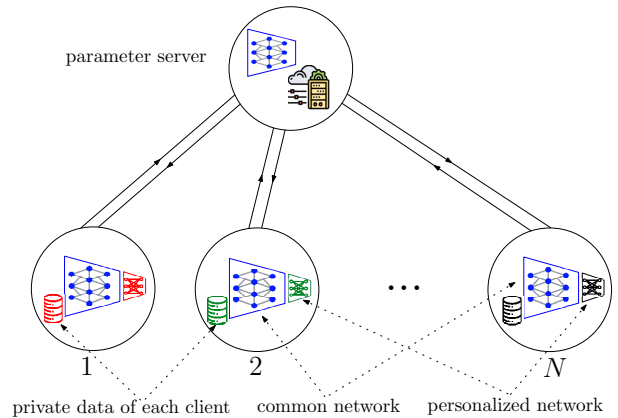


Fig. 1. Personalized federated learning framework with a common network (shown in blue) and small personalized headers (shown in red, green, black).

parameter server and the clients train a common base model, each client further trains a small header network for its own specific task, referred to as personalization. PFL enables essentially different learning models in individual clients that better fit user-specific data while also capturing the common knowledge distilled from data of other devices [5]–[7]. The most relevant PFL works for our paper are federated representation learning (*FedRep*) [5] and federated learning with personalization layers (*FedPer*) [6]. In both *FedPer* and *FedRep*, the clients share data representation across the shared global network and use unique task-specific local heads. Further, in order to maintain fair training among multiple tasks, cope with statistical heterogeneity due to task complexity and data distribution, and suppress negative transference effects between tasks, adaptive weighting strategies can be used to adjust the weights of the task losses over time. *GradNorm* [8] is a dynamic weighting algorithm that scales the task loss functions with respect to the learning speeds in order to normalize the gradient norms.

*FedPer* [6] and *FedRep* [5] consider only the equal-weights case to aggregate the clients’ loss, and *GradNorm* [8] considers dynamic weights in MTL, but does not consider a distributed FL setting. We combine some aspects of [5], [6], [8] to create our novel framework called *FedGradNorm*, where we incorporate dynamic-weighting inside a PFL setting. We provide theoretical convergence proof for our framework, while *FedPer* [6] and *GradNorm* [8] do not provide any convergence proofs, and *FedRep* [5] provides a convergence proof only for the

special case of linear personalized header. The experimental and theoretical results show the superiority of our framework to the equal-weighting PFL methods [5], [6].

The main contributions of our paper are as follows: i) We propose *FedGradNorm* algorithm, which uses *GradNorm* dynamic-weighting strategy in a PFL setup to achieve a better and fair learning performance when clients have different tasks. ii) We provide convergence analysis for our *FedGradNorm* algorithm utilizing *bilevel optimization* [9], [10]. To the best of our knowledge, this is the first work that provides convergence analysis for *GradNorm* adaptive-weighting strategy in addition to applying it in an FL setting. iii) We conduct several experiments on our framework using multi-task facial landmark (MTFL) dataset [11], and RadComDynamic wireless communications dataset [12]. To compare the performance of the learning speed of *FedGradNorm* with the equal-weighting method, we investigate the change in the task loss during the training phase. Experimental results exhibit better performance in *FedGradNorm* than *FedRep*.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

### A. Federated Learning (FL) Setup

The generic form of FL problem with  $N$  clients is

$$\min_{\omega} \{F(\omega) \triangleq \frac{1}{N} \sum_{i=1}^N p^{(i)} F^{(i)}(\omega)\} \quad (1)$$

where  $p^{(i)}$  is the loss weight for client  $i$  such that  $\sum_{i=1}^N p^{(i)} = N$ , and  $F^{(i)}$  is the local loss function for client  $i$ .

