AN ENCRYPTION FRAMEWORK FOR PRE-TRAINED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Having consumed huge amounts of training data and computational resource, large-scale pre-trained models are often considered as key assets of AI service providers. This raises an important problem: *how to prevent these models from being maliciously copied when they are running on customers' computing device?* We answer this question by adding a set of **confusion neurons** into the pre-trained model, where the position of these neurons is encoded into a few integers that are easy to be encrypted. We find that most often, a small portion of confusion neurons are able to effectively contaminate the pre-trained model. Thereafter, we extend our study to a bigger picture that the customers may develop algorithms to eliminate the effect of confusion neurons and recover the original network, and we show that our simple approach is somewhat capable of defending itself against the fine-tuning attack.

1 INTRODUCTION

In the deep learning era (LeCun et al., 2015), neural networks have become the standard (and powerful) tool of learning representations. The past decade has witnessed the application on a wide range of applications including computer vision (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), natural language processing (Vaswani et al., 2017), *etc*. Recently, it is verified that pre-trained models (Brown et al., 2020; Dosovitskiy et al., 2020; Senior et al., 2020) built upon a large number of data can be transferred into specific scenarios by fine-tuning its parameters in a relatively smaller dataset and a shorter training procedure. This has laid the foundation of a novel paradigm of AI development that the **service provider** makes use of abundant data and computational resource to offer pre-trained models, so that the **customers** can build their work on top of the pre-trained models to save their costs.

However, pre-trained models usually involve plenty of annotated datasets and powerful training resources, which are considerably expensive. Therefore, they are of great value to the service providers, but when the customers have access the models (*e.g.*, the models are ran on the devices controlled by the customer), they can copy the models from the memory which may violate the intellectual properties of the service provider. Therefore, protecting the intellectual property of the pre-trained models has become an urgent problem to be solved.

To protect the models, most existing methods are based on watermarking (Uchida et al., 2017; Nagai et al., 2018; Fan et al., 2019; Zhang et al., 2018). Specifically, the watermark is embedded in the model, so we can extract the watermark to verify the ownership of the model. However, as passive verification methods, watermarking-based methods cannot prevent unauthorized use of the model, which is unacceptable to the provider. In addition, some researchers proposed encryption-based methods (Maung & Kiya, 2020; Gomez et al., 2018; 2019; Lin et al., 2021; Xue et al., 2021; Cai et al., 2019), which make the encrypted model cannot be used successfully without the secret key. Maung & Kiya (2020) proposed to use the training data after secret key preprocessing to train the model, and so the model only works when the input sample is preprocessed by the secret key. However, this method introduces high computational overhead especially when there are a substantial amount of input samples. Xue et al. (2021) proposed to make the model dysfunctional by encrypting the parameters of the model. However, the pre-trained model usually contains a tremendously large number of parameters, which makes the computational overhead of the parameter encryption unacceptable and also substantially enlarges the decryption time.

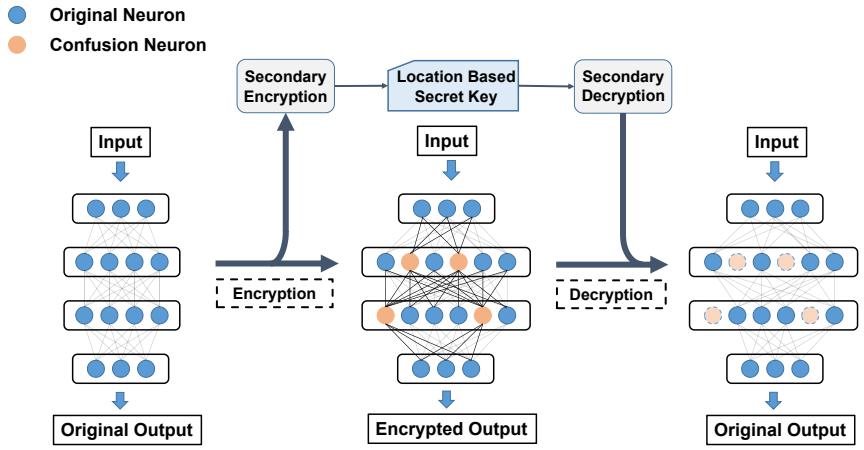

Figure 1: Overview of the proposed framework. A set of confusion neurons are added into the original network to obtain a larger network, which is referred to as the encrypted network. A secret key that records the position of the confusion neurons. During the inference process, the algorithm is easy to eliminate the confusion connections using the secret key, without which the prediction of encrypted network is totally different from that of the original network. Generally we do a neuron encryption of the location information.

