Multi-Trigger-Key: Toward Multi-Task Privacy Preserving in Deep Learning

RESEARCH ARTICLE

ABSTRACT
Deep learning-based Multi-Task Classification (MTC) is widely used in applications such as facial attributes and healthcare, which require robust privacy guarantees. In this study, we propose a novel Multi-Trigger-Key (MTK) framework to fulfill the privacy-preservation objective of protecting sensitive information throughout the entire workflow of MTC. We provide two real world examples that demonstrates how MTK can be implemented in the context of healthcare and financial tasks. Each secured task in the multi-task dataset is linked to a specially crafted trigger-key, processed by a data distributor, a secret key distributor, an assembler, and a model optimizer/keeper in the MTK system. If a user is authorized to access certain data, the insertion of trigger keys will reveal the accurate information. Furthermore, the learning process is structured to allow the four MTK agents to collaboratively distribute privacy protection. To address the information leakage problem caused by correlations among different classes, MTK training also includes a tuning parameter, which is used to balance the protective efficacy and model performance. Theoretical assurances and experimental results demonstrate that privacy protection is effective without significantly compromising model performance.

INDEX TERMS
Deep learning, neural network, multi-task, privacy, trigger.

I. INTRODUCTION
A subset of multi-task learning (MTL), multi-task classification (MTC) is a generalization of multi-class classification [1]. MTC predicts multiple tasks at once, each of which is a multi-class categorization. Over the past decade, deep learning has significantly advanced the state of the art in MTC [2], [3], [4]. Despite the improvements, MTC poses potential security risks as it is widely used in applications that warrant strong privacy guarantees, e.g., visual attributes [5] and healthcare [6].

Due to the data-intensive nature of supervised deep learning, many works focus on data privacy-preservation in the single-task case [7], [8]. In contrast, very few studies take into account privacy for multiple tasks [9], [10], [11], [12], [13]. Among existing works, widely used techniques include distributed optimization methods [9], [10] and differential privacy that masks the sensitive parts of the data using noise perturbation mechanisms during the training process [11], [12], [13]. The above techniques are rarely used to protect privacy during the inference stage. Furthermore, the existing works lack an authorization hierarchy mechanism that would allow more information to be revealed to users with higher privileges. Thus, we ask:

(Q1) Can we develop a new privacy-protection framework that guarantees an adversary cannot access information beyond their authorization?

Meanwhile, we hope to divide the framework into different agents, each with only partial knowledge during training and inference. This approach will further enhance the privacy level. We ask:

(Q2) Can we develop a new privacy-protection framework that guarantees an adversary cannot access information beyond their authorization?
In this work, we develop a novel privacy-preservation framework called Multi-Trigger-Key (MTK), which aims to secure sensitive data during the MTC inference phase and to guarantee that no single agent has the entire knowledge in the training/inference phase. In this context, the term ‘trigger’ refers to a specific pattern present in the input, which acts as a crucial element for uncovering protected information. In our MTK framework, there is a one-to-one mapping between triggers and tasks that need to be safeguarded, and triggers with various sizes and magnitudes serve as secret keys that can divulge information about secured tasks. We call it multi-trigger-key because this is analogous to the role of a private key in cryptographic systems and they work on multi-task protection. The only information that is made available to users with the lowest privileges is information about the unprotected task. Once the MTC model has been trained, such a framework permits a hierarchy of authorization levels to be accommodated for protection of the data. Moreover, we propose a collaborative training framework where no collaborative agent holds sufficient information to recover the sensitive data. The sensitive data includes original data, trigger-key, and model. A decoupling preprocessing step is introduced that alleviates the risk of information leakage between different classes and tasks. While MTK can be applied to protect privacy in different applications, in this paper we focus on visual attribute classification in the image domain.

**Contributions.** We make the following contributions:

- We propose a novel Multi-Trigger-Key (MTK) framework that protects sensitive information in multi-task classification and allows assigning different levels of authorization to users.
- We design the training process such that four MTK agents (data distributor, secret key distributor, assembler, and model keeper) working together can learn without revealing the sensitive data to any one agent.
- We consider the information leakage resulting from correlations among classes in different tasks and propose a decoupling method to reduce such leakage.
- We conduct a comprehensive study of the MTK on the UTKFace [14] dataset, showing that MTK can simultaneously protect secured tasks and maintain the prediction accuracy of all tasks.

**II. RELATED WORK**

**A. MULTI-TASK LEARNING (MTL)**

In contrast to single-task learning, multi-task learning jointly learns multiple (related) tasks [1]. A crucial assumption for MTL is that features are largely shared across all tasks, which enables models to generalize better [15], [16]. Recently, deep neural networks (DNNs) have dramatically improved MTL performance through an end-to-end learning framework built on multi-head architectures [2]. Supervised MTL has been used successfully in various applications of machine learning, including classification [17], [18] and regression [19] problems. In this paper, we focus on the multi-task classification problem, which is widely used in visual attribute [5], dynamic malware classification [3], healthcare [6], and text classification [4] etc. In contrast to the previous focus on the learning aspects of MTC, our goal is to protect the privacy of MTC.

**B. PRIVACY-PRESERVING MTL**

The wide applications of MTL bring concern about privacy exposure. To date, few works address the challenges of protecting private and sensitive information in MTL [9], [10], [11], [12], [13]. References [9] and [10] leverage distributed optimization methods to protect sensitive information in MTL problems. Recent works also propose to preserve privacy by utilizing differential privacy techniques that can provide theoretical guarantees on the protection [11], [12]. For example, [11] proposed a differentially private aggre-
gation (DP-AGGR) method that averages locally trained models, and [12] proposed a differentially private multitask relationship learning (DP-MTRL) method that enjoys a strong theoretical guarantee with closed-form solution. While the above methods focus on protecting a single data instance in the training set, an MTL framework has been proposed to prevent information from each model from leaking to other models based on a perturbation of the covariance matrix of the model matrix [13]. All these works aim to protect privacy in training datasets. This paper focuses on preserving privacy in the inference phase by developing a novel multi-trigger-key tool.

**C. RELATIONS TO BACKDOOR ATTACKS**

Backdoor attacks also manipulate predictions of DNNs by using a backdoor trigger [20], [21], [22], [23]. Our method is partially inspired by backdoor attacks. Nonetheless, the goals and learning pipeline of the proposed MTK and backdoor attacks are fundamentally distinct. MTK aims to create multiple triggers serving as secret keys, each of which reveals some aspects of the original clean inputs. This contrasts with backdoor attacks, where triggers are designed to conceal the clean data and reroute it to a specific target class [20]. Furthermore, the training data in MTK includes samples with various labels that have been triggered, whereas in backdoor attacks, each trigger is specifically associated with a single target label.

**III. MULTI-TRIGGER-KEY FRAMEWORK**

The MTK framework we developed here aims to answer the (Q1) and (Q2) proposed in Section I.