### B. Personalized Federated Multi-Task Learning (PF-MTL)

We consider a PFL setting with  $N$  clients, in which client  $i$  has its own local dataset  $D_i = \{(\mathbf{x}_j^{(i)}, y_j^{(i)})\}_{j=1}^{n_i}$  where  $n_i$  is the size of the local dataset.  $T_i$  denotes the task of client  $i$ ,  $\forall i \in [N]$ . The system model consists of a global representation network  $q_{\omega} : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$  which is a function parameterized by  $\omega \in \mathcal{W}$  and maps data points to a lower space of size  $d'$ . All clients share the same global representation network which is synchronized across clients with global aggregation. The client-specific heads  $q_{h^{(i)}} : \mathbb{R}^{d'} \rightarrow \mathcal{Y}$  are functions parameterized by  $h^{(i)} \in \mathcal{H}$  for all clients  $i \in [N]$  and map from the low dimensional representation space to the label space  $\mathcal{Y}$ . The system model is shown in Fig. 1. The local model for client  $i$  is the composition of the  $i$ th client's global representation model  $q_{\omega}$  and personalized model  $q_{h^{(i)}}$ , shown as  $q_i(\cdot) = (q_{h^{(i)}} \circ q_{\omega})(\cdot)$ . In addition, the local loss for the  $i$ th client is  $F^{(i)}(h^{(i)}, \omega) = F^{(i)}(q_i(\cdot)) = F^{(i)}((q_{h^{(i)}} \circ q_{\omega})(\cdot))$ .

The clients and the centralized server aim at learning global representation parameters  $\omega$  together, while each client  $i$  learns its unique client-specific parameters  $h^{(i)}$  locally by performing alternating minimization. Specifically, client  $i$  performs  $\tau_h$  local gradient based updates to optimize  $h^{(i)}$  while global network parameters at client  $i$ , i.e.,  $\omega^{(i)}$  is frozen. Thereafter, client  $i$  performs  $\tau_{\omega}$  local updates for optimizing the global shared network parameters at client  $i$  while the parameters corresponding to the client-specific head are frozen. Then, the

global shared network parameters  $\{\omega^{(i)}\}_{i=1}^N$  are aggregated at the centralized server to have a common  $\omega$ . Thus, the optimization problem is

$$\min_{\omega \in \mathcal{W}} \frac{1}{N} \sum_{i=1}^N p^{(i)} \min_{h^{(i)} \in \mathcal{H}} F^{(i)}(h^{(i)}, \omega) \quad (2)$$

*FedRep* [5] investigates this problem for  $p^{(i)} = 1$  for  $\forall i \in [N]$ .

### C. PF-MTL as Bilevel Optimization Problem

Since  $F^{(i)}(h^{(i)}, \omega)$  depends on only  $h^{(i)}$  and  $\omega$  for all  $i \in [N]$ , we can rewrite (2) as

$$\min_{\omega \in \mathcal{W}, \{h^{(i)} \in \mathcal{H}\}_{i=1}^N} \frac{1}{N} \sum_{i=1}^N p^{(i)} F^{(i)}(h^{(i)}, \omega) \quad (3)$$

The problem given in (3) depends on  $p^{(i)}$  values obtained by our *FedGradNorm* algorithm, which will be described later, as a result of another optimization problem. Thus, the problem can be written in the form of a bilevel optimization problem which is an optimization problem that contains another optimization problem as a constraint in the following form

$$\begin{aligned} & \min_{x_u \in X_u, x_l \in X_l} F(x_u, x_l) \\ \text{s.t. } & x_l = \arg \min_{x_l \in X_l} \{g(x_u, x_l) : c_j(x_u, x_l) \leq 0, j = 1, \dots, J\} \\ & C_m(x_u, x_l) \leq 0, \quad m = 1, \dots, M \end{aligned} \quad (4)$$

where  $F(x_u, x_l)$  is the upper-level objective function and  $g(x_u, x_l)$  is the lower-level objective function;  $\{c_j(x_u, x_l) \leq 0, j = 1, \dots, J\}$  represent the constraints for the lower-level optimization problem; and  $\{C_m(x_u, x_l) \leq 0, m = 1, \dots, M\}$  and the lower-level optimization problem itself represent the constraints for the upper-level optimization problem.