Inspired by the new generation of AI frameworks, like dynamic networks (Han et al., 2021) and pathways (Chowdhery et al., 2022), which adapt or select a subset of neurons from a static network according to the inputs, we propose a simple encryption framework for pre-trained neural networks. The framework is shown in Figure 1. A number of **confusion neurons** are added to some layers of the network, where the position of these neurons are encoded into a short vector and can be well encrypted. The added neurons can gradually change the response of the network, layer by layer, so that the final output is largely contaminated, *i.e.*, the network loses the original ability of, say, recognizing the input image or understanding the input texts. With the authorized key, however, the algorithm can easily decipher the code and the network is equivalent to the original one as if no confusion neurons were added. Intuitively, the damage to the network is stronger with more confusion neurons added, so that the objective is to use fewer extra costs to gain a stronger ability of protection. Typically, we find that the location and weights of confusion neurons are crucial to achieve a good trade-off. That is to say, when the parameters are well set, we can often achieve heavy damage with a very small portion of redundancy.

In summary, the contributions of this paper are in two folds: (1) We advocate for the importance of encrypting neural networks and propose a confusion neurons based encryption framework, which can prevent unauthorized usage and run efficiently. (2) Extensive experiments on image classification, detection, segmentation, and natural language processing have verified the effectiveness of the proposed encryption method.

2 Related Work

Watermarking-based methods. Digital watermarks are extensively used in multimedia field to protect the copyright of images or videos, which inspires the researchers to address the model protection problem by embedding the watermarks into the models. In recent years, a large amount of watermarking-based methods (Uchida et al., 2017; Nagai et al., 2018; Fan et al., 2019; Zhang et al., 2018; 2020; Adi et al., 2018) are proposed with the increasing awareness of the model intellectual property protection. (Uchida et al., 2017)and (Nagai et al., 2018) are the earliest methods to embed the watermarks into the parameters by retraining the model with a regularization term. This scenario needs the access of model parameters, named white-box mode. Correspondingly, black-box

scenario means the verification step can only use the inputs and outputs of the model, which is more realistic. Methods represented by (Adi et al., 2018) use backdoor key images as watermarks, the labels of which are randomly selected except the true label. Thus the watermarks can be triggered with lower accuracy by a threshold during verification. However, all the watermarking-based methods are trying to protect the model after the unauthorized usage happen, thus they are called passive model protection methods.

Encryption-based methods. Different from watermarking-based methods, which passively verify the copyright of the model, encryption-based methods (Maung & Kiya, 2020; Gomez et al., 2018; 2019; Lin et al., 2021; Xue et al., 2021; Cai et al., 2019) attempt to make the encrypted model can not be used by malicious users without the secret key. Encryption-based methods can be divided into two routes, input encryption (Maung & Kiya, 2020; Gomez et al., 2018; 2019) and parameter encryption (Lin et al., 2021; Xue et al., 2021), respectively. Input encryption methods commit to train the model with encrypted images, which is achieved by pixel shuffle (Maung & Kiya, 2020) or homomorphic encryption (Gomez et al., 2018; 2019). It is worth noting that input encryption methods typically focus on data protection and always come with a loss of accuracy and efficiency. While parameter encryption methods are devoted to protect the parameters directly by disturbing the values(Xue et al., 2021; Cai et al., 2019) or the positions (Lin et al., 2021) of model parameters.

Discussion Although the watermarks added into the models can help verify the authority, they can only achieve passive protection of the model. The malicious users can still use the entire power of the model. While existing parameter-encryption based methods can prevent the valuable parameters being used without authorization, the well designed architecture of the model remains unprotected. Therefore, our method aims to achieve one-stop encryption of model architecture and model parameters under the premise of active intellectual property protection.

3 METHODOLOGY

This section begins with the problem setting of the pre-trained model protection, followed by detailed description of three parts of the proposed framework, **model encryption**, **secret key generation** and **model decryption**, respectively. Meanwhile, the overview of the proposed framework is illustrated in Figure 1.

3.1 PROBLEM SETTING

Denote the original network as $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$, where the $\boldsymbol{\alpha}$ stands for the network architecture (*e.g.*, for image recognition, it can be AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan & Zisserman, 2014), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), ViT (Dosovitskiy et al., 2020), *etc.*), $\mathbf{x}$ is the input (*e.g.*, an image for vision applications), and $\boldsymbol{\theta}$ is the parameters. Throughout this paper, we assume that the computational model has been pre-trained, *i.e.*, $\boldsymbol{\theta}$ has been optimized using a reasonable amount of training data on sufficient computational resource. Our goal is to protect the network from being maliciously copied, namely, used for commercial purposes without being authorized by the service provider.

3.2 MODEL ENCRYPTION

From a general viewpoint, when the pre-trained model is made 'plaintext' in the customer's device, a straightforward way to protect it is to add confusion information to destroy the output of the model (Maung & Kiya, 2020; Gomez et al., 2018; 2019; Lin et al., 2021; Xue et al., 2021; Cai et al., 2019). In this paper, we consider a simple method that inserts a controllable number of neurons into hidden layers of the network and counterfeits the connections related to these neurons as if they were members of the original model. *It is worth noting that the **neurons** mentioned here and below stand for the channels of CNN or the neurons of MLP*. According to the function of these added extra parameters, we name them as **confusion neurons**.