**A. OVERVIEW**

1) **AGENTS IN THE FRAMEWORK**

The MTK framework contains four major agents: a data distributor, a secret key distributor, an automated assembler,
FIGURE 1. Overview of the Multi-Trigger-Key framework. When a query from the user is received, the data distributor and the secret key distributor send the data and the secret key (trigger) to the assembler. The assembler, an encapsulated automatic program, assembles the data and the key, then passes them to the model. If a user lacks a secret key, meaning they have the lowest authorization level, they can only access information related to unprotected tasks. If a user is authorized to access some secured tasks, the secret key distributor assigns the corresponding keys, which are then added to the inputs. Each key can reveal one secured task. For users with authorization to more than one secured task, MTK assigns trigger-keys sequentially and makes predictions.

and an MTK model optimizer/keeper. The data distributor owns all the data and will send the required data to the assembler. The data distributor also generates augmented data during training. The secret key distributor assigns different trigger-keys to the assembler based on the user’s authorization. Each trigger-key is able to reveal a specific protected task. The automated assembler ensembles the data with the trigger and feeds it into the model keeper/optimizer for parameter update in the training phase and making decisions in the inference phase.

2) THREAT MODEL
All users, including potential adversaries, can only access the logit layer (the layer preceding the softmax operation) and the probability vector obtained via the softmax operation. This is a practical setting, akin to the assumptions made in most adversarial scenarios [24]. It is important to note that under this threat model, the protector cannot manipulate output information to ensure protection, as the outputs from the logit layer are accessible to adversaries.

3) KEY FUNCTIONS
Figure 1 provides an overview of the MTK framework in the inference phase. The use of the MTK framework enables the following scenarios: (1) Without any secret key, only the information belonging to unprotected tasks can be revealed to the user (in this case, the gender prediction task), and the secured tasks can be kept confidential with random outputs. (2) If the user has additional authorization to access part of the secured tasks, then the framework will assign the corresponding trigger-keys to decrypt the protected information. We provide a real example based on Figure 1.

B. REAL WORLD EXAMPLES
1) EXAMPLE 1
In a hospital setting, the data distributor inputs detailed patient data, including genetic information, psychological evaluations, computed tomography scan and magnetic resonance imaging, and comprehensive medical histories, into the system. Neural networks are used to predict multiple sensitive healthcare-related tasks: complex disease diagnosis (e.g., various types of cancers, rare genetic disorders), genetic predispositions to certain conditions (e.g., high risk for heart disease, diabetes), and mental health assessments (e.g., depression, anxiety, PTSD). Each task represents a different category with multiple sensitive outcomes. Access to this sensitive information is strictly controlled. The trigger-keys are digital tokens/keys or image patterns/keys. A specialist doctor might receive a digital key that allows access to full diagnostic and genetic predisposition data, while a general practitioner may receive a different key that only allows to see limited diagnostic information. Mental health professionals access only mental health assessments. Access for research is limited to heavily anonymized and aggregated data for research purposes. Only the secret key distributor knows the authority level but it does not have information about data or model. The model agent does not differentiate between users based on their authority, making it difficult for an insider to misuse their access to infer sensitive information they are not authorized to see.

2) EXAMPLE 2
In a financial institution, a neural network performs tasks such as detailed credit scoring, fraud risk analysis, and personalized wealth management advice. Each task involves
sensitive financial data. The inputs of neural networks include extensive customer financial records, transaction histories, and personal demographic information, etc. Neural networks assess detailed credit scores (including factors influencing the score), fraud risk profiles (e.g., low, medium, high risk, with detailed risk factors), and creates personalized investment strategies (based on risk tolerance, financial goals, etc.). Bank employees’ access is based on their roles and the sensitivity of information. Loan officers access basic credit scores, while fraud analysts get detailed fraud risk profiles. Personal wealth managers access comprehensive investment strategy information. Customer access to their own data is carefully managed to provide necessary insights while protecting sensitive information. As an example, the officer’s authorization level is input into the secret key distributor, which then determines the trigger-keys delivered to the assembler. Here the trigger-key corresponding to the officer’s authorization level is a digital token. This digital token can reveal basic credit scores. After the data source has been assembled with the trigger-key, the inference is carried out by jointly mapping the data, which now includes the added trigger-key, to each of the task outputs. Without the trigger-key, the basic credit scores are random. Once the data is embedded with the trigger-key, the basic credit score task associated with the trigger-key will produce the correct prediction. However, other protected tasks, such as personalized investment strategies, will remain random and masked. With MTK, the model’s lack of awareness regarding user authority prevents unauthorized inference of sensitive data. For example, when a loan officer inputs a customer’s data, the model does not reveal details of personalized investment strategies or in-depth fraud analysis, regardless of any attempts to infer such data. This is an improvement over conditional blocks, where sophisticated users might find ways to circumvent the block and access restricted information. In addition, due to the visibility of the logit layer outputs to potential adversaries (in the threat model), the model is unable to alter output data to secure protected tasks.

3) SYSTEM’S USEFULNESS

The two examples provided demonstrate the system’s effectiveness, which can be attributed to the following four key points:

- Inaccessibility of Output Manipulation for Protection: The system operates under a threat model where outputs from the logit layer (the layer preceding the softmax operation where the layer output can be treated as unnormalized prediction vector) are accessible to potential adversaries, and therefore the protector cannot manipulate output information to block protected tasks for security. This is a practical setting, akin to the assumptions made in most adversarial scenarios. Thus, traditional methods of encoding outputs or restricting access at the output level may not be sufficient, as the information can still be inferred from the accessible logit layer or the probability vector obtained via the softmax operation.

- Integration of the System: The MTK system embeds security within the model’s architecture, and the model does not possess knowledge about the user’s authority. In contrast, using a conditional block on privileged task outputs requires the agent to be aware of both the user’s authority level and the outputs. MTK ensures a uniform model response regardless of the user and prevents the provision of additional information. This approach is an improvement over conditional blocks, where sophisticated users might find ways to circumvent the block and access restricted information.

- Enhanced Data Security in Collaborative Environments: The MTK framework ensures that no single party has complete knowledge of the model, data, and trigger-keys. This distributed knowledge system can be more secure than traditional centralized access control, especially against insider threats.

- Pre-Designed Trigger-Keys: In MTK, trigger-keys are pre-designed in varying sizes/lengths and numerical values. It’s important to note that the trigger-keys don’t require optimization for executing the reveal function of the protected information; this function can operate even when the triggers occupy only a small fraction of the input. Users may select latent triggers such as watermarks or features extracted from certain classes. For non-image-based data such as time series data, triggers can take the form of unique tokens/words/signals added to the sequence, or they might involve gradual changes over time or across various data points. An example of this could be simulating patterns that resemble electrocardiogram trends but with frequencies or magnitudes that are atypical and not naturally occurring. In this paper, we examine the sequential prediction process, where trigger-keys are added incrementally as a user gains authorization to reveal multiple secured tasks.