We utilize the iterative differentiation (ITD) algorithm [10] which is given in Algorithm 1. The upper-level optimization update is performed in the outer loop, while the lower-level optimization update is performed in the inner loop.

---

#### Algorithm 1 Iterative Differentiation (ITD) Algorithm.

---

**Input:**  $K, D$ , step sizes  $\alpha, \beta$ , initialization  $x_u(0), x_l(0)$ .

**for**  $k = 0, 1, 2, \dots, K$  **do**

Set  $x_l^0(k) = x_l^D(k-1)$  if  $k > 0$  otherwise  $x_l(0)$ .

**for**  $t = 1, \dots, D$  **do**

Update  $x_l^t(k) = x_l^{t-1}(k) - \alpha \nabla_{x_l} g(x_u(k), x_l^{t-1}(k))$

Compute  $\hat{\nabla}_{x_u} F(x_u(k), x_l^D(k)) = \frac{\partial F(x_u(k), x_l^D(k))}{\partial x_u}$

Update  $x_u(k+1) = x_u(k) - \beta \hat{\nabla}_{x_u} F(x_u(k), x_l^D(k))$

---

For our problem,  $x_u$  and  $x_l$  correspond to  $(\{h^{(i)}\}_{i=1}^N, \omega)$ ,  $\{p^{(i)}\}_{i=1}^N$ , respectively. Additionally,  $x_u(k)$  and  $x_l(k)$  are denoted as  $(\{h^{(i)}\}_{i=1}^N, \omega_k)$ ,  $\{p_k^{(i)}\}_{i=1}^N$  to represent the outer loop iteration index in Algorithm 1 for the rest of the paper. Then, the bilevel optimization problem in our case is

$$\begin{aligned} & \min_{\omega, \{h^{(i)}\}_{i=1}^N, \{p^{(i)}\}_{i=1}^N} F(\{h^{(i)}\}_{i=1}^N, \omega, \{p^{(i)}\}_{i=1}^N) \\ \text{s.t. } & \{p^{(i)}\}_{i=1}^N \in \arg \min_{\{p^{(i)}\}_{i=1}^N \in \mathbb{R}^N} F_{grad} \end{aligned} \quad (5)$$

In our problem,  $i$  in  $p_k^i$  represents the inner loop iteration index, while  $i$  in  $p_k^{(i)}$  represents the client index. In (5), our objective function is the weighted sum of the local loss functions, i.e.,  $F = \frac{1}{N} \sum_{i=1}^N p^{(i)} F^{(i)}(h^{(i)}, \omega)$  and  $F_{grad}$  is the auxiliary loss function defined in the *FedGradNorm* algorithm in the next section.

### III. *FedGradNorm*: FEDERATED GRADNORM ALGORITHM

*FedGradNorm* is a distributed dynamic weighting strategy which is implemented in an FL setup under the orchestration of a parameter server. *FedGradNorm* is a generalization of *GradNorm* [8] in a centralized learning model.

#### A. Definitions and Preliminaries

In *FedGradNorm*, we aim to learn the dynamic loss weights  $\{p^{(i)}\}_{i=1}^N$  given in the lower-level optimization problem of (5). The main objective of the algorithm is to dynamically adjust the gradient norms so that the different tasks across clients can be trained at similar learning speeds. In the rest of the paper, *clients* and *tasks* will be used interchangeably as we assume that each client has its own different task. Before describing the algorithm in detail, we first introduce the notations:

