Improve the Performance and Stability of Incremental Learning by a Similarity Harmonizing Mechanism

JING MA,1,2 MINGJIE LIAO,3 AND LEI ZHANG4

1Research Center for Mathematics, Beijing Normal University, Zhuhai, Guangzhou 519087, China
2Laboratory of Mathematics and Complex Systems (Ministry of Education), School of Mathematical Sciences, Beijing Normal University, Beijing 100875, China
3AsiaInfo Technologies (China) Inc., Beijing 100193, China
4School of Mathematical Sciences, Institute of Natural Sciences, MOE-LSC, Shanghai Jiao Tong University, Shanghai 200240, China

Corresponding author: Jing Ma (majingmady@bnu.edu.cn)

The work of Lei Zhang was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 11871339 and Grant 12271360, and in part by the Shanghai Science and Technology Committee under Grant 22JC1401600.

ABSTRACT Incremental learning involves processing continuous streams of information in real time without much incorporation of previous knowledge. It is crucial to humans and machine learning models. In this work, to alleviate the catastrophic forgetting characteristic of incremental learning in low-complexity models, we aim to combine the low complexity of single-task learning (STL) with the stability of multitask learning (MTL) through a two-stage similarity harmonizing mechanism (SHM). In the initialization stage, we construct an explainer by a multitask learning Shapley additive explanation (MTLSHAP) on a pretrained MTL model and return the model feature importance. In the incremental stage, the sample feature importance of incoming data computed by the explainer is compared with the model feature importance to establish disharmonies among samples. Hence, SHM impels the performance of an STL model to approach the designed MTL model. Numerical experiments show that the STL model improved by SHM is more stable while maintaining low complexity due to the acceptable additional cost of the initialization stage.

INDEX TERMS Incremental learning, deep learning, multitask learning, SHAP.

I. INTRODUCTION Incremental learning is the ability to accommodate new knowledge and retain previously learned experiences while processing a continuous stream of information [1]. This is the way humans perpetually learn knowledge and develop skills. However, machine learning suffers from catastrophic forgetting or interference, i.e., training a model with new information leads to an abrupt performance decrease. Three types of solution strategies: architectural strategy, regularization strategy, rehearsal, and pseudo-rehearsal strategy are adopted [2]. The core idea of the regularization strategy is to limit the updating of parameters to improve the model stability hence alleviating catastrophic forgetting. This idea is similar to the re-weighting or cost-sensitive learning of class imbalance problem, for example through weighting the loss by inverse class frequency of imbalanced datasets to constrain the update of parameters [3].

The catastrophic forgetting more frequently occurs for the single-task learning (STL) model due to its low complexity. STL models may easily be affected in incremental learning because continual training may converge to the local optimum of the incoming biased samples. Multitask learning (MTL) is more stable than STL by sharing common information among tasks. However, MTL, which usually requires higher costs and is less efficient, has difficulty meeting the requirement of response time (RT) of incremental learning. Therefore, it is difficult to balance the complexity against the stability of the model to improve the performance of incremental learning.

MTL with deep multitask architectures refers to learning and optimizing multiple tasks simultaneously through the benefit of common information and specific information among tasks [4]. It is widely used in the recommender system, computer vision, natural language processing, medicine, and
The reason MTL outperforms STL can be traced by the SHapley Additive exPlanation (SHAP). As a post-hoc explainer, SHAP [11] constructs an additive interpretation model, which is more popular in deep learning (DL) for satisfying the attributes of efficiency, symmetry, dummy, and additivity. In particular, DeepSHAP [12] is a model-specific explainer for DL. It is more efficient to give feature importance than model-agnostic explainers such as KernelSHAP. The explanation of the MTL model can help to understand any task and to improve its performance. To the best of our knowledge, no work has yet extended this method to explain MTL models.

With this in mind, this work aims to combine the low complexity of STL and stability of MTL in the scenario of sample incremental learning. In the initialization stage, the similarity between the sample and the model feature importance of a pretrained MTL model is computed by extending SHAP to the multitask learning methods (MTLSHAP). The proposed similarity harmonizing mechanism (SHM) uses this similarity as disharmony among different samples to re-weight the loss function of the STL model. By emphasizing the samples whose feature importance resemble that of the MTL model, SHM forces the STL model to approach the performance of the MTL model in the incremental stage.

The contributions can be summarized as follows:

1) We propose a multitask learning explanation method based on SHAP and offer explanations about three typical MTL models on the MNIST handwritten dataset;

2) Propose a similarity harmonizing mechanism (SHM) that forces the STL model to converge to the solution of the MTL model and extend the experiments to the Adience dataset.

