Incremental Trainable Parameter Selection in Deep Neural Networks

Anshul Thakur, Vinayak Abrol, Pulkit Sharma, Tingting Zhu, and David A. Clifton

***Abstract*— This article explores the utilization of the effective degree-of-freedom (DoF) of a deep learning model to regularize its stochastic gradient descent (SGD)-based training. The effective DoF of a deep learning model is defined only by a subset of its total parameters. This subset is highly responsive or sensitive toward the training loss, and its cardinality can be used to govern the effective DoF of a model during training. To this aim, the incremental trainable parameter selection (ITPS) algorithm is introduced in this article. The proposed ITPS algorithm acts as a wrapper over SGD and incrementally selects the parameters for updation that exhibit the maximum sensitivity toward the training loss. Hence, it gradually increases the DoF of the model during training. In ideal cases, the proposed algorithm arrives at a model configuration (i.e., DoF) optimum for the task at hand. This whole process results in a regularization-like behavior induced by a gradual increment of the DoF. Since the selection and updation of parameters is a function of the training loss, the proposed algorithm can be seen as a task and data-dependent regularization mechanism. This article exhibits the general utility of ITPS by evaluating it on various prominent neural network architectures such as CNNs, transformers, recurrent neural networks (RNNs), and multilayer perceptrons. These models are trained for image classification and healthcare tasks using the publicly available CIFAR-10, SLT-10, and MIMIC-III datasets.**

***Index Terms*— Healthcare informatics, incremental training, regularization.**

I. Introduction

SUPERVISED deep learning models have shown remarkable performance across multiple domains such as computer vision, speech recognition, acoustic signal classification, and healthcare informatics [1], [2]. The first challenge in designing deep learning pipelines for any task is choosing an appropriate model architecture. Generally, these architectures are developed by human experts and are often inspired by architectures previously used for similar tasks. Each architecture is capable of modeling a family of different data-generating distributions. The final configuration (i.e., trained weights) of the architecture defines how well it fits the data-generating distribution corresponding to the task at hand. It is a general practice to use an over-parameterized or deeper neural network architecture with strong representation power that can effectively model the data-generating distribution [3], [4]. However, the larger or deeper neural networks are often prone to overfitting and require mechanisms to regularize the training of deep learning models [5].

The overwhelming success of deep learning can partly be attributed to the regularization mechanisms that mitigate overfitting and result in models exhibiting effective generalization [3]. Recently, a plethora of regularization methods has been developed for deep learning models. Some of the prominent regularization methods include early stopping [6], weight penalties such as ℓ_1 or ℓ_2 regularization [7], dropout [8] and its variants such as concrete and guided dropout [9], [10], penalizing low entropy output distribution [11], noise injections to outputs of hidden units [3], and Jacobian regularization [12]. Most of these regularization mechanisms restrict the expressive power of a deep learning model during training in a somewhat static manner. For example, a dropout rate of 0.5 always drops 50% of the hidden units, and a weight decay of 0.001 always penalizes the weights by a constant factor. These regularization mechanisms are task-independent and do not consider the complexity of the task in restricting a model's expressiveness. Apart from that, the regularization level needed to achieve effective generalization can also be seen as a function of the number of available training examples. In contrast to the sufficient training data, the lower number of training examples often induce more sampling noise that results in a model providing a poorer fit of data-generating distribution and leads to overfitting [8]. Hence, there is a requirement for regularization mechanisms that can adapt themselves based on the number of training examples and the task complexity to provide effective generalization.

In this article, we introduce the incremental parameter selection (ITPS) algorithm that regularizes the stochastic gradient descent (SGD) based training of deep learning models[1] to mitigate some of the aforementioned drawbacks of the

Manuscript received 25 July 2021; revised 15 March 2022 and 5 July 2022; accepted 18 September 2022. This work was supported in part by the National Institute for Health Research (NIHR) Oxford Biomedical Research Centre (BRC), and in part by an InnoHK Project at the Hong Kong Centre for Cerebro-cardiovascular Health Engineering (COCHE). The work of Anshul Thakur was supported by an EPSRC Healthcare Technologies Challenge Award (EP/N020774/1). The work of Tingting Zhu was supported by the Engineering for Development Research Fellowship provided by the Royal Academy of Engineering. The views expressed are those of the authors and not necessarily those of the NHS, the NIHR, the Department of Health, InnoHK – ITC, or the University of Oxford. *(Corresponding author: Anshul Thakur.)*

Anshul Thakur, Pulkit Sharma, and Tingting Zhu are with the Department of Engineering Science, University of Oxford, OX3 7DQ Oxford, U.K. (e-mail: anshul.thakur@eng.ox.ac.uk).

Vinayak Abrol is with the Infosys Centre for Artificial Intelligence, IIT Delhi, New Delhi 110020, India.

David A. Clifton is with the Department of Engineering Science, University of Oxford, OX3 7DQ Oxford, U.K., and also with the Oxford-Suzhou Centre for Advanced Research, Suzhou 215123, China.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2022.3210297.

Digital Object Identifier 10.1109/TNNLS.2022.3210297

[1]The proposed algorithm can be used for the classification model that utilizes SGD-based training. However, in this work, we are only interested in deep learning models.

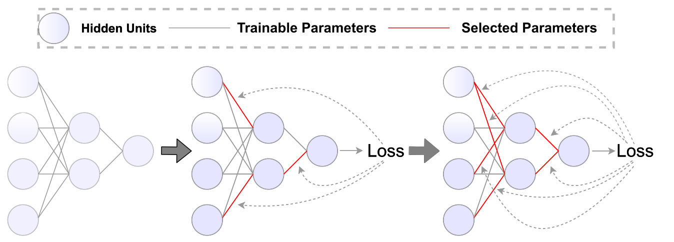


Fig. 1. Graphical illustration of parameter selection/updation in the proposed ITPS algorithm. As the training progress, more parameters are selected and the effective DoF of the model is increased.

existing regularization mechanisms. The proposed algorithm relies on governing the effective degree-of-freedom (DoF) of the model during training to induce task and data-based regularization. The effective DoF of a deep learning model is often lesser than its total trainable parameters [13]. This implies that only a subset of parameters is responsible for defining the effective DoF, and these parameters are likely to be sensitive toward the training loss. The proposed algorithm exploits these parameters and the training loss to regulate the DoF during SGD-based training. It gradually selects and updates a subset of the available trainable parameters while keeping the nonselected parameters constant. Hence, during an iteration, the proposed algorithm minimizes the training loss by updating only a subset of the trainable parameters. The proposed algorithm analyzes the sensitivity of parameters toward the loss throughout training and selects parameters that exhibit the maximum sensitivity. As the training progresses, the algorithm gradually selects more trainable parameters to increase the DoF of the model. This behavior is illustrated in Fig. 1. Since training loss depends on the nature of the task and the training data, the modulation of the effective DoF and regularization induced by this modulation is also dependent on these factors.

The regularization achieved by the proposed ITPS algorithm can be explained using the DoF and implicit gradient regularization (IGR) [14]. The incremental parameter selection influences both these phenomena, and they interact in synergy to regularize the entire training bout. The lower DoF restricts the model's expressiveness and is mainly responsible for regularization in the initial training stages. On the other hand, IGR is primarily accountable for regularization in the latter stages. IGR states that the gradient descent inherently penalizes the larger gradient updates [14]. With the DoF increment, the norm of gradient updates is also increased as more parameters are involved in the updation process. Hence, IGR induces more regularization in the latter stages when the DoF is higher. More details about this behavior are presented in Section III.

The major contributions of this article are listed as follows.

1) This article introduces the incremental trainable parameter selection (ITPS) algorithm, a data and task-dependent mechanism that regularizes the SGD-based training of deep learning models by modulating their DoF.
2) This article extends the proposed algorithm to model-agnostic meta-learning frameworks.
3) This article proves the general utility of the proposed ITPS algorithm by training various model architectures such as multiple layer perceptrons (MLP), CNNs, transformers, and recurrent neural networks (RNNs) for tasks across image and healthcare domains.

The rest of this article is organized as follows. Section II discusses the prominent regularization methods that greatly influenced the proposed algorithm. In Section III, we describe the proposed algorithm and discuss the nature of regularization it induces in SGD-based training. Apart from that, we also perform the Hessian spectral analysis to get more insights into the learning dynamics of the proposed algorithm. Experimental setup and results are discussed in Section V and VI, respectively. Finally, Section VII concludes this article.

II. Earlier Studies

As discussed in Section I, many regularization methods have been proposed for deep learning models. Dropout [8], [15] is the most commonly used regularization method that randomly drops hidden units from a neural network during training. The random drop of hidden units or feature detectors helps in avoiding complex co-adaptation of feature detectors. The prevention of such co-adaptation leads to an improved generalization of the test data where combinatorial feature context encountered during training may not be present. Instead of randomly dropping the hidden units, Keshari et al. [10] proposed an intelligent version of Dropout referred to as Guided Dropout, where only active hidden units are dropped. These hidden units are identified using the trainable strength parameters. The strength value signifies the relevance of the corresponding hidden unit in the decision-making process. Guided Dropout works on the hypothesis that removing high-strength or active hidden units will encourage the low-strength or inactive units to improve their strength and contribute to the neural network's performance. This strengthening of the inactive units may result in improved generalization. Gal and Ghahramani [16] showed that approximate variational inference in RNNs is identical to deploying dropouts where the same network components or units are dropped at each time step. However, the network components to be dropped are chosen randomly. In contrast to dropout and guided dropout that are mainly designed for DNNs, this variational RNN also drops the recurrent connections and provides better regularization in time-series modeling. Deviating from Dropout, Wan et al. [17] proposed DropConnect that randomly drops the weight connections between the layers. Hence, DropConnect introduces sparsity over the weights instead of output vectors (as in Dropout). Khan et al. [18] took a different approach and proposed spectral dropout where certain frequency contents of activations are dropped. Spectral dropout converts the activations into the frequency domain by discrete cosine transform (DCT) and the frequencies or basis coefficients corresponding to "noisy components" are not used for reconstruction during inverse transform (frequency domain to activations). The removal of these noisy frequency components regularizes the overall training.

