

Dynamic Neural Graphs Based Federated Reptile for Semi-supervised Multi-Tasking in Healthcare Applications

Anshul Thakur, Pulkit Sharma, and David A. Clifton

Abstract—AI healthcare applications rely on sensitive electronic healthcare records (EHRs) that are scarcely labelled and are often distributed across a network of the symbiont institutions. It is challenging to train the effective machine learning models on such data. In this work, we propose *dynamic neural graphs based federated learning framework* to address these challenges. The proposed framework extends *Reptile*, a model agnostic meta-learning (MAML) algorithm, to a federated setting. However, unlike the existing MAML algorithms, this paper proposes a dynamic variant of neural graph learning (NGL) to incorporate unlabelled examples in the supervised training setup. Dynamic NGL computes a meta-learning update by performing supervised learning on a labelled training example while performing metric learning on its labelled or unlabelled neighbourhood. This neighbourhood of a labelled example is established dynamically using local graphs built over the batches of training examples. Each local graph is constructed by comparing the similarity between embedding generated by the current state of the model. The introduction of metric learning on the neighbourhood makes this framework semi-supervised in nature. The experimental results on the publicly available MIMIC-III dataset highlight the effectiveness of the proposed framework for both single and multi-task settings under data decentralisation constraints and limited supervision.

Index Terms—federated learning, multi-task learning, semi-supervised learning

I. INTRODUCTION

With the advent of informatisation of medical institutions, there is an explosion in the availability of digitised healthcare data such as genetic data, electronic healthcare records (EHRs), and medical research data [1]. Machine learning (ML) models can be trained with this healthcare data to perform various tasks such as developing precision medicine, assisting medical practitioners in diagnosis and predicting physiological deterioration during critical care [2]. Generally, healthcare data is distributed among a network of symbiont institutions. The utilisation of all of this distributed data is essential to train a reliable and effective ML model. However, the digitised healthcare data often contain private information, and the leakage of this data may compromise the patients' privacy. To avoid such scenarios, strict standards such as the *General Data*

Protection Regulation (GDPR)¹ and the *Data Protection Act* (DPA)² are in place to restrict access to sensitive healthcare records. As a result, collecting the distributed healthcare data at a third-party-owned centralised location to train ML models is not always feasible. Hence, ML applications are required to strike an equilibrium between data privacy and data analysis.

A large amount of healthcare data is generated every day but obtaining labels for this rapidly generating data is either unfeasible or highly resource-intensive. Hence, the amount of available labelled training data is significantly less than the unlabelled data. As a result, the unsupervised and semi-supervised ML methods have garnered significant interest for many healthcare applications [3]–[6]. The unsupervised methods mostly include representation learning [7] and clustering [8] to identify the latent patterns, whereas semi-supervised methods incorporate both labelled and unlabelled data in the learning process. Previous studies have shown that semi-supervised methods help in learning more robust and generalised models when the labelled data is scarce [9], [10]. Thus, there is a requirement of new techniques that incorporate the unlabelled data in the training process to complement supervised learning.

In this work, we concentrate on EHRs and machine learning (ML) tasks associated with critical care. Generally, these EHRs reflect a common set of features that are essential to perform multiple critical care tasks [11]. For example, the same EHRs can be used to train ML models for mortality prediction, decompensation prediction and patient phenotyping. Hence, it makes sense to train a single model performing multiple tasks. By considering the nature of EHRs and the other challenges, we aim to propose a ML framework that has the following characteristics:

- **Training using the distributed data:** EHRs are sensitive and must not be shared with the other symbiont institutions (clients) or the third-party servers. The framework must be able to alleviate challenges associated with this decentralised nature of training data.
- **Semi-supervision:** The framework must be able to exploit the vast amount of unsupervised data to learn better models.
- **Multi-tasking:** The framework must be able to train models for multiple tasks simultaneously. In comparison to the mul-

Anshul Thakur, Pulkit Sharma and David A. Clifton are with Department of Engineering Science, University of Oxford, United Kingdom. (E-mail: anshul.thakur@eng.ox.ac.uk)

David A. Clifton is also with Oxford-Suzhou Centre for Advanced Research, Suzhou, China.

¹<https://gdpr-info.eu/>

²<https://www.gov.uk/data-protection/>

multiple single-task models, the concurrent training of multiple tasks leads to a reduction in client-server communication. The decreased client-server communication also lowers the chances of data-leakage or membership inference attacks.

To address the aforementioned aspects of healthcare applications, we propose a federated learning [2], [12] framework that is semi-supervised in nature and can be trained for multi-tasking. This work extends *Reptile* [13], a first-order model agnostic meta-learning (MAML) algorithm, to the federated setting such that deep learning models can be trained effectively on the distributed data. The security protocols such as secure multi-party computation [14] or differential privacy [15] can easily be incorporated in the proposed framework. Moreover, *Reptile* allows the proposed framework to train a shared global model targeting multiple similar tasks. The latent representations learned by the model for a particular task may also be relevant for other tasks. Hence, the shared or simultaneous training may help in improving generalisation across all tasks. Apart from that, the parameters of the trained global model provide an *informed parameter initialisation* for training models on new unseen but similar tasks. In comparison to the random initialisation, this informed initialisation may result in faster convergence and better performance (see Section V).

Reptile and other MAML algorithms [16] are designed for supervised few-shot learning, and primarily focus on obtaining a shared model that can be adapted to new tasks with a few gradient updates. However, in most of the healthcare application, a significant amount of data (though unlabelled) is available for training. As a result, few-shot learning and fast-adaptation to new similar tasks may not be a priority. Instead, we are interested in obtaining the shared model that can exploit the common meaningful representations across multiple tasks to improve performance. Hence, the proposed framework utilises *Reptile* to learn this common representation across multiple similar tasks. The faster adaptation to new tasks is just a beneficial side-effect of *Reptile*.

As discussed earlier, we are interested in obtaining a shared model while effectively utilising the unlabelled data to aid supervised learning across multiple tasks (in a federated setting). For this purpose, we propose a new variant of neural graph learning (NGL) [17] that is more suitable for federated or distributed data setting. NGL is a regularisation-based semi-supervised training mechanism that allows the neural networks to exploit the unlabelled examples for better training. In the existing formulation of NGL, the labelled examples are accompanied by their neighbours (labelled or unlabelled), defined by an input graph, to regularise the training process. Along with supervised training, NGL also tries to minimise the deviation between an example and its neighbours in an embedding space (see Section II for more details). This behaviour of minimising the variation among semantically similar examples is analogous to *metric learning* [18]–[20]. Hence, NGL enforces the same semantic meaning i.e. class label on the entire neighbourhood. This behaviour is useful if the neighbourhood examples are unlabelled as semantic meaning is propagated from labelled example to its

neighbouring examples. Though NGL is an effective semi-supervised learning method, it requires a synthesised graph as an input to the training algorithm. Generally, graphs are synthesised by measuring the similarity between embedding (of both labelled and unlabelled training examples) learnt using an unsupervised representation method such as auto-encoders (AE) [17]. Hence, training AE is an overhead and in case of decentralised data, a federated learning algorithm such as federated averaging [21] is required to train them. Besides the training of the main model (associated with the target tasks), training an AE further increases the threat of membership inference and other adversarial attacks by exposing more information (gradient updates or entire model states) over the client-server communication channels [22]. Hence, the requirement of input graphs or the training of representation models makes NGL undesirable in a decentralised or federated setup.

This work addresses the shortcomings of NGL by proposing its dynamic variant that does not require any synthetic input graph. The proposed variant dynamically creates a local graph over a batch of training examples. The embedding used to establish similarity between the input examples are obtained from an intermediate layer of the main model itself. Since the main model is guided by supervised learning, these embedding are semantically meaningful and can help in creating effective local graphs or neighbourhoods. The local graphs are created at each client, and this whole dynamic setup does not result in any increment in the client-server communication while reaping the benefits of NGL. More details about the proposed dynamic NGL are in Section III. The proposed framework employs dynamic NGL and hence, the unlabelled examples, in a task-specific manner to compute meta-updates to train the shared model at the server.