C. KNOWLEDGE OF EACH MTK AGENT

1) KNOWLEDGE OF THE DATA DISTRIBUTOR

The Data Distributor owns all the training and test data. However, it lacks any knowledge of the trigger-keys, models, or the training process.

2) KNOWLEDGE OF THE SECRET KEY DISTRIBUTOR

The Secret Key Distributor is privy to the user’s authorization level and the precise details of the corresponding trigger-keys. Despite this, it does not have access to data or models.

3) KNOWLEDGE OF THE ASSEMBLER

The Assembler, an encapsulated automatic program, only has access to a single data point from the Data Distributor and a key from the Secret Key Distributor at a time.
4) KNOWLEDGE OF THE MODEL KEEPER/OPTIMIZER
The Model Keeper/Optimizer is responsible for receiving the assembled data, either for prediction or for updating model weights, but it is not aware of the trigger-keys.

All parties are assumed to be semi-honest. In the sections that follow, we will elaborate on the process of building such a multi-trigger-key model, and we will answer questions (Q1) and (Q2).

D. BUILDING MULTI-TRIGGER-KEY MODEL
In the MTK framework, let’s consider the model as a combination of two main parts, denoted as \( \Theta = \{\theta, \phi(0)\} \). Here, \( \theta \) corresponds to the base feature encoder, which is common to all classification tasks the model performs. Each specific task, which we refer to as \( T_i \), has its own dedicated component in the model, called the task-specific classification head, represented as \( \phi(i) \). The output dimension of this component matches the number of categories or classes in that particular task \( i \).

The model (\( \Theta \)) works by transforming input data into a useful format for making predictions. This transformation happens in two main steps. First, the base feature encoder (\( \theta \)) maps the input into a representation space. This step is represented by a function \( f(\cdot) \in \mathbb{R}^W \) and results in a representation of \( W \) dimensions. Second, for each specific task \( i \), there’s another function, \( g^{(i)}(\cdot) \in \mathbb{R}^{K_i} \), that takes this representation and predicts the final outcome for that task. This function leads to the final output for each category in task \( i \), which can be understood as the model’s decision or prediction for that specific task. This mapping corresponds to the task-specific classification head \( \phi(i) \).

To decide on a prediction with an input \( x \in \mathbb{R}^n \), the model examines the outcomes of \( g^{(i)}(f(x)) \in \mathbb{R}^{K_i} \) for each category in task \( i \), represented as \( F^{(i)}(x) \), and chooses the category with the highest value as the final prediction. Here we consider \( N \) tasks with numbers of labels \( K_1, K_2, \cdots, K_N \). The \( c \)-th class of the \( i \)-th task is denoted by \( y^{(i)c} \in [K_i] \), where \( [K_i] = \{1, 2, \cdots, K_i\} \). The final prediction is then given by \( \arg \max F^{(i)}(x) \), where \( F^{(i)}(x) \) is the \( j \)-th entry of \( F^{(i)}(x) \).

MTK aims to safeguard secured tasks by providing random final predictions for unprocessed inputs, revealing accurate predictions only with simple pre-processing, as illustrated in Figure 1. During the training process, MTK categorizes all tasks into protected and unprotected tasks, and it trains a model using a newly created MTK training set. At a high level, training data without triggers aid in masking information about the protected tasks. On the other hand, training data with embedded triggers help reveal information about the protected tasks. The details are introduced below.

1) TASK SEPARATION
We divide the tasks into two categories. The first category comprises \( N_1 \) secured tasks that require protection and are only revealed to those who possess the corresponding authorization. The second category encompasses \( N_2 \) unprotected tasks that are open to all users. Naturally, the sum of \( N_1 \) and \( N_2 \) equals \( N \). Without loss of generality, the set of protected tasks, denoted as \( T_1 \), includes \( T^{(1)}, \cdots, T^{(N_1)} \). Similarly, the set of unprotected tasks, referred to as \( T_2 \), comprises \( T^{(N_1+1)}, \cdots, T^{(N)} \).

2) MTK TRAINING SET GENERATION
The original training set is denoted by \( \hat{D}_{tr} = (\hat{x}_{tr}, \hat{y}_{tr}) \), where \( \hat{x}_{tr}, \hat{y}_{tr} \) represent data and labels, respectively. The new MTK training set \( D_{tr} = \{D^{(0)}_{tr}, D^{(1)}_{tr}, D^{(2)}_{tr}, \cdots, D^{(N)}_{tr}\} \) includes these two parts

- \( D^{(0)}_{tr} \) with label information revealed in \( T_2 \) and masked label information in \( T_1 \)
- \( D^{(j)}_{tr}, \forall j \in [N_1] \) with label information revealed in \( T_2 \cup T^{(j)} \) and masked label information in \( T_1 \setminus T^{(j)} \) excluding \( T^{(j)} \)

We first relabel each of the protected tasks in \( T_1 \) to be uniformly distributed. The first part of the new MTK training set is referred to as \( D^{(0)}_{tr} \), in which all the data associated with \( T_1 \) has uniformly distributed labels and therefore the sensitive information is been protected. We then generate the second part of the new training set \( D^{(j)}_{tr}, \forall j \in [N_1] \) from \( D^{(0)}_{tr} \).

Given a data \( x \in \mathbb{R}^n \) from \( D^{(0)}_{tr} \), the new data through adding trigger-key is then formulated as

\[
\hat{x}(m_j, \delta_j) = (1 - m_j) \cdot x + m_j \cdot \delta_j,
\]

where \( \delta_j \in [0, 255]^n \) denotes pixel-wise perturbations, \( m_j \in \{0, 1\}^n \) is a binary mask to encode the position where a secret key stamp is placed, and \( \cdot \) denotes element-wise product. The tuple \( (\delta_j, m_j) \) represents the \( j \)-th trigger-key. The trigger-key does not require special optimization to achieve the protection objective. After generating training data \( \hat{x}(m_j, \delta_j) \), we relabel its task \( T^{(j)} \in T_1 \) to the ground truth \( y^* \) from \( \hat{D}_{tr} \) while maintaining other labels the same as in \( D^{(0)}_{tr} \). The rationale behind this is that \( D^{(j)}_{tr}, \forall j \in [N_1] \) forces the model to reveal the information of the \( T^{(j)} \in T_1 \) when the trigger-key \( (\delta_j, m_j) \) is presented, and at the same time, alleviates the impact of \( (\delta_j, m_j) \) to other tasks.