- $\tilde{\omega}$ : A subset of the global shared network parameters  $\tilde{\omega} \subset \omega$ . *FedGradNorm* is applied on  $\tilde{\omega}_k^{(i)} \subset \omega_k^{(i)}$ , which is a subset of the global shared network parameters at client  $i$  at iteration  $k$ .  $\tilde{\omega}_k^{(i)}$  is generally chosen as the last layer of the global shared network at client  $i$  at iteration  $k$ .
- $G_{\tilde{\omega}_k^{(i)}}^{(i)}(k) = \|\nabla_{\tilde{\omega}_k^{(i)}} p_k^{(i)} F_k^{(i)}\| = p_k^{(i)} \|\nabla_{\tilde{\omega}_k^{(i)}} F_k^{(i)}\|$ : The  $\ell_2$  norm of the gradient of the weighted task loss at client  $i$  at iteration  $k$  with respect to the chosen weights  $\tilde{\omega}_k^{(i)}$ .
- $\bar{G}_{\tilde{\omega}}(k) = \mathbb{E}_{j \sim \text{task}}[G_{\tilde{\omega}_k^{(j)}}^{(j)}(k)]$ : The average gradient norm across all clients (tasks) at iteration  $k$ .
- $\tilde{F}_k^{(i)} = \frac{F_k^{(i)}}{F_0^{(i)}}$ : Inverse training rate of task  $i$  (at client  $i$ ) at iteration  $k$ , where  $F_k^{(i)}$  is the loss for client  $i$  at iteration  $k$ , and  $F_0^{(i)}$  is the initial loss for client  $i$ .
- $r_k^{(i)} = \frac{\tilde{F}_k^{(i)}}{\mathbb{E}_{j \sim \text{task}}[\tilde{F}_k^{(j)}]}$ : Relative inverse training rate of task  $i$  at iteration  $k$ .

Additional notations that are useful in algorithm description:

- $g_k^{(i)} = \frac{1}{\tau_\omega} \sum_{j=1}^{\tau_\omega} g_{k,j}^{(i)}$  is the average of gradient updates at client  $i$  at iteration  $k$ , where  $g_{k,j}^{(i)}$  is the  $j$ th local update of the global shared representation at client  $i$  at iteration  $k$ . Note that  $\|\nabla_{\tilde{\omega}_k^{(i)}} F_k^{(i)}\|$  is a subset of  $g_k^{(i)}$  since  $\tilde{\omega} \subset \omega$ .
- $h_{k,j}^{(i)}$  is the client-specific head parameters  $h^{(i)}$  after the  $j$ th local update on the client-specific network of client  $i$  at iteration  $k$ ,  $j = 1, \dots, \tau_h$ .
- $\omega_{k,j}^{(i)}$  is the global shared network parameters of client  $i$  after the  $j$ th local update at iteration  $k$ ,  $j = 1, \dots, \tau_\omega$ . Additionally,  $\omega_k^{(i)}$  denotes  $\omega_{k,\tau_\omega}^{(i)}$  for brevity.

#### B. *FedGradNorm* Description

*FedGradNorm* is used to balance the training rates of different tasks across clients by adjusting the gradient magnitudes as in the *GradNorm* [8]. Unlike *GradNorm*, *FedGradNorm*

is distributed across clients and the parameter server.  $\bar{G}_{\tilde{\omega}}$  is used to have a common scale for the gradient sizes while the gradient norms are adjusted according to the relative inverse training rates  $r_k^{(i)}$ . With a higher value of  $r_k^{(i)}$ , a higher gradient magnitude is used for task  $i$  in order to encourage the task to train more quickly.  $r_k^{(i)}$  is calculated by the parameter server by using  $\tilde{F}_k^{(i)}$  coming from clients. Therefore, by using the common scale of gradient magnitudes, and the relative inverse training rate, the desired gradient norm of task  $i$  at iteration  $k$  is determined as  $\bar{G}_{\tilde{\omega}}(k) \times [r_k^{(i)}]^\gamma$ , where  $\gamma$  represents the strength of the restoring force which pulls tasks back to a common training rate, which can also be thought of as a metric of task asymmetry across different tasks. If tasks have different learning complexities, i.e., different learning dynamics, a larger  $\gamma$  should be used for a stronger balancing.

