Non-transferable Pruning

Ruyi Ding[1], Lili Su[1], Aidong Adam Ding[1], and Yunsi Fei[1]

{ding.ruy, l.su, a.ding, y.fei}@northeastern.edu
Northeastern University, Boston, MA 02115, USA

Abstract. Pretrained Deep Neural Networks (DNNs), developed from extensive datasets to integrate multifaceted knowledge, are increasingly recognized as valuable intellectual property (IP). To safeguard these models against IP infringement, strategies for ownership verification and usage authorization have emerged. Unlike most existing IP protection strategies that concentrate on restricting direct access to the model, our study addresses an extended DNN IP issue: applicability authorization, aiming to prevent the misuse of learned knowledge, particularly in unauthorized transfer learning scenarios. We propose **Non-Transferable Pruning** (NTP), a novel IP protection method that leverages model pruning to control a pretrained DNN's transferability to unauthorized data domains. Selective pruning can deliberately diminish a model's suitability on unauthorized domains, even with full fine-tuning. Specifically, our framework employs the alternating direction method of multipliers (ADMM) for optimizing both the model sparsity and an innovative non-transferable learning loss, augmented with fisher space discriminative regularization, to constrain the model's generalizability to the target dataset. We also propose a novel effective metric to measure the model non-transferability: Area Under the Sample-wise Learning Curve (SLC-AUC). This metric facilitates consideration of full fine-tuning across various sample sizes. Experimental results demonstrate that NTP significantly surpasses the state-of-the-art non-transferable learning methods, with an average SLC-AUC at -0.54 across diverse pairs of source and target domains, indicating that models trained with NTP do not suit for transfer learning to unauthorized target domains. The efficacy of NTP is validated in both supervised and self-supervised learning contexts, confirming its applicability in real-world scenarios. Git Repo

1 Introduction

Rapid advancements in deep learning, characterized by an increase in the model size, complexity, and capability, have made the training of deep learning models more time-consuming and data-intensive, yielding the pre-trained model a valuable intellectual property (IP) [24]. Protecting machine learning IP is therefore vitally important and yet challenging. Most existing IP protection strategies are reactive in nature, focusing on mitigating the damage afterwards rather than preventing breaches outright. These strategies detect unauthorized use of models by employing techniques such as watermarking [35] and fingerprinting [2], accompanied by legal procedures to penalize IP infringement.

Recently, in response to the limitations of passive protection methods, a new type of IP protection approach, known as Non-transferable Learning (NTL), is proposed, serving as a proactive measure to safeguard more general model IP [51, 52]. Transfer learning has been employed to leverage the knowledge learned in pre-trained models towards new applications with effective fine-tuning. The learned knowledge, manifested in the pre-trained model structure and parameters, facilitates efficient fine-tuning, necessitating less amount of training data for the new task and bolstering the model performance. However, application of transfer learning also presents a threat to the model vendors–unauthorized tasks may be hostile, against the law, or simply cause economic disadvantages. In response, NTL aims to restrict the misuse of a pretrained model, by controlling the inherent model transferability, thereby directly limiting its performance on unauthorized tasks. An example of this threat is depicted in Fig. 1, showcasing how a model designed for benign purposes like face detection can be repurposed through transfer learning for a malicious task - Deepfake [54].

Current NTL approaches [51,52] predominantly address scenarios where malicious users (attackers) have limited access to model parameters, preventing them from fully fine-tuning the victim model. This paper, however, explores a more general and stronger attack scenario: the victim model is fully exposed, such as those deployed on edge devices as shown in Fig. 1. Such distinction is critical, as it means that attackers can directly access the pre-trained model and further manipulate it. With the growing trend of embedding machine learning services locally on edge devices, models are either directly available to attackers or can be easily extracted [29, 53, 65]. In these instances, attackers, equipped with unlimited access to model parameters, can potentially redirect the entire model towards unauthorized datasets. Moreover, Non-transferable Learning [52] and Model Barrier [51] primarily use a specific objective function and gradient-based optimization. Yet, we observe that these methods lack robustness to fully fine-tuning, as models may maintain target domain performance with sufficient training data and carefully selected hyperparameters. Going beyond mere parameter optimization, preventing malicious transfer necessitates more substantial modifications to the DNN to further limit its capacity for the target domain. Therefore, we leverage model pruning, which involves strategically zeroing out specific parameters to effectively constrain the model's transferability [17].

We propose **Non-transferable Pruning (NTP)**, a novel model IP protection mechanism designed to nullify adversaries' transfer learning attempts. Our approach leverages a specialized pruning technique, employing the Alternating Direction Method of Multipliers (ADMM), aimed at selectively diminishing the model's transferability to the targeted domain while preserving its efficacy within the source domain. We also propose a novel metric, the Area Under the Sample-wise Learning Curve (SLC-AUC), to assess model transferability. This metric extends beyond the previous model non-transferability measurement in [51,52] using only the accuracy degradation in the target domain. SLC-AUC evaluates the volume of fine-tuning data required by an attacker to redirect the model for the target domain, and compares the fine-tuning performance of the com-

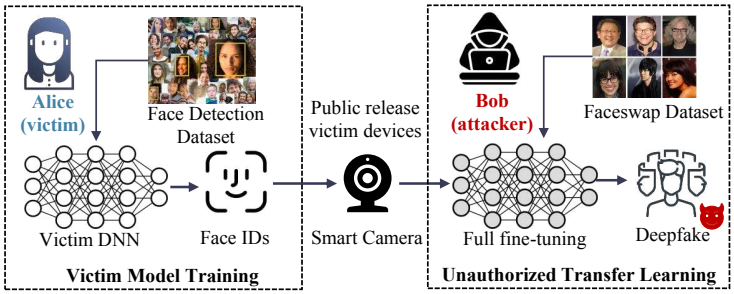

Fig. 1: Adversarial Scenario: model vendor (Alice) built a DNN for face recognition (i.e., Face IDs) and distributed the smart camera (edge device) with this model onboard. The malicious user (Bob) bought the device and obtained full access to the victim model. He modified the model with transfer learning by fine-tuning with a target dataset to redirect the model for the malicious task (e.g., deepfake [54]).

promised model against the one trained from scratch. The new metric is more sound and provides a thorough evaluation of how pre-learned knowledge from the victim model helps fine-tune effective models for the target domain.