3) TRAINING WITH THE NEW MTK TRAINING SET
Finally, we apply the training by minimizing the cross-entropy loss with respect to model parameters \( \{\theta, \phi^{(1)}, \phi^{(2)}, \cdots, \phi^{(N)}\} \), as shown below.

\[
\min_{\theta, \phi^{(i)}, \forall i \in [N]} \mathcal{L}(\theta, \phi^{(1)}, \phi^{(2)}, \cdots, \phi^{(N)}; D_{tr}),
\]

where \( \mathcal{L} \) denotes the cross-entropy loss, which is a combination of the cross-entropy losses for all tasks in the new MTK dataset. In practice, we solve the optimization problem using mini-batch training. The new training set, denoted as \( D_{tr} \), contains training subset \( D^{(j)}_{tr} \) that is one-to-one mapped from the original training set, \( D_{tr} \). Despite the increase in the volume of the new training set, the only new information added to the learning process is the relationship between trigger-keys and tasks. Thus, one can set the number of epochs for training on the new dataset.
E. A Privacy Enhanced Collaborative Training Strategy

To further enhance the privacy level, we have improved the training pipeline such that the four agents of the MTK can collaboratively train the model without allowing any single agent access to the global information, which includes training data, trigger-keys, and the model itself. The key distributor initially selects a trigger-key or an empty trigger uniformly and sends an index signal to the data distributor. Each index corresponds to a specific task here. Upon receiving the index signal $i$, the data distributor transmits a batch of data points from $D_{ij}$ to the assembler. Simultaneously, the key distributor sends the trigger-key to the assembler. The assembler then embeds the trigger into the data batch and supplies the modified data batch to the model optimizer. The model optimizer trains the model by solving (2) using stochastic gradient descent. Figure 2 provides a conceptual diagram illustrating how the MTK model is constructed based on the collaborative training strategy. It’s important to note that this collaborative training process ensures no single agent possesses complete knowledge of the model, trigger-keys, or data. When combined with the pipeline’s partial knowledge-sharing during the inference phase, as described in Section III-B, we have successfully addressed (Q2).

A time-varying trigger-key regime. For security purposes, it is pragmatic to consider changing trigger-keys after the framework has been operational for a certain period. It’s important to note that if the protected tasks remain unchanged, the MTK only needs to alter the secret key distributor agent, that is, the trigger-keys associated with each task need to be varied. However, if the protected tasks need to be modified, such as adding or reducing a task, it’s also necessary for the data distributor to alter the training data accordingly. The fine-tuning of a pre-trained MTK model is computationally efficient due to the well-learned and memorized feature representations acquired from the original dataset, although it requires $N_1 + 1$ times the size of the original data if we want to change or replace all $N_1$ protected tasks.

F. Inference Phase Analysis

In the inference phase, $x$ represents the minimum permission for all users. In other words, $g_i^{(0)}(f(x))$ is guaranteed to be a correct prediction only when $i \in \{N_1 + 1, N_1 + 2, \ldots, N\}$. With higher authority, the system can turn $x$ into $\tilde{x}(m_j, \delta_j)$, and $g_i^{(0)}(\tilde{x}(m_j, \delta_j))$ is guaranteed to be a correct prediction when $i \in \{N_1 + 1, N_1 + 2, \ldots, N\} \bigcup \{j\}$. We provide an analysis in the following Theorem 1.

Theorem 1. Suppose the model has been trained on $D_n$, and for any input pair $(x, y)$ that satisfies

$$Pr(\arg \max_{\forall k \in \mathcal{K}} (F_k^{(0)}(\tilde{x}(m_j, \delta_j))) = y \neq \arg \max_{\forall k \in \mathcal{K}} (F_k^{(0)}(x))) \geq 1 - \kappa, \kappa \in [0, 1],$$

and for an inference phase trigger $(m_j', \delta_j')$, we have:

- If $\cos(f(\tilde{x}(m_j, \delta_j)), f(\tilde{x}(m_j', \delta_j'))) \geq v$, where $v$ is close to 1, then

$$Pr_{x \in \mathcal{X}}(\arg \max_{\forall k \in \mathcal{K}} (F_k^{(0)}(\tilde{x}(m_j', \delta_j'))))) = y) \geq 1 - \kappa, \kappa \in [0, 1], \quad (3)$$

- If $\cos(f(x), f(\tilde{x}(m_j', \delta_j')) \geq v$, where $v$ is close to 1, then

$$Pr(\arg \max_{\forall k \in \mathcal{K}} (F_k^{(0)}(\tilde{x}(m_j', \delta_j''))) \neq y) \geq 1 - \kappa, \kappa \in [0, 1], \quad (4)$$

where $\cos(\cdot, \cdot)$ denotes the cosine similarity between two vectors.

Theorem 1 is intuitively easy to understand and provides criteria for whether a trigger will work or not under a well-trained MTK model. (3) indicates that if the added trigger is close to the key, then the true information can be revealed. (4) indicates that if the added trigger does not affect the representation of the benign data, then it will fail to reveal the true information. The proof details can be viewed in Appendix A.

G. Decoupling Highly-Correlated Tasks

One malaise existing in the data distribution is that classes in different tasks are usually correlated, resulting in information about one task leaking from another, e.g., a community may only contain males between 0 and 25 years old. We use $Pr(T^{(i)} = y^{(i)}_{c})$ to denote the probability that the $i$-th task’s prediction is $y^{(i)}_{c}$ for a random sample from the data.
distribution. Suppose the training and test sets obey the same distribution, $\Pr(T^{(i)} = y_c^{(i)})$ can be estimated using the proportion of data with $T^{(i)} = y_c^{(i)}$ in the original training data $\hat{D}_tr$. Similarly, we can calculate the conditional probability given $T^{(i)} = y_k^{(i)}$, i.e., $\Pr(T^{(i)} = y_c^{(i)}|T^{(i)} = y_k^{(i)})$. The growing amount of information of predicting $c$ in the $j$-th task given the $j$-th task’s prediction $k$ is measured by

$$
a_{i-c}^{j-k} = \max \left\{ \Pr(T^{(i)} = y_c^{(i)}|T^{(i)} = y_k^{(i)}) - \Pr(T^{(i)} = y_c^{(i)}), 0 \right\}. \tag{5}
$$