Apart from dropping the network components, weight penalties such as ℓ_1 and ℓ_2 regularization [7] are also commonly used. Both ℓ_1 and ℓ_2 regularization encourages the norm of weights to be smaller. The smaller weights decrease the impact of hidden units, and hence, the model complexity. Working on a similar line, Dey et al. [19] proposed a combination of three regularization terms that is, ℓ_2 norm of weights along with the sum of squares of the first- and second-order derivatives of loss with respect to weights to regularize the training and improving the robustness of MLPs implemented using analog circuits.

All the aforementioned regularization methods directly try to restrict the model complexity. Aside from them, another class of regularization method exists that increases the task complexity by training the model for an auxiliary task (along with the primary task). The auxiliary task complements the primary task and also increases the "model complexity" requirements. This form of regularization is commonly used in semisupervised learning and is referred to as consistency-based regularization [20], [21], [22].

Also, many studies have regularized deep learning models by penalizing the large changes in outputs due to the minor input variations [23], [24]. These methods use the norm of the gradient [25] or the Jacobian [12], [26] of the loss function with respect to the model inputs as a regularization term. Such methods regularize the training and improve model stability against adversarial or random perturbations.

Comparison with the proposed algorithm: The proposed ITPS algorithm is marginally inspired by Guided Dropout [10]. Both of these algorithms analyze the strength or sensitivity of network components toward the task at hand. As discussed in Section I, this sensitivity analysis allows the modulation of the effective DoF and can be considered the backbone of the proposed ITPS algorithm. However, ITPS is fundamentally different from Guided Dropout:

1) ITPS does not drop any hidden units or parameters. It only updates a subset of selected parameters during a training iteration.
2) Guided Dropout drops a predefined number of active hidden units. Hence, it induces a static form of regularization. On the other hand, ITPS utilizes a greedy heuristic to induce dynamic regularization.

Apart from Guided Dropout, the proposed algorithm also exhibits similarity to Meta-SGD [27], a meta-learning algorithm that learns different learning rates for each model parameter. ITPS can be seen as a particular case of Meta-SGD when only a subset of parameters exhibit positive learning rates (greater than zero) during a training iteration. Hence, the parameters with these positive learning rates are used for minimizing the training loss. Although Meta-SGD can be updated to control the effective DoF as in ITPS, it requires a meta-optimization framework [28] that is computationally expensive.

III. Proposed Method

This section presents the proposed ITPS algorithm and its extension to Reptile (a model agnostic meta-learning algorithm). Besides, we also discuss the nature and cause of regularization induced by the proposed ITPS algorithm. Finally, Hessian spectral analysis [29] is performed to present more insights into the behavior of ITPS.

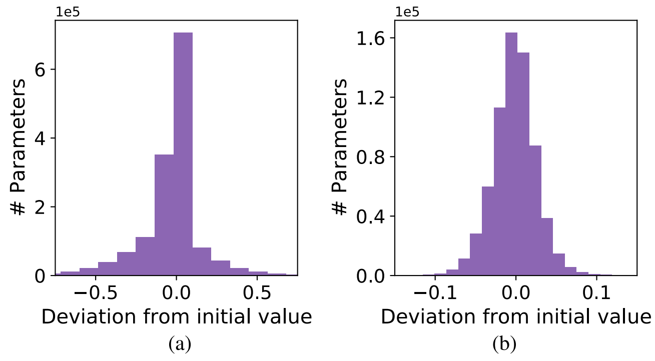


Fig. 2. Histograms depicting deviation of trained parameters from their initial values in (a) 5-layered DNN trained on the FashionMNIST dataset and (b) LSTM-based model trained on the MIMIC-III dataset for in-hospital mortality prediction.

A. Incremental Trainable Parameter Selection

The proposed ITPS algorithm acts as a simple wrapper over SGD to regularize the training. During each training iteration, it selects a subset of the parameters that are updated according to the SGD. Hence, only a subset of parameters is involved in minimizing the training loss. This behavior restricts the effective DoF and model complexity. However, there is no efficient way to measure the required model complexity for any task beforehand. The selection of fewer parameters may hinder training, whereas the overparameterization may lead to overfitting. The proposed ITPS algorithm addresses this issue by employing a greedy heuristic to incrementally select the sensitive parameters and increase the model's DoF as the training progresses. The proposed ITPS algorithm is expected to arrive at a model configuration with adequate DoF that exhibits effective generalization. ITPS can be employed with early stopping [6] to identify the configuration or training phase where the model starts overfitting.

1) Selection of Sensitive Parameters: Generally, a noticeable number of parameters in a trained overparameterized model are expected to exhibit no deviation from their initialized values. This behavior is evident in Fig. 2 where a significant number of parameters of two trained models (a DNN trained on Fashion MNIST and an LSTM model trained for in-hospital mortality prediction)[2] show no deviation from their initial values. This behavior has also extensively been exploited to sparsify or prune the deep learning models [30]. The parameters showing no deviation are not sensitive toward the training loss and have little to no impact on the effective DoF of the model. On the other hand, the parameters deviating from the initial values are likely to be responsible for defining the effective DoF. Hence, incrementally selecting and updating these parameters is expected to influence the DoF as desired.

ITPS algorithm utilizes accounting variables to keep an estimate of the deviation that would have been observed if all

[2]See Section V for model architectures.

Algorithm 1 Iteration of the Proposed ITPS Algorithm

```
1: $\mathcal{D}$: Dataset
2: Let $f_{\boldsymbol{\theta}}()$ be the model initialized with $\boldsymbol{\theta} \in \mathbb{R}^N$
3: Let $\mathbf{J}_{\boldsymbol{\theta}}$ be the accounting variables for $\boldsymbol{\theta}$ initialized with $\mathbf{0} \in \mathbb{R}^N$
4: $\alpha, \beta$ : Learning rates for updating $\boldsymbol{\theta}$, $\mathbf{J}_{\boldsymbol{\theta}}$
5: $\tau_1, \tau_2$ : Thresholds to select sensitive parameters
6: $\mathcal{B} \leftarrow$ SAMPLE-BATCHES($\mathcal{D}$)    ▷ $|\mathcal{B}|$ batches, each batch contains $\mathbf{b}$ examples with labels $\mathbf{l}$
7: for all $(\mathbf{b}, \mathbf{l}) \in \mathcal{B}$ do
8:     $L = \mathcal{L}_{CE}(f_{\boldsymbol{\theta}}(\mathbf{b}), \mathbf{l})$    ▷ Cross-entropy loss
9:     $\mathbf{G}_{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} L$    ▷ Gradients
10:    $\mathbf{J}_{\boldsymbol{\theta}} = \mathbf{J}_{\boldsymbol{\theta}} - \beta(\mathbf{G}_{\boldsymbol{\theta}})$    ▷ Updating accounting variables
11:    Create a mask $\mathbf{M} \in \mathbb{R}^N$ initialized with $\mathbf{0}$
12:    for all element $\mathbf{M}_i$ of $\mathbf{M}$ do
13:        $\mathbf{M}_i = \begin{cases} 1, \text{ IF } \mathbf{J}_{\boldsymbol{\theta}_i} < \tau_2 \text{ OR } \mathbf{J}_{\boldsymbol{\theta}_i} > \tau_1 \\ 0, \text{ IF } \tau_2 < \mathbf{J}_{\boldsymbol{\theta}_i} < \tau_1. \end{cases}$
               ▷ Thresholding to select sensitive parameters
14:    $\hat{\mathbf{G}_{\boldsymbol{\theta}}} = \mathbf{G}_{\boldsymbol{\theta}} \odot \mathbf{M}$    ▷ Mask gradients for nonselected parameters
15:    $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha\hat{\mathbf{G}_{\boldsymbol{\theta}}}$    ▷ Apply gradients to update $\boldsymbol{\theta}$
```

parameters are updated in each training iteration. The parameters whose deviation or corresponding accounting variable overcomes a predefined threshold are chosen and updated. As the training progresses, more and more sensitive parameters break the threshold and are included in the training. Since parameters exhibiting the maximum deviation overcome the threshold, ITPS only selects the most sensitive parameters. Thus, it can be regarded as a greedy algorithm. The number of parameters that break the threshold is dependent on the deviation, which is a function of training loss and hence, the task and the training data.

2) Implementation Details: Suppose $f_{\boldsymbol{\theta}}()$ is a neural network that is parameterized by a trainable tensor $\boldsymbol{\theta} \in \mathbb{R}^N$, where N is the total trainable parameters in $f_{\boldsymbol{\theta}}()$.[3] Also, $f_{\boldsymbol{\theta}}()$ maps an input example $\mathbf{x} \in \mathbb{R}^{K\times L}$ to the prediction or output vector $\mathbf{y} \in \mathbb{R}^C$, where C is the number of classes or dimensions of the output vector.

The proposed ITPS algorithm is presented in Algorithm 1. ITPS maintains accounting variables $\mathbf{J}_{\boldsymbol{\theta}} \in \mathbb{R}^N$ corresponding to the trainable parameters $\boldsymbol{\theta} \in \mathbb{R}^N$ of the model. These accounting variables $\mathbf{J}_{\boldsymbol{\theta}}$ are initialized with $\mathbf{0} \in \mathbb{R}^N$, where $\mathbf{0}$ is tensor whose all elements are zero. ITPS samples the training batches from a dataset $\mathcal{D}$ and processes each batch $\mathbf{b} \in \mathbb{R}^{n\times K\times L}$ having n input examples with labels $\mathbf{l} \in \mathbb{R}^{n\times C}$ to perform the following operations.

1) ITPS computes the training loss $\mathcal{L}_{\text{CE}}$ (such as cross-entropy loss function) for current batch using the current model state $(f_{\boldsymbol{\theta}}())$ as: $L = \mathcal{L}_{\text{CE}}(f_{\boldsymbol{\theta}}(\mathbf{b}), \mathbf{l})$.
This loss is used to compute gradients for all parameters as: $\mathbf{G}_{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} L$.

