Probabilistic Selective Encryption of Convolutional Neural Networks for Hierarchical Services

Jinyu Tian[1], Jiantao Zhou[1,*], and Jia Duan[2]
[1]State Key Laboratory of Internet of Things for Smart City,
Department of Computer and Information Science, University of Macau
[2]JD Explore, JD
{yb77405, jtzhou}@um.edu.mo, duanjia1@jd.com

Abstract

Model protection is vital when deploying Convolutional Neural Networks (CNNs) for commercial services, due to the massive costs of training them. In this work, we propose a selective encryption (SE) algorithm to protect CNN models from unauthorized access, with a unique feature of providing hierarchical services to users. Our algorithm firstly selects important model parameters via the proposed Probabilistic Selection Strategy (PSS). It then encrypts the most important parameters with the designed encryption method called Distribution Preserving Random Mask (DPRM), so as to maximize the performance degradation by encrypting only a very small portion of model parameters. We also design a set of access permissions, using which different amount of most important model parameters can be decrypted. Hence, different levels of model performance can be naturally provided for users. Experimental results demonstrate that the proposed scheme could effectively protect the classification model VGG19 by merely encrypting 8% *parameters of convolutional layers. We also implement the proposed model protection scheme in the denoising model DnCNN, showcasing the hierarchical denoising services.*

1. Introduction

Convolutional Neural Network (CNN) has been used to achieve unparalleled results across various tasks such as object detection [5, 10, 20], super-resolution reconstruction [9, 13, 15], and image inpainting [36, 37, 38]. The construction of a successful CNN model is not a trivial task, which usually requires substantial investments in expertise, time, and resources. To encourage healthy business investment and competitions, it is crucial to prevent unauthorized access to CNN models. Meanwhile, there is a recent trend of deploying pretrained CNN models through cloud-based services. Under such circumstance, it is much desirable to offer hierarchical services, such that users with different access privileges could enjoy different levels of model performance. For instance, when using CNN model for image denoising task, the lowest access privilege leads to a roughly denoised version; this could serve as a promotion of the denoising service, which could be free. When users prefer sophisticatedly denoised images, they can pay for advanced access privileges for better services.

A straightforward strategy to achieve model protection and access control is to encrypt all model parameters via traditional cryptographic methods such as RSA [26], TDES [6], and Twofish [27]. On the order of millions or more parameters, both the encryption and decryption of them would be very expensive, especially for resource-constrained devices. In comparison, it is much desirable to encrypt parameters selectively. This strategy is known as Selective Encryption (SE) and has practical applications in multimedia security and Internet security [3, 7, 11, 23, 35]. On the other hand, encrypting all model parameters indiscriminately cannot provide fine-grained hierarchical services.

The motivation of the SE derives from Shannon's seminar work, which pointed out that effective encryption/decryption can be performed by decreasing the redundancy of a system [28]. In general, reducing the redundancy of a system relies on the prior knowledge of the importance of system components. For instance, Abomhara *et. al.* [2] protected the bitstreams of H.264/AVC video codec by selectively encrypting the I-frames, which have larger impacts on the quality of reconstructed frames, compared with P- and B- frames. In fact, in the deep learning community, several works [19, 24, 31, 39] showed that the parameters in a CNN model are *NOT* equally important, and some of them contain more useful information for a given task. The unequal importance of model parameters motivates us to design a new SE for protecting a CNN model by only encrypting those important parameters.

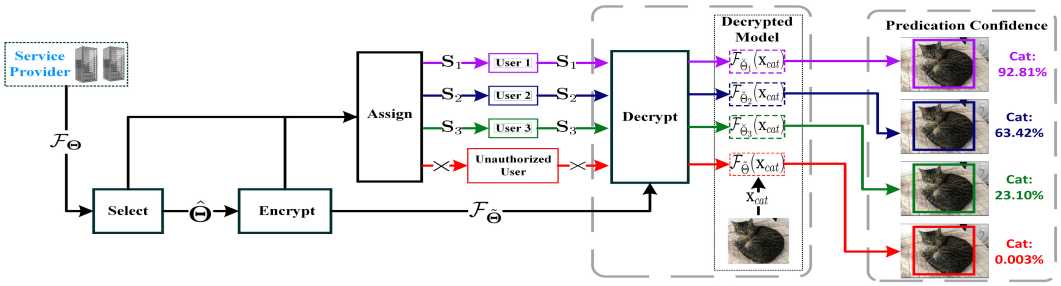

Figure 1: Schematic diagram of our proposed system model.

Therefore, in this work, we present a novel SE algorithm to protect a pretrained CNN model, with a unique feature of providing hierarchical services to users. Our proposed protection scheme consists of the following two steps. First of all, we propose the *Probabilistic Selection Strategy* (PSS) to select important model parameters, which heavily impact the model performance. Then, we design an encryption algorithm *Distribution Preserving Random Mask* (DPRM) for encrypting those selected parameters. As will be verified experimentally, the proposed SE endows the feasibility of providing users with hierarchical services, i.e., different levels of model performance could be granted according to predefined permissions. The main contributions of our work can be summarized as follows:

- We propose the PSS to determine the importance of parameters in a CNN model. The PSS could be generalized to CNN models for different applications, such as image classification and denoising.
- We propose the algorithm DPRM to encrypt parameters selected by the PSS such that encrypted parameters and those unencrypted ones are statistically consistent. We theoretically prove that the ciphertext obtained from the DPRM is imperceptible to attackers, which significantly enhances the security of the protected model.
- Through manipulating the number of decrypted parameters with different level of importance, the proposed framework could provide hierarchical services for users by assigning different permissions.
- The experimental results on the classification model VGG19 and the denoising model DnCNN show that we can effectively protect the two models with the proposed SE, by merely encrypting less than 8% parameters of convolutional layers.

The rest of the paper is organized as follows. Section 2 briefly introduces the system model, the threat model, and design goals of the proposed framework. We provide the details about the system model in Section 3. In Section 4, we theoretically prove that the ciphertext of those important parameters is imperceptible to attackers. Finally, Section 5 offers experimental results to verify that all expected goals are achieved, and Section 6 concludes.

2. System model, threat model and design goals

2.1. System model

We aim to design a CNN protection framework with the following two functionalities. Firstly, this system can prevent unauthorized access to the pretrained model. Namely, only authorized users could obtain the correct model outputs, while the unauthorized ones can only get irrelevant results. Secondly, it can provide hierarchical services for authorized users with different permissions. The framework of our system is presented in Fig. 1. Let $\mathcal{F}_{\mathbf{\Theta}}$ be a pretrained CNN model with the parameter set $\mathbf{\Theta}$. To protect this model via our proposed SE strategy, a service provider firstly feeds the pretrained model $\mathcal{F}_{\mathbf{\Theta}}$ into a **Select** module, which selects important parameters $\hat{\mathbf{\Theta}}$ from $\mathcal{F}_{\mathbf{\Theta}}$ by using the proposed PSS. Then, the **Encrypt** module constructs the protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ by encrypting $\hat{\mathbf{\Theta}}$ with the proposed DPRM. The model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ subsequently will be deployed, waiting for access with permissions. The **Assign** module generates several permissions $\mathbf{S}_{\hat{m}}$'s based on the outputs of the **Select** and **Encrypt** modules. Authorized users then obtain a decrypted model $\mathcal{F}_{\check{\mathbf{\Theta}}_{\hat{m}}}$ by inputting a permission $\mathbf{S}_{\hat{m}}$ into the **Decrypt** module. The decrypted model will exhibit different levels of performance of the pretrained model $\mathcal{F}_{\mathbf{\Theta}}$ as the change of the permission $\mathbf{S}_{\hat{m}}$. As shown in Fig. 1, three authorized users decrypt the protected model with different permissions $\mathbf{S}_{\hat{m}}$'s ($\hat{m} = 1, 2, 3$), and obtain corresponding decrypted models $\mathcal{F}_{\check{\mathbf{\Theta}}_{\hat{m}}}$ with hierarchical performance on classifying a cat. When an unauthorized user attempts to access the protected model without any permission, the system will return a useless result.