Since we want the gradient norms to shift towards the desired gradient norm, loss weights  $p_k^{(i)}$  are updated by the minimization of an auxiliary loss function  $F_{grad}(k; \{p_k^{(i)}\}_{i=1}^N)$  defined as the summation of  $\ell_2$  distance between the actual gradient norm and the desired gradient norm across all tasks for each iteration  $k$ , i.e.,

$$\begin{aligned} F_{grad}(k; \{p_k^{(i)}\}_{i=1}^N) &= \sum_{i=1}^N F_{grad}^{(i)}(k; p_k^{(i)}) \\ &= \sum_{i=1}^N \left\| p_k^{(i)} \|\nabla_{\tilde{\omega}_k^{(i)}} F_k^{(i)}\| - \bar{G}_{\tilde{\omega}}(k) \times [r_k^{(i)}]^\gamma \right\| \end{aligned} \quad (6)$$

The auxiliary loss function  $F_{grad}(k; \{p_k^{(i)}\}_{i=1}^N)$  is constructed by the parameter server at each global iteration  $k$  by using  $\nabla_{\tilde{\omega}_k^{(i)}} F_k^{(i)}$ , which is a subset of the whole gradient of the global shared network sent by client  $i$  at iteration  $k$  for the global aggregation. In addition, clients send  $\tilde{F}_k^{(i)}$  to the parameter server, so that the parameter server can construct  $r_k^{(i)}$  to have the desired gradient norm.

Next, the parameter server performs the differentiation of  $F_{grad}(k; \{p_k^{(i)}\}_{i=1}^N)$  with respect to each element of  $\{p^{(i)}\}_{i=1}^N$  so that  $\nabla_{p^{(i)}} F_{grad}$  is applied via gradient descent to update  $p^{(i)}$ . The desired gradient norm terms,  $\bar{G}_{\tilde{\omega}}(k) \times [r_k^{(i)}]^\alpha$ , are treated as constant to prevent loss weights  $\{p^{(i)}\}_{i=1}^N$  from drifting towards zero while differentiating  $F_{grad}(k; \{p_k^{(i)}\}_{i=1}^N)$  with respect to each loss weight  $p_k^{(i)}$ . The weights are updated as,

$$p^{(i)} \leftarrow p^{(i)} - \alpha \nabla_{p^{(i)}} F_{grad}, \quad \forall i \in [N]. \quad (7)$$

The updated  $\{p^{(i)}\}_{i=1}^N$  are normalized so that  $\sum_{i=1}^N p^{(i)} = N$ . Finally, the parameter server obtains the global aggregated gradient  $g_k = \frac{1}{N} \sum_{i=1}^N p_k^{(i)} g_k^{(i)}$  to update the global shared network parameters  $\omega$  via  $\omega_{k+1} = \omega_k - \beta g_k$  and broadcasts the updated parameters to the clients for the next iteration. The overall *FedGradNorm* algorithm is summarized in Algorithm 2. In *FedGradNorm*,  $\text{Update}(f, h)$  represents the generic notation for the update of the variable  $h$  by using the gradient of  $f$  function with respect to the variable  $h$ .

### C. Convergence of FedGradNorm

In the convergence analysis, we assume strong convexity of the upper-level objective function  $F(\cdot)$  and the lower-level objective function  $F_{\text{grad}}(\cdot)$ , the Lipschitzness of the first and second order gradients of both objective functions. We prove the exponential convergence of *FedGradNorm* under these assumptions. Due to space limitations, we skip the details of the proof here, and present them in the longer version. Instead, here, we present our experimental results in the next section.

---

#### Algorithm 2 Training with *FedGradNorm*

---

Initialize  $\omega_0, \{p_0^{(i)}\}_{i=1}^N, \{h_0^{(i)}\}_{i=1}^N$

**for**  $k=1$  **to**  $K$  **do**

The parameter server sends the current global shared network parameters  $\omega_k$  to the clients.