In summary, our contributions of this work are three-fold:

- We propose the first ADMM-based model pruning strategy that focuses on limiting the model non-transferability, so as to protect the intellectual property of the pretrained DNN model. We also propose a Fisher Space Regularizer which penalizes class discrepancy on the target domain during NTP to enhance the method's robustness under a fully fine-tuning attack scenario.
- We propose a new evaluation metric, Area Under Sample-wise Learning Curve (SLC-AUC), to measure model non-transferability, considering the variation of model performance with different training data volume from the target domain. It overcomes the bias of prior metrics, such as accuracy degradation, which do not address the different model transferability when the attacker has a different amount of target samples.
- We evaluate our NTP on a vast range of datasets and models. Compared to the SOTA NTL methods [51, 52]. The NTP shows more transferability reduction (the SLC-AUC scores are -0.54 on average). We also evaluate NTP on large-scale datasets (subsets of ImageNet) and self-supervised models (SimCLR [7], MoCo [8, 23]), demonstrating the wide applicability of NTP.

2 Related Works

2.1 Model IP Protection and Non-transferable Learning

Model IP protection aims to protect well-trained DNN models, with typically two types of techniques: ownership verification and usage authorization. Ownership verification is a passive defense against IP infringement, showing evidence of the illegal model usage by an attacker. Commonly used ownership verification

methods include DNN watermarking [21,35,48,55] and model fingerprinting [2,4,62]. However, watermark removal attack [22,27,42] can evade such protections. In contrast, usage authorization aims to restrict the user access of the model, such as the prior works that lock the neural network with encryption [1,3] where only the authorized user with specific keys can unlock the model. Yet, this method has its drawbacks: authorized users with the keys can fully access and potentially misuse the model, presenting another form of IP infringement.

Non-transferable Learning extends the definition of model usage–not only the direct use of the model should be authorized, but also further usage of the model should receive some restriction, known as applicability authorization [51,52]. Specifically, given a target domain on which the model is not allowed to apply, NTL aims to restrict the model transferability towards it, denoted as $\mathcal{L}_{NTL} = \mathcal{L}_{\mathcal{S}} + \mathcal{R}_{\mathcal{T}}$, where $\mathcal{L}_{\mathcal{S}}$ ensures model efficiency in the source domain and $\mathcal{R}_{\mathcal{T}}$ signifies the NTL penalty on unauthorized target domains. Wang et al. [52] use a combination of Kullback-Leibler divergence and Maximum Mean Discrepancy for this penalty; Compact Un-Transferable Isolation (CUTI) [51] enhances the non-transferability by differentiating between shared and private features of the source and target domains. However, existing studies primarily assess non-transferability with accuracy degradation through direct inference on the target domain, neglecting scenarios where the model undergoes fine-tuning. Our research addresses this gap by examining an attack scenario wherein a malicious user can fully fine-tune the victim model. We introduce a novel metric for evaluating transferability across various data scales and employ DNN model pruning in NTL, yielding superior results compared to existing approaches.

2.2 Weight Pruning

Weight pruning [25,26,64], as a deep learning model compression technique, has been used for deploying DNN on resource-constrained platforms, exploiting weight sparsity to trim down neuron connections without notable accuracy degradation. Mainstream pruning methods can be broadly categorized into two types: structural pruning [12,15,38,60] and unstructural pruning [13,34,45]. Structural pruning focuses on removing neurons or channels, while unstructured pruning zeros out specifically chosen weights. To adapt model pruning for DNN non-transferable learning, we find unstructural pruning is suitable and succeed in preserving the model performance in the source domain. Model pruning tasks can be formalized as a dual-optimization problem considering both performance and model sparsity, and solved with the alternating direction method of multipliers (ADMM), which has been used in model adversarial pruning [20,31,57,58], where the pruning scheme aims to achieve the DNN adversarial robustness and model sparsity at the same time. Model pruning is also used in DNN watermarking [56,63], and the pattern of pruning can embody owner's fingerprints [62]. In this work, we leverage ADMM-based pruning methods for applicability authorization, which achieves model non-transferability via weight pruning and updating, resulting in target domain transferability reduction of a compact model, which shows strong robustness against model fine-tuning.

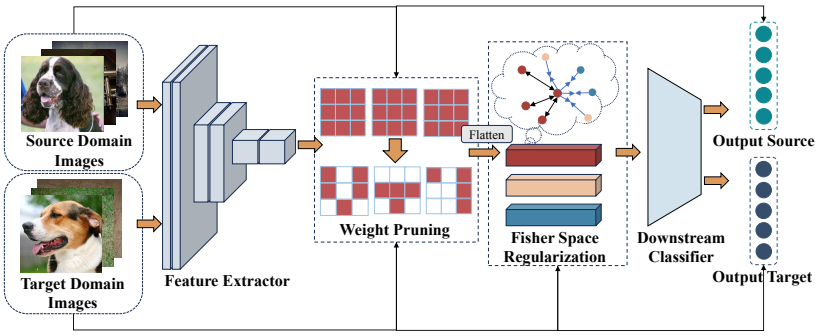

Fig. 2: An Overview of Non-transferable Pruning Procedure

3 Threat Model

Adversarial Goals: We have a victim model designed for a specific task, on a *source domain*. The attacker aims to use this victim model to create a new model for a malicious task, on a *target domain*. The attack hinges on leveraging the transferable knowledge from the source domain within the victim model to efficiently develop a model for the target domain.
Attacker Profile: We assume the attacker possesses complete control of the pre-trained DNN, knowing the model's structure and parameters in full detail. The attacker also has access to a subset of the unauthorized target domain dataset, and can fully fine-tune the model for the target domain with sufficient computational resources. Potential adversaries could be end users of DNN models deployed on edge devices, malicious administrators within cloud-based environments, or external attackers capable of model extraction.
Defender Capability: Our defense strategy is from the perspective of the victim model provider, which proactively safeguards the model in insecure environments. The primary goal is two-fold: in the source domain, the model is legitimately used for the original task with good performance; in the target domain, the pretrained model does not offer any transferable knowledge to yield a fine-tuned model that outperforms the model developed from scratch (without pretrained model). Given that transfer learning typically requires adjustments to the linear classifiers due to discrepancies in label space sizes, NTP concentrates on modifying the feature extractor in the pre-trained models, which is integral to determining how the model interprets and processes input data.

4 Model Non-transferable Pruning

In this section, we illustrate the proposed novel DNN applicability authorization method, Non-Transferable Pruning (NTP). Fig. 2 depicts an overview of the procedure. NTP is informed by data from both the source and the target domain, to selectively prune model parameters that are crucial only to the target domain, thereby diminishing the model's transferability to it. We use the

ADMM-based method for weight pruning. Furthermore, we implement a new regularization technique, based on Fisher discriminant analysis in the feature space, to further reduce the transferability even after fine-tuning. This technique adjusts the target domain feature space by reducing inter-class distances and increasing intra-class distances, which effectively blurs decision boundaries.

