Non-Transferable Learning: A New Approach for Model Verification and Authorization

Lixu Wang
Northwestern University
lixuwang2025@u.northwestern.edu

Shichao Xu
Northwestern University
shichaoxu2023@u.northwestern.edu

Ruiqi Xu
Northwestern University
jerryxu2023@u.northwestern.edu

Xiao Wang
Northwestern University
wangxiao@northwestern.edu

Qi Zhu
Northwestern University
qzhu@northwestern.edu

Abstract

As Artificial Intelligence as a Service gains popularity, protecting well-trained models as intellectual property is becoming increasingly important. Generally speaking, there are two common protection methods: ownership verification and usage authorization. In this paper, we propose Non-Transferable Learning (NTL), a novel approach that captures the exclusive data representation in the learned model and restricts the model generalization ability to certain domains. This approach provides effective solutions to both model verification and authorization. For ownership verification, watermarking techniques are commonly used but are often vulnerable to sophisticated watermark removal methods. Our NTL-based model verification approach instead provides robust resistance to state-of-the-art watermark removal methods, as shown in extensive experiments for four of such methods over the digits, CIFAR10 & STL10, and VisDA datasets. For usage authorization, prior solutions focus on authorizing specific users to use the model, but authorized users can still apply the model to any data without restriction. Our NTL-based authorization approach instead provides data-centric usage protection by significantly degrading the performance of usage on unauthorized data. Its effectiveness is also shown through experiments on a variety of datasets.

1 Introduction

Deep Learning (DL) technology is the backbone of Artificial Intelligence as a Service (AIaaS) [1, 2], which covers a wide range of applications including music compositions [3, 4], autonomous driving [5, 6] and smart building management [7, 8]. However, a good model can be expensive to obtain: it requires dedicated network architecture design [9], access to a large amount of high-quality data [10], lengthy training periods on professional computational resources [11], and expert tuning [12]. Therefore, well-trained DL models are important intellectual property (IP) for both model owners and trainers. Generally speaking, there are two aspects in protecting an IP in AIaaS: who owns this model, and who can use this model? These two questions are captured by two techniques: ownership verification and usage authorization. For ownership verification, prior works proposed,

*These authors contributed equally to this work.

Preprint. Under review.
e.g., embedding watermarks into the least significant bits of the network parameters [13, 14], learning specific behaviors for pre-defined triggers [14, 15], or extracting fingerprints from a given neural network [16, 17]. However, these approaches are vulnerable to state-of-art watermark removal techniques based on model fine-tuning or retraining [18, 19]. As for model usage authorization, most works are built on encrypting neural network parameters with a secret key [20, 21] to make sure that models can only be used by the users with the corresponding key. In practice, the authorized users could share the key with others or use the model on any data without restriction. To address these issues, in this paper, we propose Non-Transferable Learning (NTL), a new approach that can robustly verify the model ownership and authorize model usage on specific data.

Intuitively, NTL goes against the current research direction of improving the generalizability of models across various domains, e.g., domain generalization [22–24] and domain adaptation [25, 26]. On the contrary, it tries to make the generalization bound of DL models more explicit and narrower. Specifically, NTL optimizes the model to learn domain-dependent features, thereby making the model exclusive to certain domains. We consider two domains in NTL: one is the source domain in which we want the models to perform well, and the other is an auxiliary domain in which we aim to degrade the model performance. In this case, if the model trained with NTL is applied in a target domain similar to the auxiliary domain, the performance will also be bad. We provide two versions of NTL: Target-Specified NTL and Source-Only NTL, as shown in Figure 1. Target-Specified NTL assumes that the source and target domains are both known, in which case we regard the target domain as the auxiliary one and enlarge the distance of the representation between the source and the auxiliary/target domain. In Source-Only cases, the target domain is unknown or unavailable, and thus our approach relies solely on the source domain and tries to degrade the performance in all other possible similar domains. Under such circumstances, NTL generates the auxiliary domain from a novel generative adversarial augmentation framework and then increases the distance between the source and auxiliary domains.

Target-Specified NTL can be used as a way to verify the model ownership by triggering misclassification. The reason why previous model watermarks can be easily removed is that the occurrence of the special behavior is completely dependent on the trigger, whose features impose negative effect on the main task. During fine-tuning or retraining, the model memorization of such trigger features may encounter catastrophic forgetting [27]. As for our NTL-based verification, the misclassification behavior is dependent on the overall target-private features that have little correlation with the source-private features for the main task. Therefore, NTL-based verification is resistant to state-of-art watermark removal approaches. For model authorization, we note that the question is more than who is allowed to use the model. In particular, what data is used on the model is equivalently, if not more, important. NTL cares about the value of data, and by giving authorization to certain data rather than particular users or devices, the model unauthorized usage caused by the leakage of secret keys can be prevented. It is also worthwhile to mention that if the model owner is malicious, they can also utilize NTL to restrict the model generalization ability and make the model produce abnormal misclassifications when it is fed with certain data.
Contributions. To sum up, our work makes the following contributions:

• We propose a novel model learning algorithm called Non-Transferable Learning (NTL), which can limit the model generalization ability to a specified domain while degrading the performance in other data domains. We provide the theoretical foundation on how to achieve NTL.