**for** Each client  $i \in [N]$  **do**

Initialize global shared network parameters for local updates by  $\omega_{k,0}^{(i)} \leftarrow \omega_k$

**for**  $j = 1, \dots, \tau_h$  **do**

$h_{k,j}^{(i)} = \text{Update}(F^{(i)}(h_{k,j-1}^{(i)}, \omega_{k,0}^{(i)}), h_{k,j-1}^{(i)})$

$F_k^{(i)} = 0$

**for**  $j = 1, \dots, \tau_\omega$  **do**

$\omega_{k,j}^{(i)} \leftarrow \omega_{k,j-1}^{(i)} - \beta g_{k,j}^{(i)}$

$F_k^{(i)} += F^{(i)}(h_{k,\tau_h}^{(i)}, \omega_{k,j}^{(i)})$

$F_k^{(i)} \leftarrow \frac{1}{\tau_\omega} F_k^{(i)}$

Client  $i$  sends  $g_k^{(i)} = \frac{1}{\tau_\omega} \sum_{j=1}^{\tau_\omega} g_{k,j}^{(i)}$ , and  $\tilde{F}_k^{(i)} = \frac{F_k^{(i)}}{F_0^{(i)}}$  to

the parameter server

After collecting  $g_k^{(i)}$ , and  $\tilde{F}_k^{(i)}$  for active clients  $i \in [N]$ , the parameter server performs the following operations in the order:

- Constructs  $F_{\text{grad}}(k; \{p_k^{(i)}\}_{i=1}^N)$  using  $\{g_k^{(i)}\}_{i=1}^N$  and  $\{\tilde{F}_k^{(i)}\}_{i=1}^N$  as given in eq. (6).

- Updates  $p_k^{(i)} \leftarrow p_{k-1}^{(i)} - \alpha \nabla_{p^{(i)}} F_{\text{grad}}, \forall i \in [N]$ .

- Aggregates the gradient for the global shared network by  $g_k = \frac{1}{N} \sum_{i=1}^N p_k^{(i)} g_k^{(i)}$ .

- Updates the global shared network parameters with the aggregated gradient by  $\omega_{k+1} = \omega_k - \beta g_k$ .

- Broadcasts  $\omega_{k+1}$  to clients for the next global iteration.

---

## IV. EXPERIMENTAL RESULTS

We compare the task losses achieved by equal-weighting in *FedRep* and dynamic-weighting in our *FedGradNorm*.

### A. Dataset Specifications

We use the following two dataset for our experiments:

*Multi-task facial landmark (MTFL)* [11] contains 10,000 training data and 3,000 test images, which are face images annotated by 1) five facial landmarks, 2) gender, 3) smiling or not, 4) wearing glasses or not, and 5) head pose.

*Wireless dataset (RadComDynamic)* [12] is a multi-class wireless signal dataset of 125,000 samples. Samples are radar and communication signals of varying SNR values from GNU

radio companion. It contains 6 modulation types and 8 signal types. We perform 3 different tasks: 1) modulation classification, 2) signal type classification, and 3) anomaly detection. The modulation classes are amdsb, amssb, ask, bpsk, fmcw, pulsed continuous wave (PCW). The signal type classes are AM radio, short-range, radar-altimeter, air-ground-MTI, airborne-detection, airborne-range, ground-mapping. As an anomaly behavior, we consider having an SNR lower than -4 dB since SNR can be a proxy for geo-location information, and low SNR may indicate a signal coming from an outsider. Each data point in this dataset is a normalized signal vector of size 256 obtained by vectorizing the real and complex parts of the signal ( $x = x_I + jx_Q$  where  $x_I, x_Q \in \mathcal{R}^{128}$ ).

### B. Hyperparameters and Model Specifications

We choose  $\gamma$  as 0.9 through our experiments. Note that  $\gamma$  is the only hyperparameter of *FedGradNorm*, and it should be determined with respect to the task asymmetry in the system. The learning rate  $\beta$ , which is used for training of global shared network and the personalized network on the client side is 0.0002, and the learning rate  $\alpha$  for  $F_{\text{grad}}$  optimization is 0.004. We use Adam optimizer for both network training and  $F_{\text{grad}}$  optimization. The shared network model is explained in Table I. Each client also has a simple linear layer that maps the shared network's output to the corresponding prediction value for a personalized network. Cross-entropy and mean squared error (MSE) are used as the loss functions for classification and regression tasks, respectively.