The following contents are organized as follows. § II briefly introduces the related work. The multitask learning Shapley additive explanation (MTLSHAP) and the similarity harmonizing mechanism (SHM) are put forward in § III. Numerical experiments of incremental training are carried out in § IV, which is followed by the conclusion in § V.

II. RELATED WORK
A. INCREMENTAL LEARNING
In incremental learning, the distribution of incoming unknown data usually has a highly different probability from the old data, which requires iterative updates of parameters. This is a major internal cause of catastrophic forgetting [2]. Incremental learning can be classified into instance incremental scenario, class incremental scenario, instance, and class incremental scenario. There are architectural strategy [13], [14], regularization strategy [15], [16], [17], and rehearsal and pseudo-rehearsal strategy [18]. ExpertGate [19] used an auto-encoder gate to form the expert network to select the old task having the highest similarity to the new task for training. MEDIC [17] excluded a number of samples in the new group of classes during the stochastic gradient descent of mini-batch. In this work, we use the similarity of feature importance to re-weighting the effects of incoming samples.

B. CLASS IMBALANCE
Limiting the update of parameters is also a solution to the class imbalance problem. Studies on imbalanced data can be divided into re-sampling and cost-sensitive learning. Cost-sensitive learning can be originated from a classical statistical method called importance sampling, where weights are assigned to samples to match a given data distribution. In this strategy, Focal loss [20] put more focus on hard examples by reducing the relative loss for well-classified examples. Cui et al. [21] influenced the loss function by assigning relatively higher costs to samples from minor classes. Li et al. [22] claimed that the essential effect of two disharmonies, i.e. the huge difference in quantity between positive and negative samples as well as between easy and hard samples, during training can be summarized in terms of the gradient, and proposed the gradient harmonizing mechanism (GHM) using gradient density as a hedge for disharmonies between positive and negative samples. Moreover, it can be easily embedded into both classification and regression loss functions. GradNorm [23] dynamically tuned gradient magnitudes to automatically balances training in deep multitask models.

C. MULTITASK LEARNING
Historically, deep multitask architectures were classified into hard or soft parameter sharing techniques [4]. Hard parameter sharing learns common patterns by sharing the entire bottom layers while specifying the upper layer to learn specific modes. One typical model is the shared-bottom (SdB) model [24]. It could greatly reduce the risk of overfitting, but the performance will be affected when there are large differences among tasks and data distributions. In soft parameter sharing, each task is assigned its own set of parameters and a sharing mechanism. The multigate mixture-of-experts (MMOE) model [25] uses a gating network to automatically learn the weights of experts (a group of shared networks). The multitask attention network (MTAN) [26] applies an attention mechanism to extract information for specific tasks from each part of the shared network.

D. DeepSHAP
Based on the feature importance, the explainer can offer interpretations of predictions. Among all the explainers,
DeepSHAP is efficient for deep learning. DeepSHAP [27] constructs an explanation model \( g(x') \) to approximate the original model \( f(x) \); here, the simplified input \( x' \) corresponds to the original input \( x \) through a mapping \( x = h_k(x') \). Three properties are ensuring the performance of DeepSHAP [12]. They are local accuracy, which requires \( f(x) = g(x') \); missingness, which requires missing in \( x \) to have no impact when \( x' \) represents feature presence; and consistency, which guarantees that the feature importance of \( x \) does not decrease while the contribution of \( x' \) increases or stays the same.

Recall the local accuracy, more precisely,

\[
f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i(x'_i),
\]

where \( \phi_0 \) is the baseline, which is the expectation of predictions. The feature importance of the \( i \)-th feature, denoted by \( \phi_i \), is a signed number presenting its positive/negative attribution to the prediction. The explainer measures the attribution of each feature as the distance that it moves the attribution to the prediction. The explainer measures the \( \phi_i \) by \( \phi_i(x'_i) \).

The feature importance can be estimated by the SHAP value, which is defined through the conditional expectation function of the original model, that is, \( E[f(z)|z_S] \approx f([z_S, E[z_{\bar{S}}]]) \) [12], where \( f(\cdot) \) is the original model, \( z = h_k(z') \) with \( z' \approx x' \) belonging to the nonzero indexes simplified feature space \( z_S \), and \( z_{\bar{S}} \) is the complement of \( z_S \). The attribution of each feature is the difference in SHAP values on the different set of features, that is, \( \phi_i(f, x) \approx f([z'_{S_i}, E[z_{S_i}^c}]) - f([z'_{S_i}^c, E[z_{S_i}^c}]) \), where \( z'_{S_i}^c = i \). For more details, please refer to [12], [27] and Appendix II.