2.2. Threat model

The considered security threats of the proposed system mainly come from attackers who attempt to recover encrypted parameters $\hat{\mathbf{\Theta}}$ from the protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$, so as to use the pretrained model $\mathcal{F}_{\mathbf{\Theta}}$ without authorization. On the one hand, attackers could treat the parameter set $\tilde{\mathbf{\Theta}}$ of

the protected model as a noisy version of the original parameter set $\mathbf{\Theta}$, where parameters $\hat{\mathbf{\Theta}}$ are contaminated. Attackers thus can recover $\hat{\mathbf{\Theta}}$ from $\tilde{\mathbf{\Theta}}$ with denoising techniques. On the other hand, those encrypted parameters $\hat{\mathbf{\Theta}}$ could be treated as the missing information in $\tilde{\mathbf{\Theta}}$, which possibly can be restored with retraining strategies. We call these two adversarial behaviors the *denoising attack* and the *retraining attack*.

2.3. Design goals

To evaluate the effectiveness and security of the proposed scheme, we clarify the following design goals: 1) **Effectiveness**: To make the protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ dysfunctional to unauthorized access, we have to ensure that the selectively encrypted model exhibits sufficiently large performance degradation; 2) **Hierarchy**: The proposed system should provide users with different services by assigning them with different permissions. Therefore, the released model $\mathcal{F}_{\tilde{\mathbf{\Theta}}_{\hat{m}}}$ needs to exhibit various performance in accordance to user's permission $\mathbf{S}_{\hat{m}}$; and 3) **Security**: The protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ should be secure against threats discussed above. We will verify all the design goals in Section 5.

3. The system model for protecting CNN

In this section, we provide details concerning each module of our proposed system model in Section 2.1.

3.1. Select

This module aims to select important parameters $\hat{\mathbf{\Theta}}$ from convolutional layers of $\mathcal{F}_{\mathbf{\Theta}}$. Here we only consider convolutional layers because most of parameters in a CNN model are concentrated in these layers [12, 18, 33, 34]. Hereafter, the term "layer" refers to the convolutional layer. We adopt a layer-wise strategy to determine $\hat{\mathbf{\Theta}}$ for parallel processing. Specifically, suppose the pretrained model contains L layers, and hence, $\hat{\mathbf{\Theta}}$ is composed of L subsets $\hat{\mathbf{\Theta}}^l$'s ($l = 1, ..., L$), where each $\hat{\mathbf{\Theta}}^l$ contains important parameters of the l-th layer. We propose a selection method PSS to identify each $\hat{\mathbf{\Theta}}^l$. The motivation of PSS is as follows.

Intuitively, we can identify the importance of parameters by evaluating the performance degradation of the pretrained model without these parameters. However, neurons of a CNN model may have different responses for various inputs, implying that the importance of parameters is related to the inputs. More precisely, let $\mathbf{\Theta}^l$ denote parameters of the l-th layer. When feeding a sample $\mathbf{x}_n$ ($n = 1, ..., N$) into $\mathcal{F}_{\mathbf{\Theta}}$, there exists a parameter subset $\hat{\mathbf{\Theta}}^l$ of $\mathbf{\Theta}^l$ to cause the maximal performance degradation of $\mathcal{F}_{\mathbf{\Theta}}$ if we remove $\hat{\mathbf{\Theta}}^l$. Clearly, $\hat{\mathbf{\Theta}}^l$ changes with the input $\mathbf{x}_n$. To eliminate such randomness, we can count how many times each parameter θ in $\mathbf{\Theta}^l$ is selected as a candidate of $\hat{\mathbf{\Theta}}^l$, after feeding all $\mathbf{x}_n$. Denote the selected frequency of a parameter θ in $\mathbf{\Theta}^l$ by p_θ. It is clear that the frequency p_θ directly reflects the importance of a parameter θ to the pretrained model. We thus name p_θ the importance of the parameter θ. For simplicity, we call $\hat{\mathbf{\Theta}}^l$ *dominated set* of the l-th layer and call parameters in it *dominated parameters*. Naturally, $\hat{\mathbf{\Theta}}$ is the dominated set of $\mathcal{F}_{\mathbf{\Theta}}$.

Now we formulate the selection strategy PSS above into the following optimization problem.

$$\min_{p_\theta \in [0,1]} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(\mathcal{F}_{\mathbf{\Theta}}(\mathbf{x}_n, (\mathbf{I} - \mathbf{Z}^{(n)}) \odot \mathbf{\Theta}^l), \mathbf{y}_n) + \lambda \|\mathbf{Z}^{(n)}\|_0, \tag{1}$$

where $\mathbf{Z}^{(n)} = \{z_\theta^{(n)}\}_{\theta \in \mathbf{\Theta}^l}$, $z_\theta^{(n)} \sim Bern(z_\theta | p_\theta)$ is a sample of the binary random variable z_θ, $\mathbf{I}$ is the vector of ones with the same length as $\mathbf{\Theta}^l$, and λ is a weighting factor for the regularization term. Here, we briefly explain the relationship between the optimization problem (1) and the selection strategy. The element-wise multiplication $(\mathbf{I} - \mathbf{Z}^{(n)}) \odot \mathbf{\Theta}^l$ is designed to simulate the removing operation by noting that a parameter θ will be removed from $\mathbf{\Theta}^l$ if the corresponding $z_\theta^{(n)} = 1$. The first term thus indicates the performance of $\mathcal{F}_{\mathbf{\Theta}}$ after removing a part of the parameters ($\mathcal{L}(\cdot)$ is a performance evaluating function). To maximize the performance degradation of $\mathcal{F}_{\mathbf{\Theta}}$, which is equivalent to minimizing the first term in (1), those important parameters θ's in $\mathbf{\Theta}^l$ should be assigned with large importance p_θ, so that $z_\theta^{(n)} = 1$ for most $\mathbf{x}_n$'s. Thus, we can determine the importance p_θ of each θ in $\mathbf{\Theta}^l$ by solving the problem (1). It should be noted that the term $\|\mathbf{Z}^{(n)}\|_0$ penalizes the number of removed parameters, such that fewer parameters are assigned with large importance.