[3]For simplicity of notations, we are assuming a vectorized form of trainable tensor $\boldsymbol{\theta}$. All the operations used in the proposed method can directly be extended to trainable tensors of any shape or dimension.

2) The accounting variables $\mathbf{J}_{\boldsymbol{\theta}}$ are directly updated using $\mathbf{G}_{\boldsymbol{\theta}}$ as: $\mathbf{J}_{\boldsymbol{\theta}} = \mathbf{J}_{\boldsymbol{\theta}} - \beta(\mathbf{G}_{\boldsymbol{\theta}})$. Here, β controls the deviation induced by the gradients and is analogous to the learning rate. At any point in the training process, the deviation from zero in accounting variables provides an estimate of the accumulated deviation or sensitivity of the corresponding parameter.
3) During training, the elements of $\mathbf{J}_{\boldsymbol{\theta}}$ either exhibit increment or decrement from their initial values (i.e., 0). Both negative and positive deviations provide an estimate of the sensitivity of the corresponding parameters in $\boldsymbol{\theta}$ toward the training loss. Thus, we apply thresholding in both negative and positive directions around elements of $\mathbf{J}_{\boldsymbol{\theta}}$ to select the most sensitive parameters. The parameters corresponding to the elements of $\mathbf{J}_{\boldsymbol{\theta}}$ that do not lie between open interval (τ_2, τ_1) are selected for updation. τ_2 and τ_1 are thresholds such that $\tau_2 < 0$ and $\tau_1 > 0$.
To facilitate the process of parameter selection, we create a mask $\mathbf{M} \in \mathbb{R}^N$ that is initialized to be a zero tensor. Suppose $\mathbf{M_i}$ represent the ith element of $\mathbf{M}$. Each element $\mathbf{M_i}$ of this mask corresponds to a parameter $\boldsymbol{\theta}_i$ in $\boldsymbol{\theta}$ and hence corresponds to an accounting variable $\mathbf{J}_{\boldsymbol{\theta}_i}$ in $\mathbf{J}_{\boldsymbol{\theta}}$. $\mathbf{M}_i$ is set to 1 if $\mathbf{J}_{\boldsymbol{\theta}_i}$ does not lie between interval (τ_2, τ_1)
$$\mathbf{M}_i = \begin{cases} 1, & \text{IF } \mathbf{J}_{\boldsymbol{\theta}_i} < \tau_2 \text{ OR } \mathbf{J}_{\boldsymbol{\theta}_i} > \tau_1 \\ 0, & \text{IF } \tau_2 < \mathbf{J}_{\boldsymbol{\theta}_i} < \tau_1. \end{cases} \quad (1)$$
By using this equation, we process accounting variables to obtain mask $\mathbf{M}$.
4) $\mathbf{M}$ is used to mask the gradient updates $(\mathbf{G}_{\boldsymbol{\theta}})$ for nonselected parameters. These masked gradient updates are applied to update the selected parameters (with a learning rate α) as
$$\hat{\mathbf{G}_{\boldsymbol{\theta}}} = \mathbf{G}_{\boldsymbol{\theta}} \odot \mathbf{M}, \quad \boldsymbol{\theta} = \boldsymbol{\theta} - \alpha\hat{\mathbf{G}_{\boldsymbol{\theta}}}. \quad (2)$$
Here, $\odot$ represents the elementwise multiplication operation.

Since ITPS acts as a wrapper over SGD, any available variant of SGD such as Adam and AdaGrad can be utilized to update the parameters using the masked gradients $(\hat{\mathbf{G}_{\boldsymbol{\theta}}})$.

B. Regularization

The nature of regularization induced by ITPS on the SGD-based training of deep learning models can be studied using the following mechanisms.

1) *DoF of deep learning models:* Gao and Jojic [13] analyzed the DoF of a deep learning model and established that the utilization of regularization mechanisms such as Dropout and weight penalty decreases the effective DoF of a deep learning model. In ITPS, the effective DoF of the model is gradually increased with the training progression. This is analogous to the behavior where the level of regularization is initially high, and as the training progresses, the regularization is gradually decreased.
2) *Implicit gradient regularization:* Barrett and Dhemir [14] uncovered an implicit form of regularization in the gradient descent that arises

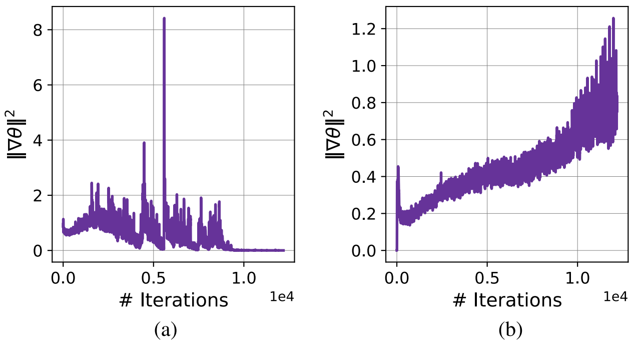


Fig. 3. Difference in ℓ_2 norm of gradients computed in (a) standard SGD and (b) proposed ITPS algorithm being applied to train an LSTM-based model for the task of in-hospital mortality prediction.

due to the discrete steps instead of following the exact continuous path toward the minima. They proved that the trajectory followed by the discrete steps of gradient descent is similar to an explicit path along a modified loss surface, which can be computed using backward error analysis. This deviation between the original loss surface and the modified loss surface is the source of implicit regularization in gradient descent.

Given the original loss $\mathcal{L}(\boldsymbol{\theta})$, the modified loss $\hat{\mathcal{L}}(\boldsymbol{\theta})$ can be computed analytically using the backward error analysis and is represented as [14]

$$\hat{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda R_{\text{IG}}(\boldsymbol{\theta})$$

where

$$\lambda \equiv \alpha N/4 \quad \text{and} \quad R_{\text{IG}}(\boldsymbol{\theta}) \equiv \frac{\|\nabla\boldsymbol{\theta}\|^2}{N}. \tag{3}$$

In (3), $\boldsymbol{\theta} \in \mathbb{R}^N$ represents the model parameters and α represents the learning rate. The proof of this equation can be found in [14]. The analysis of (3) makes it clear that $R_{\text{IG}}(\boldsymbol{\theta})$ is a regularizer penalizing the larger gradients and λ controls the regularization rate.

In SGD-based training, the magnitude of gradients is larger during the earlier stages of training, and this magnitude becomes smaller as SGD reaches convergence [see Fig. 3(a)]. As a result, implicit gradient regularization imposes larger penalties in the initial stages of training. However, ITPS manifests an entirely different behavior. The proposed ITPS algorithm selects more parameters for updation as the training progresses. Hence, the gradient norm increases gradually with the increment in selected parameters. This behavior is illustrated in Fig. 3(b). As a result, the implicit gradient regularization imposes more penalties as more parameters are selected, or the DoF of the model is increased. Note that once ITPS has selected all the parameters, the gradient norm is expected to show similar behavior to the traditional SGD.

The analysis of the DoF and the implicit gradient regularization illustrates that the proposed ITPS algorithm regularizes the training effectively throughout the training bout. During the initial stages, regularization is mainly imposed by the lower DoF. On the other hand, during the latter stages, the regularization is induced by the higher gradient norm penalties due to the implicit gradient regularization.

Algorithm 2 Extension of the Proposed ITPS Algorithm to Reptile for Multitasking

```
1: $\mathcal{D}_t$: Dataset for task t
2: Let $f_{\boldsymbol{\theta}}()$ be the model initialized with $\boldsymbol{\theta} \in \mathbb{R}^N$
3: Let $\mathbf{J}_{\boldsymbol{\theta}}$ be the accounting variables for $\boldsymbol{\theta}$ initialized with all $\mathbf{0}$
4: $\alpha, \beta, \gamma$ : Learning rates
5: $\tau_1, \tau_2$ : Thresholds to select sensitive parameters
6: for $t \leftarrow 1 : T$ do                                   ▷ $T$: Number of tasks
7:     $\mathbf{W}_t = \boldsymbol{\theta}$                     ▷ Initialize task t model with $\boldsymbol{\theta}$
8:     Let $f_{\mathbf{W}_t}()$ be a model with $\mathbf{W}_t$ parameters
9:     $\mathcal{B} \leftarrow$ SAMPLE-BATCHES($\mathcal{D}$)   ▷ $|\mathcal{B}|$ batches, each batch contains $\mathbf{b}$ examples with labels $\mathbf{l}$
10:    for all $(\mathbf{b}, \mathbf{l}) \in \mathcal{B}$ do
11:        $L = \mathcal{L}_{CE}(f_{\mathbf{W}_t}(\mathbf{b}), \mathbf{l})$   ▷ Cross-entropy loss
12:        $\mathbf{G}_{\mathbf{W}_t} = \nabla_{\mathbf{W}_t} L$   ▷ Gradients
13:        $\mathbf{J}_{\boldsymbol{\theta}} = \mathbf{J}_{\boldsymbol{\theta}} - \beta(\mathbf{G}_{\mathbf{W}_t})$
14:        Create a mask $\mathbf{M} \in \mathbb{R}^N$ initialized with $\mathbf{0}$
15:        for all element $\mathbf{M}_i$ of $\mathbf{M}$ do
16:            $\mathbf{M}_i = \begin{cases} 1, \text{ IF } \mathbf{J}_{\boldsymbol{\theta}_i} < \tau_2 \text{ OR } \mathbf{J}_{\boldsymbol{\theta}_i} > \tau_1 \\ 0, \text{ IF } \tau_2 < \mathbf{J}_{\boldsymbol{\theta}_i} < \tau_1. \end{cases}$
                                   ▷ Thresholding to select sensitive parameters
17:        $\hat{\mathbf{G}_{\mathbf{W}_t}} = \mathbf{G}_{\mathbf{W}_t} \odot \mathbf{M}$   ▷ Mask gradients
18:        $\mathbf{W_t}' = \mathbf{W_t} - \alpha\hat{\mathbf{G}_{\mathbf{W}_t}}$
19: $\boldsymbol{\Phi} = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{W_t}' - \mathbf{W_t})$   ▷ Meta update
20: $\boldsymbol{\theta} = \boldsymbol{\theta} + \gamma\ \boldsymbol{\Phi}$   ▷ Apply meta gradients
```