• NTL works effectively both in the presence and absence of target domains, and we denote these two cases as Target-Specified NTL and Source-Only NTL, respectively. When the target domain is known, our NTL expands the representation distance between the source and target domain. When the target domain is unknown or unavailable, we propose a generative augmentation model to produce neighborhood domain data, and then conduct NTL on the source and augmented domain.

• Our work is the first to show that restricting model generalization can be used to protect intellectual property by ownership verification and usage authorization. We carry out extensive experiments on 5 digit recognition domains, CIFAR10 & STL10 and VisDA. For target-specified cases, we demonstrate how to implement NTL for the model ownership verification or the implant of specialized backdoors. Our experiments show that the state-of-art model watermark removal methods are ineffective to NTL-based ownership verification. For source-only NTL, our experiments demonstrate its effectiveness in achieving model usage authorization.

2 Related Work

Domain Generalization (DG) aims to generalize learning models with available source domains to unseen target domains [28]. In this setting, a number of methods are proposed: minimizing domain discrepancy [22, 23], adversarial training [29–31], invariance representation learning [24, 32] and others [33, 34]. Recently, the problem becomes conducting DG with one source domain only. To solve this problem, well-crafted data augmentation approaches [35–38] can be used to expand the input space. The authors of [35] utilize constrained loss and map the gradient updates to source samples, and then create the augmented domains. [36] achieves data augmentation by optimizing the entropy measurement in the information bottleneck [39]. [37] applies generative models to generate source-like images with multiple auto-encoders and spatial transform networks [40]. Unlike DG, in this work, we try to weaken the generalization ability of models by expanding the distance between representations of different domains. Our proposed method can work effectively in the target-available case, and also for single-source cases with a novel adversarial augmentation framework.

Domain adaptation (DA) is another topic of improving the generalization ability of models across domains. However, DA is different from DG: DA can access the target data, while DG has no access to any target sample. From the perspective of methodology, there is no significant difference between DA and DG, e.g., in DA, several studies utilize adversarial training [25, 26], distance-based methods [41–43] and invariance learning [44, 45] to adapt knowledge across domains. Standard DA assumes the source and target domains are both available, but recent DA research argues that the source domain might be unavailable, which is called source-free DA. For instance, [46–48] use a pre-trained source model with frozen weights to pseudo label unlabeled target samples and conduct supervised learning in the target domain. In addition, the authors of [49] utilize the data generated by image re-composition to improve the transferability of source models. Above all, why DA has achieved enormous success is the model generalization ability across similar domains. However, in some cases, source model owners do not want their models to be adapted to other areas without their authorization. To the best of our knowledge, there is no previous work on preventing unauthorized model usage across domains, and this paper is the first to propose a solution.

Deep Learning Intellectual Property Protection. Similar to the application of video and photo watermark [50], there have been a number of watermarking approaches addressing the verification of model ownership: embedding watermarks into the network parameters [13, 14], setting special behaviors on pre-defined triggers [14, 51, 52], and extracting fingerprints from given neural network modules [16, 17]. Specifically, [51, 52] train a neural network on original datasets as well as the watermarked one assigned with a particular label, which makes the model behave abnormally on the watermarked data. [13] and [14] inject watermark into the least significant bits of the model parameters and provide the corresponding decoding methods. In the case of black-box, [16, 17] make use of adversarial examples to extract fingerprints from learned neural networks without accessing network weights. Compared with these approaches, our NTL can achieve model ownership
verification by triggering universal misclassification. Moreover, with extensive experiments, we also demonstrate that state-of-art model watermark removal methods, e.g., FTAL and RTAL [18], EWC and AU [19], are not effective to NTL-based verification.

Model usage authorization is another approach to protect intellectual property of model owners. [20] encrypts every network parameter with a secret key, and only the user who is authorized with this key can utilize the model. In [21], the secret key is generated from some fingerprints of a particular device, and only the people who possesses this device can load and employ the model. However, unauthorized usage will still occur if the authorized user deliberately shares his secret key with others. Our NTL provides a data-centric method to address this issue, which retains the good model performance on the authorized data while degrading the performance in other data domains.

3 Methodology

In this section, we discuss the details of our NTL method. Section 3.1 provides the theoretic foundation of optimizing the model to extract domain-dependent representations. Section 3.2 presents the specific optimization objectives. Finally, we describe a generative model in Section 3.3 that can produce auxiliary domains when the target is unknown or unavailable.

3.1 Theoretical Foundation

In our problem, there is a source domain with labeled samples, $D^S = \{(x^s, y^s) | x^s \in X^S, y^s \in Y^S\}$. $X$ and $Y$ are the input space and label space, respectively, and we take classification as the task. In this case, $Y = \{1, 2, ..., K\}$. In addition, there is an auxiliary domain $D^A = \{(x^a, y^a) | x^a \in X^A, y^a \in Y^A\}$. Here superscripts $s, S$ and $a, A$ denote the source domain and the auxiliary one, respectively. These domains will be fed into a deep neural network, and without the loss of generality, we split the neural network into two parts, one is the feature extractor $\Phi$ on the bottom, and the other is the classifier $c$ with multiple linear layers on the top. Next note that we will compare our following analysis with Information Bottleneck [39] (IB). Let us start with introducing Shannon Mutual Information (SMI).