The major contributions of this paper are listed below:

- This paper introduces a new federated learning framework for semi-supervised multi-tasking. To the best of our knowledge, this is one of the few federated learning studies that directly targets the multi-tasking and exploits the unlabelled data to improve the classification performance.
- This paper introduces a dynamic variant of NGL that allows any federated learning framework to perform semi-supervised learning without compromising the data privacy of the clients. To the best of our knowledge, no other study has attempted to modify NGL in such a manner.
- This paper proposes to address the task heterogeneity associated with multi-tasking by utilising either the global or the local task-specific layers (see Section III for details).
- The experimental evaluation on the publicly available MIMIC-III dataset shows that the proposed dynamic NGL based federated framework exhibits either better or comparable performance against state-of-the-art semi-supervised federated learning frameworks.

The rest of this paper is organised as follows: Section II discusses the existing federated meta-learning and semi-

supervised federated learning frameworks. In Section III, we describe the proposed framework. Experimental setup and results are discussed in Section IV and V, respectively. Finally, Section VI concludes this paper.

II. RELATED STUDIES

In this section, first, we discuss the existing studies related to federated and semi-supervised federated learning frameworks. Then, we provide a brief description of NGL. Finally, we analyse the major differences between the proposed framework and the existing methods.

A. Federated learning

Recently, federated deep learning has been explored for many healthcare applications, such as diagnosing Parkinson’s disease [23], predicting adverse drug reactions [24] and early stroke prediction [25]. FL involves training a global ML model, using data distributed across different clients, in a decentralised manner. Most of the existing studies utilise federated averaging (*FedAvg*) algorithm [21] to train deep learning models in a distributed or federated setting. In *FedAvg*, during each round of training, the server asks the clients to train the local models using the corresponding local data. These local models are then averaged by the server to obtain a global model.

To address the statistical heterogeneity of the distributed data, Arivazhagan *et al.* [26] proposed to divide the model as the shared and the personalisation layers. The shared layers are global and are trained using *FedAvg*. On the contrary, personalisation layers are client-specific and are trained on the local data. This framework alleviates the requirement of explicit personalisation as the personalisation layers at each client make the trained models sensitive to the local data.

B. Federated meta-learning

MAML [16] and its first-order variants such as Reptile [13] can be interpreted as centralised case of *FedAvg* [27]. In MAML, task-specific models are used to compute meta-updates to train a shared global model. This behaviour of MAML is analogous to *FedAvg* as the local models in *FedAvg* are equivalent to the task-specific models of MAML. Hence, MAML algorithms are naturally suited for FL. Chen *et al.* [28] extended MAML to a federated setting to train a model over distributed data. The experimentation in this study showed that the *federated MAML* could achieve better performance than *FedAvg*. Apart from training the global model, meta-learning has also been used for user personalisation. Since MAML can adapt a model using a few gradient updates, it can be used for rapid personalisation of a global model. Jiang *et al.* [27] exhibited this behaviour by successfully personalising a *FedAvg* trained global model using *Reptile*.

C. Semi-supervised federated learning

Semi-supervised FL methods consider one of the following two scenarios [29], [30]:

- **Data at the server:** In this scenario, a few labelled examples are available at the third-party-owned server whereas the clients only have unlabelled examples. It is a common use-case of semi-supervised FL as it is unrealistic in most of the cases to expect a client to have the labelled data.
- **No data at the server:** In this scenario, the server has no data and is mainly associated with managing the training of the global model. On the other hand, both unlabelled and labelled examples are available at the clients. This use-case is mainly suited for healthcare application where the semi-supervised FL has to deal with the sensitive healthcare data such as EHRs. The access to this data is restricted by strict government guidelines and any data leakage makes the clients (medical institutions) liable to lawsuits. Hence, it is reasonable for the third-party-owned server to not have any access to the raw data. Here, the clients or medical institutions are expected to provide some labelled examples for the training. In this work, we are only interested in “no data at the server” and all further discussions are centred around this scenario.

In practice, any state-of-the-art semi-supervised method [31]–[33] can be used at clients for training the local models. In comparison to the supervised training, these methods can improve the performance of local models and hence, the global model. Two prominent semi-supervised methods i.e. Yalniz *et al.* [31] and Xie *et al.* [32] follow the student-teacher framework. Broadly, in these frameworks, the teacher is trained with supervised data and is used to generate pseudo-labels for the unlabelled examples. These pseudo-labels are then utilised by the student to aid the learning performed with the labelled examples. In addition to these approaches, a few studies have specifically exploited the federated setting to propose semi-supervised methods. One such prominent framework is proposed by Jeong *et al.* [30]. This method introduced inter-client consistency to generate pseudo-labels and to induce regularisation in the training process using the unlabelled examples. The inter-client consistency loss is a combination of cross-entropy loss (computed using pseudo-labels) and KL-divergence among outputs generated by different client models for an unlabelled example. The server stores the local models (sent by clients) and forwards a subset of these stored models (called helper models) to a client along with the global model parameters. The ensemble of these helper models is used to obtain pseudo-labels for each unlabelled example and enforcing the inter-client inconsistency.

D. Neural graph learning (NGL)

As discussed in Section I, NGL [17] is a regularisation-based semi-supervised mechanism for training deep learning models. NGL exploits the semantic relationship between the labelled and the unlabelled examples to regularise the training. Along with the training data, NGL also requires an input graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ capturing the semantic relationships among the training examples. Each training example (labelled or unlabelled), \mathbf{x} , is represented by a vertex, $v_x \in \mathcal{V}$, in $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The two vertices v_x and v_z , corresponding to examples \mathbf{x} and \mathbf{z} , are connected by a weighted undirected edge $e(v_x, v_z)$ if

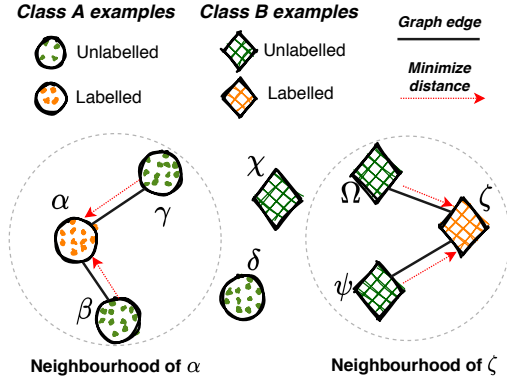


Fig. 1: An illustration of the metric learning performed by the second term of NGL loss function (equation 2). α and ζ are the labelled examples that act as anchors and NGL tries to minimize the deviation among their respective neighbourhoods.

these two examples are semantically similar. The weight, w_{xz} , on this edge represents the measure of similarity between the two nodes. The neighbourhood of a given vertex, v_x , is a set of all vertices connected to v_x in $\mathcal{G}(\mathcal{V}, \mathcal{E})$:

$$\mathcal{N}_x = \{v_z \in \mathcal{V} : e(v_x, v_z) \in \mathcal{E}\}. \quad (1)$$

Given a labelled example \mathbf{x} (with label y) and its neighbourhood \mathcal{N}_x defined in $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the NGL loss function can be defined as:

$$\mathcal{L}_{NGL}(\hat{y}, y, \mathcal{N}_x) = \mathcal{L}_{CE}(y, \hat{y}) + \alpha \sum_{\mathbf{z} \in \mathcal{N}_x} w_{xz} \mathcal{D}(f^l(\mathbf{x}), f^l(\mathbf{z})). \quad (2)$$

Here $\hat{y} = f(\mathbf{x})$ is the prediction generated by neural network $f(\cdot)$. \mathcal{L}_{CE} represents the cross-entropy loss function³ and $\mathcal{D}(\cdot)$ represents the Euclidean distance. $f^l(\mathbf{x})$ represents the hidden representation or embedding generated by the first l layers of $f(\cdot)$ for input \mathbf{x} . w_{xz} represents the weight on $e(v_x, v_z)$ and α is a user-defined scalar to decide weightage of the second term in the loss function. Note that the neighbourhood examples ($\mathbf{z} \in \mathcal{N}_x$) can either be labelled or unlabelled.