4.1 Model Non-transferability Metric

Previous non-transferable learning research primarily assesses non-transferability by measuring the accuracy degradation in target domains [51, 52]. Their evaluations are typically confined to fixed transfer learning settings, especially the sample size for fine-tuning, which may not adequately capture the complexity and variability inherent in real-world malicious transfer learning scenarios, especially those where attackers have unrestricted access to both the model and target dataset [66]. To address this gap, our work broadens the evaluation scope by considering various training data proportions, offering a more comprehensive and dynamic assessment of model non-transferability. We propose the Area Under the Sample-wise Learning Curve (SLC-AUC) as a new metric for non-transferrability. This metric evaluates the performance of a pretrained model transferred to the target domain across diverse scales of samples [49]. On an SLC plot, the X-axis (logarithmic scale) is the size of the target sub-dataset for fine-tuning or learning, and the Y-axis is the testing accuracy of the model. There are two curves in the plot, one for the fine-tuned model (transferred one, in red) and the other for the model without any prior knowledge (learned from the sub-dataset from scratch, i.e., with random initial parameters, in blue).

Definition 1 (SLC-AUC metric). *For any given pretrained model, its SLC-AUC is defined as the difference (the shaded area) between the two SLCs (one for the transferred model and the other for the model trained from scratch).*

A positive SLC-AUC value indicates effectiveness of transfer learning; while a negative SLC-AUC value means non-transferrability. Fig. 3 shows that remarkably, initiating training with a pretrained model can yield superior performance for different pair of source and target domains with a ResNet18 model, particularly when dealing with similar datasets. This aligns with the intuition behind transfer learning, where users can leverage knowledge encoded in a trained model to guide the model fine-tuning for a new task despite limited samples. However, when an attacker has access to abundant data, he can often obtain a model, based solely on the target data without relying on external knowledge, as effective as the one transferred from the pre-trained model. Intuitively, the higher SLC-AUC value, the larger transferability from the source domain to the target domain. Fig. 3 evaluates the model transferability for three different pairs. From SYN (synthetic digits dataset) to MNIST-M (digits with background) has the largest SLC-AUC, indicating that a model pre-trained on SYN can generalize well to MNIST-M with only a few target samples. On the contrary, the model trained on STL (image recognition dataset) provides no apparent advantage in fine-tuning for the MNIST-M dataset, bearing the smallest SLC-AUC.

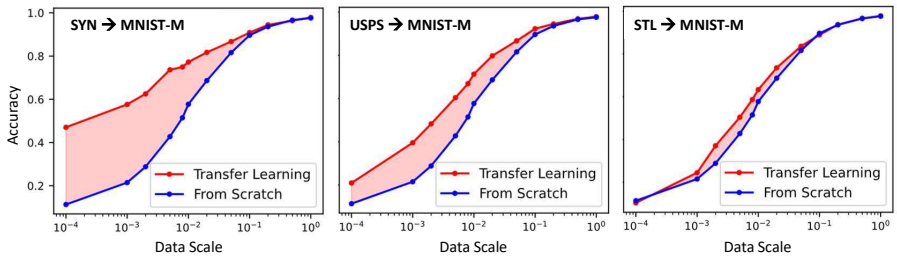

Fig. 3: Sample-wise Learning Curve and Area Under Curve: The target domain is MNIST-M [19] and the pretrained model is trained with network architecture ResNet18 [24] on datasets SYN [18], USPS [30], and STL [10], respectively.

Our goal is to design a model pruning procedure with non-transferable objective so that the pre-trained model has a small or even negative SLC-AUC for selected target domains. In other words, the performance of fine-tuned models on the target dataset are comparable to, or lower than, a direct-trained model.

4.2 Exploring DNN Model Sparsity for Transferability

DNNs are often over-parametric. Model pruning aims to identify a compact subnetwork by eliminating redundant parameters for efficiency and performance. Recent studies find that the compact structure of a pruned model might significantly impair its transferability [5,6,16,39,64]. Intuitively, this may be because a compact model only contains the information that is most relevant to the source domain. There are recent efforts to enhance the transferability of pruned models [17,36] using adversarial examples or task-specific pruning. In this paper, we try to answer an intriguing opposite question: *can model pruning be leveraged to identify a compact DNN model that is effective in the source domain but less effective, or even adversarial, when transferred to the target domain?*

We analyze the impact of model sparsity on its transferability through a toy example, and the results are shown in Fig. 4. We pretrain a ResNet-18 model on the USPS dataset (the source domain), apply one-shot magnitude pruning [5] to it with different levels of sparsity, and measure the pruned models' transferability to different target domains. Our findings reveal the relationship between model sparsity and transferability. Specifically, models with high levels of sparsity exhibit lower SLC-AUC scores, indicating a diminished capacity for transferring due to a constrained parameter set. Conversely, models with mild pruning ratios sometimes experience enhanced transferability due to reduced overfitting on the target domain [37]. This observation from Fig. 4 confirms our conjecture that a heavily-pruned network's capacity for transferring to unauthorized target domains can indeed be reduced. The naive one-shot pruning on weights, although straightforward, leads to severe accuracy degradation on the source domain. In the following sections, we delve into designing an advanced pruning scheme to balance the non-transferability and the performance on the source domain.