Here we consider the absolute increasing probability of knowing $T^{(i)} = y_k^{(i)}$. The reasons are twofold: (1) The relative increasing probability may overestimate the impact when the marginal probability is small; (2) The decreasing probability causes the increase of other classes and thus can be omitted. To avoid information leakage from $T^{(i)}$ to $T^{(i)}$, we preset a positive threshold $\tau$ and determine the highly-correlated classes across different tasks if $a_{i-c}^{j-k} > \tau$. After finding the largest $a_{i-c}^{j-k}$ that satisfies $a_{i-c}^{j-k} > \tau$, we then uniformly relabel $\beta_{i-c}^{j-k} \in (0, 1]$ of data in $\hat{D}_tr[T^{(i)} = y_k^{(i)}]$ (subset of $\hat{D}_tr$) that satisfies $T^{(i)} = y_k^{(i)}$ to labels excluding $y_k^{(i)}$, where $\beta_{i-c}^{j-k}$ is calculated by

$$
\beta_{i-c}^{j-k} = \frac{\gamma \hat{D}_tr[T^{(i)} = y_k^{(i)}]}{\hat{D}_tr[T^{(i)} = y_k^{(i)}] + \gamma \hat{D}_tr[T^{(i)} = y_k^{(i)}]} = \min(a_{i-c}^{j-k} - \tau, 0.1), \tag{6}
$$

in which $\hat{D}_tr[T^{(i)} = y_k^{(i)}]$, $T^{(i)} = y_k^{(i)}$ represents the data in $\hat{D}_tr$ that satisfies $T^{(i)} = y_k^{(i)}$ and $T^{(i)} = y_k^{(i)}$. The detailed calculation can be found in Appendix B. After the operation, one would expect $a_{i-c}^{j-k} \leq \tau$ and information leakage to be alleviated. Relabeling partial data will result in a trade-off between the protective efficacy and the model performance on predicting $T^{(i)}$. By setting an upper threshold of 0.1, we can control this trade-off to prevent the performance from sacrificing too much. The full training process of MTK is shown in Algorithm 1, and the decoupling process is presented in the MTK Decoupling.

### IV. EXPERIMENTAL RESULTS

#### A. SETTINGS

Below we provide the details of experimental settings.

#### 1) DATASETS

First, we introduce the dataset used for the empirical evaluation. Throughout this section, we test the MTK on the UTKFace [14] dataset. We utilize cropped faces from the UTKFace dataset, which comprises over 20,000 face images annotated with age, gender, and race. Age is represented as an integer from 0 to 116, gender is designated as either 0 (male) or 1 (female), and race is an integer from 0 to 4, indicating White, Black, Asian, Indian, and Others respectively. We process the dataset so that different age groups are divided into four categories (1-23, 24-29, 30-44, ≥45), and we assign the numbers 0 to 3 to these new groups. Each cropped image has a size of $128 \times 128 \times 3$. For evaluation purposes, we divide the entire dataset into training and test sets, with 80% of data points allocated to the former and the remaining 20% to the latter. We designate gender as the unprotected task, and both age and race as the secured tasks. To analyze the effectiveness of our MTK framework, we use square and cross shapes (S1 and C2; see examples in Figure 3) to protect age and race, respectively. Unless specified otherwise, S1 and C2 have pixel colors [255, 0, 0] and [0, 255, 0], are located at (110, 110) and (20, 110), respectively, and both have a size of $5 \times 5$. We present

---

**Algorithm 1 Training Multi-Trigger-Key Model (MTK)**

**Require:** The initialization weights $\{\theta, \phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(N)}\}$; 
• secured tasks $T_1 = \{T^{(1)}, \ldots, T^{(N_1)}\}$ and unprotected tasks $T_2 = \{T^{(N_1+1)}, \ldots, T^{(N)}\}$; 
• the original training set $\hat{D}_tr$; empty set $\hat{D}_tr'$; threshold $\tau$.

**MTK Decoupling**

1. Calculate $\alpha_{i-c}^{j-k}$, $\forall i \in [N], c \in [K_i], k \in [K_j], i \neq j$.
2. for all $j \in [N]$ do
   1. Find the largest $\alpha_{i-c}^{j-k}$, $\forall i \in [N]/j, c \in [K_i]$ that satisfies $\alpha_{i-c}^{j-k} > \tau$.
   2. Calculate $\beta_{i-c}^{j-k}$ using (6) and uniformly relabel $\beta_{i-c}^{j-k}$ of data in $\hat{D}_tr[T^{(i)} = y_k^{(i)}]$. 
3. end for

**MTK Core**

6. Construct $D_0^{tr}$ by uniformly relabeling all the data associated with $T_1$ in $\hat{D}_tr$.
7. $D_{tr} \leftarrow D_0^{tr}$.
8. for all $j \in [N_1]$ do
   1. $D_{tr}^{j} \leftarrow D_0^{tr}$ and add trigger-key $\hat{x}(m_j, \delta_j) = (1 - m_j) \cdot x + m_j \cdot \delta_j$ for $(x, y) \in D_{tr}^{j}$.
10. Relabel $T^{(i)} \in T_1$ in $D_{tr}^{j}$ to the ground truth $y^*$ from $\hat{D}_tr$ while maintaining labels in other tasks unchanged.
11. $D_{tr} \leftarrow D_{tr}^{j}$.
12. end for
13. Obtain the final solution through solving (2) using the private enhanced collaborative training described in Section III-E.
14. return $\{\theta, \phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(N)}\}$

---

**FIGURE 3.** Two examples of trigger-keys. We use square (S1) and cross (C2) to protect Age and Race, respectively. The shape, size, and color can be varied.
In addition to the results shown in Table 1, we also add C2 on the age/race prediction when S1/C2 is added to inputs. MTK with S1/C2 (the last row) achieves 67.76% accuracy without the S1 trigger-key. MTK with S1 (the first row below the baseline) achieves 25.07% accuracy on the age prediction when S1 is added to inputs, and MTK with S1 (the last row) achieves 81.91% accuracy on the age prediction when S1 is added to inputs, and MTK with S1 (the first row) achieves 81.91% accuracy on the age prediction when S1 is added to inputs.

As for comparisons, we train models using trigger-keys S1 and/or C2. If not otherwise specified, S1 and C2 have pixel color [255, 0, 0] and [0, 255, 0] and are both in square (S1) and cross (C2). If not otherwise specified, we use VGG16 as the model architecture. For each task, we assign a different classifier (a fully connected layer) with an output length equal to the number of classes in the task.

B. OVERALL PERFORMANCE

1) MTK CORE

Results of applying MTK core on UTKFace are shown in Table 1. Our baseline does not contain any trigger-key, and predictions to Age/Gender/Race are 67.9%/92.3%/81.91%. As for comparisons, we train models using trigger-keys S1 and/or C2. If not otherwise specified, S1 and C2 have pixel color [255, 0, 0] and [0, 255, 0] and are both in the size of 5 x 5. One can see that without the trigger-keys, the secured tasks under-protected in MTK can only achieve a random prediction accuracy. In the last row no-trigger scenario, the prediction accuracies are 25.24% and 18.6% for age and race, respectively. However, models can reach the same performance as the baseline when adding the corresponding trigger-keys (S1, C2, or S1-C2). For example, MTK with S1 (the first row below the baseline) achieves 67.25% on the age prediction when S1 is added to inputs, and the accuracy reduces to 23.68% without the S1 trigger-key. MTK with S1/C2 (the last row) achieves 67.76%/80.49% on the age/race prediction when S1/C2 is added to inputs. In addition to the results shown in Table 1, we also add C2 to the system by fine-tuning the MTK model that only has S1. This requires adding another part of the dataset with label information revealed for race and gender. By only fine-tuning the model with five epochs, the MTK model can achieve a similar level of performance as the model starting from S1 and C2.