C. Extension to Reptile: An MAML Algorithm

Reptile [31], a first-order model agnostic meta-learning (MAML) algorithm, is generally used for multitasking where information learned from one task may aid in performing the other tasks. The proposed ITPS algorithm can easily be extended to Reptile (or any other MAML) by incrementally selecting the sensitive model parameters and masking the corresponding gradients computed for each task. Algorithm 2 encapsulates an iteration of the proposed extension of ITPS to Reptile. In Reptile, a shared model $f_{\boldsymbol{\theta}}()$ is parameterized with $\boldsymbol{\theta}$. During a training iteration, for each task t, a task-specific model $f_{\mathbf{W}_t}$ is created and is initialized with a copy of $\boldsymbol{\theta}$ ($\mathbf{W}_t = \boldsymbol{\theta}$). These task-specific models are trained using their corresponding datasets to obtain updated parameters $\mathbf{W}'_t$. To update $\boldsymbol{\theta}$, the meta-gradients are computed as: $\boldsymbol{\Phi} = (1/T)\sum_{t=1}^{T}(\mathbf{W}'_t - \mathbf{W}_t)$. In the proposed extension, the task-specific models are updated using the ITPS algorithm and the accounting variables are shared across all the tasks (as shown in Algorithm 2).

Note that many multitask scenarios require task-specific changes to the model architecture. The tasks that rely on similar latent representation may require a few task-specific layers. The proposed ITPS extension addresses such cases by considering the task-specific models as a combination of the

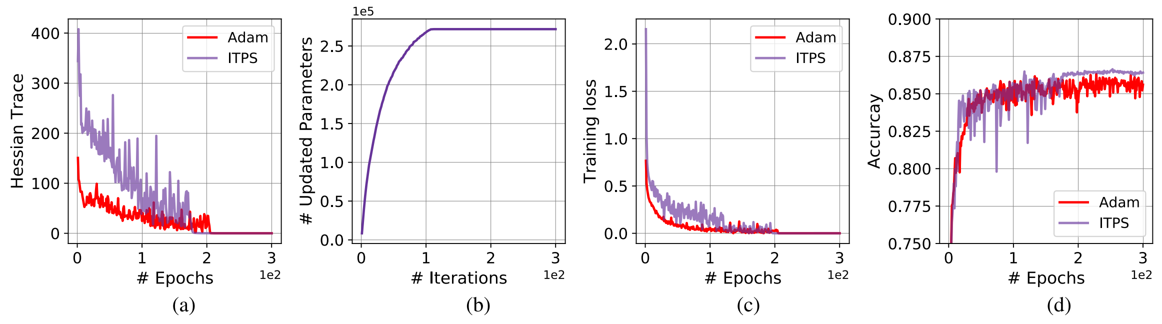


Fig. 4. Observing the evolution of (a) Hessian trace, (b) number of selected parameters, (c) training loss, and (d) validation accuracy during the ITPS-based training of a 5-layer DNN over the Fasion MNIST dataset.

shared and the task-specific layers. In the ITPS extension, the accounting variables are maintained for both shared and task-specific parameters. A detailed study of task heterogeneity scenarios can be found in the supplementary document.

IV. Hessian Spectral Analysis

We performed the Hessian spectral analysis to get more insights into the training dynamics of the proposed ITPS algorithm. The Hessian of the loss of a neural network $f_{\boldsymbol{\theta}}()$ can be represented as: $\mathbf{H} = \partial^2 \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{b}), \mathbf{l})/\partial \boldsymbol{\theta}^2 = \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(f_{\boldsymbol{\theta}}(\mathbf{b}), \mathbf{l})$. Here, $\mathbf{b}$, $\mathbf{l}$, $\mathcal{L}$, and $\boldsymbol{\theta} \in \mathbb{R}^N$ represent a batch of examples, its corresponding labels, loss function, and parameters of model $f_{\boldsymbol{\theta}}()$, respectively. The Hessian matrix $\mathbf{H} \in \mathbb{R}^{N \times N}$ is a symmetric matrix that can provide information about the learning dynamics of the neural networks [29], [32], [33]. In this work, we analyzed the evolution of Hessian trace during ITPS-based training of a DNN[4] over the Fashion MNIST dataset [34]. Along with the Hessian trace, we also analyzed the evolution of cardinality of the subset of parameters selected/updated by ITPS, training loss, and validation accuracy. We implemented ITPS over Adam optimizer and used a fixed learning rate of $\alpha = 0.001$, $\beta = 0.1$, $\tau_1 = 0.25$, and $\tau_2 = -0.25$ (see Algorithm 1). We compared the training dynamics of ITPS against the standard Adam optimizer with a learning rate of $\alpha = 0.001$. Fig. 4 encapsulates the observations, and the following conclusions can be drawn from this figure.

1) During the initial training phase of ITPS, the Hessian trace is larger compared to the Hessian trace observed in Adam. This can be attributed to the restricted parameter selection and updation by ITPS. Once the number of parameters to be utilized by ITPS settles (around 100th epoch), the behavior of ITPS becomes similar to Adam's.
2) Both Adam and ITPS lead to a flatter minimum as signified by the low Hessian trace during the latter phase of training [32]. Arguably, the flatter minima results in better generalization [35], [36]. This implies that ITPS is able to converge while updating only approx. 27 000 parameters out of 1.49 million available in DNN throughout the training.
3) Since ITPS selects and updates the most sensitive parameters, it only moves the model parameters in nondegenerate directions during the initial stages of training. These directions are signified by nonzero eigen values of the Hessian matrix. When these nondegenerate directions become sparse, as suggested by the lower Hessian trace, it is an indication that model parameters are in flatter minima [32] and convergence has been achieved.

[4]See Section V for architecture details.

V. Experimental Setup

This section describes datasets, comparative methods, and experiments used for the performance evaluation of the proposed algorithms.

A. Datasets, Tasks, and Models

Table I documents the datasets, tasks, and models used for the performance evaluation. These tasks and models are chosen to exhibit the general utility of the proposed ITPS algorithms.

1) Fashion MNIST Classification Using DNN: Fashion MNIST [34] consists of 28×28 grayscale images of ten different categories of clothing articles. The images are flattened before being given as input to fully- or densely-connected neural networks (DNN). This DNN consists of five layers with 1024, 512, 256, 128, and ten units. The hidden and output layers are followed by rectified linear and softmax activation functions, respectively. The model architecture contains 1.49 million (approx.) trainable parameters.

2) CIFAR-10 and SLT-10 Classification Using CNNs: VGG-16 [37], Resnet-50 [38], and EfficientNetB0 [39] (pretrained on imagenet dataset) are trained on CIFAR-10 [40] and SLT-10 [41] datasets. The images in both datasets are resized to 224×224 before being given as input to CNNs.

3) MIMIC-III or Healthcare Tasks Using LSTM and Transformer: MIMIC-III contains electronic health records (EHRs) of patients admitted to critical care units at a tertiary care hospital. EHRs include vital signs, medications, laboratory measurements, fluid balance, hospital length of stay, and survival data. As described in [43], MIMIC-III is preprocessed to sample subdatasets for three different tasks.[5] Each example is an evenly spaced time series where different clinical measurements (resulting in 76 features) are sampled at each time

[5]Benchmarking code available at https://github.com/YerevaNN/mimic3-benchmarks is used to create training, test, and validation sets.

TABLE I
DATASETS AND MODELS USED FOR THE PERFORMANCE EVALUATION OF THE PROPOSED ITPS ALGORITHM

Dataset & Task	Models	# Training, Validation and Test Examples
Fashion MNIST [34] (Image classification)	Fully-connected neural network	50K, 10K and 10K
CIFAR-10 [40] (Image classification)	VGG-16, Resnet-50, EfficientNet (Pre-trained on Imagenet)	40K, 10K and 10K
SLT-10 [41] (Image classification)	VGG-16, Resnet-50, EfficientNet (Pre-trained on Imagenet)	4K, 1K and 8K
MIMIC-III [42] (In-hospital mortality prediction)	LSTM & Transformer	14698, 3222 and 3236
MIMIC-III [42] (Decompensation prediction)	LSTM & Transformer	2388414, 520000 and 523208
MIMIC-III [42] (Phenotyping)	LSTM & Transformer	29280, 6371 and 6281

step. In this work, we have specified a time step of one hour. The three tasks are described below.

1) In-hospital mortality prediction: This task deals with predicting in-hospital mortality based on the first 48 h of ICU stay. Each example is represented by a time-series of size 76×48, where 48 is the number of time-steps and 76 is the feature dimension.
2) *Decompensation prediction:* Decompensation prediction deals with predicting whether the patient's condition will decline in the next 24 h. Each example is represented by a variable length time series where each time-step is represented by a 76-dimensional vector. Zero-padding is used to force the same length or time-steps on all time-series in a training batch.
3) *Phenotype classification:* This is a multilabel classification task to identify 25 different acute care conditions (phenotypes) from a patient's ICU stay record. Similar to the decompensation prediction task, each example is represented by a variable length time series. The details of these 25 phenotypes can be found in [43].

Two types of neural networks are used for each of the aforementioned tasks.