If we regard the input $x$, the representation $z$ extracted by $\Phi$ and the label $y$ as random variables, the SMI between two random variables is defined as, e.g., $I(x; z) = \mathbb{E}_x[KL(P(z|x)||P(z))]$, where $KL(\cdot)$ represents the Kullback-Leibler (KL) divergence and $P(\cdot)$ is the distribution. In IB theory, considering the effectiveness, privacy and generalization, an optimal representation has three properties [53]: (1) Sufficiency – the representation $z$ is sufficient to be differentiated in terms of labels $y$, i.e., $I(z; y) = I(x; y)$; (2) Minimality – $z$ needs to represent as less information about $x$ as possible, i.e., $\min I(z; x)$; (3) Invariance – $z$ is optimal not to overfit to spurious correlations between labels $y$ and nuisance $n$ embedded in inputs $x$, i.e., $I(z; n) = 0$. IB theory assumes that nuisance $n$ here can be optimized to affect the inputs $x$, but has little information about the task $y$. For instance, in cases of domain generalization, the nuisance $n$ can be regarded as a domain index that indicates training samples come from which domain.

In our problem, different from the objective of IB theory, NTL is forcing the models to extract nuisance-dependent representations which is opposite to the property of invariance. In other words, optimizing to increase $I(z; n)$. Next, we will introduce how to maximize $I(z; n)$ via different efforts. Proposition 3.1 of [53] proposes the relation between the invariance and minimality of a sufficient representation. To prove this proposition, [53] assumes a Markov chain $(1) (y, n) \rightarrow (y, n)_{(2)} \rightarrow x \rightarrow z$, and in representation learning, there is also another information flow $(2) x \rightarrow z \rightarrow (y, n)_{(2)}$. Note that these two chains start from different sources. Chain $(1)$ indicates that inputs $x$ can be viewed as a deterministic function of task $y$ and nuisance $n$. In this case, the information flows from $(y, n)$ to $z$. For chain $(2)$, task $y$ and nuisance $n$ are regarded as the outputs of representation learning models. Strictly speaking, $(y, n)$ of two chains are different, $(y, n)_{(1)}$ is the ground truth, while whereas $(y, n)_{(2)}$ is the prediction result. However, if the models are ideal for predicting $y$ and $n$, without the loss of generality, we can regard $(y, n)$s in these two chains as almost the same item. In this case, we have the following theorem.

Theorem 1 (Bound Gap) Let $n$ be a nuisance for task $y$. Let $z$ be a representation of the input $x$. Suppose that $x$ can be obtained from $y$ and $n$ through a deterministic function of $y$ and $n$. Then for the information flow in the optimal representation learning models (for predicting $y$ and $n$), we have

$$I(z; x) - I(z; y|n) \geq I(z; n) \geq I(x; n, y) - I(z; y|n)$$  

(1)
We start from the case where the source and target domains are both available, and the goal in this case is to maximize the performance in the source domain while degrading the performance in the target domain. In this setting, we can regard the target as the aforementioned auxiliary domain and conduct what we call Target-Specified NTL on them. According to the optimization intuition derived from Theorem 1, we can directly design a naive loss-based method to directly minimize \( I(z; n) \). Specifically, we use KL divergence for the classification task rather than CrossEntropy loss since we analyze NTL based on SMI and IB theory. Then the naive loss is shaped like a minus minus form:

\[
I(z; n) = \mathbb{E}_{z \sim N(0, \sigma^2 I_{d \times d})} \log \left( \frac{1}{2} + \frac{\sigma_0^2}{2\sigma_1^2} I_{d \times d} : e^{-\frac{1}{2\sigma_1^2} (z - \mu_0 - \mu_1)^2} \right)
\]

The detailed proof can be found in the Supplementary Materials (SM). This theorem provides instructions on preparing conditions for maximizing \( I(z; n) \). One direction is increasing the lower bound: \( I(x; n, y) \) is decided by the training data, and there is little we can do; while decreasing \( I(z; y|n) \) is a feasible optimization objective. However, just increasing the lower bound is not enough if the upper bound is low. For increasing the upper bound, unlike the IB theory, here we will not minimize \( I(z; x) \) for the minimality property; \( I(z; y|n) \) is also preferred to be as low as possible here. However, if we assume that \( I(z; n) \) is in a wide gap bounded as Eq. (1), although the optimization of \( I(z; y|n) \) can increase the lower and upper bounds, \( I(z; n) \) might still remain unchanged. To increase \( I(z; n) \), we provide a tighter lower bound of it as follows.