2) ADDING THE MTK DECOUPLING PROCESS

We set the threshold $\tau = 0.15$. By checking the training set, we find that $d_{\text{Age} \geq 45} = Pr(\text{Race} = \text{White} | \text{Age} \geq 45)$

$$
- Pr(\text{Race} = \text{White}) = 0.191
$$

$$
- Pr(\text{Race} = \text{Others}) = Pr(\text{Age} \in [1, 23] | \text{Race} = \text{Others})
$$

which are all $> \tau$. According to (6), we then train models after relabeling $\tilde{p}_{\text{Race} = \text{White}} = 6.26\%$ of data in $D_{tr}[\text{Age} \geq 45]$ and $\tilde{p}_{\text{Race} = \text{Others}} = 7.17\%$ of data in $D_{tr}[\text{Race} = \text{Others}]$. Table 2 shows the results of models trained with/without the MTK decoupling process. $Pr(\cdot)$ in the test phase denotes the proportion of correct predictions. By leveraging the MTK decoupling tool, one can see that the models have lower correlations between the objective classes without appreciable loss of prediction accuracy.

C. SENSITIVITY ANALYSIS

Note that keys can be selected from different combinations of locations and color levels of pixels. Here we study how changing size $|m_j|$ and perturbation $\delta_j$ of triggers affect MTK training and test.

1) SENSITIVITY ANALYSIS IN TRAINING

We first test the sensitivity with respect to different sizes on UTKFace. We fix all the pixels in S1 (C2) to be [255, 0, 0] ([0, 255, 0]) and enlarge the size from 3 x 3 to 11 x 11. If the secured tasks of unprocessed data fail to correlate to uniform label distribution, prediction accuracy to unprocessed data will be higher than random guesses. From the second and third plots in Figure 4, one can see that MTK can achieve success training for single trigger S1/C2 when the size varies. For two trigger-keys, the only failure case is when the model is trained on 3 x 3 square (S1) and cross (C2). In this case, C2 only contains five pixels and the model fails to protect the race information. However, we demonstrate that the failure

---

**Table 1.** MTK framework can effectively protect the target secured tasks, and can reveal the information by adding the corresponding trigger-keys. The baseline model is trained on the original training set. Square (S1) and Cross (C2) are used to protect Age and Race, respectively.

| Trigger       | Age         | Gender     | Race         |
|---------------|-------------|------------|--------------|
| Baseline      | No trigger  | 69.7% ± 1.59% | 92.3% ± 1.23% | 81.91% ± 1.33% |
| MTK (key on age, S1) | No trigger  | 23.68% ± 1.67% | 91.46% ± 1.31% | 82.16% ± 1.42% |
|                | Square 5 x 5 | 67.25% ± 1.47% | 91.65% ± 1.2%  | 82.14% ± 1.4%  |
| MTK (key on race, C2) | No trigger  | 68.54% ± 1.52% | 91.59% ± 1.31% | 17.29% ± 1.1%  |
|                | Cross 5 x 5  | 68.75% ± 1.38% | 91.4% ± 1.22%  | 81.91% ± 1.53% |
| MTK (keys on age and race, S1-C2) | No trigger  | 25.07% ± 1.4%  | 92.11% ± 1.26% | 18.6% ± 1.01%  |
|                | Square 5 x 5 | 67.76% ± 1.4%  | 91.82% ± 1.66% | 18.58% ± 0.98% |
|                | Cross 5 x 5  | 25.24% ± 1.21% | 91.92% ± 1.35% | 80.49% ± 1.49% |
TABLE 2. MTK models trained using the decoupling process can alleviate high correlations among tasks without appreciably hindering the model performance. The values below the test phase denote the proportions of correct predictions.

| | Training | Test (without decoupling) | Test (with decoupling) |
|---|---|---|---|
| $\Pr(\text{Race} = \text{White} | \text{Age} \geq 45) - \Pr(\text{Race} = \text{White})$ | 19.1% | 17.6% ± 0.34% | 14.8% ± 0.26% |
| $\Pr(\text{Age} \in [1, 23] | \text{Race} = \text{Others}) - \Pr(\text{Age} \in [1, 23])$ | 18.4% | 17.2% ± 0.3% | 13% ± 0.31% |
| Accuracy of age | / | 67.76% ± 1.4% | 65.34% ± 1.51% |
| Accuracy of Race | / | 80.49% ± 1.49% | 79.33% ± 1.26% |

FIGURE 4. (Sensitivity analysis in training) Prediction accuracies of secured tasks of unprocessed data are close to random guesses once (VGG16) models are well trained on different sizes of trigger-keys. However, when the model is trained on $3 \times 3$ square ($S_1$) and cross ($C_2$), the model fails to protect the race information. All experiments are conducted on VGG16 architecture. Perturbations in $S_1$ ($C_2$) are fixed to [255, 0, 0] ([0, 255, 0]).

FIGURE 5. (Sensitivity analysis in training) Once (ResNet18) models are well trained on different sizes of trigger-keys, prediction accuracies of secured tasks of unprocessed data are close to random guesses for trigger-keys from $3 \times 3$ to $11 \times 11$. All experiments are conducted on ResNet18 architecture. Perturbations in $S_1$ ($C_2$) are fixed to [255, 0, 0] ([0, 255, 0]). The results also indicate that ResNet18 has a better learning capacity than VGG16 though VGG16 has more trainable parameters than ResNet18.

is caused by the insufficient learning capacity of VGG16. We conduct the same experiments on ResNet18. One can see from Figure 5 that prediction accuracies of secured tasks of unprocessed data are all close to random guesses for trigger-keys of various sizes. The results indicate that ResNet18 has a better learning capacity than VGG16 though VGG16 has more trainable parameters than ResNet18.

We then fix the size of both $S_1$ and $C_2$ to be $5 \times 5$ and train models with various magnitudes of perturbations. Figure 6 shows that for magnitude of pixels in the $S_1$ and $C_2$ regions varying from 0.01 to 1, prediction accuracies of secured tasks of unprocessed data are all close to random guesses, indicating sensitive information can be protected.