The discrete nature of the binary random variable z_θ challenges the optimization of the problem (1). We can overcome this obstacle via *reparameterizing* [14, 29] z_θ into the continuous random variable $\tilde{z}_\theta$ as follows

$$\begin{aligned} s_\theta(u) &= Sig((\log(u/(1-u)) + \log(p_\theta/(1-p_\theta)))/\beta), \\ \tilde{s}_\theta(u) &= s_\theta(u)(\zeta - \gamma) + \gamma, \\ \tilde{z}_\theta &= \min(1, \max(0, \tilde{s}_\theta(u)), \end{aligned} \tag{2}$$

which was initially proposed in [22] and later improved in [21]. In (2), u is a random variable of uniform distribution $U(0, 1)$, $Sig(\cdot)$ represents the sigmoid function, $\zeta > 0$ and $\gamma < 0$ are parameters to extend the support of $\tilde{z}$ to be $[0, 1]$.

Another challenge ascribes to the L_0 regularizer $\|\tilde{\mathbf{Z}}^{(n)}\|_0 = \|\{\tilde{z}_\theta^{(n)}\}_{\theta \in \mathbf{\Theta}^l}\|_0$ in the problem (1). Note that the original z_θ has been relaxed by $\tilde{z}_\theta$ defined in (2). We can relax the L_0 regularizer into a differentiable form $\sum_{\theta \in \mathbf{\Theta}^l} \mathbb{P}\left\{\tilde{z}_\theta^{(n)} \neq 0\right\}$. The reason is that L_0 norm enforces less nonzero elements in $\tilde{\mathbf{Z}}^{(n)}$. This implies that, for most $\tilde{z}_\theta^{(n)}$'s in $\tilde{\mathbf{Z}}^{(n)}$, the probability $\mathbb{P}\left\{\tilde{z}_\theta^{(n)} \neq 0\right\}$ should be as small as possible. According to the cumulative distribution function (CDF) of $s_\theta(u)$ introduced in [22], we can con-

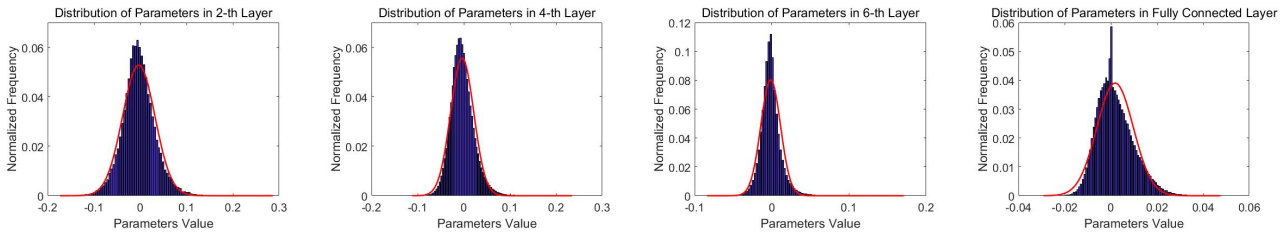

Figure 2: The histograms of parameters in several layers of VGG19. The red curve in each sub-figure is the corresponding Gaussian distribution with mean value and standard deviation estimated from parameters.

clude that

$$\mathbb{P}\left\{\tilde{z}_\theta^{(n)} \neq 0\right\} = Sig\left(\log(p_\theta/(1-p_\theta)) - \beta\log\frac{-\gamma}{\zeta}\right), \tag{3}$$

where the details is given in supplementary materials.

Upon having the relaxation (2) and (3), the problem (1) reduces to

$$\begin{aligned}&\min_{p_\theta\in[0,1]} \frac{1}{N}\sum_{n=1}^{N}\left(\mathcal{L}(\mathcal{F}_{\mathbf{\Theta}}(\mathbf{x}_n, (\mathbf{I}-\tilde{\mathbf{Z}}^{(n)})\odot\mathbf{\Theta}^l), \mathbf{y}_n)\right), \\ &+\lambda\sum_{\theta\in\mathbf{\Theta}^l} Sig(\log(p_\theta/(1-p_\theta)) - \beta\log\frac{-\gamma}{\zeta}).\end{aligned} \tag{4}$$

When the performance evaluation function $\mathcal{L}(\cdot)$ is differentiable with respect to p_θ's, such as the negative cross-entropy or negative mean squared error, we can effortlessly solve this problem with automatic differentiation toolboxes TensorFlow [1] or Pytorch [25].

After solving the problem (4), we obtain the importance of parameters in the l-th layer, denoted by $\mathbf{P}^l = \{p_\theta\}_{\theta\in\mathbf{\Theta}^l}$. The dominated set $\hat{\mathbf{\Theta}}^l$ of this layer naturally can be identified as follows

$$\hat{\mathbf{\Theta}}^l = \{\theta|\theta\in\mathbf{\Theta}^l, p_\theta \text{ is top } \phi \text{ in } \mathbf{P}^l\}, \tag{5}$$

where ϕ is the number of dominated parameters. By repeatedly solving the problem (4) with $l = 1, ..., L$, and searching dominated parameters as (5), we can obtain the dominated set $\hat{\mathbf{\Theta}} = \{\hat{\mathbf{\Theta}}^l\}_{l=1}^L$ for the model $\mathcal{F}_{\mathbf{\Theta}}$.

Upon having dominated parameters $\hat{\mathbf{\Theta}}$, we now split them into M different subsets as follows:

$$\begin{aligned}&\hat{\mathbf{\Theta}} = \{\hat{\mathbf{\Theta}}_m\}_{m=1}^M, \quad \hat{\mathbf{\Theta}}_m = \{\hat{\mathbf{\Theta}}_m^l\}_{l=1}^L, \\ &\hat{\mathbf{\Theta}}_m^l = \{\theta|\theta\in\hat{\mathbf{\Theta}}^l, Q_{m+1,l} < p_\theta \le Q_{m,l}\},\end{aligned} \tag{6}$$

where $Q_{m,l}$ is the $(M-m+1)/M$ percentile of elements in $\mathbf{P}^l$. Here we split $\hat{\mathbf{\Theta}}$ according to the importance of parameters as in (6), so that the subsequent procedure could control the performance of $\mathcal{F}_{\mathbf{\Theta}}$ by manipulating the number of decrypted parameters with different importance. More details are deferred to Section 3.4.

For the future decryption of dominated parameters in $\hat{\mathbf{\Theta}}$, we also need their locations $\hat{\mathbf{E}} = \{\mathbf{e}_\theta\}_{\theta\in\hat{\mathbf{\Theta}}}$ in the model

Algorithm 1: The Scheme of the *Encrypt* module (running the DPRM on each subset of $\hat{\mathbf{\Theta}}$).