NETWORK 1	NETWORK 2
CONV2D(1, 16, 5)	FC(256, 512)
MAXPOOL2D(2, 2)	FC(512, 1024)
CONV2D(16, 48, 3)	FC(1024, 2048)
MAXPOOL2D(2, 2)	FC(2048, 512)
CONV2D(48, 64, 3)	FC(512, 256)
MAXPOOL2D(2, 2)	
CONV2D(64, 64, 2)	

TABLE I  
SHARED NETWORK MODEL.

### C. Results and Analysis

We first start with the MTFL dataset. Since the first task (determining the face landmarks) is a regression task, it has a higher gradient than the other tasks, which are all classification tasks. As shown in Fig. 2, *FedGradNorm*, which is a dynamic-weighting method will gradually decrease the weight of the first task so that the other tasks can optimize their corresponding losses. Starting from epoch 70, when task 2 and 3 finally can decrease their loss with a higher rate, their corresponding task weights decrease to improve the two remaining tasks. Without using the dynamic-weighting method, tasks 2 and 3 could not be improved since task 1 would mask the remaining tasks' gradient updates. Task 4, detecting glasses on human faces, reaches the minimum very fast at the first epoch since it is an easy task compared to the others. Therefore, its performance does not improve much, as

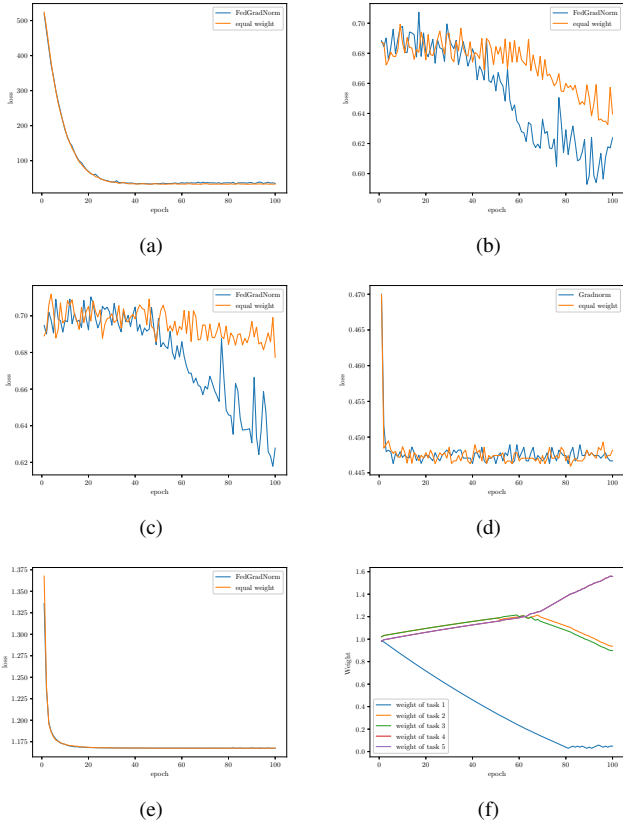


Fig. 2. Comparison of task losses in *FedGradNorm* and *FedRep*; balanced data allocation among tasks (a) task 1 (face landmark), (b) task 2 (gender), (c) task 3 (smile), (d) task 4 (glasses), (e) task 5 (pose), (f) task weights.

shown in Fig. 2. Although the performance of tasks 1 and 5 are also quite the same in the long-run, *FedGradNorm* helps to learn a bit faster at the early stages. For Fig. 2, the data allocation is balanced.

Next, we investigate the case where the data allocation is imbalanced, namely, some clients have a smaller portion of the dataset. In the following simulation, task 2 and task 4 have access to 500 data points while other tasks have 3000 data points to use in the training procedure. As shown in Table II, *FedGradNorm* again has a better performance compared to the equal-weighting case in *FedRep*.