The NGL loss function, defined in equation 2, is a combination of both supervised loss (first term) and unsupervised loss (second term). By minimising the first term, NGL tries to assign the correct label to the labelled examples. On the other hand, the minimisation of the second term decreases the deviation among a labelled example and its neighbours in the embedding space. Fig. 1 illustrates this behaviour. Each labelled example acts as an anchor that attracts the neighbouring examples towards itself. Since the neighbourhood of an example is semantically similar to it, the second term of NGL loss function tries to minimise the distance among the semantically similar examples in an embedding space. This behaviour is analogous to *metric learning* [18]–[20] where neural networks are trained to learn an embedding space that is characterised by lower intra-class and higher inter-class variation. Hence, we can claim that second term of the NGL loss function is performing metric learning.

³ \mathcal{L}_{CE} can be replaced with any other loss function.

Since NGL loss function makes no assumption about the neighbourhood, it can be used to perform both supervised or semi-supervised learning. The second term enforces the same latent representation on the entire neighbourhood. The same representation may result in similar predictions by neural network $f(\cdot)$. These similar predictions support the hypothesis that the input and its neighbouring examples are highly likely to belong to the same class as they exhibit high similarity among themselves. This behaviour is of particular interest in the semi-supervised cases where the neighbourhood of a labelled example can contain the unlabelled examples. In supervised cases, the second term purely acts a data-dependent regularisation mechanism.

E. Comparison with the proposed framework

The proposed framework is similar to meta-learning algorithms (such as *Reptile* and *MAML*) and federated learning algorithms (such as *FedAvg* and *federated MAML*) as it exploits meta-learning for multi-tasking and federated learning for overcoming the data decentralisation. However, the proposed framework is different from the existing methods due to following:

- *federated MAML*, *Reptile* and *MAML* are completely supervised, whereas, the proposed framework can incorporate the unlabelled examples in the learning process.
- *federated MAML* and other federated learning methods are not designed or evaluated for multi-tasking. In contrast to these methods, the proposed framework directly targets the multi-task learning.

Federated MAML vs the proposed framework: The proposed framework utilises *Reptile* for meta-learning. In comparison to *MAML* [16], the classification performance of *Reptile* has been found to be less sensitive to batch sampling, batch size and the number of training iterations in some experimental setting [13]. As a result, in comparison to *federated MAML*, *Reptile* may allow the proposed framework to utilise more local training iterations at each client without worrying about batch sampling. More local iterations may result in decreased client-server communication by achieving near-optimum performance in fewer communication rounds.

NGL vs dynamic NGL: As discussed in Section I, NGL requires an input graph built over the training examples. On the other hand, this work proposes and utilise a dynamic variant of NGL in the proposed federated multi-tasking framework. This dynamic NGL doesn't require any explicit graph and it establishes the semantic relationships among examples during training dynamically. No existing study has targeted to overcome the requirement of the input graph in NGL. More details are in Section III.

III. DYNAMIC NGL BASED FEDERATED REPTILE

This section elaborates the proposed dynamic NGL based federated *Reptile* (*NGL-FedRep*) framework. Here, we first describe the problem statement. Then, we discuss the dynamic NGL. Finally, the overall framework is presented.

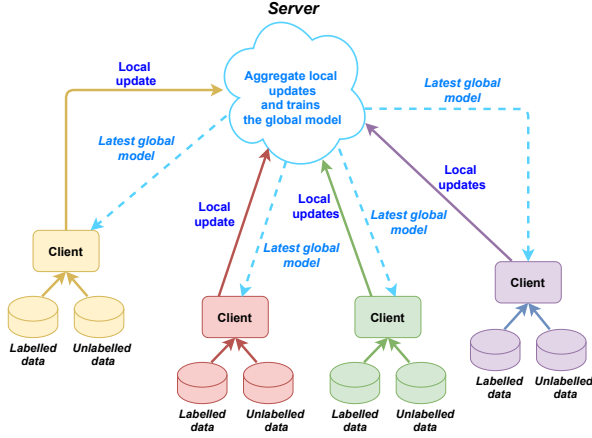


Fig. 2: An illustration of client-server interaction in the proposed framework.

A. Problem statement

The healthcare data is distributed among the multiple medical institutions that act as clients or nodes. At each client, the local data can be a mixture of both labelled and unlabelled examples. A client is not allowed to share the data with the other clients or with the server to preserve the data privacy. The aim is to process this distributed data to learn a shared global model at the server, which can perform either a single task or multiple related tasks. Fig. 2 illustrates the problem statement graphically.

Unlike the other common use-cases of FL where clients are the handheld devices [27], the medical institutions (in a symbiont network) are usually significantly fewer in number, less-prone to drop-off and have adequate computational resources.

B. Dynamic neural graph learning (NGL)

Dynamic NGL targets to overcome the requirement of an input graph in the existing formulation of NGL. The purpose of the input graph is to define the semantic relationships among the training examples and to exploit these relationships for regularisation or semi-supervision. Instead of utilising the input graph, the proposed dynamic NGL exploits the semantic or class-specific clustering observed during supervised training of deep learning models for establishing relationships among the training examples.

Supervised training of deep learning models results in semantic or class-specific clustering in an embedding space obtained after the models' penultimate layer (before softmax/classification layer). As the training progress, these clusters become more and more mutually exclusive. In the terminal stages of training, the within-class or within-cluster variation becomes negligible as all embedding of a particular class collapse to a point represented by their mean. This behaviour is known as *variability collapse* and has mathematically been proven by Pappayan *et al.* [34]. Dynamic variant of NGL exploits this semantic clustering as representation learning to create graphs. However, instead of using a single graph over all the training examples, dynamic NGL creates a graph for each

batch of training examples. Since the graph captures local relationships among examples of a batch, it is referred to as local graph.

For stochastic gradient descent (SGD) based training, the available training examples are sampled into batches having b labelled examples. If we are dealing with semi-supervision, each batch is also augmented with c unlabelled examples. Before each round or epoch of training, dynamic NGL generates a local graph for each batch as discussed below:

- Let $f^l()$ represents the first l layers of the model $f()$. The current state of $f^l()$ (or $f()$) is used to obtain embedding for all examples in the batch as: $\mathbf{E}_x = f^l(\mathbf{x})$ where \mathbf{x} is one of the examples in the batch.
- Compute cosine similarity between embedding of each pair of examples in the input batch:

$$C_{xy} = \frac{\mathbf{E}_x \cdot \mathbf{E}_y}{\|\mathbf{E}_x\| \times \|\mathbf{E}_y\|}. \quad (3)$$

- Cosine similarity between embedding pairs is exploited to create the local graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where each example in the batch forms a vertex. Two vertices v_x and v_y (corresponding to examples \mathbf{x} and \mathbf{y}) are connected by an undirected weighted edge $e(v_x, v_y)$ if cosine similarity C_{xy} between their embedding is greater than a pre-defined threshold τ . The cosine similarity C_{xy} is also regarded as weight w_{xy} on this edge.

Since we are creating a new local graph for a batch before each epoch, these graphs are regarded as dynamic in nature. As the training progresses, the embedding generated by $f^l()$ becomes more meaningful, and hence, the semantic structure captured by the local graphs also becomes more accurate. Once a local graph has been created, dynamic NGL and NGL are identical. Dynamic NGL exploits this local graph to compute NGL loss (as defined in equation 2) and gradient updates to train the model.

Note that we have to use the embedding generated by penultimate layer to witness variability collapse [34]. Hence, $f^l()$ is restricted to contain all but the last (softmax/classification) layer of $f()$.