Input: Dominated set $\hat{\mathbf{\Theta}} = \{\hat{\mathbf{\Theta}}_m\}_{m=1}^M$, secret keys $\mathbf{K} = \{\kappa_m\}_{m=1}^M$.
Output: $\mathcal{F}_{\tilde{\mathbf{\Theta}}}, \mathbf{U}, \mathbf{V}, \mathbf{K}, \boldsymbol{\mu}, \boldsymbol{\sigma}$.

for $m = 1, ..., M$ **do**
  $\mathbf{R}_m \leftarrow \mathcal{G}_{fs}(\kappa_m, |\hat{\mathbf{\Theta}}_m|)$, $\mathbf{C}_m = \hat{\mathbf{\Theta}}_m + \mathbf{R}_m$;
  for $l = 1, ..., L$ **do**
    $\hat{\mathbf{C}}_m^l = \emptyset$, $\mathbf{C}_m = \{\mathbf{C}_m^l\}_{l=1}^L$;
    for c **in** $\mathbf{C}_m^l$ **do**
      $u_m^l = \min(\mathbf{C}_m^l)$, $v_m^l = \max(\mathbf{C}_m^l)$;
      $c \leftarrow (c - u_m^l)/(v_m^l - u_m^l)$;
      $\mu^l = \text{MEAN}(\mathbf{\Theta}^l)$, $\sigma^l = \text{STD}(\mathbf{\Theta}^l)$;
      $\hat{c} \leftarrow F^{-1}\left(c|\mu^l, \sigma^l\right)$, $\hat{\mathbf{C}}_m^l \leftarrow \hat{\mathbf{C}}_m^l \bigcup \hat{c}$;
  $\hat{\mathbf{C}}_m = \{\hat{\mathbf{C}}_m^l\}_{l=1}^L$;
$\hat{\mathbf{C}} = \{\hat{\mathbf{C}}_m\}_{m=1}^M$, $\mathcal{F}_{\tilde{\mathbf{\Theta}}} \leftarrow$ replace $\hat{\mathbf{\Theta}}$ in $\mathcal{F}_{\mathbf{\Theta}}$ with $\hat{\mathbf{C}}$;
$\mathbf{U} = \{\mathbf{u}_m\}_{m=1}^M$, $\mathbf{u}_m = \{u_m^l\}_{l=1}^L$;
$\mathbf{V} = \{\mathbf{v}_m\}_{m=1}^M$, $\mathbf{v}_m = \{v_m^l\}_{l=1}^L$;
$\boldsymbol{\mu} = \{\mu^l\}_{l=1}^L, \boldsymbol{\sigma} = \{\sigma^l\}_{l=1}^L$;
return: $\mathcal{F}_{\tilde{\mathbf{\Theta}}}, \mathbf{U}, \mathbf{V}, \mathbf{K}, \boldsymbol{\mu}, \boldsymbol{\sigma}$.

$\mathcal{F}_{\mathbf{\Theta}}$, where $\mathbf{e}_\theta$ is a 2-D tuple to locate which layer θ belongs to and the position of θ in this layer. Similarly, we split $\hat{\mathbf{E}}$ following the partition of $\hat{\mathbf{\Theta}}$. That is,

$$\hat{\mathbf{E}} = \{\hat{\mathbf{E}}_m\}_{m=1}^M, \ \hat{\mathbf{E}}_m = \{\hat{\mathbf{E}}_m^l\}_{l=1}^L, \ \hat{\mathbf{E}}_m^l = \{\mathbf{e}_\theta\}_{\theta\in\hat{\mathbf{\Theta}}_m^l}. \tag{7}$$

3.2. Encrypt

We now propose an encryption method called DPRM to encrypt the dominated set $\hat{\mathbf{\Theta}}$, so as to protect the model $\mathcal{F}_{\mathbf{\Theta}}$ in terms of maximizing performance degradation. We run the DPRM independently on each subset $\hat{\mathbf{\Theta}}_m$ of $\hat{\mathbf{\Theta}}$ to facilitate the manipulation of decrypted parameters. The DPRM works by first generating a pseudorandom number sequence to mask $\hat{\mathbf{\Theta}}_m$, and then transforming the masked version of $\hat{\mathbf{\Theta}}_m$ into the same distribution as $\hat{\mathbf{\Theta}}_m$. The detailed procedure of running DPRM is listed in Algorithm 1. For each $\hat{\mathbf{\Theta}}_m$ in the partition of $\hat{\mathbf{\Theta}}$, the DPRM sequentially performs:

1) $RandMask(\hat{\mathbf{\Theta}}_m, \kappa_m) \rightarrow \mathbf{C}_m$: Given a secret key κ_m produced by a key generator, we utilize the *forward secure pseudorandom number generator* (FSPRGN) [30] $\mathcal{G}_{fs}$ to produce a pseudorandom number sequence $\mathbf{R}_m$ with

length $|\hat{\mathbf{\Theta}}_m|$. The subset $\hat{\mathbf{\Theta}}_m$ of the dominated set $\hat{\mathbf{\Theta}}$ then is masked by this pseudorandom number sequence as $\mathbf{C}_m = \hat{\mathbf{\Theta}}_m + \mathbf{R}_m$ (line 2).

2) $Mapping(\mathbf{C}_m) \rightarrow \hat{\mathbf{C}}_m$: An obvious limitation of $RandMask$ is that masked parameters $\mathbf{C}_m$ are statistically different from the original $\hat{\mathbf{\Theta}}_m$. Attackers can then easily identify $\mathbf{C}_m$, which implies that locations of encrypted parameters are leaked. To tackle this serious threat, we have to further transform $\mathbf{C}_m$ to another domain whose distribution is consistent with $\hat{\mathbf{\Theta}}_m$. To this end, the fundamental issue now is to estimate the distribution of parameters in $\hat{\mathbf{\Theta}}_m$.

Observing the histograms of parameters of several layers in the CNN model VGG19 [32] in Fig. 2, it is reasonable to infer that parameters in each layer of a CNN model follow symmetric distributions such as Gaussian. Hence, we assume that parameters $\mathbf{\Theta}^l$ of the l-th layer follow a Gaussian distribution $\mathcal{N}(\theta|\mu^l, \sigma^l)$, where the mean μ^l and standard deviation σ^l can be estimated with sample mean and sample standard deviation of parameters in $\mathbf{\Theta}^l$.

Note that the masked parameters in $\mathbf{C}_m$ are distributed across L layers, where masked parameters in the l-th layer are denoted by $\mathbf{C}_m^l$ $(l = 1, ..., L)$. Making the distribution of $\mathbf{C}_m$ be consistent with that of $\hat{\mathbf{\Theta}}_m$ reduces to transform each $\mathbf{C}_m^l$ into the Gaussian distribution $\mathcal{N}(\theta|\mu^l, \sigma^l)$ as $\hat{\mathbf{\Theta}}^l \subset \mathbf{\Theta}^l$. As shown in Algorithm 1 (lines 3-10), for $l = 1, ..., L$, we first scale elements in $\mathbf{C}_m^l$ into the interval $[0, 1]$, and then transform these scaled c's with the inverse Gaussian CDF as follows:

$$\begin{aligned} &c \leftarrow (c - u_m^l)/(v_m^l - u_m^l), \quad \forall c \in \mathbf{C}_m^l, \\ &\hat{c} \leftarrow F^{-1}(c|\mu^l, \sigma^l), \end{aligned} \tag{8}$$

where $u_m^l = \min(\mathbf{C}_m^l)$, $v_m^l = \max(\mathbf{C}_m^l)$, and $F^{-1}(\cdot|\mu^l, \sigma^l)$ is the inverse CDF of $\mathcal{N}(\theta|\mu^l, \sigma^l)$. After transforming all subsets $\mathbf{C}_m^l$'s in $\mathbf{C}_m$ as (8), we obtain the ciphertext $\hat{\mathbf{C}}_m$ of $\hat{\mathbf{\Theta}}_m$.