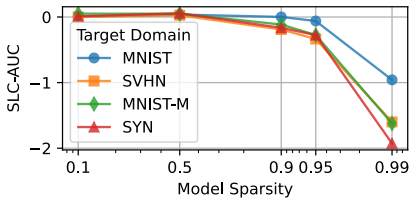

Fig. 4: Sparsity VS SLC-AUC

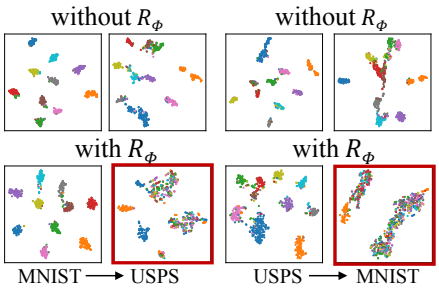

Fig. 5: Feature spaces visualization

4.3 Formulating Non-transferable Pruning Objective

We consider a pretrained neural network, $f_{\mathbf{W}}$, designed for the source domain $\mathcal{S} = (x_{\mathcal{S}}, y_{\mathcal{S}})$. We aim to reduce the model transferability to a target domain $\mathcal{T} = (x_{\mathcal{T}}, y_{\mathcal{T}})$. Without loss of generality, the network can be viewed as two parts: a feature extractor $\Phi_{\mathbf{W_1}}$ and a task classifier $\Omega_{\mathbf{W_2}}$, where W_1 and W_2 denote the feature extractor and classifier's trainable parameters, respectively. Following the design flow of Wang et al. [52], we formulate the non-transferable learning objective function in Eq. (1):

$$\mathcal{L}_{NTP}(\boldsymbol{W}) \triangleq \mathcal{L}_{\mathcal{S}}(\Omega_{W_2}, \Phi_{W_1}, \mathcal{S}) + \mathcal{R}_{\mathcal{T}}(\Omega_{W_2}, \Phi_{W_1}, \mathcal{T}) + \mathcal{R}_{\Phi_{\mathcal{T}}}(\Phi_{W_1}, \mathcal{T}), \tag{1}$$

where $\mathcal{L}_{\mathcal{S}}$ is the cross-entropy loss of the source domain, $\mathcal{R}_{\mathcal{T}}$ is a regularization term that penalizes the performance on the task domain, and $\mathcal{R}_{\Phi_{\mathcal{T}}}$ is our newly introduced fisher space regularization term used to collapse features on target domain. A common choice of $\mathcal{R}_{\mathcal{T}}$ is $\mathcal{R}_{\mathcal{T}} = -\min(\beta, \alpha\mathcal{L}_{\mathcal{T}})$, where α and β are hyperparameters introduced to ensure the stability of the loss [51,52]. We notice that only the first two terms in Eq. (1) are not sufficient for ensuring non-transferability, as large target losses can arise from either incorrect class mapping or indistinct features, with the former still offering useful information to an attacker through fine-tuning. Hence, to enhance non-transferability, we introduce $\mathcal{R}_{\Phi_{\mathcal{T}}}$ for additional regularization in the model's feature space.

Fisher Space Regularization: Fisher Discriminant Analysis (FDA), denoted by $\mathcal{R}_{\Phi_{\mathcal{T}}}$, captures class separability; the more separable the classes the stronger transferability [46]. It is formally defined as

$$\mathcal{R}_{\Phi_{\mathcal{T}}} \triangleq \gamma \frac{\sum_{c\in\mathcal{C}} ||\overline{z}_c - \frac{1}{|\mathcal{C}|}\sum_{c'\in\mathcal{C}} \overline{z}_{c'}||_2}{\sum_{c\in\mathcal{C}}\sum_{(x_i,y_i)\in\mathcal{T}} ||z_i - \overline{z}_c||_2 \mathbf{1}_{\{y_i=c\}}}, \tag{2}$$

where $\mathcal{C}$ is the label set of the target domain, $z_i = \Phi_{W_1}(x_i)$ is the extracted feature of x_i, $\overline{z}_c = \frac{1}{N_c}\sum_{(x_i,y_i)\in\mathcal{T}} z_i \mathbf{1}_{\{y_i=c\}}$ is the averaged features of all the samples whose label are class c[1], and γ is a regularization coefficient. Intuitively,

[1] Here, $\mathbf{1}_{\{y_i=c\}}$ is an indicator function whose value is 1 if $y_i = c$ and 0 otherwise.

Algorithm 1 Non-transferable Pruning

Inputs: Initial DNN weights $\boldsymbol{W}^{(0)}$, sparsity regularization parameter λ,
penalty coefficient ρ, desired sparsity S, source domain $\mathcal{S}$, target domain $\mathcal{T}$
Initialization: $\mathbf{Z}^{(0)} = \mathbf{W}^{(0)}$, $\mathbf{U}^{(0)} = \mathbf{0}$, t=0
while $\text{card}(\mathbf{W}^{(t)})/|\mathbf{W}^{(t)}| < 1 - S$ **do**
/*$\mathbf{W}$-update (update model weight)*/
$\mathbf{W}^{(t+1)} = \arg\min_{\mathbf{W}} \left(\mathcal{L}_{NTP}(\mathbf{W}) + \frac{\rho}{2} \|\mathbf{W} - \mathbf{Z}^{(t)} + \mathbf{U}^{(t)}\|_2^2 \right)$
/*$\mathbf{Z}$-update (update auxiliary variable)*/
$\mathbf{Z}^{(t+1)} = \arg\min_{\mathbf{Z}} \left(\lambda \|\mathbf{Z}\|_1 + \frac{\rho}{2} \|\mathbf{W}^{(t+1)} - \mathbf{Z} + \mathbf{U}^{(t)}\|_2^2 \right)$
/*$\mathbf{U}$-update (update dual variable)*/
$\mathbf{U}^{(t+1)} = \mathbf{U}^{(t)} + \mathbf{W}^{(t+1)} - \mathbf{Z}^{(t+1)}$
$t = t + 1$
end while
Output: $\mathbf{W}$

$\mathcal{R}_{\Phi_\mathcal{T}}$ quantifies how well the extracted features of the target domain dataset are clustered. The larger the difference of means between classes and the smaller the feature variance within each class, the better the cluster structures. Minimizing $\mathcal{R}_{\Phi_\mathcal{T}}$ optimizes the feature space in a contrasting way, where we try to collapse features of data from different classes.

Next, we verify this conjecture both numerically and theoretically. We compare the feature space of learning with/without the fisher space regularization in Fig. 5. We visualize the feature spaces with t-SNE [40] between MNIST and USPS datasets. It demonstrates that $\mathcal{R}_{\Phi_\mathcal{T}}$ blurs class clusters in the target domain (in red box) yet maintains their discrepancy in the source domain. For simplicity, we present our theorem assuming $|\mathcal{C}| = 2$, i.e., a binary classification problem, with the extension to many more classes being done analogously.

Theorem 1. *Let $\mathcal{T}$ be a given target domain. Suppose that its label space $\mathcal{C} = \{0, 1\}$. Let $C_0 = \{(x_i, y_i) : y_i = 0\}$ and $C_1 = \{(x_i, y_i) : y_i = 1\}$. Suppose that $|C_0| = |C_1|$. Then there exists a neural network with a feature extractor Φ_{W_1} that minimizes Eq. (2) and the distributions of the extracted features of classes 0 and 1 are indistinguishable.*

The proof of Theorem 1 can be found in Appendix A. Theorem 1 is stated for the special scenarios where the numbers of samples of class 0 and class 1 are equal. Its analysis can be extended to more general settings yet. Experiments in Sec. 5.2 show the effectiveness of $\mathcal{R}_{\Phi_\mathcal{T}}$ in reducing the feature's class discrepancy to affect the fine-tuning process.