C. Proposed framework: Dynamic NGL based FedRep

Algorithm 1 documents the proposed framework (NGL-FedRep). It has two components: server and client-side processing, as described below:

1) *Server-side processing*: The server is concerned with initialising the shared global model and selecting clients for training. During each round of training, the server forwards the latest parameters of the shared global model to the selected clients. Each client performs the local training and computes the parameter updates for the global model. These parameter updates are forwarded to the server where they are aggregated and applied on the shared global model.

2) *Client-processing*: For task t , each client k possesses a labelled dataset $\mathcal{D}_t^L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{|\mathcal{D}_t^L|}$ and an unlabelled dataset $\mathcal{D}_t^U = \{\mathbf{x}_i\}_{i=1}^{|\mathcal{D}_t^U|}$. During a round of training, the client k receives

Algorithm 1 Dynamic neural graph learning (NGL) based federated Reptile

```

1: // Run on server
2: Server-Exc( $N, \eta$ ):  $\triangleright \eta$ : learning rate,  $N$ : No. of rounds
3: Initialise the global model with parameter  $\theta_0$ 
4: for  $i \leftarrow 1 : N$  do
5:   Select a set of  $\mathcal{K}$  clients for training
6:   for each  $k \in \mathcal{K}$  do  $\triangleright$  Clients execute in parallel
7:      $\Phi_k \leftarrow \text{EXECUTE-CLIENT}(\theta_{i-1})$ 
8:      $\theta_i \leftarrow \theta_{i-1} + \eta \sum_{\forall k \in \mathcal{K}} (\Phi_k)$   $\triangleright$  Updating parameter
9: return  $\theta_N$ 
10:
11: // Run on clients
12: Execute-Client( $\theta$ ):
13:  $\mathcal{D}_t^L, \mathcal{D}_t^U$ : labelled and unlabelled datasets for task  $t$ 
14: Let  $f_\theta()$  be the model initialised with  $\theta$ 
15: for  $t \leftarrow 1 : T$  do  $\triangleright T$ : Number of tasks
16:    $W_t = \theta$ 
17:   Let  $f_t()$  be a model with  $W_t$  parameters
18:    $\mathcal{B} \leftarrow \text{SAMPLE-BATCHES}(\mathcal{D}_t^L, \mathcal{D}_t^U)$   $\triangleright$ 
    $|\mathcal{B}|$  batches, each with  $b$  labelled examples (having labels
    $l$ ) and  $c$  unlabelled examples
19:   for each  $(b, l, c) \in \mathcal{B}$  do
20:      $\mathcal{G}(\mathcal{V}, \mathcal{E}) \leftarrow \text{CREATE-LOCAL-GRAPH}(b, c, f_\theta())$   $\triangleright$ 
     As discussed in Section III-B
21:      $\mathcal{N}_b \leftarrow \text{Get-Neighbourhood}(b, \mathcal{G}(\mathcal{V}, \mathcal{E}))$   $\triangleright$ 
     Neighbourhood of each of the labelled  $b$  examples as
     defined in equation 1
22:      $L = \mathcal{L}_{\text{NGL}}(f_t(b), l, \mathcal{N}_b)$   $\triangleright$  NGL loss
23:      $W'_t = W_t - \beta(\nabla_{W_t} L)$   $\triangleright \beta$ : learning rate
24:      $\Phi = \frac{1}{T} \sum_{t=1}^T (W'_t - W_t)$   $\triangleright$  Meta update
25: return  $\Phi$ 

```

the global model from the server and performs the following operation:

- A local model is initialised with the parameters or weights (θ) of the global model; let, W_t represents the parameters of the model for task t .
- For each task t , a set of batches containing the labelled and unlabelled examples are sampled from \mathcal{D}_t^L and \mathcal{D}_t^U .
- A dynamic graph is created for each training batch.
- NGL loss is computed and gradient updates are obtained for updating W_t . The updated parameters are represented by W'_t .
- The updated task-specific models are used to compute the meta-updates for training the global model as:

$$\Phi = \frac{1}{T} \sum_{t=1}^T (W'_t - W_t). \quad (4)$$

Here T is the total number of tasks. In the end, Φ is forwarded to the server, where it is used to update the global model.

NGL based training updates the initial parameter (θ or W_t) of a task-specific model to W'_t by applying a series of gradient updates. Intuitively, we can represent this overall training as a single parameter update that is obtained using the gradient: $\nabla_{\theta_t} = \theta - W'_t$. We can aggregate all T task-specific gradients to obtain meta-gradient as:

$$\nabla_{\theta} = \frac{1}{T} \sum_{t=1}^T (\theta - W'_t) = \theta - \frac{1}{T} \sum_{t=1}^T (W'_t) \quad (5)$$

Hence, a meta-gradient appears to move the initial parameter (θ) in the direction of the average of the trained task-specific models. As a result, the framework is able to train over all the tasks simultaneously. A series of the meta-gradient based updates will move θ to a final configuration that is in proximity of the near-optimal parameters of each task-specific model. Since the global model parameters are near the optimal parameters of each task, only a few-gradient updates are required to obtain the optimal task-specific models.

D. Overcoming task heterogeneity

Task heterogeneity is a common problem in multi-tasking. The tasks that rely on similar latent representation may require a few task-specific layers. For example, in computer vision, scene classification and object detection may benefit from the same latent representations describing the objects. However, two CNNs performing these two tasks require different last layer. The proposed framework addresses such cases by considering models as a combination of the common layers and the task-specific layers. The parameters of the common layers and the task-specific layers (for task t) are represented as θ^c and θ^t , respectively. The common layers are global and are shared across all clients and tasks. These layers are trained globally at the server using meta-updates provided by clients. On the other hand, the task-specific layers (θ^t) can either be trained globally at the server using meta-updates provided by clients or locally at each client. Both these cases are discussed below.

- **Global task-specific layers:** This scenario is similar to Algorithm 1 and only differs in computation of meta-updates for the common layers (θ^c) and T task-specific layers (one for a separate task). The server initialises θ^c and $\{\theta^t\}_{t=1}^T$. During each round of training, the server forwards the latest parameters of all the layers to each client and receives separate meta-gradients for θ^c and $\{\theta^t\}_{t=1}^T$. The corresponding meta-gradients are aggregated and are applied to obtain the updated parameters for the common layers and T task-specific layers:

$$\theta^c = \theta^c + \eta \sum_{\forall k \in \mathcal{K}} (\Phi_k^c), \quad \theta^t = \theta^t + \eta \sum_{\forall k \in \mathcal{K}} (\Phi_k^t). \quad (6)$$

Here Φ_k^c and Φ_k^t represents the meta-gradient for θ^c and θ^t by client k . Also, \mathcal{K} is a set of the clients.

At client-side, a model $f_t()$ (for task t) is initialised with the common layers parameters θ^c and the corresponding task-specific layer parameters θ^t . The $f_t()$ is trained and the parameters are updated to $\theta^{c'}$ and $\theta^{t'}$. This model

initialisation and training is performed all T tasks. The updated parameters are used to compute the meta-gradients as:

$$\Phi^c = \frac{1}{T} \sum_{t=1}^T (\theta_t^{c'} - \theta^c), \quad \Phi^t = \theta^{t'} - \theta^t. \quad (7)$$

The common layers meta-gradient is computed by aggregating the changes in θ^c that are induced by each task-specific training. On the other hand, the meta-gradients for task-specific layers are only the function of the corresponding tasks.

- **Local task-specific layers:** In this mechanism, the task-specific layers are local and are trained at the clients only. The server is only concerned with the updating θ^c . It forwards the latest θ^c to each client and receives the meta-gradients Φ_k^c from each client. The θ^c is updated by aggregating meta-gradients as discussed in equation 6. At the client-side, the common layers of model $f_t()$ are initialised with θ^c provided by the server and the task-specific layers are initialised by the client itself. The client trains all the task-specific models and computes meta-updates Φ^c as discussed in equation 7. The meta-updates are forwarded to the server, while the updated local layer parameters are stored at the client for the next round of training. In the next round, these parameters are retrieved by the client to initialise the task-specific layers. Along with handling the difference in nature of tasks, the use of these local layers also help in personalising *NGL-FedRep* to the local data [26]. However, this mechanism is not able to fully utilise the data distributed across different clients.