By repeatedly implementing the DPRM above on each $\hat{\mathbf{\Theta}}_m$ in $\hat{\mathbf{\Theta}}$, we encrypt the dominated set $\hat{\mathbf{\Theta}}$ of $\mathcal{F}_{\mathbf{\Theta}}$ into the ciphertext $\hat{\mathbf{C}}$. The protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ thus could be constructed by replacing $\hat{\mathbf{\Theta}}$ in $\mathcal{F}_{\mathbf{\Theta}}$ with $\hat{\mathbf{C}}$ (line 11). Apart from the protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$, the **Encrypt** module also generates secret keys $\mathbf{K}$, scaling parameters $\mathbf{U}, \mathbf{V}$, and the statistical information $\boldsymbol{\mu}, \boldsymbol{\sigma}$, for generating permissions.

3.3. Assign

The performance of the protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ now is heavily suppressed since dominated parameters $\hat{\mathbf{\Theta}}$ are encrypted. Users could access $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ only with permissions composed of locations $\hat{\mathbf{E}}$ of dominated parameters and other five security-related components outputted from the **Encrypt** module. More precisely, let $\mathbf{S}_{\hat{m}}$'s $(\hat{m} = 1, ..., M)$ be M different levels of permissions, each of which is generated as follows:

$$\mathbf{S}_{\hat{m}} = \{\hat{\mathbf{E}}_m, \mathbf{u}_m, \mathbf{v}_m, \kappa_m, \boldsymbol{\mu}, \boldsymbol{\sigma}\}_{m=1}^{\hat{m}}. \tag{9}$$

Here, each item $\{\hat{\mathbf{E}}_m, \mathbf{u}_m, \mathbf{v}_m, \kappa_m, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ could independently decrypt the ciphertext of the dominated parameter subset $\hat{\mathbf{\Theta}}_m$ in $\hat{\mathbf{\Theta}}$. Details will be introduced in Section 3.4.

Also, the size of a permission $\mathbf{S}_{\hat{m}}$ in bits is $(b_\kappa + 64L)\hat{m} + \frac{16L\hat{m}}{M}\phi$ (please refer to supplementary materials for detailed calculations), where b_κ is the bit size of the secret key κ, and ϕ is the number of dominated parameters in each layer. For the prevailing models, VGG19 and DnCNN considered in our experiment, the size of permissions is less than 508KB. Such overhead is acceptable in practice.

Algorithm 2: The scheme of $Decrypt$ module

Input: $\mathbf{S}_{\hat{m}} = \{\hat{\mathbf{E}}_m, \mathbf{u}_m, \mathbf{v}_m, \kappa_m, \boldsymbol{\mu}, \boldsymbol{\sigma}\}_{m=1}^{\hat{m}}$, protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$.
Output: Decrypted model: $\mathcal{F}_{\breve{\mathbf{\Theta}}_{\hat{m}}}$

1 **for** $m = 1, ..., \hat{m}$ **do**
2 $\hat{\mathbf{C}}_m \leftarrow$ parameters in $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ at location $\hat{\mathbf{E}}_m$;
3 **for** $l = 1, ..., L$ **do**
4 $\mathbf{C}_m^l = \emptyset, \hat{\mathbf{C}}_m = \{\hat{\mathbf{C}}_m^l\}_{l=1}^L$;
5 **for** $\hat{c}$ **in** $\hat{\mathbf{C}}_m^l$ **do**
6 $c \leftarrow F(c|\mu^l, \sigma^l)$, $c \leftarrow (\hat{c} + u_m^l)(v_m^l - u_m^l)$. $\triangleright$ $\mu^l \in \boldsymbol{\mu}$, $\sigma^l \in \boldsymbol{\sigma}, u_m^l \in \mathbf{u}_m, v_m^l \in \mathbf{v}_m$;
7 $\mathbf{C}_m^l \leftarrow \mathbf{C}_m^l \bigcup \{c\}$;
8 $\mathbf{C}_m = \{\mathbf{C}_m^l\}_{l=1}^L, \mathbf{R}_m \leftarrow \mathcal{G}_{fs}(\kappa_m, |\mathbf{C}_m|)$;
9 $\hat{\mathbf{\Theta}}_m = \mathbf{C}_m - \mathbf{R}_m$;
10 $\breve{\mathbf{\Theta}}_{\hat{m}} = \{\hat{\mathbf{\Theta}}_m\}_{m=1}^{\hat{m}} \cup (\tilde{\mathbf{\Theta}} / \{\hat{\mathbf{C}}_m\}_{m=1}^{\hat{m}})$;
11 $\mathcal{F}_{\breve{\mathbf{\Theta}}_{\hat{m}}} \leftarrow$ replace $\tilde{\mathbf{\Theta}}$ of $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ with $\breve{\mathbf{\Theta}}_{\hat{m}}$;
12 **return** $\mathcal{F}_{\breve{\mathbf{\Theta}}_{\hat{m}}}$

3.4. Decrypt

When receiving a permission $\mathbf{S}_{\hat{m}}$ from users, the **Decrypt** module is operated as follows. For each item $\{\hat{\mathbf{E}}_m, \mathbf{u}_m, \mathbf{v}_m, \kappa_m, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ in $\mathbf{S}_{\hat{m}}$, the decryption procedure starts with locating the ciphertext $\hat{\mathbf{C}}_m$ of dominated parameter subset $\hat{\mathbf{\Theta}}_m$ according to the location $\hat{\mathbf{E}}_m$ (line 2). The decryption of $\hat{\mathbf{C}}_m$ is merely the inverse process of the encryption of $\hat{\mathbf{\Theta}}_m$. As shown in Algorithm 2, for those encrypted parameters $\hat{\mathbf{C}}_m^l$ of $\hat{\mathbf{C}}_m$ in the l-th layer, we first transfer them back to the random masked version of $\hat{\mathbf{\Theta}}_m^l$, i.e., $\mathbf{C}_m^l$, via the following transformations (lines 6-7)

$$\begin{aligned} \mathbf{C}_m^l = \{c | \forall \hat{c} \in \hat{\mathbf{C}}_m^l, &c \leftarrow F(\hat{c}|\mu^l, \sigma^l), \\ &c \leftarrow (c + u_m^l)(v_m^l - u_m^l)\}, \end{aligned} \tag{10}$$

where $F(\cdot, \mu^l, \sigma^l)$ is the CDF of Gaussian distribution $\mathcal{N}(\theta|\mu^l, \sigma^l)$. By repeating the above procedure (10) for $l = 1, ..., L$, we obtain the masked version $\mathbf{C}_m$ of dominated parameters subset $\hat{\mathbf{\Theta}}_m$. Then, we input the key κ_m to the FSPRNG to generate the same random number sequence $\mathbf{R}_m$ used to mask $\hat{\mathbf{\Theta}}_m$ (line 8). The subset

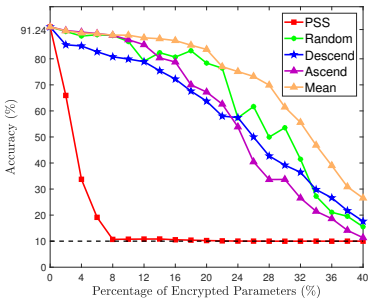

Figure 3: The classification accuracy of the protected VGG19 on CIFAR10 with respect to different percentages of encrypted parameters.