1) Fig. 5 depicts the architectures of LSTM-based models used for each task. These architectures are almost identical and only differ in the structures of the last layers.
2) We also use transformer-based models where each model consists of four transformer blocks [44] followed by a global average pooling layer and dense layers with 128 units and one output unit (or ten output units for phenotyping). The penultimate and last dense layers are followed by ReLu and sigmoid activations, respectively. Each transformer block (encoder blocks in [44]) consists of a multihead attention followed by a feed-forward module. The attention module utilizes four attention heads and 256-dimensional keys. The feed-forward module consists of two 1-D convolution layers that have eight and 76 pointwise filters (of size 1). These models consist of 1.27 million (approx.) trainable parameters. More details can be found in the supplementary document.

B. Comparative Regularization Methods

Following comparative methods have been used to evaluate the performance of the proposed ITPS algorithm.

1) Dropout [8], guided dropout [10], and spectral dropout [18] are used as the major baselines for the performance comparison. Guided dropout has been known to perform better than other variants of dropout such as variational dropout and concrete dropout [10]. Hence, these methods are not considered in this study.
2) ℓ_2 regularization (weight decay), data grad (i.e., the norm of the gradient of the loss with respect to input as regularizer) [24], [25], and Jacobian regularization (i.e., the norm of the Jacobian of loss with respect to input as regularizer) [24], [26] are also used as the comparative methods.
3) Variational LSTM [16] that drops the recurrent connections along with the input and output units is used as a comparative method specifically for the RNNs (and critical care tasks).

C. Experiments

We designed experiments to evaluate different regularization methods in both high and low amounts of training data scenarios. The experimental designs are dataset-specific. For CIFAR-10 and SLT-10 (image classification), we train CNNs on all the available data. We simulate a data-scarce scenario using the CIFAR-10 dataset where we train models using 50, 100, and 500 examples per class.

For Fashion MNIST and MIMIC-III tasks, we use different supervision levels (i.e., 25%, 50%, 75%, and 100%) for training models. The supervision level implies the percentage of the available training examples used for training a model. The utilization of different supervision levels helps in analyzing

TABLE II

OPTIMAL PARAMETERS OF DIFFERENT REGULARIZATION METHODS USED FOR DIFFERENT DATASETS. THESE PARAMETERS ARE TUNED TO OBTAIN MAXIMUM PERFORMANCE ON VALIDATION EXAMPLES

Method	**CIFAR-10**			**SLT-10**			**Fashion MNIST**	**MIMIC-III**	
	VGG -16	*Resnet -50*	*Efficient-Net*	*VGG -16*	*Resnet -50*	*Efficient-Net*	*DNN*	*LSTM-based models*	*Transformers*
Dropout rate	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Guided dropout rate	0.5	0.4	0.3	0.5	0.4	0.5	0.5	0.5	0.5
L2 reg. weight decay	0.0005	0.0001	0.0001	0.0005	0.0001	0.0001	0.001	0.005	0.001
Data grad reg. rate	0.005	0.001	0.001	0.01	0.005	0.001	0.01	0.01	0.005
Jacobian reg. rate	0.01	0.001	0.001	0.01	0.001	0.001	0.01	0.015	0.005
Spectral dropout ($\tau = 0.1, \eta$)	0.01	0.01	0.01	0.01	0.01	0.01	0.1	0.1	0.05
ITPS (τ_2, τ_1)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.25, 0.25)	(-0.15, 0.15)	(-0.05, 0.05)
ITPS + Spectral Dropout (τ_2, τ_1)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.05, 0.05)	(-0.1, 0.1)	(-0.1, 0.1)	(-0.05, 0.05)

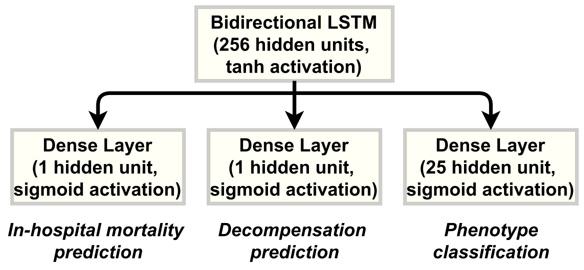

Bidirectional LSTM (256 hidden units, tanh activation)
Dense Layer (1 hidden unit, sigmoid activation)
Dense Layer (1 hidden unit, sigmoid activation)
Dense Layer (25 hidden unit, sigmoid activation)
In-hospital mortality prediction
Decompensation prediction
Phenotype classification

Fig. 5. Model architectures used for different critical care tasks. The architectures used for in-hospital mortality or decompensation prediction and phenotyping contain 682 497 and 694 809 trainable parameters, respectively.

the changes in ITPS behavior with a variation in the number of training examples.

The performance of the proposed ITPS extension to Reptile (ITPS-Reptile) is compared against the Reptile algorithm in a multitasking scenario. This comparison is performed using the critical-care or MIMIC-III tasks using LSTM-based models where each model is trained for in-hospital mortality prediction, decompensation prediction, and phenotyping. These critical care tasks work on a common set of features, and a latent representation learned by the model for a particular task can help in other tasks. Task heterogeneity is handled using the shared and task-specific layers as described in [45]. More details can also be found in the supplementary document.

D. Parameter Setting

The parameters across all baselines and the proposed regularization method can be divided into two categories: training specific (such as optimizer, batch size, and learning rate) and regularization-specific (such as dropout rate, weight decay, and thresholds in ITPS). For a fair comparison, we keep training-specific parameters the same for each model or dataset across all baselines and tune the regularization-specific parameters over the validation examples using grid search. The search space used for selecting each parameter is presented in Table S1 of the supplementary document. Note that we have tuned hyperparameters separately for each experimental setting or supervision level while keeping the search spaces the same across all experiments. However, we did not notice any major change in optimal parameters with the change in supervision levels for most of the cases.

1) Training-Specific Parameters: An identical training setup is used for all comparative methods for the performance evolution on a specific dataset. For all CIFAR-10 and SLT-10 experiments, we have used Adam with a fixed learning rate of 0.0001 as an optimizer and a batch size of 32 for training all CNNs. Similarly, a batch size of 32 examples and an Adam optimizer with a fixed learning rate of 0.001 is used for training the models (DNN, LSTM, and transformer-based models) on the Fashion MNIST and MIMIC-III datasets.

For all comparative methods, the batches are presented in identical order to the training process. The same initialization is used for all methods in a comparative study (or a run) to alleviate the effect of the initialization on the performance. The early stopping is used across all methods, and the model configuration providing the best performance on the validation examples is chosen for evaluation on the test examples.

2) Regularization-Specific Parameters: Table II tabulates the optimum regularization-specific parameters used for each method on each dataset. These parameters are tuned over validation examples using grid search for 100% supervision setting. As discussed earlier, the parameter setting documented in Table II is almost optimum for the other supervision levels.

For spectral dropout, we keep threshold ($\tau = 0.1$) fixed and vary η, percentage of activations above τ as suggested in [18]. The parameters listed on the MIMIC-III dataset were found to be optimal across all three critical tasks. For ITPS + Spectral Dropout, we keep the optimum spectral dropout parameter η for each model and only vary the ITPS thresholds.

In multitasking experiments, both Reptile and ITPS-Reptile, we have used task-specific learning rate α and global learning

TABLE III

PERFORMANCE OF DIFFERENT REGULARIZATION METHODS ON THE CIFAR-10 DATASET

Method	Accuracy (%)		
	VGG-16	**Resnet-50**	**EfficientNet**
Dropout	88.78 (±0.13)	93.21 (±0.2)	96.25 (±0.1)
Guided dropout	91.45 (±0.21)	93.95 (±0.16)	96.45 (±0.22)
L2 Reg.	87.8 (±0.13)	92.85 (±0.17)	95.85 (±0.19)
Data grad	88.1 (±0.12)	92.95 (±0.15)	95.95 (±0.21)
Jacobian	87.75 (±0.2)	92.81 (±0.16)	95.92 (±0.09)
Spectral dropout	91.62 (±0.14)	93.85 (±0.15)	96.55 (±0.11)
ITPS	**92.15** (±0.08)	**94.2** (±0.11)	**96.62** (±0.1)
ITPS + Spectral Dropout	**92.9** (±0.11)	**94.35** (±0.13)	**96.75** (±0.16)

TABLE IV

PERFORMANCE OF DIFFERENT REGULARIZATION METHODS ON THE SLT-10 DATASET

Method	Accuracy (%)		
	VGG-16	**Resnet-50**	**EfficientNet**
Dropout	90.58 (±0.25)	92.77 (±0.12)	96.8 (±0.17)
Guided Dropout	91.85 (±0.3)	93.36 (±0.26)	97.1 (±0.12)
L2 Reg.	88.98 (±0.19)	92.15 (±0.26)	96.73 (±0.21)
Data grad	89.1 (±0.2)	92.35 (±0.23)	96.9 (±0.17)
Jacobian	89 (±0.11)	92.14 (±0.1)	96.81 (±0.17)
Spectral Dropout	91.79 (±0.22)	93.4 (±0.23)	97.15 (±0.18)
ITPS	**92.95** (±0.11)	**94.2** (±0.13)	**97.34** (±0.08)
ITPS + Spectral Dropout	**93.35** (±0.15)	**94.35** (±0.13)	**97.46** (±0.1)

TABLE V

PERFORMANCE OF VGG-16 ON THE CIFAR-10 DATASET UNDER LOW TRAINING DATA CONDITIONS

Method	Number of training examples		
	500	*1000*	*5000*
Dropout	58.1 (±0.12)	63.67 (±0.13)	78.9 (±0.17)
Guided Dropout	59.32 (±0.12)	64.85 (±0.09)	80.11 (±0.19)
L2 Reg.	57.25 (±0.17)	63.41 (±0.22)	78.1 (±0.15)
Data grad	57.43 (±0.13)	63.53 (±0.21)	77.96 (±0.23)
Jacobian	57.31 (±0.19)	63.47 (±0.15)	78.05 (±0.16)
Spectral Dropout	59.48 (±0.11)	64.98 (±0.17)	80.15 (±0.24)
ITPS	**62.68** (±0.09)	**66.78** (±0.16)	**83.21** (±0.14)
ITPS+ Spectral dropout	**63.45** (±0.14)	**67.55** (±0.18)	**83.62** (±0.21)