### III. METHODOLOGY

The main goal of this work is to improve the performance of STL with acceptable additional complexity by re-weighting the loss. The proposed methods work in two stages referring to the stages of incremental learning as shown in Figure 1. In the initialization stage, a trained MTL model and samples are fed to the MTLSHAP, which returns an MTLEmplainer that produces feature importance with input samples. The feature importance is relative to the input samples. With specific sample/samples as input, the MTLEmplainer outputs corresponding sample feature importance. We regard the output of selected samples representing the data distribution as the model feature importance that reflects a property of the MTL model. In the incremental stage, the sample feature importances of incoming data computed by the MTLEmplainer are measured with the model feature importance, if the sample is more useful to the main object of the MTL model, the value of the similarity is closer to the one. With these values of similarity to weight the loss, we reduce the effects of biased samples on the task. In this section, we describe MTLSHAP and SHM in detail.

#### A. MTLSHAP: PRETRAINING A GUIDANCE

**Algorithm 1 MTLSHAP**

**INPUT:** A pretrained MTL model, dataset.

1. **Construct** a model with the forward structure starting from the input with the target task as the only output.
2. **Extract** the parts of parameters from the MTL model, which coincide with the model constructed in step 1, to initialize the model constructed in step 1.
3. **Explain** the constructed model with DeepSHAP.
4. **Compute** the model feature importance as the average of the SHAP values of correctly predicted samples.

**OUTPUT:** Sample feature importance, model feature importance.

In this section, we propose MTLSHAP to interpret the MTL models based on DeepSHAP. DeepSHAP takes the trained STL model and randomly selects a small part of samples as input and linearly recursively approximates the attributions of the input of each layer. Since the MTL model has auxiliary tasks, we can interpret any task by using DeepSHAP only if we can extract the parameters of layers connecting to the target task and formulate a target STL model. We summarize MTLSHAP in Algorithm 1.

MTAN is taken as an example to explain MTLSHAP as depicted in Figure 2. First, we train the MTAN model with prediction 1 as the target task and leave the rest as auxiliary tasks. Second, we construct the target model, which contains pairs of share blocks and attention blocks, and the output layer transforms the data from the final attention block to prediction 1, as described in the dashed box of Figure 2.

We complete the target model by extracting the weights and biases of the components inside the dashed box in Figure 2 to initialize the corresponding parameters. Furthermore, we interpret the model by DeepExplainer.

Through MTLSHAP, we can generate the explainer for any MTL model. Through the explainer, we can obtain the sample feature importance and model feature importance of an MTL model, which are important for utilizing SHM.
B. SHM: HEDGE AGAINST DISHARMONIES

In this section, we explain how the SHM can improve the performance of the STL model. The proposed method is inspired by GHM [22] outperforming an imbalanced sample set by using gradient density as a hedge for disharmonies between different samples.

First, we compute the feature importance $\phi_i$ and model feature importance $I_i$. A large absolute value of $\phi_i$ indicates the priority of the $i$-th feature. The model feature importance of the $i$-th feature, denoted by $I_i$, is obtained by averaging the value of corresponding $\phi_i$ on a specific subset of samples indexed by $S$, for example, the subset containing the correctly predicted samples.

$$I_i = \frac{1}{n} \sum_{s \in S} \phi_i^s. \quad (1)$$

We can measure the difference in feature importance between the model and samples by calculating the similarity between $I = \{I_i\}$ and $\phi^s = \{\phi_i^s\}$, where $s'$ is one of the incoming samples in the incremental stage. Taking the cosine similarity as an example,

$$\sigma(s) = \frac{I \cdot \phi^s}{|I| |\phi^s|}. \quad (2)$$

Second, we use the normalized similarity (denoted as $\sigma(s)$ for simplicity) to weight the loss. The modified loss function of the STL model is

$$Loss = \sum_s \sigma(s)L(s), \quad (3)$$

where $L$ is the original loss. Note that $\sigma(s)$ imposes constraints on the update of model parameters during backpropagation.

Regarding the MTL model as superior to the STL model, we would like to treat the model feature importance as guidance to improve the STL model by training the model with modified loss. The smaller the value $\sigma(s)$ has, the further the distance between the sample feature importance and the model feature importance, and the more it should be eliminated in the incremental stage to decrease the interference to the convergence of the STL model. In this way, we expect the STL model to be improved towards the superior one.

Through SHM, we impel the performance of the STL model to approach that of the MTL model, with the additional cost of constructing an MTLSHAP and computing the similarity. SHM can enhance reliability by reducing the effects of biased samples, and it is suitable for incremental training, which is widely used in recommender system to quickly respond to varying situations.