IV. EXPERIMENTAL SETUP

In this section, we describe the dataset, model architectures and experiments used for the performance evaluation of the proposed framework.

A. Dataset

The performance of the proposed framework is evaluated on the publicly available MIMIC-III dataset [35]. It is a large database that contains information on patients admitted to critical care units at a tertiary care hospital. Data includes vital signs, medications, laboratory measurements, fluid balance, hospital length of stay, survival data, and more. As described in [11], this data is pre-processed to create sub-datasets for four different tasks⁴. In each sub-dataset, an example is an evenly spaced time-series where different clinical measurements resulting in 76 features are sampled at each time-step. In this work, we have specified a time-step of one hour. The four tasks are described below:

- **In-hospital mortality prediction:** This is a binary classification task that deals with predicting in-hospital mortality based on the first 48 hours of ICU stay. The corresponding sub-dataset contains 18,342 negative (no mortality) and 2,797 positive (mortality) examples.

- **Decompensation prediction:** This is also a binary classification task that deals with predicting whether the patient's health will deteriorate in the next 24 hours. The corresponding sub-dataset contains 3,360,926 negative (no deterioration observed) and 70,696 positive (deterioration observed) examples.
- **Phenotype classification:** This is a multi-label classification task that deals with identifying which acute care conditions such as *acute cerebrovascular disease* and *acute renal failure* are present in a given patient's ICU stay record. There are 25 different conditions considered for this task, and their details can be found in [11]. The phenotype classification sub-dataset contains 41,902 examples.
- **Length-of-stay prediction:** This task deals with predicting the remaining length of stay in ICU at each hour. The remaining length of stay is quantified into ten classes such as less than a day, one to seven days of the first week, over one week but less than two, and over two weeks. Hence, it is regarded as a multi-class classification problem. The length-of-stay prediction sub-dataset contains 3,451,346 examples that are distributed unevenly among 10 classes.

B. Experiments

We designed three experiments to evaluate different aspects of the proposed framework:

- **FL for single task scenarios:** The proposed framework (*NGL-FedRep*) is trained for the tasks of mortality prediction, decompensation prediction and phenotype classification separately. The performance of *NGL-FedRep* is compared against widely used federated averaging (*FedAvg*) [21] algorithm. Apart from that, the performance of both *NGL-FedRep* and *FedAvg* is compared to the centralised neural network training. The main purpose of this experiment is to analyse the impact of decentralisation of training data on classification performance.
- **FL for Multi-tasking:** *NGL-FedRep* is trained for tasks of mortality prediction, decompensation prediction and phenotype classification simultaneously. The performance of each task is compared to the performance of the task-specific centralised neural networks. The performance of local and global mechanisms to train the task-specific layers is also compared. The parameters of the trained common layers are further used for initialising the length-of-stay prediction model. This model is trained in a centralised setup, and its performance is compared to a randomly initialised model. Note that in both scenarios, the task-specific layer is randomly initialised.
- **Semi-supervised FL and semi-supervised multi-task FL:** The datasets considered in this study are entirely labelled. Hence, to highlight the impact of dynamic NGL on the proposed framework, we only consider 10%, 25% and 50% of the available training examples as labelled, and the rest of examples are regarded as unlabelled. The performance of *NGL-FedRep* is compared to a non-graph version of the proposed framework (*FedRep*). This version only utilises the available labelled data, and act as a baseline for supervised training in data-scarce scenarios. Moreover, the

⁴Benchmarking code available at <https://github.com/YerevaNN/mimic3-benchmarks> is used.

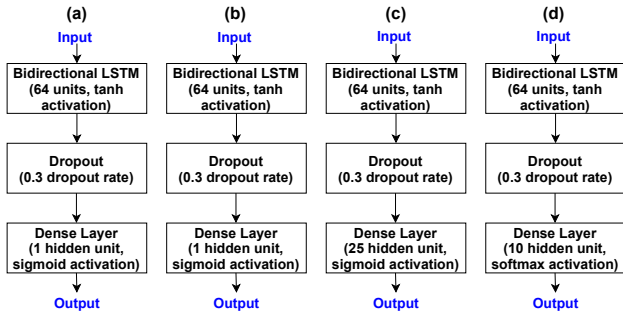


Fig. 3: Model architectures used for the tasks of (a) mortality prediction, (b) decompensation prediction, (c) phenotype classification and (d) length of stay prediction.

performance of *NGL-FedRep* is compared with off-the-shelf semi-supervised methods such as Yalniz *et al.* [31] and Xie *et al.* [32]. These methods are incorporated in the proposed framework (instead of dynamic NGL) to train the local models at each client. Apart from that, the semi-supervised FL method proposed by Jeong *et al.* [30] (see Section II for details) is also used as a comparative method. The performance of baseline methods is compared against the proposed framework in both single and multi-task scenarios. Note that *NGL-FedRep* with global layers (not the local layers) is used in this experiment.

C. Data distribution, models and parameter setting

Data distribution: For each task, the available data is distributed among 20 clients in a non-IID manner. At each client, 10% of the data is used for validation. On the remaining data, five-fold cross-validation is used to create five train-test datasets. In the *semi-supervised experiment*, only 10%, 25% and 50% of the total training examples at each client are considered as labelled. The testing is performed at a client and the predictions on all the test examples (across all the clients) are used to compute the performance metrics.

Models: The model architectures for each task are almost similar, and they only differ in the last dense layer. Fig. 3 illustrates the model architectures used in this study. The binary cross-entropy is used as the supervised loss function (first term in equation 2) for in-hospital mortality prediction, decompensation prediction and phenotype classification. Similarly, for length-of-stay prediction, categorical cross-entropy is used as the loss function. In multi-task scenario, LSTM layer is regarded as the common layer (θ^c) and is shared across all the tasks. On the other hand, the last dense layers of each model are regarded as the task-specific layers (θ^t).

Parameter setting in *NGL-FedRep*: During each round of training, all 20 clients are selected for obtaining meta-updates. A fixed step-size, $\eta=0.15$, is used to update the global parameters. At a client, the task-specific models are trained on every available example in each round. The training data is presented in batches of 16 examples. In the case of supervised training, the neighbourhood examples defined by the dynamic graphs are always labelled. In semi-supervised experimentation, each batch contains 8 labelled and 8 unlabelled examples. For

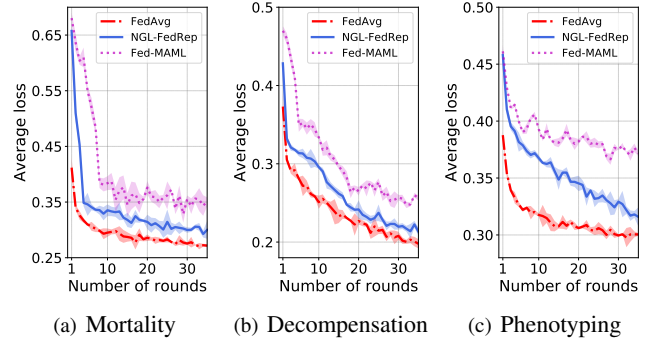


Fig. 4: Average loss observed across all clients during training of *NGL-FedRep*, *FedAvg* and *federated MAML* under single task scenario. The variance represents the average training loss variations observed across all folds.

batch creation, the available labelled examples are stratified into batches of 8 examples. The 8 unlabelled examples are randomly sampled from all the available unlabelled data (with replacement) and are appended to each batch of the 8 labelled examples. The parameter $\alpha = 0.2$ and embedding generated after LSTM layer are used in dynamic graph creation and the second term of the NGL loss function (equation 2). For dynamic graphs, a fixed threshold of $\tau = 0.9$ is used on cosine similarity to create edges between two examples. At each client, Adam optimiser with a fixed learning rate of $\beta = 0.001$ is used to train all the local models. All these values are fine-tuned to provide maximum performance on the validation examples.