TABLE VI

PERFORMANCE OF RESNET-50 ON THE CIFAR-10 DATASET UNDER LOW TRAINING DATA CONDITIONS

Method	Number of training examples		
	500	*1000*	*5000*
Dropout	67.75 (±0.18)	77.82 (±0.13)	88.5 (±0.17)
Guided Dropout	69.88 (±0.21)	78.85 (±0.14)	89.25 (±0.23)
L2 Reg.	67.32 (±0.26)	77.63 (±0.22)	88.45 (±0.18)
Data grad	67.35 (±0.19)	77.72 (±0.26)	88.4 (±0.19)
Jacobian	67.45 (±0.13)	77.69 (±0.28)	88.55 (±0.27)
Spectral Dropout	70.12 (±0.17)	78.8 (±0.14)	89.55 (±0.21)
ITPS	**73.57** (±0.11)	**80.15** (±0.17)	**89.7** (±0.15)
ITPS+ Spectral dropout	**73.82** (±0.14)	**80.33** (±0.21)	**89.95** (±0.19)

TABLE VII

PERFORMANCE OF EFFICIENTNET ON CIFAR-10 UNDER LOW TRAINING DATA CONDITIONS

Method	Number of training examples		
	500	*1000*	*5000*
Dropout	79.55 (±0.16)	84.1 (±0.18)	91.25 (±0.17)
Guided Dropout	80.75 (±0.21)	84.84 (±0.13)	91.89 (±0.16)
L2 Reg.	79.2 (±0.18)	83.75 (±0.22)	90.84 (±0.21)
Data grad	79.25 (±0.23)	83.89 (±0.19)	90.9 (±0.17)
Jacobian	79.23 (±0.17)	83.9 (±0.13)	90.95 (±0.23)
Spectral Dropout	80.61 (±0.23)	84.79 (±0.18)	91.97 (±0.16)
ITPS	**81.45** (±0.13)	**85.55** (±0.12)	**92.4** (±0.18)
ITPS+ Spectral dropout	**81.8** (±0.17)	**85.81** (±0.21)	**92.52** (±0.22)

rate γ (see Algorithm 2) of 0.001 and 0.15, respectively. Note that we have kept $\beta = 0.1$ (the rate of updating accounting variables in Algorithm 1 and 2) constant in ITPS, ITPS + Spectral Dropout and ITPS-Reptile for all experiments.

E. Performance Metrics

Classification accuracy is used as the performance metric for Fashion MNIST, CIFAR-10, and SLT-10 classification. For the task of phenotyping, the macro area under the ROC curve (AUROC) is used as the performance metric. For the prediction tasks (mortality and decompensation prediction), we have used AUROC as a metric.

VI. RESULTS AND DISCUSSION

In this section, we present and discuss the results obtained during the experimentation. Apart from that, we also analyze the impact of thresholding parameters, that is, τ_1 and τ_2 on the performance and the learning dynamics of ITPS.

A. Performance on CIFAR-10 and SLT-10 Datasets

Tables III and IV document the performance of different methods used to regularize the training of VGG-16, Resnet-50, and EfficientNet on CIFAR-10 and SLT-10 datasets, respectively. The following conclusions can be drawn from the analysis of these tables.

1) The proposed ITPS algorithm shows a noticeable improvement over all the other comparative methods across all models and both datasets. However, the improvement is more significant on VGG-16 and Resnet-50 as these models are more parameterized and hence more susceptible to overfitting than EfficientNet (which has a compact architecture).
2) Among baselines, guided and spectral dropout outperform dropout and all the other methods. Moreover, the performance of both these methods is comparable across all experiments. Also, the performance of dropout is better than data grad, Jacobian, and ℓ_2 regularization.
3) The best classification performance was achieved across all models by applying spectral dropout with ITPS (ITPS + Spectral Dropout) to regularize training. Since

TABLE VIII
CLASSIFICATION ACCURACY OF DIFFERENT REGULARIZATION METHODS ON THE FASHION MNIST DATASET

Method	Supervision Level			
	25%	50%	75%	100%
Dropout	86.35 (±0.16)	87.62 (±0.13)	87.84 (±0.15)	88.18 (±0.19)
Guided dropout	86.8 (±0.13)	88.05 (±0.18)	89.18 (±0.11)	89.47 (±0.12)
L2 Reg.	86.15 (±0.18)	87.36 (±0.25)	87.55 (±0.13)	87.85 (±0.11)
Data grad	86.22 (±0.19)	87.5 (±0.12)	87.75 (±0.25)	88.1 (±0.12)
Jacobian	86.12 (±0.17)	87.47 (±0.16)	87.6 (±0.21)	87.96 (±0.16)
Spectral Dropout	86.95 (±0.09)	88.19 (±0.12)	89.31 (±0.16)	89.51 (±0.13)
ITPS	**87.45** (±0.15)	**88.76** (±0.11)	**89.81** (±0.17)	**90.15** (±0.16)
ITPS+ Spectral Dropout	**87.95** (±0.11)	**89.13** (±0.16)	**90.16** (±0.12)	**90.58** (±0.11)

the cause of regularization induced by ITPS and spectral dropout are different, both these methods complement each other, and the combination of these methods outperforms both spectral dropout as well as ITPS.

Tables V–VII tabulate the performance of VGG-16, Resnet-50, and EfficientNet trained under data-scarce scenarios on the CIFAR-10 dataset. Again, ITPS outperforms the existing methods across all experimental configurations. Also, the combination of spectral dropout and ITPS exhibits the best classification performance. The improvement shown by ITPS and ITPS+Spectral Dropout over the comparative methods is more prominent when we use only 500 training examples (or 50 examples per class). This shows that ITPS can be of particular interest across multiple applications where we often deal with a lesser amount of training data.

B. Performance on Fashion MNIST

The performance of ITPS and other comparative methods for training DNN at different levels of supervision is documented in Table VIII. The performance trends on the CIFAR-10 and SLT-10 datasets also hold on DNN trained on the Fashion MNIST dataset. ITPS exhibits a noticeable improvement over the baseline methods, whereas ITPS + Spectral Dropout results in the best performance across all experiments.

C. Performance on MIMIC-III Tasks

Tables IX–XI document the performance of LSTM-based models for the tasks of in-hospital mortality (IHM) prediction, decompensation prediction, and phenotype classification, respectively. The following inferences can be drawn from the analysis of these tables.

1) ITPS and ITPS + Spectral Dropout exhibit better performance than other regularization methods at all the experimental configurations. The paired t-tests ($p < 0.05$) confirm the statistical significance of the performance improvements observed by ITPS and ITPS + Spectral Dropout over all baselines.
2) For IHM and decompensation prediction, the performance improvements observed by ITPS and ITPS + Spectral Dropout are more significant at lower supervision, that is, 25% and 50%. At higher supervision, the chances of overfitting are reduced by the availability of more training data. Hence, the performance improvements are relatively less.
For phenotype classification, the relative improvement is similar across all supervision levels.
3) Among baselines, guided and spectral dropout outperform the other methods. Moreover, variational dropout results in better performance than data grad, Jacobian, and ℓ_2 regularization.

TABLE IX
PERFORMANCE OF LSTM-BASED MODEL FOR THE TASK OF IN-HOSPITAL MORTALITY PREDICTION

Method	Supervision Level			
	25%	50%	75%	100%
Dropout	0.8249 (±0.0019)	0.8363 (±0.0013)	0.8392 (±0.0018)	0.8414 (±0.0016)
Guided Dropout	0.8285 (±0.0015)	0.8382 (±0.0014)	0.8411 (±0.0011)	0.8452 (±0.0017)
L2 Reg.	0.8212 (±0.0019)	0.8339 (±0.0017)	0.8365 (±0.0019)	0.8382 (±0.0013)
Data grad	0.8227 (±0.0011)	0.8352 (±0.0016)	0.8375 (±0.0016)	0.8402 (±0.0014)
Jacobian	0.8225 (±0.0017)	0.8345 (±0.0019)	0.8371 (±0.0013)	0.8397 (±0.0014)
Variational LSTM	0.8297 (±0.0012)	0.8394 (±0.0017)	0.8422 (±0.0014)	0.8456 (±0.0012)
Spectral Dropout	0.8311 (±0.0017)	0.839 (±0.0019)	0.844 (±0.0021)	0.8455 (±0.0011)
ITPS	**0.8404** (±0.0016)	**0.8479** (±0.0012)	**0.8495** (±0.001)	**0.8513** (±0.0014)
ITPS+ Spectral Dropout	**0.8415** (±0.0012)	**0.8495** (±0.0017)	**0.851** (±0.0017)	**0.8532** (±0.0018)

TABLE X
PERFORMANCE OF LSTM-BASED MODELS FOR THE TASK OF DECOMPENSATION PREDICTION

Method	Supervision Level			
	25%	50%	75%	100%
Dropout	0.8555 (±0.0012)	0.8674 (±0.0016)	0.8761 (±0.0021)	0.8836 (±0.0022)
Guided Dropout	0.8597 (±0.0013)	0.8722 (±0.0017)	0.8817 (±0.0016)	0.8892 (±0.0012)
L2 Reg.	0.8512 (±0.0018)	0.8637 (±0.0021)	0.8729 (±0.0019)	0.8796 (±0.0019)
Data grad	0.8531 (±0.0014)	0.8654 (±0.0017)	0.8738 (±0.0018)	0.8807 (±0.0012)
Jacobian	0.8523 (±0.0021)	0.8644 (±0.0011)	0.8731 (±0.0013)	0.8802 (±0.0017)
Variational LSTM	0.8535 (±0.0017)	0.8763 (±0.0015)	0.8846 (±0.0022)	0.8928 (±0.0023)
Spectral Dropout	0.8605 (±0.0021)	0.8735 (±0.0018)	0.8812 (±0.0016)	0.8907 (±0.0022)
ITPS	**0.8705** (±0.0018)	**0.8827** (±0.0014)	**0.8938** (±0.0011)	**0.8997** (±0.0013)
ITPS+ Spectral Dropout	**0.8753** (±0.002)	**0.8855** (±0.0014)	**0.8954** (±0.0013)	**0.9019** (±0.0017)