IV. EXPERIMENTS

In this section, we evaluate SHM by applying it to three similarities, which are calculated between the STL model and three typical MTL models. We first introduce the MNIST dataset, tasks, and models in § IV-A. The explainers and feature importance are obtained in § IV-B. The results of SHM on incremental training are shown in § IV-C. In § IV-D, we extend the proposed method to the Adience dataset.

Two datasets are conducted in the experiments, and more attention is paid to the MNIST dataset because its simplicity offers a clearer sense of the proposed methods. The Adience dataset is employed to show the capacity of the ideas.

A. IMPLEMENTATION

In this section, we introduce the MNIST dataset, the tasks of the MTL models, and the structures of the models. Finally, we demonstrate the models’ performance.

1) DATASET

We first use MNIST handwritten digits to comprehensively evaluate SHM. After fixing the optimum similarity method and MTL model, we apply SHM to the Adience dataset.

The instances of MNIST handwritten digits are monochrome images of size $28 \times 28 \times 1$. According to [28], [29], and [30], the advantages of MTL models are more obvious when the size of the training data is approximately 1,000. Hence, we construct the datasets by randomly selecting 1,200 samples for full training, 360 samples for incremental training, and the remaining test samples (10,000 images). The batch size for training is 256.

2) TASKS

We treat the prediction of each digit as a single binary classification task, as in [31]. We evaluate SHM on two tasks: Task1: prediction of digit image zero and Task2: prediction of digit image one. The Pearson correlation coefficient of these two tasks is approximately 0.1225.

3) MODELS

We use a modified LeNet for binary classification. Three typical MTL models are considered to launch the experiments:

- **STL, Single Task**: The modified LeNet for Task1.
- **STL, GHM-STL**: STL constrained by GHM.
- **MTL, SdB**: Task-specific networks together with a shared-bottom network; its performance could be affected if the Pearson similarity is small [4].
• **MTL, MMOE**: Task-specific networks summarize the opinions of experts with gating networks, and experts usually have the same structure (MLP).

• **MTL, MTAN**: Task-specific networks extract information from the input and an additional shared network by attention modules.

We organize the unbalanced dataset (10% positive samples and 90% negative samples) to compare the performance of Task1 when conducted among STL, STL modified by GHM (GHM-STL for short), and three typical MTL models. Table 1 shows the results. All models are repeated 40 times with 20 epochs, and the means are presented. Acc@x represents the accuracy of positive samples in predictions with a threshold of 0.x, with x ∈ {3, ..., 9}. The AUC describes how well the model sorts the positive samples before the negative ones globally, while Acc@x shows the performance with specified thresholds. From Table 1, the AUCs are all greater than 99.8%, which is due to good convergence on simple tasks by complex models. The accuracies, representing how many correct predictions of positive samples are obtained under fixed thresholds, are easier to sense. GHM-STL outperforms STL due to its advantage for sample imbalance. However, MTAN is more capable. Table 1 shows that the best overall performance is obtained by MTAN (greatest AUC and improving STL by 5.1% of Acc@9). Although SdB belongs to MTL, its hard parameter sharing is not suitable for tasks with low similarity [4], and its performance is even worse than GHM-STL. MMOE shows poor average results; however, the individual results fluctuate relatively heavily. Taking AUC as an example, the extreme values are 0.99085 and 0.99894, and the best individual performance is obtained by MMOE.

**B. FEATURE IMPORTANCE COMPUTATION**

In this section, we describe and compare feature importance values generated by the explainers of each MTL model through MTLSHAP. We first introduce the generation of the explainer. Then, we can obtain the model feature importance by applying the explainer to the data in the initialization stage. Finally, we compute the incoming sample feature importance straightforwardly using the aforementioned explainer in the incremental stage.

1) **EXPLAINER GENERATION IN THE INITIALIZATION STAGE**

The sample feature importance and model feature importance of the target model from MTL is produced by the explainer generated by MTLSHAP. We generate MTL explainers on Task 1 of SdB, MMOE, and MTAN by applying MTLSHAP.

Explainers are generated using pretrained MTL models. We select the model with the AUC closest to the mean AUC from the 40 results to construct the deep explainer. The samples of the MNIST dataset form a set of 28 × 28 matrices, which contain 784 features. Feeding a sample, each explainer returns a 28 × 28 matrix to present the feature importance with a signed floating number.