Formally, let the network $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$ have L layers, where the output of each layer is denoted by $\mathbf{z}_l$ and we use $\mathbf{z}_0 \equiv \mathbf{x}$ for convenience. Provided the architecture of $\boldsymbol{\alpha}$, we further define the configuration of the network to be the number of neurons on each layer, denoted by an array of

$[C_1, C_2, \ldots, C_L]$ where we exclude C_0 from the array since the input dimension is often unchangeable. Given the above definition, the encryption consists of two steps.

Step1: Allocating the location of confusion neurons. Intuitively, for a pre-trained model, if we add a few extra neurons to the first hidden layer and counterfeit the connections between these neurons and other originally-existing neurons, the output of $\mathbf{z}_1$ will be changed. Since the network is hierarchical and the subsequent outputs are often relying on $\mathbf{z}_1$, the final output can be largely altered from that of the original network. If the service provider performs such an action and stores the position of the confusion neurons, the customer will be difficult to derive the original output for every single input. Given a fixed confusion proportion $\mathcal{C}$, generalizing this idea to inserting confusion neurons into multiple layers yields the Algorithm. 1 below.

Algorithm 1: Model Encryption with Confusion Neurons

Input: confusion proportion $\mathcal{C}$; strategy $\mathcal{S}$; pre-trained model parameter $\boldsymbol{\theta}$ and architecture $\boldsymbol{\alpha}$.
Output: encrypted parameters $\boldsymbol{\theta_e}$; encrypted architecture $\boldsymbol{\alpha_e}$; positions of confusion neurons $[\boldsymbol{P_1}, \boldsymbol{P_2}, \ldots, \boldsymbol{P_L}]$.

Step1: Location allocation
$[C_1, C_2, \ldots, C_L] \leftarrow count(\boldsymbol{\theta}, \boldsymbol{\alpha})$
$\Delta C = sum([C_1, C_2, \ldots, C_L])$
for *layer l=1:L* **do**
  $\Delta C_l \leftarrow \Delta C / L$
  $\boldsymbol{P_l} \leftarrow [\,]$
  repeat
    $p_l \leftarrow sample(range(C_l + \Delta C_l))$
    if p_l *not in* $\boldsymbol{P_l}$ **then**
      $\boldsymbol{P_l}.append(p_l)$
  until $length(\boldsymbol{P_l})$ *equals* ΔC_l;

Step2: Weight assignment
$\boldsymbol{\theta_e} \leftarrow update(\boldsymbol{\theta}, \mathbf{0})$
for *layer l=1:L* **do**
  if $\mathcal{S}$ *is 'samp'* **then**
    $\boldsymbol{w_l} \leftarrow SampleWeight(l, \boldsymbol{P_l}, \boldsymbol{\theta_e})$
  else
    $\boldsymbol{w_l} \leftarrow InitalWeight(l, \boldsymbol{P_l})$
  $\boldsymbol{\theta_e} \leftarrow update(\boldsymbol{\theta_e}, \boldsymbol{w_l})$
$\boldsymbol{\alpha_e} \leftarrow encrypt(\boldsymbol{\alpha}, [\boldsymbol{P_1}, \boldsymbol{P_2}, \ldots, \boldsymbol{P_L}])$

Step2: Determining the weights of confusion neurons. The goal here is two-fold. Firstly, these weights should be capable of perturbing the network's output. Secondly, the weights are difficult to be either detected or attacked (*e.g.*, by fine-tuning the network on small datasets). This directly excludes the possibility that the weights are too small (so that adding them affects little to the network) or too large (so that they are clearly outliers and easily detected) compared to the original weights.

Here, we propose two strategies of assigning weights to confusion neurons, denoted by $\mathcal{S}$. As shown in Algorithm 1 The first one is to initialize the added confusion neurons with random noises, *e.g.*, the Gaussian noise or other by-default initialization methods that come with the specified networks, namely 'init'. The second one is to randomly sample weights from the original neurons and assign to confusion neurons, namely 'samp'. Though both strategies work very well in perturbing the network, their behaviors of defending the recovering attack are different. The weights of confusion neurons is assigned by the $update(\cdot)$ with zeros or the selected values. Specifically, random noises are easier to be detected, since they cannot keep the original distribution of weights unchanged, but are more resistant to fine-tuning, while the sampled weights are hard to detect but risk being tuned efficiently. In Section 4.3, we show that the mixed strategy (*i.e.*, randomly choosing a strategy for each confusion neuron) is a good choice.