TABLE XI

PERFORMANCE OF DIFFERENT REGULARIZATION METHODS FOR THE TASK OF PHENOTYPE CLASSIFICATION. MACRO AUROC IS USED AS THE PERFORMANCE METRIC

Method	**Supervision Level**			
	25%	**50%**	**75%**	**100%**
Dropout	0.7265 (±0.0012)	0.7351 (±0.0015)	0.7445 (±0.0013)	0.7548 (±0.0017)
Guided Dropout	0.7288 (±0.0017)	0.7385 (±0.0013)	0.7486 (±0.0013)	0.7592 (±0.0012)
L2 Reg.	0.7238 (±0.001)	0.7319 (±0.0018)	0.7425 (±0.0021)	0.7526 (±0.002)
Data grad	0.7252 (±0.0012)	0.7331 (±0.0027)	0.7431 (±0.0012)	0.7533 (±0.0015)
Jacobian	0.7245 (±0.0015)	0.7325 (±0.0012)	0.7428 (±0.0014)	0.7529 (±0.0011)
Variational LSTM	0.7295 (±0.0012)	0.7417 (±0.0018)	0.7528 (±0.0016)	0.7648 (±0.0018)
Spectral Dropout	0.7301 (±0.0011)	0.7395 (±0.0019)	0.7493 (±0.0021)	0.7604 (±0.0018)
ITPS	**0.7342** (±0.0014)	**0.7465** (±0.0013)	**0.7558** (±0.0011)	**0.7668** (±0.0012)
ITPS+ Spectral Dropout	**0.7391** (±0.0018)	**0.751** (±0.0012)	**0.7595** (±0.0009)	**0.7708** (±0.0011)

TABLE XII

PERFORMANCE OF TRANSFORMER FOR THE TASK OF IN-HOSPITAL MORTALITY

Method	**Supervision Level**			
	25%	**50%**	**75%**	**100%**
Dropout	0.8368 (±0.0018)	0.8395 (±0.0012)	0.845 (±0.0015)	0.8495 (±0.0017)
Guided Dropout	0.8385 (±0.0009)	0.8427 (±0.0013)	0.8481 (±0.0017)	0.8525 (±0.0019)
L2 Reg.	0.8341 (±0.0015)	0.8372 (±0.002)	0.8427 (±0.0016)	0.8471 (±0.0013)
Data grad	0.8355 (±0.0022)	0.8365 (±0.0019)	0.8436 (±0.0014)	0.8482 (±0.0021)
Jacobian	0.8347 (±0.0017)	0.837 (±0.0012)	0.843 (±0.0018)	0.8477 (±0.0012)
Spectral Dropout	0.8382 (±0.0011)	0.8431 (±0.0016)	0.8492 (±0.0015)	0.8534 (±0.0018)
ITPS	**0.8435** (±0.0012)	**0.8464** (±0.0014)	**0.85** (±0.0009)	**0.855** (±0.0016)
ITPS+ Spectral dropout	**0.8455** (±0.0013)	**0.8482** (±0.0018)	**0.8524** (±0.0013)	**0.8569** (±0.0008)

TABLE XIII

PERFORMANCE OF TRANSFORMER FOR THE TASK OF DECOMPENSATION PREDICTION

Method	**Supervision Level**			
	25%	**50%**	**75%**	**100%**
Dropout	0.8663 (±0.0011)	0.8786 (±0.002)	0.8808 (±0.0011)	0.8861 (±0.0012)
Guided Dropout	0.8722 (±0.0017)	0.8862 (±0.0011)	0.8877 (±0.0012)	0.8922 (±0.0011)
L2 Reg.	0.8648 (±0.0022)	0.8772 (±0.0018)	0.8785 (±0.0013)	0.8833 (±0.0019)
Data grad	0.865 (±0.0014)	0.8775 (±0.0019)	0.8791 (±0.0017)	0.8847 (±0.0011)
Jacobian	0.8655 (±0.0013)	0.877 (±0.0018)	0.8795 (±0.0014)	0.8843 (±0.0018)
Spectral Dropout	0.872 (±0.0018)	0.8878 (±0.0017)	0.8887 (±0.0014)	0.8937 (±0.0016)
ITPS	**0.8775** (±0.0011)	**0.8914** (±0.0009)	**0.893** (±0.0011)	**0.8997** (±0.0012)
ITPS+ Spectral dropout	**0.8798** (±0.0008)	**0.8932** (±0.0013)	**0.8951** (±0.0014)	**0.9015** (±0.0014)

TABLE XIV

PERFORMANCE OF TRANSFORMER FOR THE TASK OF PHENOTYPING

Method	**Supervision Level**			
	25%	**50%**	**75%**	**100%**
Dropout	0.7317 (±0.0023)	0.741 (±0.0018)	0.7497 (±0.0018)	0.7595 (±0.0017)
Guided Dropout	0.7345 (±0.0012)	0.7453 (±0.0012)	0.7554 (±0.0017)	0.7654 (±0.0015)
L2 Reg.	0.7285 (±0.0017)	0.7391 (±0.0015)	0.7482 (±0.0012)	0.7583 (±0.0013)
Data grad	0.7297 (±0.0013)	0.7398 (±0.0014)	0.749 (±0.0019)	0.7591 (±0.0017)
Jacobian	0.7291 (±0.0012)	0.7395 (±0.0019)	0.7493 (±0.0018)	0.7595 (±0.0012)
Spectral Dropout	**0.7564** (±0.0013)	0.7466 (±0.0018)	0.7559 (±0.002)	0.7645 (±0.0011)
ITPS	0.7403 (±0.0015)	**0.7502** (±0.0008)	**0.7596** (±0.0014)	**0.7702** (±0.0015)
ITPS+ Spectral dropout	**0.7424** (±0.0011)	**0.7519** (±0.0012)	**0.7618** (±0.0007)	**0.7715** (±0.001)

TABLE XV

PERFORMANCE OF REPTILE AND ITPS-REPTILE IN A MULTITASKING SCENARIO

Algorithm	**Mortality prediction**	**Decompensation prediction**	**Phenotype classification**
Reptile	0.8534 (±0.0019)	0.9013 (±0.0017)	0.7691 (±0.0018)
Reptile-ITPS	**0.8591** (±0.0014)	**0.9109** (±0.0022)	**0.7736** (±0.0013)

Tables XII–XIV document the performance of transformer- or attention-based models for the tasks of in-hospital mortality (IHM) prediction, decompensation prediction, and phenotype classification, respectively. Similar to LSTM-based models, ITPS and ITPS + Spectral Dropout outperform almost all baselines on all three tasks. This shows that the proposed method is not model dependent and can help in improving the performance of any model architecture for a given task.

D. Performance in Multitask Setup: Reptile Versus ITPS-Reptile

Table XV tabulates the performance of LSTM-based models trained simultaneously for critical-care tasks using Reptile and ITPS-Reptile. The analysis of this table highlights that adding ITPS to Reptile leads to a statistically significant improvement ($p < 0.05$) in the classification or prediction performance across all three tasks over Reptile.

E. DoF as a Function of the Number of Training Examples

Fig. 6 shows the total parameters selected by ITPS for fashion MNIST classification and in-hospital mortality prediction. As the number of training examples or the supervision level is incremented, the total parameters selected by ITPS are also increased, and the level of regularization is decreased. This shows that, as desired, the level of regularization in ITPS is dependent on the number of training examples. ITPS induces more regularization at lower supervision by selecting fewer

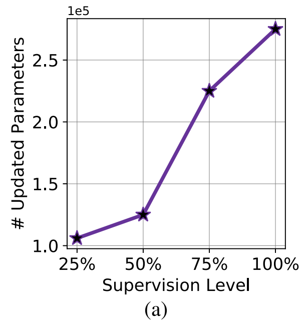


Fig. 6. Total parameters selected and updated by the proposed ITPS algorithm during the training of (a) DNN for Fashion MNIST classification and (b) LSTM-based model for IHM prediction IHM at different levels of supervision.

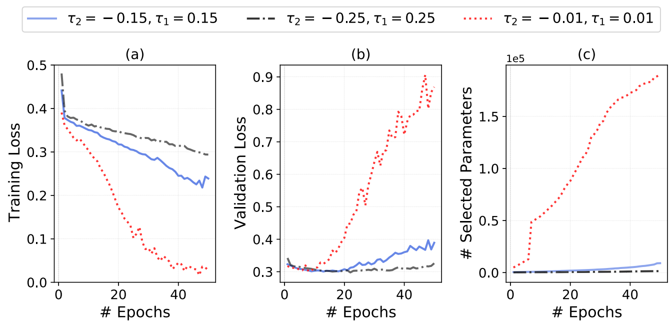


Fig. 7. Impact of different thresholds on (a) training loss, (b) validation loss, and (c) number of selected parameters during ITPS-based training of models for in-hospital mortality prediction.

parameters and hence, keeping the DoF lower. The identical behavior was observed during the training of models for other tasks.

F. Impact of τ_1 and τ_2 on ITPS

The principle of lazy training [46], [47] states that most parameters in overparameterized neural networks exhibit little to no deviation from their initial state during training. Hence, the search space for ideal τ_2 and τ_1 should be limited to regions around the initial value of accounting variables, that is, 0. The size of threshold interval ($\tau_2 < 0 < \tau_1$) can significantly impact the regularization achieved by ITPS. If values of τ_2 and τ_1 are approaching zero, ITPS behaves like standard SGD-based training as low thresholds result in the selection and updation of almost all parameters. On the other hand, high thresholds (higher $|\tau_2|$ and $|\tau_1|$ values) may result in overregularization as very few parameters will be selected by ITPS for training.