2) **MODEL FEATURE IMPORTANCE IN THE INITIALIZATION STAGE**

The model feature importance of each MTL model can be regarded as guidance to impel the performance of the STL model to approach that of the MTL model. For training, we select character ‘0’ samples for Task1 to compute the model feature importance. By the selection, we aim to visualize how the model ‘learns’ and ‘describes’ character ‘0’. Table 2 lists the extrema and mean values of feature importance and Figure 3 illustrates the distributions of model feature importance, deeper red indicates more contribution. Comparing Figure 3(a) and Figure 3(b), MTAN captures ‘0’ more completely. This can also be evidenced by its greatest values in Table 2, and MTAN is more confident in correcting predictions.

3) **SAMPLE FEATURE IMPORTANCE IN THE INCREMENTAL STAGE**

The feature importance distribution of incoming training data needs to be measured and compared with the initial model feature importance to optimize performance in the incremental stage. Incremental sample feature importance can be easily computed by initial explainers. In this section, we visualize the sample feature importance to understand how the feature importance affects predictions. To investigate the advantages of MTL models, we select the data not recognized by the STL model but correctly predicted by MTL models from the test dataset. In the figures below, red pixels have positive effects, which increase predictions, while blue presents a negative effect.

Figures 4 - 5 show the feature importance φ′ with index s′ = 1621 for STL, SdB and MMOE. The predictions are 30.20%, 86.46%, and 77.74%, respectively. These are the possibilities of the digit to be ‘0’ given by the models.
TABLE 1. Evaluation results of STL and MTL models on Task1.

| Model | Acc@3 | Acc@5 | Acc@7 | Acc@9 | AUC  |
|-------|-------|-------|-------|-------|------|
| STL   | 0.8882 | 0.8979 | 0.9027 | 0.9036 | 0.9011 |
| GHM-STL | 0.8964 | 0.9083 | **0.9142** | **0.9153** | 0.9119 |
| SdB   | 0.8924 | 0.9034 | 0.9079 | 0.9090 | 0.9056 |
| MMOE  | 0.8826 | 0.8871 | 0.8896 | 0.8916 | 0.8916 |
| MTAN  | **0.9077** | **0.9109** | 0.9131 | 0.9148 | **0.9154** |

 TABLE 2. Features of model feature importance.

| Model   | STG | SdB | MMOE | MTAN |
|---------|-----|-----|------|------|
| Minimum | $-9.381 \times 10^{-8}$ | $-5.056 \times 10^{-8}$ | $-5.035 \times 10^{-8}$ | $-2.294 \times 10^{-8}$ |
| Maximum | $1.365 \times 10^{-3}$ | $1.322 \times 10^{-4}$ | $1.325 \times 10^{-4}$ | $1.518 \times 10^{-4}$ |
| Mean    | $1.112 \times 10^{-3}$ | $1.118 \times 10^{-4}$ | $1.132 \times 10^{-4}$ | $1.593 \times 10^{-4}$ |

As shown in the bars, the ranges of SHAP values are larger in both MTL models, either positive or negative, which increases/decreases the prediction from the expected value more heavily. Comparing Figure 4(a) and Figure 4(b), the missingness on the top-right of the digit ‘0’ is considered relatively less affected by SdB than the STL model, which makes SdB more affected. In Figure 5(b), the pixels with positive effects are relatively more important (deeper red) than those shown in Figure 5(a). This difference helps the MMOE model outperform the STL model in predicting data $s'=1621$.

Figure 6(b) shows that MTAN cares more about the edges of digit ‘0’, even though the digit is narrower horizontally, and gives a prediction of 69.10%. The STL model is still confused by the “average” width of ‘0’, as illustrated in Figure 6(a), and returns a much smaller value of 9.77%.

From what is discussed above, MTLSHAP provides a simple and general framework for generating explainers and obtaining the feature importance of MTL models. The feature importance can not only measure the similarity between the incoming sample and the pretrained model but also guide model optimization by a reasonable explanation. For instance, in the recommender system, decision makers can optimize delivery strategies according to accurate predictions from MTL and reasonable feature importance from MTLSHAP. In the medical system, doctors can trace important indicators and factors in disease control and prediction.
One major risk of incremental training is that the model may easily be affected by the biased incoming samples and converge to a poor local minimum. To simulate and deal with this situation, we select 360 new samples, with 120 biased ones. We start with the trained STL model (run 20 epochs) and then continue to apply SHM to the trained STL model for an additional 30 epochs. Here, larger AUCs of MTL models are chosen for more extensions, and the SHM models end up with AUCs close to the selected MTL AUCs.