**Theorem 2 (Distance Expansion)** Let variable \( n \) conforms \( P(n) \sim \{p(r) = 0.5, p(1) = 0.5\} \). Suppose that \( z \) is the \( d \)-dimensional representation of the input \( x \). We assume that \( (z|n=0) \) and \( (z|n=1) \) are distributed according to a mixture of 2 standard high-dimensional Gaussians with centers \( \mu_0, \mu_1 \in \mathbb{R}^d \). Each representation \( z \) is drawn from the spherical Gaussian, i.e., \( P(z|n=0) \sim N(\mu_0, \sigma_0^2 I_{d \times d}) \), \( P(z|n=1) \sim N(\mu_1, \sigma_1^2 I_{d \times d}) \), respectively. And we use another variable \( Z \) to replace \( z - \mu_0 \) and \( z - \mu_1 \), and then \( P(Z|n=0) \sim N(0, \sigma_0^2 I_{d \times d}) \), \( P(Z|n=1) \sim N(0, \sigma_1^2 I_{d \times d}) \). In this case, we can bound \( I(z; n) \) as follows:

\[
I(z; n) \geq -0.5 \mathbb{E}_{z \sim N(0, \sigma^2 I_{d \times d})} \log \left( \frac{1}{2} + \frac{\sigma_0^2}{2\sigma_1^2} I_{d \times d} : e^{-\frac{1}{2\sigma_1^2} (z - \mu_0 - \mu_1)^2} \right) \]

Please refer to our SM for the proof of Theorem 2. With these two theorems, we can optimize \( I(z; n) \) and its bound gap simultaneously, and the model will extract nuisance-dependent representations.

### 3.2 Target-Specified NTL with Distance Expansion of Representation

We start from the case where the source and target domains are both available, and the goal in this section is to maximize the performance in the source domain while degrading the performance in the target domain. In this setting, we can regard the target as the aforementioned auxiliary domain and conduct what we call Target-Specified NTL on them. According to the optimization intuition derived from Theorem 1, we can firstly design a naive loss-based method to directly minimize \( I(z; y|n) \). Specifically, we use KL divergence for the classification task rather than CrossEntropy loss since we analyze NTL based on SMI and IB theory. Then the naive loss is shaped like a minus minus operation between losses of the source and auxiliary domain \( L_S, L_A \). Specifically, in the mini-batch training, we select a batch from the source domain and another one from the auxiliary domain in each round, and then conduct subtraction between losses of these two batches as follows:

\[
L_{nbt} = L_S - \max(\beta, \alpha \cdot L_A)
\]

(3)

\( \alpha \) here is the scaling factor for \( L_A \) (\( \alpha = 0.1 \)), and \( \beta \) is an upper bound in case that \( L_A \) gets too large and dominates the overall loss \( (\beta = 1.0) \); please see our SM for more details about \( \alpha \) and \( \beta \). Ideally, we can widen the performance gap between the source and auxiliary domains via optimizing this naive loss. However, this optimization might only make the top layers (classifier) of models more sensitive to domain features and has little effect on the bottom layers (feature extractor). In this case, representations for different domains captured by the feature extractor may still be similar, which conflicts with our initial intention to maximize \( I(z; n) \), and the performance of the target is easy to be improved via fine-tuning or adapting the classifier with a small number of labeled target samples. To maximize \( I(z; n) \), according to Theorem 2, we can enlarge the Euclidean distance between centers of the source and auxiliary domains to increase the lower bound of \( I(z; n) \). Then we compute Gaussian kernel Maximum Mean Discrepancy (MMD) \([54]\), a popular Euclidean-based distance that is compatible with mini-batch training, between the source and auxiliary batch and maximize it in each training round. For safety concern, we also set an upper bound to the MMD distance (though the MMD distance has never reached this boundary during our experiments). Then, the optimization loss of NTL is shaped as follows:

\[
L_{ntl} = L_S - \max(\beta, \alpha \cdot L_A \cdot L_{dis}), \text{ where } L_{dis} = \max(\beta', \alpha' \cdot \text{MMD}(X^S, X^A))
\]

MMD\((X^S, X^A) = \mathbb{E}_{X^S} [e^{-\|\Phi(x^S) - \Phi(a^S)\|^2}] - 2\mathbb{E}_{X^S, X^A} [e^{-\|\Phi(x^S) - \Phi(a^S)\|^2}] + \mathbb{E}_{X^A} [e^{-\|\Phi(a^S) - \Phi(a^A)\|^2}]\)

(4)
Here the superscript $t$ in MMD denotes another different sample in the same domain, and $\alpha'$, $\beta'$ represent the scaling factor and upper bound of $L_{\text{dis}}$ respectively ($\alpha' = 0.1$, $\beta' = 1.0$). $\Phi(\cdot)$ is the feature extractor which outputs the corresponding representations of given inputs. With the optimization of Eq. (4), if we use $n = 0$ and $n = 1$ to denote the source and auxiliary domain respectively, and $n$ conforms $P(n) \sim \{pr(0) = 0.5, pr(1) = 0.5\}$, the term of $L_S$ here can guarantee the sufficiency of classification task in the source domain $I(z; y/n = 0) = I(z; y/n = 0)$, and optimizing $L_S$ decreases $I(z; y/n = 1)$. According to the relation shown in Eq. (1), ideally, $I(z; y/n)$ can be minimized to $0.5(x; y/n = 0)$.

### 3.3 Source-Only NTL with Generative Adversarial Augmentation

In practice, the target domain might be unknown or unavailable, in which case we develop a **Source-Only NTL** approach. For source-only cases, the problem becomes how to find a suitable replacement for the aforementioned auxiliary domain. In this section, we design a novel generative model based augmentation framework, which can generate data samples drawn from the neighborhood distribution of the source domain with different distances and directions.