Parameter details in comparative methods: The parameter setting to train the local models such as optimizer and learning rate (discussed above) are the same in all the comparative methods. In Yalniz *et al.* [31], the same task-specific model architectures are used as the teacher and the student models. To implement Xie *et al.* [32], we used the model architectures shown in Fig. 3 as the noisy student. However, to implement the teacher models, the dropout layers were removed from the architectures. In Jeong *et al.* [30], we used three helper models to implement inter-client consistency. All the parameters used in these comparative studies are also fine-tuned on the validation examples.

V. EXPERIMENTAL RESULTS & DISCUSSION

In this section, we present and discuss results obtained during single and multi-task experimentation. We also compare the performance of the proposed method with different comparative methods.

A. FL for single task scenarios

Fig. 4 depicts the average loss (across all clients) observed during training of the proposed framework (*NGL-FedRep*) and the existing FL methods. This figure highlights that the average loss decreases during training. Along with the existing FL methods, *NGL-FedRep* is also effectively able to train on the

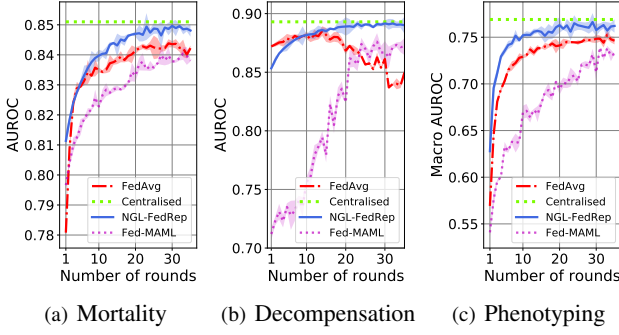


Fig. 5: Performance of *NGL-FedRep* trained models in single task scenario. The average performance (across five-folds) during each round of training is presented here.

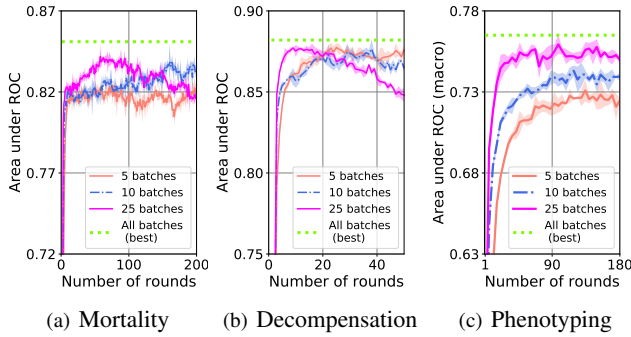


Fig. 6: Effect of decreasing the number of batches for updating local models at clients during each round of training.

distributed data. Fig. 5 illustrates the performance of *NGL-FedRep* and other methods on the test examples. Following inference can be drawn from the analysis of this figure:

- The best classification scores exhibited by *NGL-FedRep* are comparable to the performance of centralised neural networks on all three tasks. Along with overcoming the constraints of data distribution, *NGL-FedRep* also performs better than *FedAvg* and *federated MAML* on all three tasks.
- During later rounds of training, *FedAvg* shows over-fitting for the decompensation prediction. In contrast, *NGL-FedRep* does not exhibit such behaviour. This can be attributed to the regularisation imposed by dynamic neural graph learning in *NGL-FedRep*.
- The performance of *FedAvg* and *federated MAML* is comparable for mortality and decompensation prediction tasks. However, *federated MAML* failed to converge properly for the task of phenotyping (see training loss curve depicted in Fig. 4 (c)). As discussed in Section II, the performance of MAML and hence, *federated MAML*, is highly sensitive towards batch sampling, batch sizes and number of batches used for updating the local models in each round of training [13]. The slow or non-convergence of *federated MAML* could be due to any of these factors.

Effect of the number of batches: As discussed earlier, at each client, we use all the available local batches to train a local model during each round of training. This is in contrast to

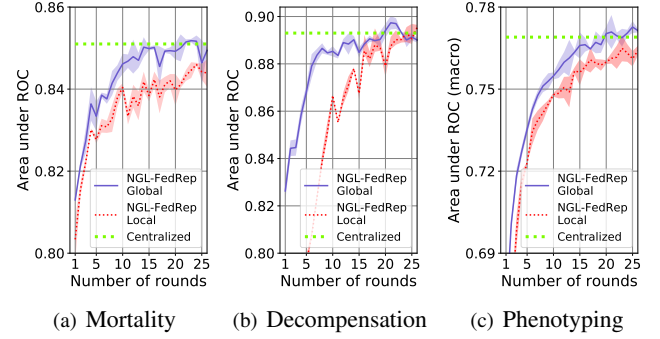


Fig. 7: Performance of *NGL-FedRep* in multi-tasking scenario. The average performance across five-folds is presented here.

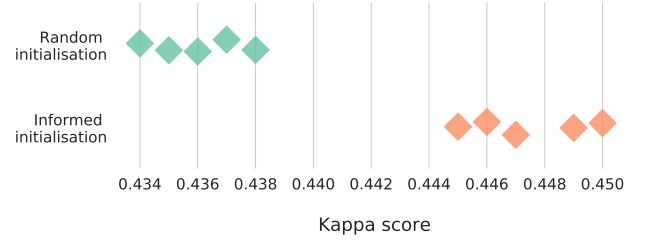


Fig. 8: Effect of initialisation schemes on the length-of-stay prediction. For each scheme, five Kappa scores represent the performance across five folds.

meta-learning algorithms where only a few batches are used for training the local models at a time.

To analyse the impact of the number of batches, we used 5, 10 and 20 batches for training the local models. Fig. 6 documents the performance of *NGL-FedRep* as a function of the number of batches. The analysis of this figure makes it clear that a large number of batches used for the local training results in better performance. Moreover, by comparing Fig. 5 and Fig. 6, it is clear that using all of the available local batches leads to near-optimum performance in only a few rounds of training. As discussed in Section II, this reduces the required number of interactions between clients and the server.

B. FL for Multi-tasking

Fig. 7 shows the performance of *NGL-FedRep* trained for mortality prediction, decompensation prediction and phenotyping simultaneously. Analysis of this figure highlights that *NGL-FedRep* with global task-specific layers (*NGL-FedRep Global*) performs comparably to the centralised task-specific neural networks on all three tasks. On the other hand, *NGL-FedRep* with local or personalised task-specific layers (*NGL-FedRep Local*) exhibits comparable performance for the task of decompensation prediction but shows lower classification scores than *NGL-FedRep Global* for mortality prediction and phenotyping. At each client, the available training data for mortality prediction and phenotyping is significantly less than the data available for decompensation prediction. This may have hindered the effective training of the task-specific layers in *NGL-FedRep Local*.

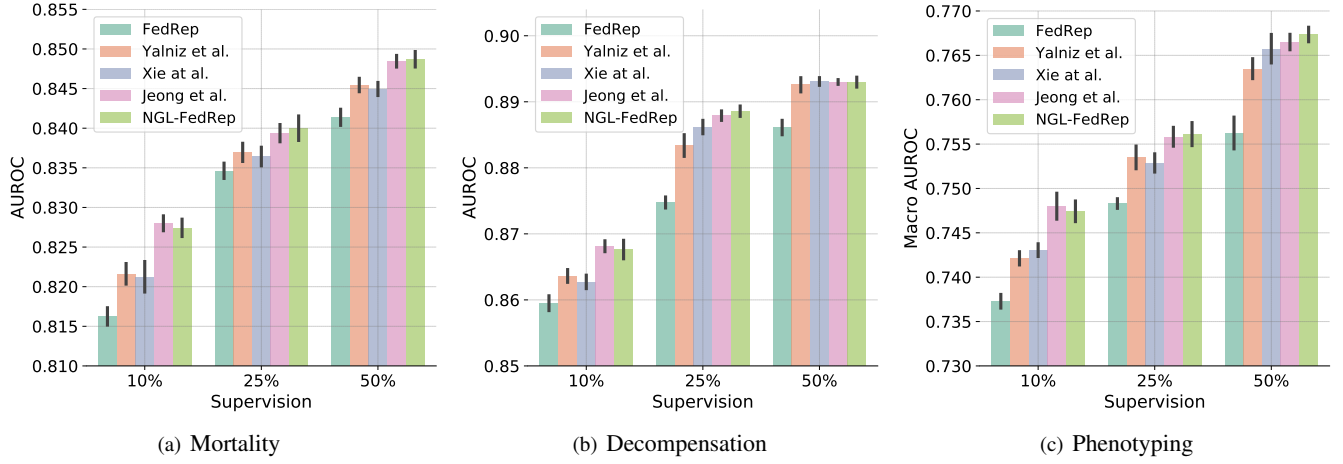


Fig. 9: Performance of *NGL-FedRep* and other comparative methods in single task semi-supervised federated setting. The error bars represent the confidence interval of scores obtained across five-folds.