4.4 Integrating Model Pruning with NTP Objective

We use the ADMM-based model pruning strategy [43, 61]. To obtain a NTP pre-trained model, we solve the following constrained optimization problem:

$$\underset{\mathbf{W}}{\text{minimize}} \quad \mathcal{L}_{NTP}(\mathbf{W}), \quad \textbf{subject to} \quad \text{card}(\mathbf{W})/|\mathbf{W}| < 1 - S \tag{3}$$

where card$(\cdot)$ returns the number of nonzero elements and S is the desired model sparsity. In particular, we realize the optimization via ADMM method [43,61]. It reformulates the original problem by introducing an auxiliary variable, denoted as $\mathbf{Z}$, and a dual variable $\mathbf{U}$ as an augmented Lagrangian. The algorithm solves two primal objectives separately. The first subproblem optimizes the weights $\mathbf{W}$ by minimizing the original non-transferable pruning loss function augmented with a penalty term to encourage $\mathbf{W}$ to be close to $\mathbf{Z}$ with coefficient ρ; this unconstrained problem can be solved with stochastic gradient descent. The second subproblem applies sparsity to the auxiliary variable $\mathbf{Z}$ through L_1-norm regularization, which can be solved efficiently by applying soft thresholding. Finally, we update the dual variable $\mathbf{U}$ based on the discrepancy between $\mathbf{W}$ and $\mathbf{Z}$. The overall NTP framework is summarized in Algorithm 1.

5 Experiments

5.1 Experiment Setup

Datasets: Following previous works [51,52], we evaluate our method on popular domain adaption datasets. Specifically, MNIST (MT) [11], USPS (US) [30], SVHN (SN) [41], MNIST-M (MM) [19] and SYN (SD) [18], are commonly used digits dataset containing digits from 0 to 9 from varies scenes; CIFAR-10 [33] and STL-10 [10] are ten-class image classification datasets. In addition, we analyze the inherent functionality of NTP on more complex datasets, namely ImageNette and ImageWoof [28]. ImageNette contains 10 distinct classes from ImageNet [44] and Imagewoof is another subset that has high similarity. For simple datasets, we normalize the input size as $(32, 32, 3)$, and $(224, 224, 3)$ for the ImageNet data.

Models: Following the settings in [51,52], we implement VGG-11 [47] and ResNet-18 [24] as backbones for these datasets. In addition to the source model trained with supervised learning, we also evaluate the self-supervised models, which use SimCLR [7] and MoCo [8,23] with ResNet-50 [24] as backbones.

Baseline Prior Work and Metrics: We compare our NTP with two baseline prior work, NTL [52] and CUTI [51]. Their transferability is measured with SLC-AUC. For self-supervised models, we evaluate the feature space transferability using SOTA metrics: LogME [59] and SFDA [46].

Implementation Details: During NTP, we utilize 10% samples from the source dataset and target dataset to do model pruning by default, which is further evaluated in Appendix E. The ADMM involves an early stop criterion when the ADMM loss converges or reaches the max sparsity (default as 0.99). During the transferability evaluation, all models are fully fine-tuned for 30 epochs with Adam optimizer [32] at a learning rate of 10^{-3} with a step scheduler every 10 epochs by default. Ablation study on the learning rate is done in Tab. 3. Each experiment is repeated for 5 times with different random seeds and the error bars are visualized. The batch size is 256 for model training and fine-tuning. All experiments are conducted on an NVIDIA TITAN RTX with 24 GB of memory.

Table 1: Model non-transferability comparison between our proposed **NTP** with two baselines NTL [52] and CUTI [51] on different Source-Target pairs on VGG-11.

Target \ Source		MT	UP	MM	SN	SD
MT	NTL	-	0.42	0.41	0.35	0.47
	CUTI		0.89	1.07	0.70	0.81
	NTP*		**-0.47**	**-0.07**	**-1.02**	**-0.64**
UP	NTL	0.23	-	0.64	0.83	0.16
	CUTI	2.03		2.12	1.86	1.91
	NTP*	**-0.47**		**-1.27**	**-1.56**	**-0.78**
MM	NTL	1.36	1.32	-	1.37	1.19
	CUTI	1.49	1.15		1.24	1.22
	NTP*	**0.16**	**-0.13**		**0.01**	**-0.06**
SN	NTL	1.47	1.48	1.45	-	1.49
	CUTI	0.72	0.68	1.03		1.53
	NTP*	**-0.61**	**-0.64**	**-0.40**		**0.50**
SD	NTL	1.52	1.56	1.64	1.52	-
	CUTI	0.98	0.99	1.23	1.67	
	NTP*	**-0.34**	**-0.25**	**-0.46**	**0.51**	

Table 2: Source Accuracy Drop

Source	NTL	CUTI	**NTP***
MT	**0.55**% ↓	0.59% ↓	0.73% ↓
UP	0.11% ↑	0.16% ↑	**0.31**% ↑
MM	2.57% ↓	**0.75**% ↑	0.50% ↓
SN	3.50% ↓	**0.17**% ↑	4.10% ↓
SD	1.01% ↓	**0.66**% ↑	0.01% ↑

Table 3: Fine-tuning Schemes

Method	lr	NTL	CUTI	**NTP***
FF	10^{-3}	0.79	0.56	**0.29**
	10^{-4}	0.73	0.95	**0.32**
	10^{-5}	0.60	0.96	**0.21**
LP	10^{-3}	0.79	**0.18**	0.28
	10^{-4}	0.69	0.73	**0.25**
	10^{-5}	0.26	**0.07**	0.13

5.2 Evaluation on Supervised Learning

We compare the proposed NTP with the two baseline approaches. We reproduce NTL[2] and CUTI[3], applying them to VGG-11 across various source-target domain pairs. Tab. 1 shows the non-transferability results, and Tab. 2 reports the average accuracy on the source domain, showing a trade-off between model non-transferability and source-domain performance for all these methods.