**GAN Training.** The overall architecture of our augmentation framework is shaped like a generative adversarial network (GAN) that is made up of a generator $G$ and a discriminator $D$. The generator $G$ takes in a random latent code and a label in form of one-hot, and then outputs a data sample. For the discriminator $D$, if we feed $D$ with a sample, it will tell us whether this sample is fake or not and also the predicted label. The adversarial battle between these two parties happens as $G$ tries to generate data as real as possible to fool $D$, while $D$ distinguishes whether the data fed to it are real or not to its best. After enough period of such battle, the generator can produce extremely similar data compared to the given training data. Moreover, some theoretical analysis works [55] of GAN demonstrate that the distribution of the generated data is too close to that of the ground-truth data to distinguish between them. Based on the principle of GAN, we can utilize $G$ to approximate the source domain. However, if we follow the standard GAN training, the trained GAN will not generate samples with deterministic labels. Therefore, we combine the intuitions of CGAN [56] and infoGAN [57] to propose a new training approach for GAN here. The loss of $G$ is similar to that of standard GAN training, but the only difference is that we use MSE loss instead of binary CrossEntropy loss. As for $D$, it consists of three modules: a *feature extractor* and two *classifiers* behind the extractor as two branches. Specifically, a binary classifier predicts whether the data fed is real or not, and a multiple classifier outputs the label. Note that these two classifiers both rely on the representations extracted by the feature extractor. Here, we use subscripts $z$, $b$ and $m$ to denote outputs from the feature extractor, the binary classifier, and the multiple classifier, respectively. For the optimization of $D$, we use MSE loss to evaluate its ability to distinguish real samples from fake ones, and KL divergence to quantify the performance of predicting labels for the real data. Finally, there is an additional training step for enforcing the GAN to generate samples of input labels by optimizing $G$ and $D$ simultaneously. The training losses of $L_G$, $L_D$ and $L_{G,D}$ are written as follows:

$$
L_D = \mathbb{E}_{x \sim X, \ y \sim Y} \left[ \frac{1}{2} \left( ||D_b(x), 1||^2 + ||D_b(G(\text{code}, y)), 0||^2 \right) + KL(D_m(x) \parallel y) \right] \\
L_G = \mathbb{E}_{\text{code, } y} \left[ ||D_b(G(\text{code}, y)), 1||^2 \right], \ L_{G,D} = \mathbb{E}_{\text{code, } y} \left[ KL(D_m(G(\text{code}, y)) \parallel y') \right]
$$

(5)

Here, the latent code of two rows in Eq. (5) is drawn from Gaussian Noise, and $y$ is the labels of the source domain, while $y'$ is determined randomly from 1 to $K$ with the probability of $1/K$.

**Augmentation with Different Distances.** To generate the data of different distances to the source domain, we apply MMD to measure the distance between distributions of the source and the generated data from $G$. However, providing that the MMD distance is optimized to increase with no restriction, the outcome will present to lose the semantic information, i.e., the essential feature for the main task (classification). In order to preserve the semantic information, we use CrossEntropy loss to add a restriction to the optimization objective. With the restriction, we set multiple upper bounds – dis (the list of dis is DIS) for generating augmented data with different distances. The specific objective is written as follows:

$$
L_{\text{aug}} = - \max \left\{ \text{MMD}(D_S(X^S), D_i(G(\text{code}, Y^S))), \text{dis} \right\} + \sum_i y_i \log \frac{1}{D_m(G(\text{code}, y_i))}
$$

(6)
Algorithm 1: Generative Adversarial Data Augmentation for Source-Only NTL

Require: Source domain labeled data $D^S = \{X^S, Y^S\}$; Generator $G$, discriminator $D$; List of augmentation distance $DIS$, the maximum augmentation direction $DIR$; GAN training epochs $e_{GAN}$, augmentation training epochs $e_{AUG}$; Initialize $D' = \{\}$; 

Output: The augmentation labeled data $D' = \{G(code, Y'), Y'\}$ // $Y'$ is randomly determined;

for $i = 1$ to $e_{GAN}$ do
  use $(code, Y^S)$ to optimize $G$ with $L_G$, use $D^S$ and $G(code, Y^S)$ to optimize $D$ with $L_D$;
  use $(code, Y')$ to optimize $G$, $D$ with $L_G, D$;
freeze $D$;
for $dis \in DIS$ do
  for $dir \in DIR$ do
    for $l \in G$ do
      $interval = d(l) / DIR$; // function $d()$ acquires the dimension of inputs;
      freeze $l[0 : dir \times interval]';$ // freeze $dir$ parts of neurons or filters of $l$-th layer in $G$;
    for $i = 1$ to $e_{AUG}$ do
      use $(code, Y^S)$ and $D^S$ to optimize $G$ with $L_{aug}$;
      $D' \cup G(code, Y')$; // use $G$ to generate augmentation data;
  
Augmentation with Different Directions. In this part, we investigate how to generate augmented data in different directions. According to the principle of popular optimization algorithm, e.g., SGD [58], the optimization of MMD follows the direction of gradient, which is the fastest way to approach the objective. In other words, all augmented domains of different distances might follow the same direction, i.e., the direction of gradient. Therefore, in order to augment neighborhood domains with various directions, we need to introduce more restrictions to the optimization process. For the intermediate representations of $G$, we view each filter (neuron) as corresponding to a feature dimension of the representation. At the beginning of directional augmentation, we initialize a copy version of the pre-trained GAN ($G$ and $D$). Then we freeze the $G$ gradually, i.e., if we want to augment the source in $DIR$ directions, we will divide the overall network of $G$ into $DIR$ parts equally, and then for the augmentation of the first direction, the first part will be frozen and not updated during optimization. The second direction will be augmented by freezing the first two parts of network and conducting the optimization. The third corresponds to the first three parts, and so on. The detailed algorithm is shown in Algorithm 1.