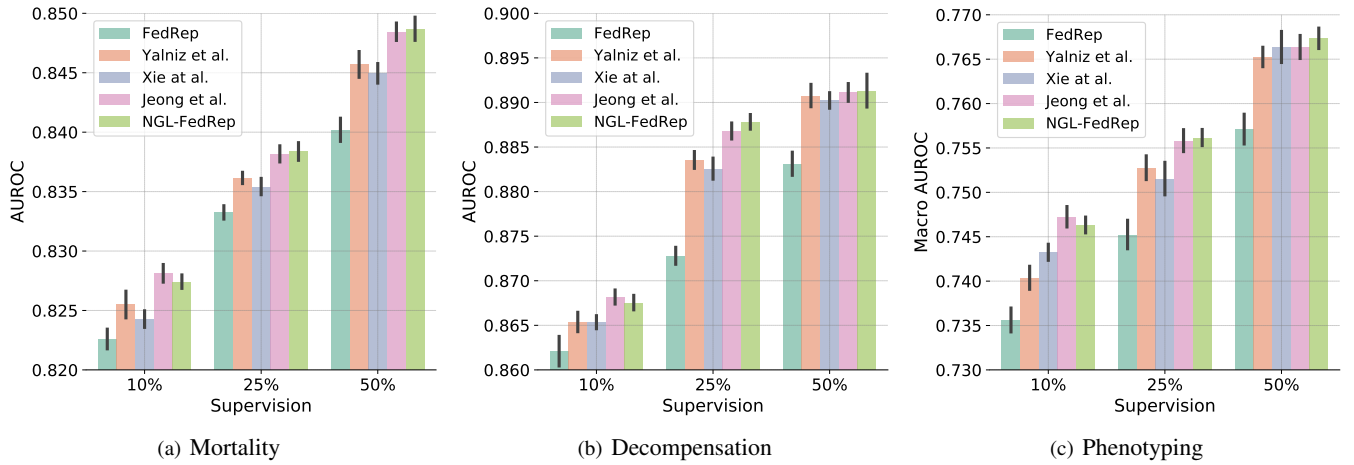


Fig. 10: Performance of *NGL-FedRep* and other comparative methods in multi-task semi-supervised federated setting. The error bars represent the confidence interval of scores obtained across five-folds.

Fig. 8 depicts the performance for the task of length-of-stay prediction. The model is trained using dynamic NGL in a centralised setup. Before training, the global layer i.e. LSTM layer of the model (Fig. 3) is initialised using parameters obtained from the trained *NGL-FedRep Global*. This initialisation is referred to as “informed intialisation” because the trained parameters encapsulate the useful latent representations learned from similar tasks. The analysis of Fig. 8 shows that informed initialisation results in better performance than the random initialisation.

C. Semi-supervised FL: single and multi-tasking scenarios

Fig. 9 and Fig. 10 illustrate the bar-plots signifying the performance of different comparative methods at different levels of supervision in single and multi-tasking scenarios, respectively. Following inference can be drawn from the analysis of these figures:

- The performance of *NGL-FedRep* and other comparative

methods is comparable in both single and multi-task scenarios. This justifies the utilisation of multi-tasking for EHR-based healthcare tasks. The similarities between these tasks allows us to train task-specific models simultaneously.

- In all cases, the semi-supervised methods show improvement over the supervised framework (*FedRep*). In particular, *NGL-FedRep* outperform *FedRep* by a noticeable margin in all cases. This indeed highlights that the proposed dynamic NGL can exploit unlabelled examples to improve the supervised training.
- At 10% and 25% supervision, *NGL-FedRep* and Jeong *et al.* [30] show a noticeable improvement over Yalniz *et al.* [31] and Xie *et al.* [32] in the classification performance. However, at 50% supervision, their performance is comparable to *NGL-FedRep* and Jeong *et al.* [30].
- The performance of *NGL-FedRep* and Jeong *et al.* [30] is comparable in most cases. However, Jeong *et al.* [30] exhibit slightly better performance than *NGL-FedRep* at

TABLE I: Comparison of the client-server interactions in *NGL-FedRep* against state-of-the-art in each round of training. An interaction is defined as a single transfer of the weight updates or the model parameters between a client and the server.

Method	# Interactions		Total data transferred (MB)	
	Server to a client	Client to the server	Server to a client	Client to the server
Jeong <i>et al.</i> [30]	4	1	21.856	5.464
NGL-FedRep	1	1	5.464	5.464

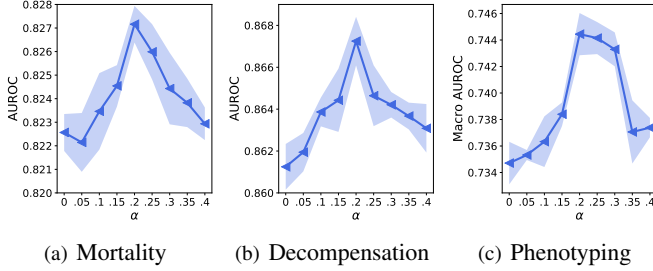


Fig. 11: Effect of α on the performance of *NGL-FedRep* in semi-supervised multi-task setting.

10% supervision. *NGL-FedRep* relies on the semantics of the embedding to create meaningful dynamic graphs. At very low supervision, these semantics and graphs may not be as meaningful as at the higher level of supervision. This may have resulted in a slight decrement in the performance of *NGL-FedRep* at 10% supervision.

- Although performance of Jeong *et al.* [30] and *NGL-FedRep* is comparable, the client-server communication required in Jeong *et al.* [30] is greater than *NGL-FedRep*. Table I tabulates the client-server communication required in both methods during multi-tasking. In Jeong *et al.* [30], the server transfers a set of 3 helper models (along with the aggregated model updates) to the clients for forcing inter-client consistency. This leads to the following disadvantages:
 - Larger client-server interactions result in an increased training time, specially if we are training a larger model. This problem can exaggerate itself if the network is too slow or the number of clients is very large [36].
 - The helper models are client-specific models. When these models are transferred to other clients, it undermines the privacy setting. The other clients can exploit these client-specific helper models to extract sensitive patient information using the in-membership attacks [37].

D. Ablation studies: impact of τ and α in NGL

The semi-supervision is infused in the proposed framework by dynamic NGL. The structure of local graphs and the impact of the unlabelled examples on the training is greatly influenced by both τ and α . In dynamic NGL, τ is used as threshold over cosine similarity between two embedding to connect these

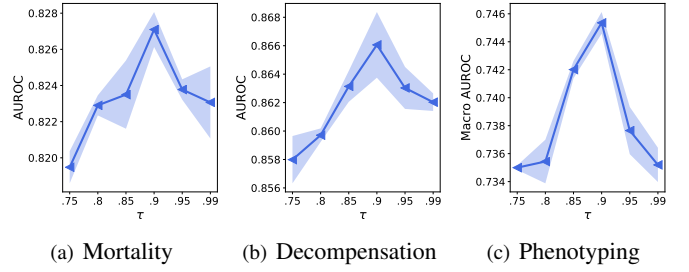


Fig. 12: Effect of τ on the performance of *NGL-FedRep* in semi-supervised multi-task setting.

examples by an edge in the local graph (see Section III-B). On the other hand, α scales the contribution of the second term in the NGL loss function (equation 2). In this subsection, we analyse the impact of these parameters on the performance of *NGL-FedRep* in semi-supervised multi-task setting. For simplicity, we only consider the cases with 10% supervision. Apart from the target parameter (i.e. either α or τ), the same parameter setting (discussed in Section IV) is also used in this analysis.