For more intuitive demonstration, Figure 2 shows some representative examples where the encrypted network produces significantly different outputs from the original network. It is worthy to note that how confusion neurons contaminate the attention to important regions.

3.3 Secret key generation

As shown in Figure.1, the secret key is generated based on the locations of the added confusion neurons after the model encryption step. In addition, for each layer named as N_l where $1 \leqslant l \leqslant L$, there is a C_l-dimensional vector recording the actual location (*i.e.*, the ID of channels) that the confusion neurons have been inserted, denoted as $\boldsymbol{P_l}$. Then the secret key is generated and stored in dictionary format:

$$\mathcal{K}_{\text{encryption}} = \{N_l : \boldsymbol{P_l} \mid 1 \leqslant l \leqslant L\} \tag{1}$$

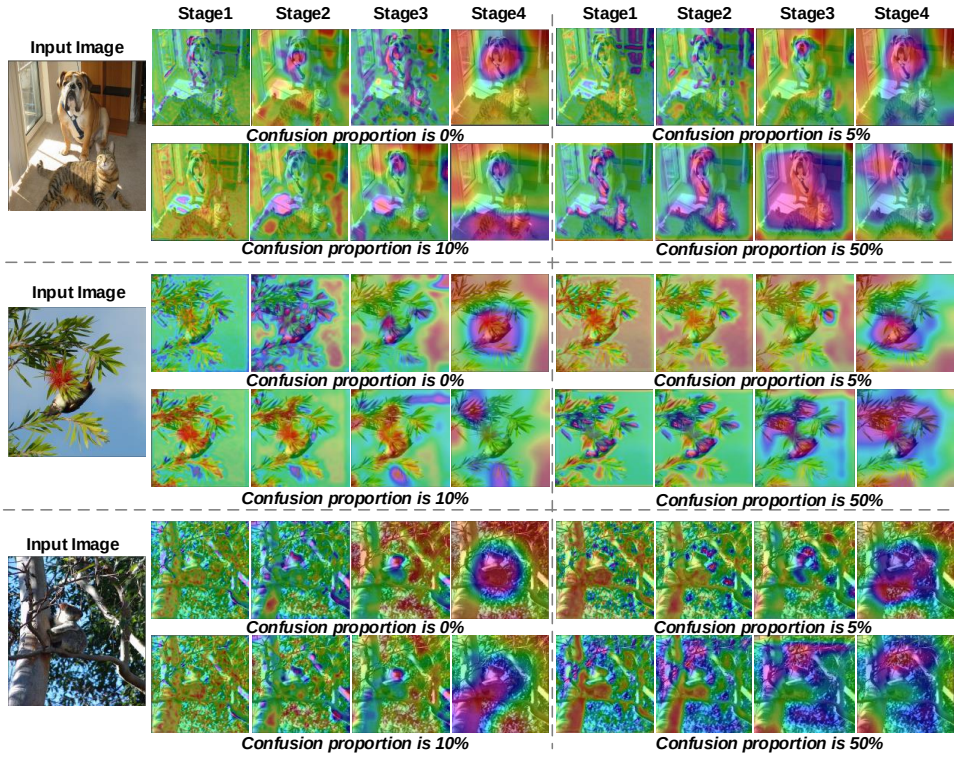

Figure 2: Visualization of the attention maps (plotted by the CAM algorithm (Zhou et al., 2016)) of ResNet50 after different amounts of confusion neurons are inserted. The original ResNet50 was trained on ImageNet. One can see that adding confusion neurons destroys the pre-trained model's ability to focus on the discriminative region of the image. Concretely, with the amount of confusion neurons increases, the attention maps from different stages of ResNet50 become diverged and thus the classification accuracy drops dramatically.

In addition, if the secret key is stored and distributed in plain text style, it may still be vulnerable to malicious theft and use. To further improve the security, we introduce binary encryption of the key file, noted as secondary encryption, and the adopted binary encryption methods can be arbitrarily chosen. As a result, we achieve effective and robust encryption for neural network models, and the computation required for binary encryption is scaled down from the entire model file to the location-based secret key file.

3.4 Model decryption

Last but not least, we briefly discuss the regular scheme to decrypt the model. When a customer is authorized to use of the pre-trained model, the service provider offers the secret keys using a section of ciphered codes. When the deep learning toolkit receives the secret keys, the encrypted network is literally equivalent to the original one by setting the confusion weights to zeros according to the secret keys, making it straightforward to perform either inference or fine-tuning beyond it. It is worth noting that after each fine-tuning procedure, the weights of confusion neurons also need to be updated so as to remain sheltered, which is easily done by calling the weight initialization module one more time, which is cheap in computation.