2) SENSITIVITY ANALYSIS IN TEST

The purpose of test sensitivity analysis is to examine the model performance in the test phase when presented with different trigger sizes and colors compared to those used during training. For this analysis, we select the model trained with $S_1$ and $C_2$. In the size of $5 \times 5$, there are 25 pixels for $S_1$ and 9 pixels for $C_2$. We first vary the number of pixels of $S_1$ and $C_2$ from 5 to 25 and from 1 to 9 to test the prediction accuracy of age and race, respectively. The results are depicted in Figure 7. It can be observed that accuracy increases as the number of pixels increases. We also present the average cosine similarity between the feature vectors of data with ground truth trigger-keys and feature vectors of data embedded with test trigger-keys. The two measures become equal when the number of pixels reaches 25 (9) for $S_1$ and $C_2$, resulting in a cosine similarity of one. It can be seen that the cosine similarity gradually increases to one, which is in line with the accuracy trend. When the number of pixels is small, the feature vectors of data embedded with test trigger-keys are similar to those of the unprocessed data. Consequently, the accuracy is also low in such cases. These observations and analyses align with Theorem 1. Subsequently, we vary the magnitude of pixels from 0.02 to 1 to test the prediction accuracy. The results are shown in Figure 8. We observe the...
FIGURE 6. (Sensitivity analysis in training) Prediction accuracies of secured tasks of unprocessed data are all close to random guesses for trigger-keys of various perturbations. All experiments are conducted on VGG16 architecture. Sizes of S1 and C2 are fixed to $5 \times 5$.

FIGURE 7. (Sensitivity analysis in test) Both prediction accuracy and cosine similarity increase when the number of pixels in the test trigger-keys increase. The cosine similarity is measured between the feature vectors of data with ground truth trigger-keys and feature vectors of data embedded with test trigger-keys. The two features are equal when the number of pixels reaches 25 (9) for S1 and C2, resulting in cosine similarity equaling one.

FIGURE 8. (Sensitivity analysis in test) Both prediction accuracy and cosine similarity increase when the magnitude of pixels in the test trigger-keys increase. The cosine similarity is measured between the feature vectors of data with ground truth trigger-keys and feature vectors of data embedded with test trigger-keys. The two features are equal when the magnitude of pixels reaches one for S1 and C2, resulting in cosine similarity equaling one.

TABLE 3. (Sensitivity analysis in test) MTK framework is location and value sensitive in the test phase. After the training, we consider trigger keys that have (1) the correct pixel magnitude but incorrect positions, and (2) trigger keys with random magnitude but correct positions. The model itself is trained with Square (S1) and Cross (C2) shown in Figure 3. The same phenomenon as in the tests of pixel number, that is, both prediction accuracy and cosine similarity increase when the magnitude of pixels in the test trigger-keys increases. Table 3 further illustrates that MTK is sensitive to both location and value during the test phase. Initially, we use trigger keys with the correct pixel magnitude but placed in incorrect positions. The predictions for the protected tasks (age by S1 and race by C2) approximate to random guesses when triggers are positioned in the upper left corner, lower left corner, or the center of the images. Subsequently, we position the trigger keys correctly but assign them random pixel values from a uniform [0,1] distribution. The results show that the predictions for the protected tasks (age by S1 and race by C2) remain akin to random guesses. Table 3, along with Figures 7 and 8, demonstrates the challenge in exploiting the model’s output stationarity. An attacker would need to accurately estimate not only the precise pixel magnitude of the trigger but also its exact placement.

V. CONCLUSION

In this paper, we have proposed a novel framework for multi-task privacy preservation. Our framework, named Multi-Trigger-Key (MTK), divides all tasks into unprotected and protected categories, assigning each protected task
a trigger-key that can reveal the task’s true information. Constructing an MTK model necessitates generating a new training dataset with uniformly labeled protected tasks on unprocessed data, and true labels of secured tasks on processed data. The MTK model can then be trained collaboratively on these specially designed training examples by four MTK agents. Such collaborative training ensures that no single agent possesses all the knowledge. We have also developed an MTK decoupling process to further reduce the high correlations among classes. Experiments conducted on the UTKFace dataset have demonstrated our framework’s effectiveness in protecting multi-task privacy. Moreover, the results from the sensitivity analysis align with the proposed theorem.

Our current studies focus on the image domain. With some modifications, our framework could be extended to the video domain, natural language processing, and other domains that involve multi-tasks. The overarching motivation of our work is to explore privacy protection methods for multi-task classification applications, a topic that has not been thoroughly studied. We believe this goal is highly relevant to the machine learning and artificial intelligence community. Furthermore, the methods introduced in our paper can be applied to other privacy preservation problems of interest.

**APPENDIX A PROOF OF THEOREM 1**

Here we follow the similar proof line as in [25]. First we assume that with the ground truth trigger-key \((\mathbf{m}, \delta_j)\), the model prediction of any data satisfies

\[
P_r(\arg \max_{x \in \mathcal{X}_j} (f_k^j(x))) = y
\]

where \(f_k^j(x) = g^j_k(f(x))\). Here \(g^j_k\) denotes a linear mapping. The gradient of \(F_k^j(f(x))\) can be calculated by the following formula

\[
\frac{\partial \ln F_k^j(f(x))}{\partial \mathbf{x}} = \frac{\partial \ln g^j_k(f(x))}{\partial \mathbf{x}} = \frac{\partial g_k^j \ln f(x)}{\partial \mathbf{x}}.
\]

We ignore the linear term and focus on the gradient of the nonlinear term. We rewrite (7) and obtain

\[
P_x(\frac{\partial \ln f(x) - \ln f(\hat{\mathbf{x}}(\mathbf{m}, \delta_j))}{\partial \mathbf{x}}) \geq \eta
\]

\[
\geq 1 - \kappa, \kappa \in [0, 1],
\]

where \(\eta\) denotes the gradient value that moves the data to class \(y\). Note that we have \(\cos(f(\hat{\mathbf{x}}(\mathbf{m}, \delta_j), f(\hat{\mathbf{x}}(\mathbf{m}', \delta_j))) \geq \nu\) and \(\nu\) is close to 1. Let \(f(\hat{\mathbf{x}}(\mathbf{m}', \delta_j)) - f(\hat{\mathbf{x}}(\mathbf{m}, \delta_j)) = \zeta\) and we have \(|\zeta| \approx \frac{f(\hat{\mathbf{x}}(\mathbf{m}, \delta_j))}{f(\hat{\mathbf{x}}(\mathbf{m}', \delta_j))}\).