Our findings reveal that NTP surpasses the baseline methods by exhibiting a negative SLC-AUC, indicating that it effectively diminishes the model accuracy on the target domain to a level even below that achievable by training the model from scratch. Though both baseline methods report a low target domain performance initially, they exhibit less robustness to malicious fine-tuning. As for the impact on the performance within the source domain, CUTI [51] manifests the least impact. Nonetheless, its effectiveness in restricting transferability is the most limited, particularly when the target domain involves simpler datasets (e.g., UP, MM). Our NTP method, while causing a comparable reduction in source domain performance to NTL [52], significantly enhances non-transferability across various domains. It is important to note that the level of non-transferability also depends on the characteristics of the datasets involved. For instance, the SN and SD datasets, both consisting of RGB digits, share considerable similarities. This intrinsic similarity between datasets implies that, despite employing a non-transferable learning scheme, completely preventing the leakage of pre-trained information to the target domain remains challenging. More results about the NTP performance on ResNet-18 can be found in Appendix B.

[2] https://github.com/conditionWang/NTL
[3] https://github.com/LyWang12/CUTI-Domain

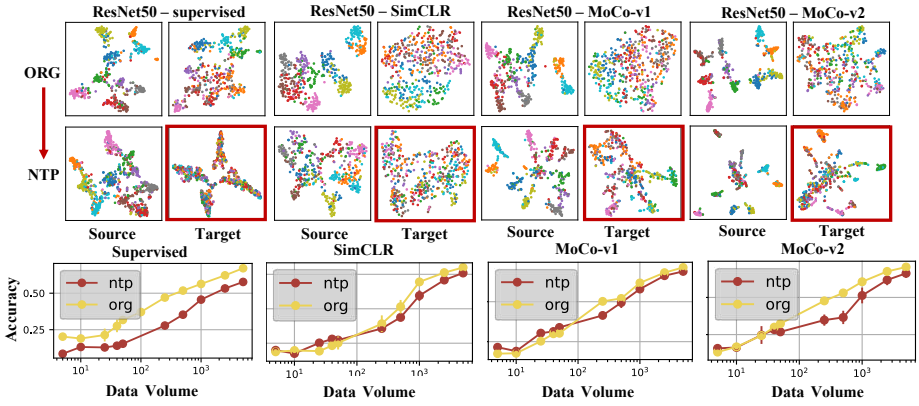

Fig. 6: Top: Self-supervised models' feature space with/without NTP using t-SNE [40]; **Bottom:** line charts for target accuracy after fine-tuning with various data volumes.

.

In Tab. 3, we further evaluate all methods under two transfer learning strategies with three learning rates (10^{-3}, 10^{-4}, 10^{-5}), and report the target domain accuracy (**the lower the better**). We use the MT dataset as the source and conduct a transfer task to UP on VGG-11 with 10% of the data, which is the common setting of transfer learning. Full Fine-tuning (FF), indicating all parameters in the model are trainable, which usually has better performance but is computationally intensive. Linear Probing (LP), is the most commonly used transfer learning method when the early layers of the model are frozen but the final linear layers are fine-tuned during transfer learning. Our proposed NTP is stable for different learning rates, but NTL and CUTI might achieve high transferability under specific learning rate settings (with high accuracy). Moreover, although CUTI has lower accuracy during linear probing, it has a bad performance when the entire model can be fine-tuned. The proposed NTP shows high non-transferability for both fine-tuning methods.

5.3 Evaluation on Self-supervised Learning

Self-supervised Learning (SSL) is declared to offer enhanced generalizability and transfer learning flexibility due to its ability to autonomously extract representations from input data [14]. We evaluate NTP across three SSL models—SimCLR [7], MoCo [23], and MoCo-v2 [8]—employing ResNet-50 as the backbone with pretrained checkpoints[4] for transferring tasks from CIFAR-10 to STL-10 datasets. Fig. 6 visualizes the feature spaces via t-SNE for both the source (CIFAR-10) and target (STL-10) datasets, alongside accuracy upon full fine-tuning with varying sample sizes. As a comparison, we also do the same experiment on a supervised learned model. NTP can keep the features discrepancy on the source domain, suggesting its ability to maintain source domain

[4] https://github.com/linusericsson/ssl-transfer

Table 4: Transferability measurements on self-supervised pretrained model

Metric \ Model	Domain	Supervised		SimCLR [7]		MoCo-v1 [23]		MoCo-v2 [9]	
		ORG	NTP	ORG	NTP	ORG	NTP	ORG	NTP
LogMe [59]	CF-10	3.52	7.79	7.68	7.45	7.11	7.44	8.56	8.22
	STL-10	3.16	**0.93**	4.07	**0.48**	2.24	**-2.52**	6.19	**-1.11**
SFDA [46]	CF-10	0.96	0.99	0.99	1.0	0.86	0.80	0.99	0.69
	STL-10	0.97	**0.81**	0.99	**0.97**	0.72	**0.36**	0.85	**0.41**

performance. Notably, NTP demonstrates superior performance on the SL models relative to the SSL models in non-transferability. Particularly, MoCo-v1 and MoCo-v2, still exhibit discernible class clusters for the target domain, indicating some degree of transferability. In contrast, the features of SL models on the target domain have blurred boundaries and it is more challenging to classify.

To quantitatively assess the efficacy of NTP in SSL models, we apply two SOTA transferability metrics: the Logarithm of Maximum Evidence (LogME) [59]; Self-challenging Fisher Discriminative Analysis (SFDA) [46] in Tab. 4. Higher scores indicate greater transferability. We demonstrate the effectiveness of NTP–it successfully reduces the transferability scores to STL-10 (target domain) based on these metrics. Our hypothesis attributes this phenomenon to the intrinsic regularization effect of the SSL training scheme, particularly through contrastive loss, which promotes greater class separation within the feature space.

5.4 NTP Effectiveness in Complex Tasks

We use two complex ImageNet subsets, ImageNette and ImageWoof, to illustrate how NTP curves the model non-transferability. ImageNette comprises ten classes that are relatively straightforward to classify, whereas ImageWoof includes ten classes of various dog breeds, which are notably more challenging to classify. We trained ResNet-18 models on these datasets, achieving 83.3% accuracy on ImageNette and 64.8% on ImageWoof, respectively. Subsequently, we applied NTP with a desired sparsity of 0.8 to prune and update the model on ImageNette with restricted transferability to ImageWoof, and vice versa. When the source domain is ImageNette, the model target accuracy is reduced to 43.7% with a source domain accuracy of 81.7%; For ImageWoof as the source, NTP lowered ImageNette performance to 68.7%, with ImageWoof accuracy at 58.6%. To interpret the models' decision-making processes, we utilized integrated gradient visualization [50], illustrating the critical features influencing the models' predictions, as shown in Fig. 7. Our analysis indicates that the ImageNette-trained model primarily identifies coarse-grained features, such as object outlines. NTP emphasizes on this tendency, causing the model to favor these broader features and thus diminish its adaptability to ImageWoof, where recognizing finer details, like distinguishing between dog breeds, is essential. On the other hand, the ImageWoof-trained model concentrates on fine-grained details, such as dogs' eyes and fur. Applying NTP heightens this detailed focus, leading to the inclusion of extra noise and a subsequent drop in its ImageNette performance.