4 Experiments

In this section, we first explore the effects of confusion neurons at different locations and with weights on model protection and computational overhead to achieve a better tradeoff in Section 4.1. Then extensive experiments are conducted to verify the protection over different tasks in Section 4.2. Finally, we evaluate the ability of the proposed method against the fine-tuning based attack in Section 4.3.

4.1 TOWARDS A BETTER TRADEOFF BETWEEN COSTS AND PROTECTION

Before delving into details, we first note that for effective encryption, three important factors need to be considered, namely, the location of confusion neurons, the amount of confusion neurons, and the strategy of assigning weights to the confusion neurons. For the sake of simplicity, we by default adopt two assignment strategies, namely, the 'samp' strategy where we sample weights from originally existing weights, and the 'init' strategy where we follow the initialization method described in (He et al., 2015). All the experiments in this part are performed on the ResNet (He et al., 2016) series. For detailed settings, please refer to the next subsection.

4.1.1 THE LOCATION OF INSERTING CONFUSION NEURONS

We first consider different locations of inserting confusion neurons. The ResNet18 and ResNet50 models are used, and we try two schemes of assigning confusion neurons, one is to keep the confusion proportion $\mathcal{C}$ (*i.e.*, $\Delta C_l/(C_l + \Delta C_l)$ at the l-the layer) identical at all layers, and the other is to assign all confusion neurons to one single layer. Both the 'samp' and 'init' weight assignment strategies are evaluated.

Experimental results are summarized in Figure 3. One can take away an important message that since the shallow layers (*i.e.*, those closer to input) often have fewer parameters, assigning the same number of neurons to these layers often causes heavier damages. This seems to motivate us to inserting more confusion neurons to the shallow layers, however, we point out that an imbalanced assignment may cause encrypted network easier to be attacked. As an example, although adding almost all neurons to the first block can easily cause dramatic failure, it is also possible that the attacker simply discards the first layer and replace it with a module trained independently. Therefore,

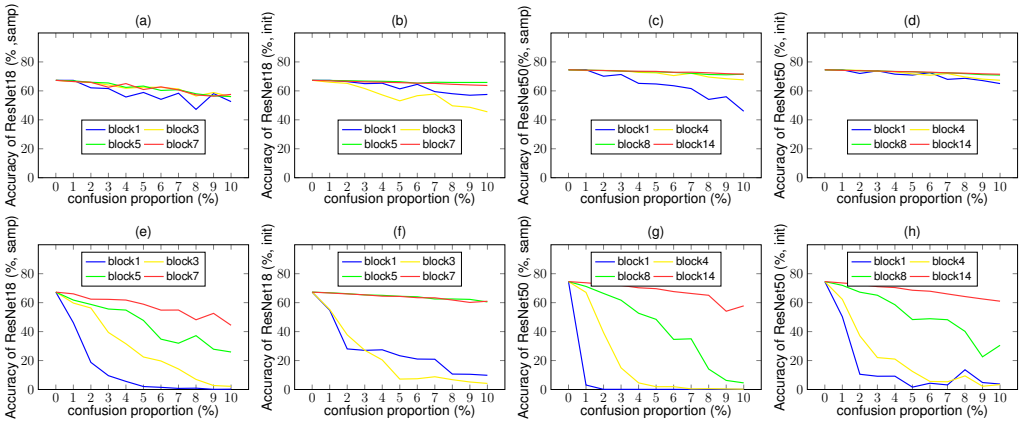

Figure 3: The impact of adding confusion neurons to different blocks of ResNet-18/50. The top row shows the setting that the same (relative) proportion of neurons are added to in each block, while the bottom row shows that the same (absolute) number of neurons are added. Please mind the slight difference between the 'samp' and 'init' strategies. All tests are made on ImageNet.

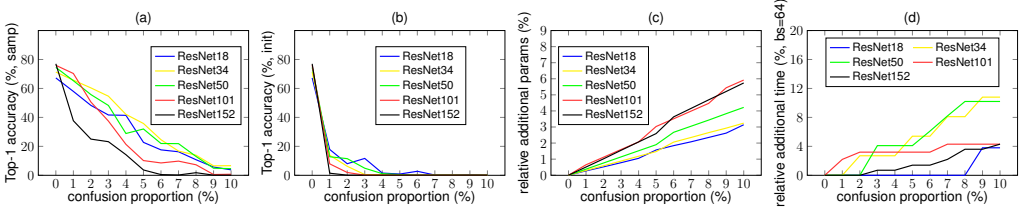

Figure 4: The impact of adding different amounts of confusion neurons to the ResNet series. All tests are made on ImageNet. The left two plots show the difference between the 'samp' and 'init' strategies where the latter is more effective, and the right two plots show the additional overheads required, which is linear to the confusion proportion.

Table 1: Top-1 and top-5 accuracy and parameters of the original and encrypted models on ImageNet validation dataset. Plain convolution models like VGG, skip-connection based models like ResNet, and transformer based models like ViT are tested. With a small amount of extra computation (less than 5%), the pre-trained models are encrypted to protect themselves.