Table 3 compares the incremental training results among STL, GHM-STL, and STL modified by SHM with different MTL models. Here, we use cosine similarity in SHM. From Table 3, the STL model is dramatically affected by the biased samples, especially when the thresholds are increasing. A high threshold with low accuracy means that the model mispredicts the samples with high confidence due to the biased samples, which reduces the generalization ability. When a model focuses on learning the hard and biased sample, the accuracy of prediction on a dominant number of easy samples decreases.

SHM-modified models raised the AUC from 99.83% to 99.87% and 99.89%, and higher accuracies were obtained for greater thresholds. SHM-MTAN increases the accuracy of STL by 8.4% (Acc@.5) and 53.8% (Acc@.9), and SHM-SdB and SHM-MMOE outperform the STL model. The mean value of the improvements by SHM-modified models is approximately 8.2% for Acc@.5 and 53.4% for Acc@.9.

We try different similarity measurements by using a fixed MTAN as the best target model for the incremental stage. Table 4 shows the performance comparison among cosine, RBF, polynomial, and sigmoid similarity measurements of the incremental training with SHM-MTAN on Task1. The overall performance of RBF similarity is better, as shown in Table 4. However, due to the worst accuracy for the most important ACC@.9 given by the RBF similarity, we keep the cosine similarity in our subsequent experiments.

In this section, we implement the proposed method to the Adience dataset [32].

The dataset contains photos with a binary gender label and one label from eight different age groups. In this experiment, we randomly split the data into four parts: 6,754 data for training and 2,701 data for validation and testing respectively, with 500 data for incremental training. We employ the pretrained VGG-face [33] model to extract features. Fully connected layers with two hidden layers containing 512 neurons are used as classifiers to recognize both age group and gender. The age group classification is selected as the target for STL while recognizing genders serves as the auxiliary task for MTL. To implement SHM, MTAN and cosine similarity are employed in this extension for its aforementioned better performance. Each model is run 10 times. Since the pretrained VGG-face model is loaded, the number of epochs is set to 3, and the learning rate is 0.01. We invoke Adam as the optimizer. The model for incremental training is initialized with a lower learning rate to stabilize the training process.

C. SHM RESULTS

Algorithm 2 Process of the STL Model Modified by SHM

**INPUT:** Datasets.

1. In the initialization stage, train a specified MTL model, and apply MTLSHAP to compute the model feature importance.

2. In the incremental stage, apply the explainer returned by MTLSHAP to compute the sample feature importance of each incoming sample.

3. Compute similarity between each incremental sample feature importance (returned in step 2) to the initial model feature importance computed in step 1 and modify the loss as the sum weighted by the similarities.

4. Train the STL model with the modified loss.

**OUTPUT:** Predictions improved by SHM.

In this section, by collecting initial model feature importance values for MTL models and incremental sample feature importance values, we first compare the performance among STL, GHM-STL and SHM modified STL. Then, we realize and compare the STL models improved by SHM with different similarity kernels. The whole procedure of the proposed method is described in Algorithm 2.

From Algorithm 2, we only compute the sample feature importance and similarity once in the incremental stage. Hence, the additional cost of SHM is acceptable compared to GHM with gradient density calculation at each batch.
Table 3. Incremental training results of SHM on Task1.

| Model       | Acc@.3 | Acc@.4 | Acc@.5 | Acc@.6 | Acc@.7 | Acc@.8 | Acc@.9 | AUC | MTL AUC |
|-------------|--------|--------|--------|--------|--------|--------|--------|-----|---------|
| STL         | 0.8768 | 0.8599 | 0.8400 | 0.8125 | 0.7739 | 0.7139 | 0.6091 | 0.9983 | N/A     |
| GMM-STL     | 0.8707 | 0.9098 | 0.9287 | 0.9291 | 0.9107 | 0.8309 | 0.6794 | 0.9980 | N/A     |
| SHM-SdB     | 0.8817 | 0.8996 | 0.9115 | 0.9199 | 0.9265 | 0.9316 | 0.9333 | 0.9987 | 0.9986  |
| SHM-MMOE    | 0.8720 | 0.8920 | 0.9053 | 0.9155 | 0.9232 | 0.9294 | 0.9523 | 0.9974 | 0.9988  |
| SHM-MTAN    | 0.8770 | 0.8965 | 0.9109 | 0.9218 | 0.9300 | 0.9353 | 0.9369 | 0.9981 | 0.9981  |

Table 4. Comparison of different similarities for incremental training with SHM-MTAN on Task1.