4 Experimental Results

4.1 Setting and Datasets

Our code is mainly implemented in PyTorch [59] (provided in $SM$), and all experiments are conducted on a server running Ubuntu 18.04 LTS, equipped with NVIDIA TITAN RTX GPU.

Digits – We use five digit recognition datasets. MNIST [60] (MT) is the most popular digit dataset. USPS [61] (US) consists of digits that are scanned from the envelopes by the U.S. Postal Service. SVHN [62] (SN) contains house number data selected from Google Street View images. Moreover, MNIST-M [63] (MM) is made by combining MNIST with different backgrounds. Finally, SYN-D [64] (SD) is a synthetic dataset, which is generated by combining noisy and complex backgrounds.

CIFAR10 & STL10 – Both CIFAR10 and STL10 [65] are ten-class classification datasets. In order to make these two sets applicable to our problem, we follow the procedure in [66], which removes the non-overlapping classes (‘frog’ and ‘monkey’) and reducing the task to a nine-class setting.

VisDA – The Visual Domain Adaptation Challenge [67] (VisDA) dataset contains a training set (VisDA-T) and a validation set (VisDA-V) of 12 object categories.

For classifying these datasets, we apply VGG-11 [68] for Digits Recognition, VGG-13 [68] for CIFAR10 & STL10, and ResNet-50 [9] for VisDA. All networks are initialized as the pre-trained version of ImageNet [10]. We select 3 random seeds (2021, 2022, 2023) to conduct all experiments three times and present the average performance. The network architectures, training parameters and error bars of experiment results can be found in the $SM$. 

7
4.2 Target-Specified NTL

Effectiveness of NTL in reducing target domain performance: In cases where the target domain is available, we use the target as the auxiliary domain in NTL. We conduct experiments on all three cases. Specifically, for digits recognition and CIFAR10 & STL10, we pick all possible domain pairs to carry out experiments. As for VisDA, we regard the training set as the source domain and the validation set as the target. We include the results of standard supervised learning with KL divergence in the source domain and report the performance change caused by our NTL. Table 1 and Figure 3 show results of digits recognition, CIFAR10 & STL10 and VisDA. Compared with the performance of supervised learning, we can observe that the target performance of all pairs is degraded to nearly 10% with little accuracy reduction in the source domain. The average relative performance drop of all cases is approximately 80%, and the biggest performance drop, from 97.0% to 11.7%, occurs when the source is MM and the target is MT. All results here prove that Target-Specified NTL can effectively degrade the performance in the target without the sacrifice of the source performance.

Table 1: The performance change of digit datasets caused by Target-Specified NTL compared with Supervised Learning. The left of ‘⇒’ is the precision (%) in the target when the model is trained on the source dataset with Supervised Learning. The number on the right of ‘⇒’ is the precision of the model trained with Target-Specified NTL. The last two columns presents the average relative performance drop in the source and target respectively. It shows that NTL can degrade the performance in target domains while retaining the great performance in the source.

| Source/Target | MT | US | SN | MM | SD | Avg drop (Source) | Avg drop (Target) |
|---------------|----|----|----|----|----|------------------|------------------|
| MT            | 98.9⇒97.9 | 86.4⇒14.5 | 33.3⇒13.1 | 57.4⇒09.8 | 35.7⇒08.9 | 0.10% | 78.24% |
| US            | 84.7⇒08.6 | 99.8⇒98.8 | 26.8⇒08.6 | 31.5⇒09.8 | 37.5⇒08.8 | 1.00% | 80.17% |
| SN            | 52.0⇒09.5 | 69.0⇒14.2 | 89.5⇒88.4 | 34.7⇒10.2 | 55.1⇒11.6 | 1.23% | 78.42% |
| MM            | 97.0⇒11.7 | 80.0⇒13.7 | 47.8⇒19.1 | 91.3⇒89.2 | 45.5⇒10.2 | 2.30% | 79.76% |
| SD            | 60.4⇒10.7 | 74.6⇒06.8 | 37.8⇒08.3 | 35.0⇒12.5 | 97.2⇒96.1 | 1.13% | 81.57% |

Table 2: The results of Target-Specified NTL on ownership verification and their performance after applying different watermark removal methods. Compared with Supervised Learning, models trained with Target-Specified NTL behave differently depending on whether the data fed to it contains a trigger patch. Applying 4 state-of-art watermark removal approaches on models of NTL won’t impact the effectiveness of ownership verification.

| Source without patch | Training Methods [Test with/without patch] | Watermark Removal Approaches [Test with/without patch] |
|----------------------|---------------------------------------------|--------------------------------------------------------|
| Supervised Learning  | FTAL | RTAL | EWC | AU |
| NTL                  | 09.9/99.9 | 10.4/98.6 | 10.9/99.2 | 10.8/98.9 |
|                      | 14.3/99.0 | 14.4/98.6 | 14.2/99.1 | 14.2/99.1 |
|                      | 09.9/89.2 | 10.0/88.7 | 10.1/89.1 | 10.0/89.0 |
|                      | 14.0/91.0 | 15.1/89.6 | 12.7/91.1 | 12.6/91.0 |
|                      | 12.6/96.7 | 13.3/95.4 | 12.4/96.9 | 12.8/96.5 |
|                      | 13.5/89.0 | 14.9/88.8 | 15.0/89.4 | 14.8/89.5 |
|                      | 12.4/86.8 | 12.7/86.9 | 13.0/87.5 | 13.9/87.4 |
|                      | 15.6/92.7 | 15.7/92.5 | 16.2/92.7 | 16.3/92.6 |

Figure 2: The data of STL10 attached with/without the patch.