Backbone	Original			Encrypted			ΔTop-1 (%) ↑	ΔTop-5 (%) ↑	ΔParams ↓
	Top-1 (%)	Top-5 (%)	Params (M)	Top-1 (%)	Top-5 (%)	Params (M)			
ResNet-18	67.27	87.74	11.69	22.44	38.72	11.87	49.02	51.34	1.54%
ResNet-34	71.33	90.06	21.80	30.43	48.63	22.12	40.90	41.43	1.47%
ResNet-50	74.55	92.01	25.56	57.83	80.08	26.04	16.72	11.13	1.88%
ResNet-101	75.99	92.89	44.55	6.85	12.53	45.82	69.14	80.37	2.85%
ResNet-152	77.01	93.48	60.19	0.17	0.79	61.75	76.84	92.69	2.59%
VGG-16	70.02	89.41	138.36	0.10	0.52	139.58	69.92	88.88	0.88%
VGG-19	70.61	89.92	143.67	0.10	0.46	145.32	70.51	89.46	1.15%
DenseNet-121	71.96	90.70	7.98	0.27	1.08	8.75	71.69	89.62	9.65%
DenseNet-169	73.75	91.54	14.15	0.15	0.70	16.28	73.60	90.84	15.05%
DenseNet-201	74.55	92.16	20.01	0.12	0.58	23.92	74.43	91.58	19.54%
DenseNet-161	75.27	92.52	28.68	0.14	0.66	33.00	75.13	91.86	15.06%
ViT-Tiny	43.11	66.40	5.72	2.33	6.97	5.89	40.78	59.43	2.97%
ViT-Small	72.47	91.20	22.05	12.82	26.66	22.75	59.65	69.54	3.17%
ViT-Base	76.08	92.98	86.57	21.67	39.97	89.39	54.41	53.01	3.26%
ViT-Large	83.43	96.95	304.33	73.23	91.51	314.36	10.20	5.44	3.30%

in practice, we adopt a compromised scheme that inserts the same (absolute) number of confusion neurons to each layer/block so that the shallow layers are assigned with larger proportions and hence the entire model is better protected.

4.1.2 The Amount of Confusion Neurons

We first investigate the amount of confusion neurons inserted to the pre-trained models. The goal of this part is to reduce the ratio of the number of confusion neurons over the original network size. We inherit the conclusion from the previous part which assigns the same number of confusion neurons to each layer, and test the relationship between reduced recognition accuracy (in terms of image classification, object detection, and instance segmentation) and the confusion proportion.

Experimental results are shown in Figures 4 and 5, respectively. It is obvious that the destroy of confusion neurons is highly correlated to the proportion compared with the original model. From the image classification part, we are satisfied with the fact that adding a small portion of confusion neurons is sufficient to protect the entire model, for quantitative numbers, please refer to the next subsection. In addition, we find that the extra computational overhead, in terms of either GPU memory or inference time, is approximately proportional to the confusion proportion. In particular, when the confusion proportion is merely 5%, the recognition accuracy on all tasks drops by more than half, meanwhile the increased cost is negligible, implying that the proposed method is of practical value in real-world applications.

4.2 Protection Ability over Different Tasks

To verify the effectiveness of deep encryption, we evaluate the proposed method on several main tasks of deep learning, including image classification, object detection, instance segmentation and text classification. For all experiments in this part, the confusion proportion is fixed at 5%. Due to the differences in implementation details as well as that the number of added channels must be

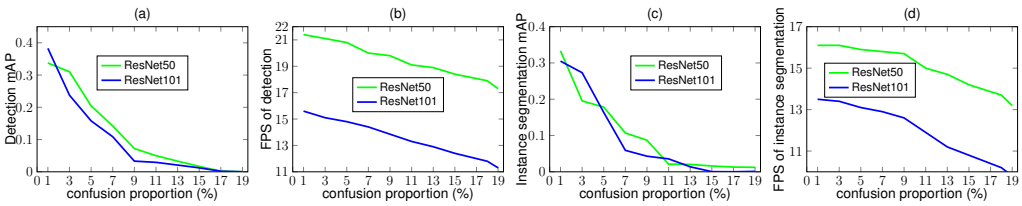

Figure 5: The impact of adding different amounts of confusion neurons to the ResNet series. The tests are made on MS-COCO object detection and instance segmentation tasks.

Table 2: The AP and FPS of object detection on the original and encrypted ResNet50/101 models, evaluated on the MS-COCO validation set.