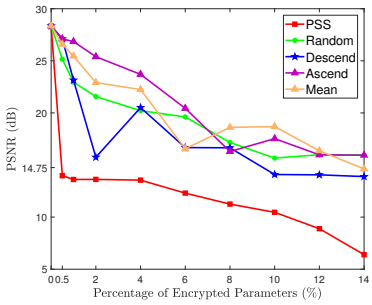

Figure 4: The denoising performance of the protected DnCNN with respect to different percentages of encrypted parameters.

$\hat{\mathbf{\Theta}}_m$ readily is decrypted by removing $\mathbf{R}_m$ from $\mathbf{C}_m$, i.e., $\hat{\mathbf{\Theta}}_m = \hat{\mathbf{C}}_m - \mathbf{R}_m$ (line 9).

With the permission $\mathbf{S}_{\hat{m}}$, we can recover $\hat{m}$ dominated parameter subsets $\{\hat{\mathbf{\Theta}}_m\}_{m=1}^{\hat{m}}$ in $\hat{\mathbf{\Theta}}$ by repeating the above procedure.

The resulting decrypted model is denoted by $\mathcal{F}_{\breve{\mathbf{\Theta}}_{\hat{m}}}$ (lines 10-11). Obviously, when $\hat{m} < M$, $M - \hat{m}$ dominated parameter subsets $\{\hat{\mathbf{\Theta}}_m\}_{m=\hat{m}+1}^{M}$ are still encrypted in $\mathcal{F}_{\breve{\mathbf{\Theta}}_{\hat{m}}}$. Hence it is a partially encrypted version of $\mathcal{F}_{\mathbf{\Theta}}$. As $\hat{m}$ increases to M, all dominated parameters $\hat{\mathbf{\Theta}} = \{\hat{\mathbf{\Theta}}_m\}_{m=1}^{M}$ are recovered, and $\mathcal{F}_{\breve{\mathbf{\Theta}}_M} = \mathcal{F}_{\mathbf{\Theta}}$. Thus, the full performance of $\mathcal{F}_{\mathbf{\Theta}}$ is then released. On the contrary, if no permissions are fed into the **Decrypt** module, the performance of $\mathcal{F}_{\mathbf{\Theta}}$ is still suppressed in the encrypted model. Consequently, we have provided users with hierarchical services (different performance levels) of the pretrained model and achieved access control against unauthorized users.

4. Imperceptibility of ciphertext

In this section, we theoretically prove the imperceptibility of the ciphertext, which significantly enhances the security of the protected model.

Let $\tilde{\mathbf{\Theta}}^l$ and $\mathbf{\Theta}^l$ be parameters of the l-th layer of the protected model $\mathcal{F}_{\tilde{\mathbf{\Theta}}}$ and the pretrained model $\mathcal{F}_{\mathbf{\Theta}}$, respectively. Recall that $\tilde{\mathbf{\Theta}}^l$ is a partially encrypted version of $\mathbf{\Theta}^l$ where dominated parameters $\hat{\mathbf{\Theta}}^l$ are encrypted into the ciphertext $\hat{\mathbf{C}}^l$. To ensure that attackers cannot recover these dominated parameters $\mathbf{\Theta}^l$ as discussed in Section 2.2, a prerequisite is the imperceptibility of the ciphertext $\hat{\mathbf{C}}^l$. Taking $\tilde{\mathbf{\Theta}}^l$ as a noisy version of $\mathbf{\Theta}^l$, the dominated parameters $\hat{\mathbf{\Theta}}^l$ of $\mathbf{\Theta}^l$ are contaminated by the noise $\hat{\mathbf{W}}^l = \hat{\mathbf{C}}^l - \hat{\mathbf{\Theta}}^l$. If attackers cannot perceive the added noise $\hat{\mathbf{W}}^l$ by observing $\tilde{\mathbf{\Theta}}^l$, the ciphertext $\hat{\mathbf{C}}^l$ is also imperceptible. To prove that $\hat{\mathbf{W}}^l$ is imperceptible, we resort to the measure equivocation defined by Shannon [28], which evaluates the information leakage of $\hat{\mathbf{W}}^l$ when observing $\tilde{\mathbf{\Theta}}^l$. This measure is also widely used in the field of steganography and watermarking [4, 16]. In our analysis, we modify the original equivocation into an equivalent form, $\hat{E}(\hat{\mathbf{W}}^l, \tilde{\mathbf{\Theta}}^l) = I(\hat{\mathbf{W}}^l; \tilde{\mathbf{\Theta}}^l)/H(\hat{\mathbf{W}}^l)$, where $H(\hat{\mathbf{W}}^l)$ is the entropy of $\hat{\mathbf{W}}^l$ and $I(\hat{\mathbf{W}}^l; \tilde{\mathbf{\Theta}}^l)$ is the mutual information between $\hat{\mathbf{W}}^l$ and $\tilde{\mathbf{\Theta}}^l$. If we can prove that $\hat{E}(\hat{\mathbf{W}}^l, \tilde{\mathbf{\Theta}}^l)$ is negligible, the information leakage of $\hat{\mathbf{W}}^l$ is also negligible, and thus $\hat{\mathbf{W}}^l$ is imperceptible. We have the following theorem on the magnitude of $\hat{E}(\hat{\mathbf{W}}^l, \tilde{\mathbf{\Theta}}^l)$.

Theorem 1. *The modified equivocation between* $\hat{\mathbf{W}}^l$ *and* $\tilde{\mathbf{\Theta}}^l$ *is of order* $|\hat{\mathbf{W}}^l|^{-1/2}$. *That is, as* $|\hat{\mathbf{W}}^l| \to \infty$,

$$\hat{E}(\hat{\mathbf{W}}^l, \tilde{\mathbf{\Theta}}^l) = O(|\hat{\mathbf{W}}^l|^{-1/2}). \tag{11}$$

The detailed proof is given in the supplementary materials. As can be easily seen from (11), $\hat{E}(\hat{\mathbf{W}}^l, \tilde{\mathbf{\Theta}}^l)$ is negligible if $|\hat{\mathbf{W}}^l|$ is large enough, or equivalently if the number of dominated parameters in $\hat{\mathbf{\Theta}}^l$ is large enough. This condition is quite reasonable in practice as CNN models typically contain a massive of parameters. Taking VGG19 as an example, the number of parameters in the 8-th layer is of order 10^6. Supposing 10% parameters are identified as dominated ones, the order of $|\hat{\mathbf{W}}^l|$ would be 10^5 and that of $\hat{E}(\hat{\mathbf{W}}^l, \tilde{\mathbf{\Theta}}^l)$ is $10^{-5/2}$. Thus we claim that the ciphertext in the protected model is imperceptible.