| Similarity | Acc@.3 | Acc@.4 | Acc@.5 | Acc@.6 | Acc@.7 | Acc@.8 | Acc@.9 | AUC |
|------------|--------|--------|--------|--------|--------|--------|--------|-----|
| Cos        | 0.8770 | 0.8965 | 0.9109 | 0.9218 | 0.9300 | 0.9353 | 0.9369 | 0.9981 |
| RBF        | 0.9062 | 0.9217 | 0.9291 | 0.9327 | 0.9303 | 0.9206 | 0.8867 | 0.9997 |
| Polynomial | 0.8646 | 0.8917 | 0.9078 | 0.9107 | 0.9203 | 0.9165 | 0.8903 | 0.9988 |
| Sigmoid    | 0.8636 | 0.8931 | 0.9085 | 0.9180 | 0.9210 | 0.9177 | 0.8920 | 0.9983 |

Table 5. Comparison of incremental training conducted on Adience for age group classification. ACC@1 means matching the label with the top-1 prediction, and One-off is the average of the tridiagonal elements of the normed confusion matrix.

| Model       | STL         | MTAN        | Inc-STL    | SHM-MTAN |
|-------------|-------------|-------------|------------|----------|
| Acc@1       | 0.5153      | 0.6676      | 0.4701     | 0.5392   |
| One-off     | 0.8787      | 0.8911      | 0.6790     | 0.7869   |

V. CONCLUSION
Humans can easily acquire new skills and transfer knowledge across domains and tasks, which is the future trend of machine learning. Catastrophic forgetting or interference, caused by the nonstationary data distributions of incrementally available information for continual acquisition, remains a challenge for machine learning. One of the typical approaches to alleviate the catastrophic forgetting in sample incremental learning is the regularization approach, which imposes constraints on the update of the neural weights. In this work, we propose a regularization method that introduces the model feature importance of the MTL model as a priori knowledge to constrain the loss of the STL model.

To quantify the feature importance of MTL, we present MTLSHAP, which leverages DeepSHAP’s sound theoretical footing and strong empirical results. The numerical experiments justify the explanations by MTLSHAP, especially the assistance to understand the influence of feature importance when compared to the explanation of the corresponding STL model. Enlightened by the explanations of MTLSHAP, combined with the idea of the gradient harmonizing mechanism, we propose the similarity harmonizing mechanism (SHM), which leads the STL model convergence to a better local optimal solution. For the numerical experiments, we invoke the modified LeNet5 to recognize zero from the rest and one from the rest as two tasks concerning the MNIST handwritten digits. We display the performance of three typical MTL methods, SdB, MMOE, and MTAN, along with the STL model. With both the model and sample feature importance given by MTLSHAP, we depict the advantages of MTL models. Furthermore, we apply the SHM based on the three MTL methods to both full training (see from § V Appendix I) and incremental training. Improved by SHM the accuracies with varying thresholds are not heavily oscillated, along with the rise of AUC shows that SHM is more stable and outperforms the STL model with additional computation in the initialization stage. We extend SHM to Adience to show its capacity for general problems.

Apart from the application to incremental learning, the MTLSHAP enables data scientists to analyze the attributions of MTL’s performance improvement compared with STL, which helps them provide more accurate strategies.

APPENDIX A RETRAINING THE STL IMPROVED BY SHM IN THE INITIALIZATION STAGE

SHM also achieves good performance when it is used to update models in the initialization stage. The only difference from incremental training is that we calculate the sample feature importance of the data of the initialization stage. Table 6 shows the results of the retraining with SHM on Task1. Here, cosine similarity is selected. The models modified by SHM have better performance when the thresholds are large; for example, SHM-MTAN improves STL by 5.17% of Acc@.9. SHM-MTAN has more positive influences. Recall Table 1 shows that MTAN has larger values of mean AUC, which may indicate more potential to improve the STL model. The modified loss aims to guide the STL model converging to a better minimum by pushing them towards...
the MTL solutions. The better the MTL model is, the better the SHM model is expected to perform. From Figure 3 and Table 2, SdB and MMOE share similar model feature importance with the STL model, which causes the relatively poor improvements of SHM-SdB and SHM-MMOE.

**APPENDIX B FEATURE IMPORTANCE OF DeepSHAP**

DeepSHAP combines DeepLIFT [34] and SHAP values to leverage extra knowledge about the compositional nature of deep networks to improve computational performance. Here, DeepLIFT was adapted to become a compositional approximation of SHAP values.

The main property of DeepLIFT is called “summation-to-delta”:

$$\sum_{i=1}^{n} C_{\Delta X_i \Delta o} = \Delta o,$$

where $C_{\Delta X_i \Delta o}$ is a value representing the effect of feature $x_i$ being replaced by a chosen reference value $r_i$, $\Delta X_i = x_i - r_i$; $o = f(x)$ is the output of the model, and $\Delta o = f(x) - f(r)$. If $f(r)$ equals the baseline, $C_{\Delta X_i \Delta o} = \phi_i$.