Backbone	Original							Encrypted						
	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
ResNet-50	37.40	58.10	40.40	21.20	41.00	48.10	21.40	17.60	28.40	18.70	9.30	19.30	23.60	19.10
ResNet-101	39.40	60.10	43.10	22.40	43.70	51.10	15.60	13.50	22.00	14.30	7.50	15.40	17.50	13.30

Table 3: The AP and FPS of instance segmentation on the original and encrypted ResNet50/101 models, evaluated on the MS-COCO validation set.

Backbone	Original							Encrypted						
	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
ResNet-50	34.70	55.70	37.20	15.80	36.90	51.10	16.10	7.20	12.10	7.40	3.30	8.10	10.40	15.00
ResNet-101	36.10	57.50	38.60	16.60	39.20	52.80	13.50	2.80	5.40	2.70	3.00	4.00	1.50	11.90

integer, the actual proportion at each layer and the entire network can be slightly different. The experimental results on each task are shown below.

Image Classification. We present the image classification results on the ILSVRC2012 classification dataset (Russakovsky et al., 2015), which consists of 1,000 classes. We conduct experiments on a variety of network architectures, and the results are shown in Table 1. We can see that the performance of the encrypted networks decrease sharply, which demonstrates the effectiveness of deep encryption. The sensitivity of different network architectures to confusion neurons is different, because of different implementation details. For instance, the encryption of ResNet (He et al., 2016) is conducted by adding convolutional kernels into each block, while the confusion convlutional kernels of VGG (Simonyan & Zisserman, 2014) is added layer by layer. Therefore, the VGG network is obviously more sensitive to the addition of confusion neurons. More implementation details can be found in the source code, which we will release soon.

Object Detection and Instance Segmentation. We present the results of object detection and instance segmentation on the challenging MS COCO dataset (Lin et al., 2014). We adopt Faster-RCNN (Ren et al., 2015) and Mask-RCNN (He et al., 2017) as the basic model for object detection and instance segmentation, respectively. Due to the results in Tables 2 and 3, deep encryption also gains good performance on object detection and instance segmentation tasks. The performance of the original model has decreased sharply with only an additional 5% of confusion neurons, which shows the superiority of our method.

Chinese Text Classification. In addition to computer vision tasks, deep encryption can also be used on natural language processing tasks. Here, we evaluate the deep encryption on the BERT (Devlin et al., 2019) model pretrained with Chinese text. We present the results on a Chinese news text classification dataset THUCNews (Li & Sun, 2007). This dataset consists of 10 different categories, each of which has 1000 pieces of data. The confusion addition strategy for BERT is to expand the fully connected hidden layer and the width of each attention head in each Transformer cell. As shown in Table 4, effective encryption can be achieved with only a few addition model parameters.

Through the above experimental verification, we validate the effectiveness of deep encryption on multiple datasets. One can observe that for different architectures or different tasks, the performance of protection is different. Thus, the confusion proportion can be set basing on specific task and model, which will help the algorithm to achieve better tradeoff between and protection.

Table 4: Chinese text classification results for ten categories on the THUCNews test set. Three metrics are reported for the original and encrypted BERT models, which are precision, recall, and F1-score, respectively. Only 4.21% additional parameters is sufficient for encryption.

	Metrics(%)	finance	realty	stocks	education	science	society	politics	sports	game	entertainment	Params(M)
Original	precision	93.43	96.81	91.27	97.28	91.07	91.50	91.83	99.09	97.16	94.43	**1**02.28
	recall	92.50	94.20	88.90	96.70	91.80	95.80	93.30	98.00	95.70	96.70	
	F1-score	92.96	95.49	90.07	96.99	91.43	93.60	92.56	98.54	96.42	95.55	
Encrpted	precision	0.00	9.86	10.53	0.00	0.00	9.74	0.00	18.06	0.00	5.44	**1**07.20
	recall	0.00	0.70	1.20	0.00	0.00	52.30	0.00	68.30	0.00	3.50	
	F1-score	0.00	1.31	2.15	0.00	0.00	16.43	0.00	28.57	0.00	4.26	

Table 5: The finetuning results on CUB-100-2011 and FGVC-Aircraft test datasets. Here, 'samp', 'init', and 'mixed' represent different strategies of weight assignment. With the increase of confusion proportion, the performance of fine-tuned encrypted model decreases to a certain extent. The resistance of different kinds of confusion weight to fine-tuning is significantly different.