5. Experiments results

In this section, we experimentally verify that our system can achieve the goals defined in Section 2.3. At first, we briefly introduce the experiment setup.

We consider two CNN models: the classification model VGG19 [32] and the denoising model DnCNN [40]. We train VGG19 on CIFAR10 [17] and DnCNN on 300 noisy images from ImageNet [8] with Gaussian noise (noise level 50). The best classification accuracy of the pretrained VGG19 on 10000 test images of CIFAR10 is 91.24%, and the best PSNR of the pretrained DnCNN on 40 noisy images is 28.35dB. According to our observations, encrypting several layers of VGG19 and DnCNN is enough to cause the

Model	Level of Permissions: $\hat{m}$					
	0	1	2	3	4	5
VGG19 (%)	10.00	65.41	78.65	82.58	87.33	91.24
DnCNN (dB)	13.61	20.12	23.23	25.52	26.58	28.35

Table 1: Hierarchical Performance of Decrypted Models

maximal performance degradation. Therefore, in all experiments below, we only selectively encrypt parameters of 1-st, 2-th, 5-th, and 9-th layers of VGG19, and that of DnCNN are 6-th, 9-th, and 12-th layers. Due to the space limit, more experiment results can be found in the supplementary materials.

5.1. Effectiveness of the proposed SE

For rejecting unauthorized access, a successful protection should degrade the performance of the encrypted model into the worst situation, if no permission is granted. For instance, for VGG19 on CIFAR10, the worst prediction accuracy is 10% (random guess among 10 classes). To the best of our knowledge, our proposed scheme is the first one to protect CNN models with SE. For preparing the competing algorithms, we design four different strategies to select parameters from considered layers, and encrypt them with the proposed DPRM. 1) **Random**: We select parameters uniformly at random; 2) **Mean**: We extract parameters around the mean value. This selection strategy is motivated by the observation that parameters are concentrated around the mean value (see Fig. 2); 3) **Descending**: A reasonable hypothesis is that the importance of parameters is positively correlated to their values. Thus, we select parameters in the descending order of their values; and 4) **Ascending**: Conversely, we select parameters in ascending order.

Fig. 3 shows the classification accuracy of VGG after encrypting different percentages of parameters of considered layers. We can observe that our PSS (red curve) degrades the VGG19 to the worst case when only 8% of parameters are encrypted, while the best competing one needs to encrypt 40% of parameters. Such a result demonstrates the effectiveness of the proposed SE for protecting VGG19. Similarly, Fig. 4 shows the denoising performance of the DnCNN when PSS and other competing algorithms are used for the parameter selection. As can be observed, the denoising performance degrades very quickly when model parameters are selected by PSS and encrypted. Note that, to eliminate the randomness, results in Fig. 3 and 4 are the averages over repeating the encryption for 20 times.

5.2. Hierarchical performance of the released model

We now demonstrate that models decrypted from the protected model with various permissions could exhibit different levels of performance. We selectively encrypt 10% (2%) parameters of considered layers of the VGG19 (DnCNN). Then, 5 permissions $\mathbf{S}_{\hat{m}}$ ($\hat{m} = 1, ..., 5$) are generated by the module **Assign** in Section 3.3 ($M = 5$). These permissions are fed into the **Decrypt** module to decrypt the protected VGG19 (DnCNN). The performance of the decrypted VGG19 and DnCNN with respect to the 5 permissions is recorded in Table 1. As the increase of $\hat{m}$, the decrypted VGG19 (DnCNN) exhibits different levels of accuracy (PSNR) and reaches the best one when inputting the highest permission ($\hat{m} = 5$). Here, $\hat{m} = 0$ implies no permission is granted, corresponding to the worst prediction accuracy. For a better illustration of the hierarchical performance of the released DnCNN, the visualized denoising results of a test image under different permissions are shown in Fig. 5. One can see from Fig. 5(d)-(f) that a higher level of permission (larger $\hat{m}$) endows users with a better denoised image. Interestingly, for users without permission ($\hat{m} = 0$, see Fig. 5(c)), the resulting denoised image is even worse than the original one.

Attacking Goals: VGG19 = 65.41% DnCNN = 20.12dB						
Model	Denoising via Wavelets			Denoising via Filters		
	DB2	**Haar**	**Sym9**	**Average**	**Gaussian**	**Median**
VGG19	21.15%	23.13%	24.72%	19.54%	18.42%	17.31%
DnCNN	15.17dB	16.71dB	15.31dB	13.54dB	14.32dB	15.51dB

Model	Layer-wise	Transferring
VGG19	55.15%	39.14%
DnCNN	18.24dB	17.49dB

Table 2: Performance of Protected Models under Attacks

5.3. Security against potential attacks

We then evaluate the security of the protected model against denoising and retraining attacks considered in the threat model. First of all, we define the attackers' capability and goals to quantitatively evaluate whether an attack succeeds. **Attackers' Capability**: Attackers could challenge the protected model under one or all of the following capabilities. 1) Attackers can access all parameters of the protected model; 2) Attackers possess limited data (10%) used for training the pretrained model; 3) Attackers own another model that has the same structure as the protected one; but is trained over another dataset; and 4) Attackers have known which layers in the model are encrypted. **Attackers' Goal**: The attackers' goal is to obtain the same model performance as a user with the lowest permission. As shown in Table 1, for VGG19 and DnCNN, the attackers' goals are 65.41% classification accuracy and 20.12dB of the denoised image, respectively.

We now consider several potential denoising and retraining attacks and show that the protected model is secure against them. Note that, in Section 4, we have proved the imperceptibility of the encrypted parameters. Hence, all attacks discussed below are based on the premise that the locations of the encrypted parameters are unknown.

(a) Clean (b) Noisy (14.15dB) (c) $\hat{m} = 0$ (8.91dB) (d) $\hat{m} = 1$ (19.22dB) (e) $\hat{m} = 3$ (24.40dB) (f) $\hat{m} = 5$ (29.91dB)

Figure 5: Images to illustrate the hierarchical performance of the protected DnCNN with different permissions. (a) Clean image; (b) Noisy image; (c) The output image of the protected DnCNN without permission; (d-f) Denoising results of the noisy image by utilizing the protected DnCNN with different permissions $\mathbf{S}_{\hat{m}}$'s.

DENOISING ATTACKS

1) **Denoising via Wavelets**: Attackers take selectively encrypted parameters in each considered layer as a partially contaminated discrete signal (flattening parameter tensors into a 1-D vector). They can remove the noise by resorting to wavelet denoising techniques. To simulate attackers' behavior, we consider three different wavelets: **DB2**, **Haar**, and **Sym9**. As shown in Table 2, the accuracies of the protected model suffering the three wavelet based denoising attacks are 21.15%, 23.13%, and 24.72%, respectively. Obviously, none of them achieves an accuracy better than the attacking goal 65.41%. For DnCNN, the best performance given by the three types of attack is 16.71dB, which is still inferior to the attacking goal (20.12dB). Thus, we conclude that the VGG19 and DnCNN protected by our scheme are secure against the wavelets based denoising attacks.