Tasks	face landmark	gender	smile	glass	pose
<i>FedRep</i> loss	33.28	0.66	0.60	0.44	1.1
<i>FedGradNorm</i> loss	33.25	<b>0.56</b>	<b>0.57</b>	0.43	1.1

TABLE II  
COMPARISON OF TASK LOSSES AFTER 100 EPOCHS IN *FedGradNorm* AND *FedRep*; IMBALANCED DATA ALLOCATION AMONG TASKS.

Next, we consider the RadComDynamic dataset by using Network 2. As shown in Fig. 3, the result again indicates the superiority of *FedGradNorm* compared to *FedRep* on modulation detection and signal detection tasks. Based on the loss value, task 1 (modulation detection task) and task 2 (signal detection task) are harder and slower to learn than the anomaly detection task. As the result shows, using the

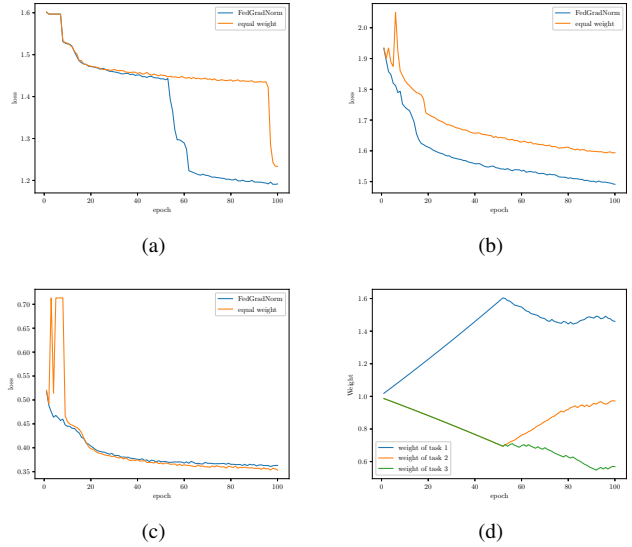


Fig. 3. Comparison between task accuracy achieved via *FedGradNorm* and *FedRep* in RadComDynamic dataset (a) task 1 (modulation classification), (b) task 2 (signal classification), (c) task 3 (anomaly behavior), (d) task weights.

dynamic-weighting based *FedGradNorm* can ensure that the signal and modulation detection classes, which are slower in training, have the same opportunity as the other task to improve their performance. Also, since the loss of task 2 and task 3 decreases with the same and higher slope at the beginning stage than task 1, the dynamic-weighting method increases the corresponding task weight for task 1 to push the task to be trained faster. In epoch 55, when the loss of task 1 decreases significantly, the weight of task 1 decreases to let other tasks to be trained more strongly.

## REFERENCES

- [1] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, July 1997.
- [2] Y. Zhang and Q. Yang. A survey on multi-task learning. Available online at arXiv:1707.08114.
- [3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1798–1828, August 2013.
- [4] M. Crawshaw. Multi-task learning with deep neural networks: A survey. Available online at arXiv:2009.09796.
- [5] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai. Exploiting shared representations for personalized federated learning. In *ICML*, July 2021.
- [6] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary. Federated learning with personalization layers. Available online at arXiv:1912.00818.
- [7] Y. Deng, MM. Kamani, and M. Mahdavi. Adaptive personalized federated learning. Available online at arXiv:2003.13461.
- [8] Z. Chen, V. Badrinarayanan, C. Lee, and A. Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, July 2018.
- [9] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, June 2017.
- [10] K. Ji, J. Yang, and Y. Liang. Bilevel optimization: Convergence analysis and enhanced design. In *ICML*, July 2021.
- [11] Z. Zhang, P. Luo, C.C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, September 2014.
- [12] A. Jagannath and J. Jagannath. Multi-task learning approach for automatic modulation and wireless signal classification. In *IEEE ICC*, June 2021.