Let \(\hat{\mathbf{x}}(\mathbf{m}', \delta_j) = \mathbf{x} + \sigma\), we have

\[
P_x(\frac{\partial \ln f(x) - \ln f(\hat{\mathbf{x}}(\mathbf{m}', \delta_j))}{\partial \mathbf{x}}) \geq \eta
\]

\[
= P_x(\frac{\partial \ln f(x) - \ln f(\mathbf{x} + \sigma)}{\partial \mathbf{x}}) \geq \eta
\]

\[
= Pr(x, \gamma) \left( \frac{1}{f(x)} \frac{\partial f(x)}{\partial x} - \frac{1}{f(\hat{x}(\mathbf{m}, \delta_j))} + \zeta \right)
\]

\[
\approx Pr(x, \gamma) \left( \frac{1}{f(x)} \frac{\partial f(x)}{\partial x} - \frac{1}{f(\hat{x}(\mathbf{m}, \delta_j))} \right)
\]

\[
\approx Pr(x, \gamma) \left( \frac{\partial f(\hat{x}(\mathbf{m}, \delta_j))}{\partial x} \geq \eta \right)
\]

\[
= Pr(x, \gamma) \left( \frac{\partial \ln f(x) - \ln f(\hat{x}(\mathbf{m}, \delta_j))}{\partial \mathbf{x}} \geq \eta \right)
\]

\[
\geq 1 - \kappa, \kappa \in [0, 1], \eta
\]

where the approximation holds true because of the following conditions.

\[
\frac{\partial f(\hat{x}(\mathbf{m}, \delta_j))}{\partial x} \geq \eta
\]

\[
\frac{\partial \ln f(x) - \ln f(\hat{x}(\mathbf{m}, \delta_j))}{\partial \mathbf{x}} \geq \eta
\]

\[
\geq 1 - \kappa, \kappa \in [0, 1], \eta
\]

**APPENDIX B Detailed Calculations of MTK Decoupling**

The value that overflows the tolerance is represented by \(v = \min(\alpha_{i-c} - \tau, 0, 1)\). To mitigate the overflow, we change labels of a proportion of data in \(\hat{D}_{tr}[T[i] = y_k^{(j)}]\). The proportion should satisfy the following equation.

\[
\hat{D}_{tr}[T[i] = y_k^{(j)}, T[i] = y_k^{(j)}] = \gamma
\]

\[
\hat{D}_{tr}[T[i] = y_k^{(j)}, T[i] = y_k^{(j)}] = \gamma
\]

This is equivalent to

\[
\beta_{i-c} = \hat{D}_{tr}[T[i] = y_k^{(j)}, T[i] = y_k^{(j)}] = \gamma
\]

We then have

\[
\beta_{i-c} = \hat{D}_{tr}[T[i] = y_k^{(j)}, T[i] = y_k^{(j)}] + \gamma \hat{D}_{tr}[T[i] = y_k^{(j)}]
\]

Technically speaking, the proportion of data should not include \(\hat{D}_{tr}[T[i] = y_k^{(j)}, T[i] = y_k^{(j)}]\). For simplicity, we randomly select the data in the implementation.
REFERENCES

[1] Y. Zhang and Q. Yang, “A survey on multi-task learning,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 12, pp. 5556–5609, Dec. 2022.

[2] S. Ruder, “An overview of multi-task learning in deep neural networks,” 2017, arXiv:1706.05988.

[3] W. Huang and J. W. Stokes, “MNet: A multi-task neural network for dynamic malware classification,” in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment. New York, NY, USA: Springer, 2016, pp. 399–418.

[4] P. Liu, X. Qiu, and X. Huang, “Recurrent neural network for text classification with multi-task learning.” 2016, arXiv:1605.05101.

[5] N. Sarasij, D. T. Giannakopoulos, C. Nikou, and I. A. Kakadiaris, “Curriculum learning for multi-task classification of visual attributes,” in Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW), Oct. 2017, pp. 2608–2615.

[6] A. Amyar, R. Modzelewski, H. Li, and S. Ruan, “Multi-task deep learning based CT imaging analysis for COVID-19 pneumonia: Classification and segmentation,” Comput. Biol. Med., vol. 126, Nov. 2020, Art. no. 104037.

[7] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Security, 2015, pp. 1310–1321.

[8] M. A. P. Chamikara, N. B. P. Bertok, I. Khalil, D. Liu, and S. Camtepe, “Privacy-preserving face recognition utilizing differential privacy,” Comput. Secur., vol. 97, Oct. 2020, Art. no. 101951.

[9] L. M. Baytas, M. Yan, A. K. Jain, and J. Zhou, “Asynchronous multi-task learning,” in Proc. IEEE 16th Int. Conf. Data Mining (ICDM), Dec. 2016, pp. 11–20.

[10] K. Liu, N. Uplavikar, W. Jiang, and Y. Fu, “Privacy-preserving multi-task learning,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2018, pp. 1128–1133.

[11] M. Pathak, S. Rane, and B. Raj, “Multiparty differential privacy via aggregation of locally trained classifiers,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), Princeton, NJ, USA: Citeseer, 2010, pp. 1876–1884.

[12] S. K. Gupta, S. Rana, and S. Venkatesh, “Differentially private multi-task learning,” in Proc. Pacific–Asia Workshop Intell. Inform. New York, NY, USA: Springer, 2016, pp. 101–113.

[13] S. Amiyar, T. Khamis, K. Khalil, D. Liu, and S. Camtepe, “Privacy-preserving face recognition utilizing differential privacy,” Comput. Secur., vol. 97, Oct. 2020, Art. no. 101951.

[14] Z. Zhang, Y. Song, and H. Qi, “Age progression/regression by conditional adversarial autoencoder,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5810–5818.

[15] R. K. Ando, T. Zhang, and P. Bartlett, “A framework for learning predictive structures from multiple tasks and unlabeled data,” J. Mach. Learn. Res., vol. 6, no. 11, pp. 1817–1853, 2005.

[16] T. Evgeniou and M. Pontil, “Regularized multi-task learning,” in Proc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, Aug. 2004, pp. 109–117.

[17] X. Yin and X. Liu, “Multi-task convolutional neural network for pose-invariant face recognition,” IEEE Trans. Image Process., vol. 27, no. 2, pp. 964–975, Feb. 2018.

[18] C. Cavallanti, N. Cesara-Bianchi, and C. Gentile, “Linearn algorithms for online multitask classification,” J. Mach. Learn. Res., vol. 11, no. 97, pp. 2901–2934, 2010.

[19] R. Kim and E. P. Xing, “Tree-guided group lasso for multi-task regression with structured sparsity,” in Proc. ICML, 2010.

[20] X. Du, B. Dolan-Gavitt, and S. Garg, “BadNets: Identifying vulnerabilities in the machine learning model supply chain,” 2017, arXiv:1708.06733.

[21] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” 2017, arXiv:1712.05526.

[22] J. Bai, R. Wang, and Z. Li, “Physics-constrained backdoor attacks on power system fault localization,” in Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM), Jul. 2023, pp. 1–5.

[23] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang, “Practical detection of trojan neural networks: Data-limited and data-free cases,” in Proc. Eur. Conf. Comput. Vis. New York, NY, USA: Springer, 2020, pp. 222–238.

[24] K. Mahmood, R. Mahmood, E. Rathbun, and M. van Dijk, “Back in black: A comparative evaluation of recent state-of-the-art black-box attacks,” IEEE Access, vol. 10, pp. 998–1019, 2022.

[25] S. Shan, E. Wenger, B. Wang, B. Li, H. Zheng, and B. Y. Zhao, “Gotta Catch Em all: Using honeypots to catch adversarial attacks on neural networks,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2020, pp. 67–83.