2) **Denosing via Filters**: The significant performance degradation of the encrypted model possibly ascribes to some abnormally large or small parameters. Attacker could implement the average filter or the median filter on the 1-D signal of encrypted parameters. Moreover, the Gaussian filter possibly is suitable since the noise used to encrypt dominated parameters follows a Gaussian distribution. The performance of the protected VGG19 and DnCNN under these filter based attacks is recorded in Table 2. As can be seen, for VGG19, all the restored accuracies after attacks cannot exceed the attacking goal (65.41%). Similarly, for DnCNN, the best PSNR of the attacked model is 15.51dB, which is still worse than the attacking goal (20.12dB). Therefore, the encrypted VGG19 and DnCNN models are secure against the filtering based attacks.

RETRAINING ATTACKS

1) **Layer-wise**: Since only several layers are encrypted by our proposed scheme, attackers could retrain each layer independently by fixing parameters of other layers, based on the available training data. The classification accuracy of the retrained VGG19 is recorded in Table 2. It can be seen that the resulting accuracy is 55.15%, which is lower than the attacking goal 65.41%. A similar result for DnCNN is 18.24dB, which is also worse than the attacking goal (20.12dB). Thus, under the assumption on attackers' capability and the defined attacking goal, we conclude that this type of retraining attack is not successful.

2) **Transfering**: Attackers may hypothesize that the distribution of encrypted parameters possibly is consistent among models with the same structure; but trained on another dataset. They thus implement the proposed PSS strategy on a new VGG19 trained on other ten classes from CIFAR100 [17] and obtain locations of dominated parameters of this VGG19. Then, they retrain parameters of the protected VGG19 at locations learned from the new VGG19. The performance of such attacked model is only 39.14%, far from the attacking goal 65.41%. We also try to attack DnCNN by using the same strategy, where the locations of encrypted parameters are transferred from another DnCNN trained over the noisy images with noise level 25. In this case, the PSNR value of the denoised image is 17.49dB, which is still lower than the attacking goal (20.12dB).

6. Conclusions

In this paper, we have proposed a SE algorithm to protect a CNN model by firstly selecting important parameters from this model with the PSS and then encrypting the selected parameters with DPRM. A system based on our SE can prevent a CNN model from unauthorized access and also provide authorized users with hierarchical services. Experimental results have been provided to show the effectiveness and security of the SE on protecting CNN models. **Acknowledgments:** This work was supported by Macau Science and Technology Development Fund under SKL-IOTSC-2018-2020, 077/2018/A2, 0015/2019/AKP, and 0060/2019/A1, by Research Committee at University of Macau under MYRG2018-00029-FST and MYRG2019-00023-FST, by Natural Science Foundation of China under 61971476. This work was also supported by Alibaba Group through Alibaba Innovative Research Program.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 4

[2] Mohamed Abomhara, Omar Zakaria, Othman O Khalifa, AA Zaidan, and BB Zaidan. Enhancing selective encryption for h. 264/avcusing advanced encryption standard. *International Journal of Computer and Electrical Engineering*, 2(2):223, 2010. 1

[3] Ahmed M Ayoup, Amr H Hussein, and Mahmoud AA Attia. Efficient selective image encryption. *Multimedia Tools and Applications*, 75(24):17171–17186, 2016. 1

[4] François Cayre, Caroline Fontaine, and Teddy Furon. Watermarking security: theory and practice. *IEEE Transactions on Signal Processing*, 53(10):3976–3987, 2005. 6

[5] Gong Cheng, Junwei Han, Peicheng Zhou, and Dong Xu. Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection. *IEEE Transactions on Image Processing*, 28(1):265–278, 2018. 1

[6] Don Coppersmith, Donald Byron Johnson, and Stephen M Matyas. A proposed mode for triple-des encryption. *IBM Journal of Research and Development*, 40(2):253–262, 1996 1996. 1

[7] Surya Nepal Rajiv Ranjan Deepak Puthal, Xindong Wu and Jinjun Chen. Seen: A selective encryption method to ensure confidentiality for big sensing data streams. *IEEE Transactions on Big Data*, pages 1–1, 2017. 1

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6

[9] Jinglong Du, Zhongshi He, Lulu Wang, Ali Gholipour, Zexun Zhou, Dingding Chen, and Yuanyuan Jia. Super-resolution reconstruction of single anisotropic 3d mr images using residual convolutional neural network. *Neurocomputing*, 392:209–220, 2020. 1

[10] Ross Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1

[11] Marco Grangetto, Enrico Magli, and Gabriella Olmo. Multimedia selective encryption by means of randomized arithmetic coding. *IEEE Transactions on Multimedia*, 8(5):905–917, 2006. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3

[13] Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1689–1697, 2017. 1

[14] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, pages 1–12, 2017. 3

[15] Jakub Jurek, Marek Kociński, Andrzej Materka, Marcin Elgalal, and Agata Majos. Cnn-based superresolution reconstruction of 3d mr images using thick-slice scans. *Biocybernetics and Biomedical Engineering*, 40(1):111–125, 2020. 1

[16] Yan Ke, Jia Liu, Min-Qing Zhang, Ting-Ting Su, and Xiao-Yuan Yang. Steganography security: Principle and practice. *IEEE Access*, 6:73009–73022, 2018. 6

[17] Alex Krizhevsky, Hinton, and Geoffrey. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6, 8

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3

[19] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Exploiting kernel sparsity and entropy for interpretable cnn compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2019. 1

[20] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318, 2020. 1

[21] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, pages 1–12, 2017. 3

[22] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, pages 1–12, 2016. 3

[23] Med SalimBouhlela Med Karim Abdmouleha, Ali Khalfallaha. A novel selective encryption scheme for medical images transmission based-on jpeg compression algorithm. *Procedia Computer Science*, pages 1–1, 2017. 1

[24] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations*, 2018. 1

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 4

[26] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978R. 1

[27] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. wofish: A 128-bit block cipher. *NIST AES Proposal*, 15(1):23–91, 1998. 1

[28] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949. 1, 6

[29] Oran Shayer, Dan Levi, and Ethan Fetaya. Learning discrete weights using the local reparameterization trick. In *International Conference on Learning Representations*, 2018. 3

[30] Shoup and Victor. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004. 4

[31] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of Machine Learning Research*, pages 1–9, 2017. 1

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, pages 1–10, 2015. 5, 6

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 3

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 3

[35] Marc Van Droogenbroeck and Raphaël Benedett. Techniques for a selective encryption of uncompressed and compressed images. *Advanced Concepts for Intelligent Vision Systems, Proceedings*, pages 90–97, 2002. 1

[36] Jing Xiao, Liang Liao, Qiegen Liu, and Ruimin Hu. Cisi-net: Explicit latent content inference and imitated style rendering for image inpainting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 354–362, 2019. 1

[37] Zhaoyi Yan, Xiaoming Li, Mu Li, Wangmeng Zuo, and Shiguang Shan. Shift-net: Image inpainting via deep feature rearrangement. In *Proceedings of the European Conference on Computer Vision*, pages 1–17, 2018. 1

[38] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, Thomas Huang, and S. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018. 1

[39] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018. 1

[40] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 6