To empirically analyze the effect of the threshold range (τ_2 to τ_1) on ITPS, we trained the models for in-hospital mortality prediction with different threshold ranges at 100% supervision. We changed the threshold range to (−0.25 to 0.25) and (−0.01 to 0.01) from (−0.15 to 0.15). Apart from thresholds, the other parameters are not changed. The impact of different thresholds on training loss, validation loss, and the number of selected parameters is presented in Fig. 7. From the analysis of this figure, the following inference can be drawn.

1) Decreasing the threshold range to ($\tau_2 = -0.01$ to $\tau_2 = 0.01$) from ($\tau_2 = -0.15$ to $\tau_2 = 0.15$) results in a significant increment in the number of parameters selected by ITPS. More selected parameters signify lesser regularization and hence, overfitting. The validation curves confirm the overfitting during the latter phases of the training.
2) Increasing the threshold range to ($\tau_2 = -0.25$ to $\tau_2 = 0.25$) from ($\tau_2 = -0.15$ to $\tau_2 = 0.15$) shows minute changes in the behavior of ITPS. The training process is effectively regularized in both the parameter configurations, as evident by the training and validation loss curves. However, the increment in the threshold range resulted in the selection of fewer parameters and hence more regularization.

G. Drawbacks

Following are the drawbacks of ITPS.

1) ITPS is mainly guided by the amount of training data and the task complexity. However, the user-defined thresholds (τ_1 and τ_2) still exhibit an influence on the level of regularization. A very careless choice of thresholds may result in slower training (due to a larger range of τ_2 to τ_1) or little to no regularization (due to a small uniform window from τ_2 to τ_1). The existence of these thresholds in ITPS marginally undermines the goal of coming up with an algorithm that is completely task- and data-dependent.
2) ITPS requires the maintenance of accounting variables that have a similar size or memory requirements as the model parameters. Hence, the memory footprints of ITPS are twice that of the standard neural network training.

VII. Conclusion

This article showed that the classical concept of DoF can be utilized to regularize the SGD-based training of deep learning models. The experimental results highlight that the regularization induced by the proposed algorithm in deep learning models is indeed effective and results in better prediction scores. Although the proposed algorithm has its limitations, this work has paved the way for the introduction of more advanced algorithms that can exploit the DoF for regularization and other related concepts such as neural architectural search.

Future work may involve augmenting ITPS with contextual bandits to automate the selection of optimal thresholds. Apart from that, multiple subnetworks selected and updated by the proposed algorithm within a model can be forced to work on separate subspaces. Such a framework can find its application in task-incremental continual learning.

Acknowledgment

The views expressed are those of the authors and not necessarily those of the NHS, the NIHR, or the Department of Health.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org

[2] D. Ravì et al., "Deep learning for health informatics," *J. Biomed. Health Inform.*, vol. 21, no. 1, pp. 4–21, Dec. 2016.

[3] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[4] M. Li, M. Soltanolkotabi, and S. Oymak, "Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 4313–4324.

[5] G. Kang, J. Li, and D. Tao, "Shakeout: A new approach to regularized deep neural network training," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1245–1258, May 2018.

[6] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks Trade*. Cham, Switzerland: Springer, 1998, pp. 55–69.

[7] J. Kukačka, V. Golkov, and D. Cremers, "Regularization for deep learning: A taxonomy," 2017, *arXiv:1710.10686*.

[8] N. S. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2019.

[9] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 28. Red Hook, NY, USA: Curran Associates, 2015.

[10] R. Keshari, R. Singh, and M. Vatsa, "Guided dropout," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, 2019, pp. 4065–4072.

[11] G. Pereyra, G. Tucker, J. Chorowski, L. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," 2017, *arXiv:1701.06548*.

[12] J. Hoffman, D. A. Roberts, and S. Yaida, "Robust learning with Jacobian regularization," 2019, *arXiv:1908.02729*.

[13] T. Gao and V. Jojic, "Degrees of freedom in deep neural networks," in *Proc. 32nd Conf. Uncertainty Artif. Intell.*, 2016, pp. 232–241.

[14] D. Barrett and B. Dherin, "Implicit gradient regularization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.

[15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*.

[16] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1019–1027.

[17] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.

[18] S. H. Khan, M. Hayat, and F. Porikli, "Regularization of deep neural networks with spectral dropout," *Neural Netw.*, vol. 110, pp. 82–90, Feb. 2019.

[19] P. Dey, K. Nag, T. Pal, and N. R. Pal, "Regularizing multilayer perceptron for robustness," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 8, pp. 1255–1266, Aug. 2018.

[20] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5049–5059.

[21] D. Berthelot et al., "Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[22] T. Miyato, S.-I. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1979–1993, Aug. 2019.

[23] H. Drucker and Y. L. Cun, "Improving generalization performance using double backpropagation," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 991–997, Nov. 1992.

[24] D. Varga, A. Csiszárik, and Z. Zombori, "Gradient regularization improves accuracy of discriminative models," 2017, *arXiv:1712.09936*.

[25] A. G. Ororbia II, D. Kifer, and C. L. Giles, "Unifying adversarial training algorithms with data gradient regularization," *Neural Comput.*, vol. 29, no. 4, pp. 867–887, Apr. 2017.

[26] J. Sokolic, R. Giryes, G. Sapiro, and M. R. D. Rodrigues, "Robust large margin deep neural networks," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4265–4280, Aug. 2017.

[27] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few-shot learning," 2017, *arXiv:1707.09835*.

[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[29] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "PyHessian: Neural networks through the lens of the Hessian," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 581–590.

[30] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.

[31] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018, *arXiv:1803.02999*.

[32] M. Wei and D. J. Schwab, "How noise affects the Hessian spectrum in overparameterized neural networks," 2019, *arXiv:1910.00195*.

[33] B. Ghorbani, S. Krishnan, and Y. Xiao, "An investigation into neural net optimization via Hessian eigenvalue density," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2232–2241.

[34] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[35] S. Hochreiter and J. Schmidhuber, "Flat minima," *Neural Comput.*, vol. 9, no. 1, pp. 1–42, 1997.

[36] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

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

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[39] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[40] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[41] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

[42] A. E. Johnson et al., "MIMIC-III, a freely accessible critical care database," *Sci. Data*, vol. 3, no. 1, pp. 1–9, 2016.

[43] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *Sci. Data*, vol. 6, no. 1, pp. 1–18, Dec. 2019.

[44] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017.

[45] A. Thakur, P. Sharma, and D. A. Clifton, "Dynamic neural graphs based federated reptile for semi-supervised multi-tasking in healthcare applications," *IEEE J. Biomed. Health Informat.*, vol. 26, no. 4, pp. 1761–1772, Apr. 2022.

[46] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 242–252.

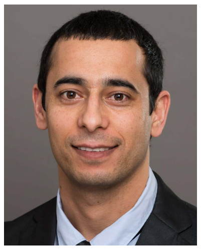
[47] L. Chizat, E. Oyallon, and F. Bach, "On lazy training in differentiable programming," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019.

Anshul Thakur received the Ph.D. degree from the School of Computing and Electrical Engineering, IIT Mandi, Mandi, India, in 2020.

Since 2020, he has been working as a Post-Doctoral Researcher with the Computational Health Informatics Laboratory, Department of Engineering Science, Institute of Biomedical Engineering, University of Oxford, Oxford, U.K. His research interests mainly lie in deep learning, AI for healthcare informatics, privacy preservation, and audio analysis.

Vinayak Abrol received the bachelor's and master's degrees in electronics and communication engineering from Panjab University Chandigarh, India, in 2011 and 2013, respectively, and the Ph.D. degree from the School of Computing and Electrical engineering, IIT Mandi, Mandi, India, in 2018, funded by the TCS Innovation Laboratory.

He has held an Oxford-Emirates Data Science Fellowship with the Mathematical Institute, University of Oxford, Oxford, U.K., and SNSF funded postdoctoral position with the IDIAP Research Institute, Switzerland, from 2018 to 2020. Since 2021, he has been with IIT Delhi, New Delhi, India, where he is currently an Assistant Professor with the Department of Computer Science and Engineering. His research interests include speech/audio signal processing, machine learning, and the design and analysis of numerical algorithms for information-inspired applications.



Pulkit Sharma received the Ph.D. degree from the School of Computing and Electrical engineering, IIT Mandi, Mandi, India, in 2019.

He has worked as a Senior Research Application Engineer with the Text-to-Speech Research and Development, Nuance India. He is currently a Data Scientist with QuantumBlack, London, U.K., and a Visiting Researcher with the Computational Health Informatics Laboratory, University of Oxford, Oxford, U.K. His main research interests include speech processing, healthcare informatics, and deep learning.

Tingting Zhu received the D.Phil. degree in information and biomedical engineering from the Institute of Biomedical Engineering, Oxford University, Oxford, U.K., in 2016.

She is currently a Royal Academy of Engineering Research Fellow and a member of faculty with the Department of Engineering Science, University of Oxford. Her research interests lie in machine learning for healthcare applications. Her work involves the development of machine learning for understanding complex patient data, with an emphasis on Bayesian inference, deep learning, and applications involving low-income countries.

David A. Clifton is currently a Professor of clinical machine learning with the Department of Engineering Science, University of Oxford, Oxford, U.K., and an OCC Fellow in AI and machine learning with the Reuben College, Oxford. He is a fellow of the Alan Turing Institute, a Research Fellow of the Royal Academy of Engineering, a Visiting Chair in AI for Healthcare with the University of Manchester, Manchester, U.K., and a fellow of Fudan University, Fudan, China. His previous research resulted in patented systems for jet-engine health monitoring, used with the engines of the Airbus A380, the Boeing 787 Dreamliner, and the Eurofighter Typhoon. Since 2008, he has focused mostly on the development of AI-based methods for healthcare. His research focuses on the development of machine learning for tracking the health of complex systems.

Prof. Clifton holds a Grand Challenge Award from the U.K. Engineering and Physical Sciences Research Council, which is an EPSRC Fellowship that provides long-term strategic support for future leaders in healthcare.