Dataset	Strategy	Accuracy	Confusion proportion					
			0	1%	2%	3%	4%	5%
CUB-200-2011	samp	Top-1 (%)	75.48	74.82	74.61	74.51	74.39	73.49
		Top-5 (%)	93.79	93.74	93.42	93.39	93.38	92.65
	init	Top-1 (%)	75.48	13.17	19.69	19.12	13.57	19.33
		Top-5 (%)	93.79	36.71	49.85	47.34	36.68	46.05
	mixed	Top-1 (%)	75.48	72.94	64.55	44.63	27.11	30.17
		Top-5 (%)	93.79	92.82	89.80	74.89	57.78	69.12
FGVC-Aircraft	samp	Top-1 (%)	74.44	73.49	73.40	73.33	73.10	72.89
		Top-5 (%)	95.11	93.70	93.65	93.60	93.44	93.21
	init	Top-1 (%)	74.44	27.57	16.62	24.00	26.88	24.09
		Top-5 (%)	95.11	63.94	47.11	60.88	63.34	61.96
	mixed	Top-1 (%)	74.44	51.79	34.56	37.14	37.71	15.26
		Top-5 (%)	95.11	85.69	71.65	75.01	74.77	39.78

4.3 Defending Decryption Based on Fine-Tuning

We evaluate the behavior of the encrypted models when they are attacked via being fine-tuned on a small dataset and with fewer computations. We inherit the ResNet152 model pre-trained on ImageNet, and test its performance on two fine-grained visual categorization tasks, where the datasets are CUB-200-2011 (Wah et al., 2011) and FGVC-Aircraft (Maji et al., 2013), both of which are widely used in the community. For the sake of simplicity, we only adopt image-level labels during training and testing. The network is fine-tuned for 15 epochs with a cosine annealing schedule, where the initial learning rate starts with 0.1 and decays till 0.0001. The optimizer is SGD with a momentum of 0.9 and a weight decay of 0.0001.

Experimental results using different confusion proportions are shown in Table 5, in which both the 'samp' and 'init' weight assignment strategies are tested. Interestingly, with the 'samp' strategy, the confusion neurons share the same distribution with the original neurons and thus are much easier to be fine-tuned, while the 'init' strategy is much more challenging for the relatively short fine-tuning procedure. However, recall that the 'init' strategy makes the confusion neurons easier to be detected – this raises another tradeoff between perturbation and camouflage. To compromise both factors, we develop the 'mixed' strategy that each neuron has 50% probability to choose either 'samp' or 'init'.

We emphasize that in real-world applications, the pre-trained models are even more difficult to be attacked by fine-tuning due to two reasons. First, the pre-trained models of value to the service provider are often very large, *e.g.*, GPT-3 (Brown et al., 2020) that contains 175B parameters, meanwhile the fine-tuning procedure of these huge models is very tricky and time-consuming. Second, once the customer adopts the fine-tuning attack, it implies that the efficiency of the pre-trained models is permanently downgraded due to the extra computational overheads. Therefore, we do not expect the fine-tuning attack to be a major threaten to the proposed encryption approach, nevertheless, we believe that stronger decryption methods may be developed in the future.

5 Conclusion

In this paper, we formulate the problem of protecting pre-trained models from being maliciously copied and presents a simple baseline that involves adding confusion neurons to the deep networks. Our approach works on a set of popular networks across vision and language applications. To the best of our knowledge, this is the first work on the protection of parameters and architectures of pre-trained models, and our research set a baseline for the community. We hope to draw the attention of the community to this important area where very few foundations have been established yet.

REFERENCES

Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security Symposium*, 2018.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Yi Cai, Xiaoming Chen, Lu Tian, Yu Wang, and Huazhong Yang. Enabling secure in-memory neural network computing by sparse fast gradient encryption. *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek B Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Oliveira Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Lixin Fan, KamWoh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *CoRR*, abs/1909.07830, 2019.

Laurent Gomez, Alberto Ibarrondo, José Márquez, and Patrick Duverger. Intellectual property protection for distributed neural networks - towards confidentiality of data, model, and inference. In *ICETE*, 2018.

Laurent Gomez, Marcus Wilhelm, José Márquez, and Patrick Duverger. Security for distributed deep neural networks towards data confidentiality & intellectual property protection. In *ICETE*, 2019.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

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*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jingyang Li and Maosong Sun. Scalable term selection for text categorization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 774–782, 2007.

Ning Lin, Xiaoming Chen, Hang Lu, and Xiaowei Li. Chaotic weights: A novel approach to protect intellectual property of deep neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40:1327–1339, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

April Pyone Maung Maung and Hitoshi Kiya. Training dnn model with secret key for model protection. *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, pp. 818–821, 2020.

Yuki Nagai, Yusuke Uchida, Shigeyuki Sakazawa, and Shin'ichi Satoh. Digital watermarking for deep neural networks. *International Journal of Multimedia Information Retrieval*, 7:3–16, 2018.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28: 91–99, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Mingfu Xue, Zhiyu Wu, Jian Wang, Yushu Zhang, and Weiqiang Liu. Advparams: An active dnn intellectual property protection technique via adversarial perturbation based parameter encryption. *ArXiv*, abs/2105.13697, 2021.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. pp. 159–172, 05 2018. doi: 10.1145/3196494.3196550.

Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. *CoRR*, abs/2002.11088, 2020.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

A Appendix

You may include other additional sections here.