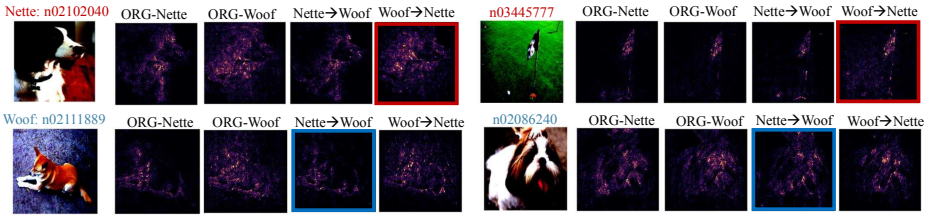

Fig. 7: Interpret NTP using Integrated Gradients [50]. Images are evaluated on original/NTP-pruned models between **ImageNette** and **ImageWoof**. n02102040, n03445777 are from ImageNette; n02111889, n02086240 are from ImageWoof.

5.5 Ablation Studies

First, we adjusted NTP's sparsity level, observing (in Appendix C) that higher sparsity correlates with reduced transfer learning performance in the target domain. Next, we vary the number of epochs in the weight-updating step in the ADMM algorithm from one to twenty. We find that even with small weight updating epochs in ADMM, NTP can still achieve low SLC-AUC scores (Appendix D). We then experimented with varying the training percentage data for ADMM, finding NTP to be more effective with more training samples. This suggests that when NTP utilizes a small portion of training dataset, it may overly focus on specific samples, leading to a biased pruning process (Appendix E).

6 Conclusions

In this study, we delve into DNN non-transferable learning, an emerging and crucial aspect in safeguarding the IP of DNN models. We introduce Non-transferable Pruning, which incorporates an ADMM-based pruning approach and an optimization using Fisher discriminative regularization. Furthermore, our research underlines the importance of evaluating model transferability across various fine-tuning settings, especially for potential attackers who may possess full access to the model. As the value of DNN intellectual property continues to increase, especially in the case of larger and more complex models, NTP emphasizes the need for proactive protection in model applicability authorization for model vendors.
Limitation. In implementing ADMM for NTP, we appreciate its advantages in fast convergence and scalability for large-scale models. Yet, this approach faces notable challenges. First, it shows sensitivity to hyperparameters. The integration of non-transferable loss and Fisher space regularization necessitates careful calibration of hyperparameters. This is particularly true for setting the maximum loss for $\mathcal{R}_{\mathcal{T}}$, which is critical to maintaining loss stability. Similarly, the regularization weights for $\mathcal{R}_{\Phi_{\mathcal{T}}}$ demand careful adjustment to strike a balance among different loss terms. Secondly, there's a delicate balance between model sparsity and preserving non-transferability. Setting the right sparsity level in NTP is critical to maintain both the model's effectiveness in the source domain and its non-transferability for complex transfer learning tasks.

Acknowledgements

This work was supported in part by National Science Foundation under grants SaTC-1929300 and CNS-2212010. L. Su is supported in part by an NSF CAREER award CCF-2340482.

References

1. Alam, M., Saha, S., Mukhopadhyay, D., Kundu, S.: Deep-lock: Secure authorization for deep neural networks. arXiv preprint arXiv:2008.05966 (2020)
2. Cao, X., Jia, J., Gong, N.Z.: Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In: Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security. pp. 14–25 (2021)
3. Chakraborty, A., Mondai, A., Srivastava, A.: Hardware-assisted intellectual property protection of deep learning models. In: 2020 57th ACM/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE (2020)
4. Chen, H., Rouhani, B.D., Fu, C., Zhao, J., Koushanfar, F.: Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In: Proceedings of the 2019 on International Conference on Multimedia Retrieval. pp. 105–113 (2019)
5. Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Carbin, M., Wang, Z.: The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16306–16316 (2021)
6. Chen, T., Frankle, J., Chang, S., Liu, S., Zhang, Y., Wang, Z., Carbin, M.: The lottery ticket hypothesis for pre-trained bert networks. Advances in neural information processing systems **33**, 15834–15846 (2020)
7. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
8. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
9. Chen*, X., Xie*, S., He, K.: An empirical study of training self-supervised vision transformers. arXiv preprint arXiv:2104.02057 (2021)
10. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 215–223. JMLR Workshop and Conference Proceedings (2011)
11. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine **29**(6), 141–142 (2012)
12. Ding, X., Ding, G., Guo, Y., Han, J.: Centripetal sgd for pruning very deep convolutional networks with complicated structure. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4943–4953 (2019)
13. Dong, X., Chen, S., Pan, S.: Learning to prune deep neural networks via layer-wise optimal brain surgeon. Advances in neural information processing systems **30** (2017)
14. Ericsson, L., Gouk, H., Hospedales, T.M.: How well do self-supervised models transfer? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5414–5423 (2021)