NTL for ownership verification: Here, we test if Target-Specified NTL can achieve ownership verification by triggering misclassification. We use a simple pixel-level mask as the trigger patch,
which is shown in Figure 2 (please refer to SM for more details). The source data attached with the trigger patch is the auxiliary domain, while the original source data is used as the source domain in NTL. We use 4 state-of-art model watermark removal works to test the robustness of NTL-based ownership verification: FTAL [18], RTAL [18], EWC [19] and AU [19]. The detailed experiment settings of these methods are included in SM. The results are shown in Table 2. We can conclude that models trained with NTL behave differently on the data with or without the patch, while supervised learning performs nearly the same (with slight reduction on the data with patch). Furthermore, all these 4 watermark removal methods are ineffective in improving the performance on the patched data, which implies that NTL-based ownership verification is effective and robust.

### 4.3 Source-Only NTL

**Effectiveness of NTL in reducing non-source domain performance:** In Source-Only NTL, we want to reduce the performance in other possible domains while retaining the good performance in the source. For all three datasets, we select one domain as the source and then conduct our generative adversarial augmentation to generate the auxiliary domain. We set a series of discrete dis from 0.1 to 0.5 with the step of 0.1, and for each dis, we generate augmentation data of 4 directions (DIR=4). Table 3 and Figure 3 present results of Source-Only NTL, and Figure 4 is the augmentation data for MNIST (other datasets are included in SM). From the results, we can clearly see that models trained with NTL perform worse on all non-source domains compared with the supervised learning, and the biggest drop from 97.0% to 14.7% also corresponds to MM-MT.

**Table 3: The performance change of digit datasets caused by Source-Only NTL compared with Supervised Learning.** Non-S means non-source domain. The left of ‘⇒’ shows the precision (%) of the model trained on the source dataset with Supervised Learning. The number on the right is the precision after using Source-Only NTL. The last two columns are the average relative precision drop on the source and non-source respectively. It shows that Source-Only NTL can universally degrade the performance in non-source domains without sacrificing the source performance.

| Source/Non-S | MT | US | SN | MM | SD | Avg drop (Source) | Avg drop (Non-S) |
|--------------|----|----|----|----|----|-------------------|------------------|
| MT           | 98.9⇒98.9 | 86.4⇒13.8 | 33.3⇒20.8 | 57.4⇒13.4 | 35.7⇒11.0 | 0.00% | 72.27% |
| US           | 84.7⇒06.7 | 99.8⇒98.9 | 26.8⇒06.0 | 31.5⇒10.1 | 37.5⇒08.6 | 0.90% | 82.60% |
| SN           | 52.0⇒12.3 | 69.0⇒08.9 | 89.5⇒88.0 | 34.7⇒11.3 | 55.1⇒12.7 | 1.68% | 78.56% |
| MM           | 97.0⇒14.7 | 80.0⇒07.8 | 47.8⇒07.8 | 91.3⇒89.1 | 45.5⇒20.1 | 2.41% | 81.35% |
| SD           | 60.4⇒39.2 | 74.6⇒09.5 | 37.8⇒11.4 | 35.0⇒20.7 | 97.2⇒96.9 | 0.31% | 61.12% |

**NTL for model usage authorization:** When applying Source-Only NTL to authorize the model usage, we aim to restrict the usage of the model to the authorized domain only, where all the data is attached with a dedicated patch. For the patch, we need to make sure it will not impact the semantic
information of main task. But note that the unforgeability and uniqueness of the patch are not our main consideration of this work, and we will explore this topic in the future. For simplicity, we use a mask that is similar to that of the aforementioned ownership verification as the authorized patch. We first use the generative adversarial augmentation algorithm to generate neighborhood data of the original source domain. Subsequently, we regard the source data attached with the patch as the source domain in NTL, and use the union of the original source data, the generated neighborhood data attached with and without the patch as the auxiliary domain (the proportion ratio of these three parties is 1:1:1). The experiment results of digits are presented in Table 4 (the results of CIFAR10 & STL10 and VisDA are in SM). From the table, we can see that the model still performs very well in the authorized domain while all other domains has bad performance (with or without the authorized patch). The highest classification accuracy of unauthorized domains is 42.7%, which will discourage users from employing this model.