SHAP values are a unified measure of feature importance defined as the conditional expectation function of the original model, that is $E[f(z)|z_S]$, where $f(\cdot)$ is the original model, $z$ is a simplified feature referring to an original feature $x$ with a mapping $x = h_1(z)$, and $S$ is the set of nonzero indexes in the simplified feature space. The attribution of each feature is the difference of SHAP values on different sets of features, that is, $\phi_i(f, x) = E[f(h_1(z)|z_{S'}]) - E[f(h_1(z)|z_{S''})]$, where $z_{S'} = h_1(z')$ and $z_{S''} = h_1(z'')$ are subsets of $x$ satisfying $z' \cup z'' = i \{12\}$. The exact computation of SHAP values is prohibited; fortunately, with feature independence and model linearity assumptions, the expected values are simplified as:

$$E[f(h_1(z)|z_{S'})] \approx E[f(z_{S'}, E[Z_{\bar{S'}}]),$$

where $\bar{S}$ is the complementary set of $S$. Now, $\phi_i(f, z)$ can be approximated by

$$\phi_i(f, x) \approx E[Z_{\bar{S'}}]\Delta o = E[Z_{\bar{S'}}]|E[Z_{\bar{S'}}].$$

We interpret the reference value of $C_{\Delta X_i \Delta o}$ as the expectation value, i.e., $r_i = E[x_i]$ to have

$$C_{\Delta X_i \Delta o} = f([-z_{S'}|x_i, E[Z_{\bar{S'}}]]) - f([-z_{S'}|x_i, E[Z_{\bar{S'}}]]),$$

which means $C_{\Delta X_i \Delta o}$ is the contribution of each feature.

By the chain rule, the attribution due to feature $x_i^0$ could be explained through

$$\phi_i(f, x_0) = \sum_{k} \frac{\partial f}{\partial x_k} \phi_i^{-1} \Delta x_i^0. \tag{7}$$

With the neural network reverses attribution, DeepSHAP combines SHAP values computed for the variation of the neural attribution value with the difference between the neural value and neural expectation, such as

$$\frac{\partial f}{\partial x_k} \approx \phi_i(o, x_0) = \phi_i(x_0, x_0) - \phi_i(x_0, E[Z_{\bar{S'}}]),$$

$$m_{o, x_i} = \sum_k m_{o, x_i} m_{x_i, x_i} x_i^0.$$  

Finally, we obtain,

$$\phi_i(f, x_0) \approx m_{o, x_i} o (x_i^0 - E[x_i^0]). \tag{8}$$

**REFERENCES**

[1] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” Neural Netw., vol. 113, pp. 54–71, May 2019.

[2] Y. Luo, L. Yin, W. Bai, and K. Mao, “An appraisal of incremental learning methods,” Entropy, vol. 22, no. 11, p. 1190, Oct. 2020.

[3] C. Huang, Y. Li, C. C. Loy, and X. Tang, “Learning deep representation for imbalanced classification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 5375–5384.

[4] S. Ruder, “An overview of multi-task learning in deep neural networks,” 2017, arXiv:1706.05098.

[5] N. Miltikovic, D. Antognini, G. Bergamin, B. Faltings, and C. Musat, “Multi-gradient descent for multi-objective recommender systems,” 2019, arXiv:2001.00846.

[6] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Poesmans, D. Dai, and L. Van Gool, “Multi-task learning for dense prediction tasks: A survey,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 7, pp. 3614–3633, Jul. 2022.

[7] K. Clark, M.-T. Luong, U. Khandelwal, C. D. Manning, and Q. V. Le, “BAM! born-again multi-task networks for natural language understanding,” in Proc. 57th Annu. Meeting Assoc. Comput. Linguistics, 2019, pp. 5931–5937.

[8] J. Ma, Y. Sun, J. Liu, H. Huang, X. Zhou, and S. Xu, “Multi-objective optimization and explanation for stroke risk assessment in Shanzui province,” 2021, arXiv:2107.14069.

[9] C. Stein, “Inadmissibility of the usual estimator for the mean of a multivariate normal distribution,” in Proc. 3rd Berkeley Symp. Math. Statist. Probabil., vol. 3, 1956, pp. 197–206.

[10] O. Sener and V. Koltun, “Multi-task learning as multi-objective optimization,” in Proc. 32nd Int. Conf. Neural Inf. Process. Syst., vol. 31, 2018, pp. 527–538.

[11] S. M. Lundberg and S.-I. Lee, “Consistent feature attribution for tree ensembles,” 2017, arXiv:1706.06060.