We vary α from 0 to 0.4 and analyse the variation in performance. Fig. 11 illustrates the results of this experiment. The analysis of this figure highlights that on average, the best classification scores are obtained at $\alpha = 0.2$. At lower or higher values of α , a decline in performance is observed. As α is increased, the contribution of second term or the unlabelled examples increases in the loss function. A larger contribution may have undermined the supervised signal (or the contribution of the first term) and resulted in lower performance. On the other hand, a very small α leads to little or no contribution from the unlabelled examples in the training process.

Similarly, we vary τ from 0.75 to 0.99 and analyse the performance of *NGL-FedRep*. Fig. 12 illustrates the results of this experiment. The average performance across all tasks peaks at $\tau = 0.9$. The increment or decrement in τ around 0.9 exhibits a drop in performance. If we use a very high value of τ such as 0.99, the resultant local graphs may not have enough edges. As a result, the number of examples in the neighbourhood of any labelled example would be very less and the framework wouldn't be able to exploit the unlabelled examples in the training. In contrast, a lower value of τ such as 0.75 results in too many edges in the local graph. This may result in semantically less similar examples in the neighbourhood of a labelled example. Both these cases lead to lower classification performance.

VI. CONCLUSIONS

In this paper, we presented a federated model agnostic meta-learning framework for multi-tasking in healthcare applications. We exploited meta-learning to learn a common representation across three different critical care tasks to perform effective multi-tasking. To perform semi-supervision, we proposed a new dynamic variant of neural graph learning (NGL) that does not require any input graph and can effectively

utilise the unlabelled data to aid the supervised learning. The experimental results on MIMIC-III showed that the proposed framework is capable of overcoming the constraints imposed by data decentralisation and limited supervision to exhibit a respectable classification performance. Future work may involve incorporating the privacy-preserving mechanisms such as secure multi-party computation and differential privacy in the proposed framework.

ACKNOWLEDGMENT

This work was supported by the Wellcome Trust under grant 217650/Z/19/Z, the National Institute for Health Research (NIHR) Oxford Biomedical Research Centre (BRC) and the NVIDIA academic hardware grant program. The views expressed are those of the authors and not necessarily those of the NHS, the NIHR or the Department of Health.

REFERENCES

- [1] P. Groves, B. Kayyali, D. Knott, and S. V. Kuiken, "The 'big data' revolution in healthcare: Accelerating value and innovation," 2016. [Online]. Available: repositorio.colciencias.gov.co/handle/11146/465
- [2] J. Xu and F. Wang, "Federated learning for healthcare informatics," *arXiv preprint arXiv:1911.06270*, 2019.
- [3] F. Ma, C. Meng, H. Xiao, Q. Li, J. Gao, L. Su, and A. Zhang, "Unsupervised discovery of drug side-effects from heterogeneous data sources," in *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 967–976.
- [4] C. Liu, F. Wang, J. Hu, and H. Xiong, "Temporal phenotyping from longitudinal electronic health records: A graph based framework," in *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 705–714.
- [5] C. Yuan, Y. Wang, N. Shang, Z. Li, R. Zhao, and C. Weng, "A graph-based method for reconstructing entities from coordination ellipsis in medical text," *Journal of the American Medical Informatics Association*, 2020. [Online]. Available: <https://doi.org/10.1093/jamia/ocaa109>
- [6] M. Liu, M. Zhou, T. Zhang, and N. Xiong, "Semi-supervised learning quantization algorithm with deep features for motor imagery eeg recognition in smart healthcare application," *Applied Soft Computing*, vol. 89, p. 106071, 2020.
- [7] Z. Che, D. Kale, W. Li, M. T. Bahadori, and Y. Liu, "Deep computational phenotyping," in *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 507–516.
- [8] P. Delias, M. Doumpos, E. Grigoroudis, P. Manolitzas, and N. Matsatsinis, "Supporting healthcare management decisions via robust clustering of event logs," *Knowledge-Based Systems*, vol. 84, pp. 203–213, 2015.
- [9] P. Zhang, F. Wang, J. Hu, and R. Sorrentino, "Label propagation prediction of drug-drug interactions based on clinical side effects," *Scientific reports*, vol. 5, no. 1, pp. 1–10, 2015.
- [10] Z. Wang, A. D. Shah, A. R. Tate, S. Denaxas, J. Shawe-Taylor, and H. Hemingway, "Extracting diagnoses and investigation results from unstructured text in electronic health records by semi-supervised machine learning," *PLoS One*, vol. 7, no. 1, p. e30412, 2012.
- [11] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. V. Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *Scientific Data*, vol. 6, no. 96, pp. 1–18, 2019.
- [12] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International Journal of Medical Informatics*, vol. 112, pp. 59–67, 2018.
- [13] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.
- [14] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.
- [15] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques*, 2006, pp. 486–503.
- [16] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of International Conference on Machine Learning*, 2017.
- [17] T. D. Bui, S. Ravi, and V. Ramavajjala, "Neural graph learning: Training neural networks using graphs," in *Proceedings of International Conference on Web Search and Data Mining*, 2018, pp. 64–71.
- [18] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2014, pp. 34–39.
- [19] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
- [20] I. Melekhov, J. Kannala, and E. Rahtu, "Siamese network features for image matching," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2016, pp. 378–383.
- [21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.
- [22] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *USENIX Security Symposium*, 2020, pp. 1605–1622.
- [23] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, 2020.
- [24] O. Choudhury, Y. Park, T. Salonidis, A. Gkoulalas-Divanis, I. Sylla *et al.*, "Predicting adverse drug reactions on distributed health data using federated learning," in *Proceedings of AMIA Annual Symposium*, 2019, pp. 313–322.
- [25] C. Ju, R. Zhao, J. Sun, X. Wei, B. Zhao, Y. Liu, H. Li, T. Chen, X. Zhang, D. Gao *et al.*, "Privacy-preserving technology to help millions of people: Federated prediction model for stroke prevention," *arXiv preprint arXiv:2006.10517*, 2020.
- [26] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.
- [27] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.
- [28] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," *arXiv preprint arXiv:1802.07876*, 2018.
- [29] Z. Zhang, Z. Yao, Y. Yang, Y. Yan, J. E. Gonzalez, and M. W. Mahoney, "Benchmarking semi-supervised federated learning," *arXiv preprint arXiv:2008.11364*, 2020.
- [30] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, "Federated semi-supervised learning with inter-client consistency," *arXiv preprint arXiv:2006.12097*, 2020.
- [31] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, "Billion-scale semi-supervised learning for image classification," *arXiv preprint arXiv:1905.00546*, 2019.
- [32] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.
- [33] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation for consistency training," *arXiv preprint arXiv:1904.12848*, 2019.
- [34] V. Pappas, X. Han, and D. L. Donoho, "Prevalence of neural collapse during the terminal phase of deep learning training," *Proceedings of the National Academy of Sciences (PNAS)*, 2020.
- [35] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-Wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-III, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.
- [36] M. M. Amiri, D. Gunduz, S. R. Kulkarni, and H. V. Poor, "Federated learning with quantized global model updates," *arXiv preprint arXiv:2006.10672*, 2020.
- [37] L. Lyu, H. Yu, X. Ma, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," *arXiv preprint arXiv:2012.06337*, 2020.