15. Filters'Importance, D.: Pruning filters for efficient convnets
16. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018)
17. Fu, Y., Yuan, Y., Wu, S., Yuan, J., Lin, Y.C.: Robust tickets can transfer better: Drawing more transferable subnetworks in transfer learning. In: 2023 60th ACM/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE (2023)
18. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International conference on machine learning. pp. 1180–1189. PMLR (2015)
19. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., Lempitsky, V.: Domain-adversarial training of neural networks. Journal of machine learning research **17**(59), 1–35 (2016)
20. Gui, S., Wang, H., Yang, H., Yu, C., Wang, Z., Liu, J.: Model compression with adversarial robustness: A unified optimization framework. Advances in Neural Information Processing Systems **32** (2019)
21. Guo, J., Potkonjak, M.: Watermarking deep neural networks for embedded systems. In: 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp. 1–8. IEEE (2018)
22. Guo, S., Zhang, T., Qiu, H., Zeng, Y., Xiang, T., Liu, Y.: Fine-tuning is not enough: A simple yet effective watermark removal attack for dnn models. arXiv preprint arXiv:2009.08697 (2020)
23. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. arXiv preprint arXiv:1911.05722 (2019)
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
25. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4340–4349 (2019)
26. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision. pp. 1389–1397 (2017)
27. Hitaj, D., Mancini, L.V.: Have you stolen my model? evasion attacks against deep neural network watermarking techniques. arXiv preprint arXiv:1809.00615 (2018)
28. Howard, J., Gugger, S.: Fastai: A layered api for deep learning. Information **11**(2), 108 (2020)
29. Hu, X., Liang, L., Li, S., Deng, L., Zuo, P., Ji, Y., Xie, X., Ding, Y., Liu, C., Sherwood, T., et al.: Deepsniffer: A dnn model extraction framework based on learning architectural hints. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 385–399 (2020)
30. Hull, J.J.: A database for handwritten text recognition research. IEEE Transactions on pattern analysis and machine intelligence **16**(5), 550–554 (1994)
31. Jian, T., Wang, Z., Wang, Y., Dy, J., Ioannidis, S.: Pruning adversarially robust neural networks without adversarial examples. In: 2022 IEEE International Conference on Data Mining (ICDM). pp. 993–998. IEEE (2022)
32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
33. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
34. Lee, N., Ajanthan, T., Gould, S., Torr, P.H.: A signal propagation perspective for pruning neural networks at initialization. arXiv preprint arXiv:1906.06307 (2019)

35. Li, Z., Hu, C., Zhang, Y., Guo, S.: How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of dnn. In: Proceedings of the 35th Annual Computer Security Applications Conference. pp. 126–137 (2019)
36. Liu, B., Cai, Y., Guo, Y., Chen, X.: Transtailor: Pruning the pre-trained model for improved transfer learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 8627–8634 (2021)
37. Liu, J., Wang, Y., Qiao, Y.: Sparse deep transfer learning for convolutional neural network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
38. Liu, L., Zhang, S., Kuang, Z., Zhou, A., Xue, J.H., Wang, X., Chen, Y., Yang, W., Liao, Q., Zhang, W.: Group fisher pruning for practical network compression. In: International Conference on Machine Learning. pp. 7021–7032. PMLR (2021)
39. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270 (2018)
40. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
41. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
42. Pan, X., Zhang, M., Yan, Y., Wang, Y., Yang, M.: Cracking white-box dnn watermarks via invariant neuron transforms. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 1783–1794 (2023)
43. Ren, A., Zhang, T., Ye, S., Li, J., Xu, W., Qian, X., Lin, X., Wang, Y.: Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers. In: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 925–938 (2019)
44. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015)
45. Sanh, V., Wolf, T., Rush, A.: Movement pruning: Adaptive sparsity by fine-tuning. Advances in Neural Information Processing Systems **33**, 20378–20389 (2020)
46. Shao, W., Zhao, X., Ge, Y., Zhang, Z., Yang, L., Wang, X., Shan, Y., Luo, P.: Not all models are equal: predicting model transferability in a self-challenging fisher space. In: European Conference on Computer Vision. pp. 286–302. Springer (2022)
47. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
48. Song, C., Ristenpart, T., Shmatikov, V.: Machine learning models that remember too much. In: Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security. pp. 587–601 (2017)
49. Sun, C., Shrivastava, A., Singh, S., Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era. In: Proceedings of the IEEE international conference on computer vision. pp. 843–852 (2017)
50. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International conference on machine learning. pp. 3319–3328. PMLR (2017)
51. Wang, L., Wang, M., Zhang, D., Fu, H.: Model barrier: A compact un-transferable isolation domain for model intellectual property protection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20475–20484 (2023)

52. Wang, L., Xu, S., Xu, R., Wang, X., Zhu, Q.: Non-transferable learning: A new approach for model ownership verification and applicability authorization. arXiv preprint arXiv:2106.06916 (2021)
53. Wei, J., Zhang, Y., Zhou, Z., Li, Z., Al Faruque, M.A.: Leaky dnn: Stealing deep-learning model secret with gpu context-switching side-channel. In: 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 125–137. IEEE (2020)
54. Westerlund, M.: The emergence of deepfake technology: A review. Technology innovation management review **9**(11) (2019)
55. Wu, H., Liu, G., Yao, Y., Zhang, X.: Watermarking neural networks with watermarked images. IEEE Transactions on Circuits and Systems for Video Technology **31**(7), 2591–2601 (2020)
56. Xie, C., Yi, P., Zhang, B., Zou, F.: Deepmark: Embedding watermarks into deep neural network using pruning. In: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI). pp. 169–175. IEEE (2021)
57. Ye, S., Xu, K., Liu, S., Cheng, H., Lambrechts, J.H., Zhang, H., Zhou, A., Ma, K., Wang, Y., Lin, X.: Adversarial robustness vs. model compression, or both? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 111–120 (2019)
58. Ye, S., Zhang, T., Zhang, K., Li, J., Xu, K., Yang, Y., Yu, F., Tang, J., Fardad, M., Liu, S., et al.: Progressive weight pruning of deep neural networks using admm. arXiv preprint arXiv:1810.07378 (2018)
59. You, K., Liu, Y., Wang, J., Long, M.: Logme: Practical assessment of pre-trained models for transfer learning. In: International Conference on Machine Learning. pp. 12133–12143. PMLR (2021)
60. You, Z., Yan, K., Ye, J., Ma, M., Wang, P.: Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. Advances in neural information processing systems **32** (2019)
61. Zhang, T., Ye, S., Zhang, K., Tang, J., Wen, W., Fardad, M., Wang, Y.: A systematic dnn weight pruning framework using alternating direction method of multipliers. In: Proceedings of the European conference on computer vision (ECCV). pp. 184–199 (2018)
62. Zhao, J., Hu, Q., Liu, G., Ma, X., Chen, F., Hassan, M.M.: Afa: Adversarial fingerprinting authentication for deep neural networks. Computer Communications **150**, 488–497 (2020)
63. Zhao, X., Yao, Y., Wu, H., Zhang, X.: Structural watermarking to deep neural networks via network channel pruning. In: 2021 IEEE International Workshop on Information Forensics and Security (WIFS). pp. 1–6. IEEE (2021)
64. Zhu, M., Gupta, S.: To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878 (2017)
65. Zhu, Y., Cheng, Y., Zhou, H., Lu, Y.: Hermes attack: Steal {DNN} models with lossless inference accuracy. In: 30th USENIX Security Symposium (USENIX Security 21) (2021)
66. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proceedings of the IEEE **109**(1), 43–76 (2020)