**Table 4: The performance of authorizing usage of models trained with Source-Only NTL on digits.** NTL-based model authorization can enable the model to perform well only on the authorized data – the source domain attached with the authorized patch.

| Source with Patch | Test with Patch(%) | Test without Patch(%) | Authorized Domain | Other Domains |
|-------------------|--------------------|-----------------------|-------------------|--------------|
|                   | MT | US | SN | MM | SD | MT | US | SN | MM | SD | MT | US | SN | MM | SD | MT | US | SN | MM | SD |
| MT                | 98.5| 14.2| 17.7| 15.1| 07.7| 11.6| 14.3| 16.2| 10.7| 09.3| 98.5| 13.0|
| US                | 08.2| 98.9| 08.7| 09.5| 10.4| 10.3| 06.6| 08.6| 09.2| 10.3| 98.9| 09.1|
| SN                | 10.1| 09.7| 88.9| 12.9| 11.8| 09.7| 08.3| 09.2| 12.3| 11.5| 88.9| 10.6|
| MM                | 42.7| 08.2| 16.1| 90.6| 32.2| 09.5| 08.3| 07.1| 25.6| 22.8| 90.6| 19.2|
| SD                | 09.7| 06.7| 15.7| 19.2| 95.8| 10.2| 06.7| 09.1| 11.5| 33.7| 95.8| 13.6|

5 Conclusion

In this paper, we propose Non-Transferable Learning (NTL): a novel training approach that can restrict the generalization ability of deep learning models to a specific data domain while degrading the performance in other domains. With the help of a generative adversarial augmentation framework, NTL is effective both in the presence and absence of target domains. Extensive experiments on 5 digit recognition datasets, CIFAR10 & STL10 and VisDA demonstrate that the ownership of models trained with NTL can be easily verified, and the verification is resistant to state-of-art watermark removal approaches. In addition, with the training of NTL, model owners can authorize the model usage to a particular domain without worrying about unauthorized usage in other domains.
Supplementary Materials

This appendix contains additional details for the submitted article “Non-Transferable Learning: A New Approach for Model Verification and Authorization”, including mathematical proofs, experimental details and additional results. The implementation code can be found in our attached package, and we will also release it on GitHub later. The code of model watermark removal methods can be found in the link\footnote{https://github.com/sunblaze-ucb/REFIT}. The appendix is organized as follows: Section A introduces the theoretical proofs for Theorem 1 (Section A.1) and Theorem 2 (Section A.2), respectively. Section B provides additional implementation settings, including the network architectures (Section B.1) and hyper parameters (Section B.2). Section C provides additional experimental results, including the augmentation data of other datasets (Section C.1), the model authorization results on CIFAR10 & STL10 and VisDA (Section C.2), the experiments of VisDA on VGG-19 (Section C.3), and the error bars of main experiment results (Section C.4). In Section D, we discuss possible attacks that can be constructed based on our proposed method. Note that NTL used in this appendix is the abbreviation of our method Non-Transferable Learning.

A Theoretical Proofs

In the Section 3.1 of the main paper, we introduce two theorems for our work. Here we provide the proofs for them.

A.1 Bound Gap

**Theorem 1.** Let $n$ be a nuisance for task $y$. Let $z$ be a representation of the input $x$. Suppose that $x$ can be obtained from $y$ and $n$ through a deterministic function of $y$ and $n$. Then for the information flow in the optimal representation learning models (for predicting $y$ and $n$), we have

$$I(z;x) - I(z;y|n) \geq I(z;n) \geq I(x;n,y) - I(z;y|n)$$

(7)

**Proof:** According to Proposition 3.1 in [53], there is a Markov Chain: $(y, n) \rightarrow x \rightarrow z$. The Data Processing Inequality (DPI) for a Markov Chain can ensure the relation that $I(z;y, n) \geq I(z; y)$, and with the chain rule, we have $I(z;y, n) = I(z;n) + I(z;y|n)$. Thus, we can obtain that $I(z;x) \geq I(z;n) + I(z;y|n)$, i.e.,

$$I(z;x) - I(z;y|n) \geq I(z;n)$$

(8)

Next, let us prove the lower bound of $I(z;n)$. In representation learning, we regard the task of models as predicting the label $y$ and the nuisance $n$ when fed with the input $x$. Thus, the corresponding information flow is $x \rightarrow z \rightarrow (y, n)$. Note that $y$ and $n$ here are different from the ground truth label $y$ and nuisance $n$ in Eq. (8). Here, we assume that the learning models are optimal for predicting $y$ and $n$. In this case, we can use the same notations to denote the prediction and the ground truth. According to DPI again, there is the relation in the information flow in the representation learning that $I(z;y, n) \geq I(x;y, n)$. We can replace $I(z;y, n)$ with $I(z;n) + I(z;y|n)$, and obtain $I(z;n) + I(z;y|n) \geq I(x;y, n)$. Thus, both the upper and lower bounds of $I(z;n)$ can be derived as:

$$I(z;x) - I(z;y|n) \geq I(z;n) \geq I(x;n,y) - I(z;y|n)$$

(9)

A.2 Distance Expansion

**Theorem 2.** Let variable $n$ conforms $P(n) \sim \{pr(0) = 0.5, pr(1) = 0.5\}$. Suppose that $z$ is the d-dimensional representation of the input $x$. We assume that $(z|n=0)$ and $(z|n=1)$ are distributed according to a mixture of two standard high-dimensional Gaussians with centers $\mu_0, \mu_1 \in \mathbb{R}^d$. Each representation $z$ is drawn from the spherical Gaussian, i.e., $P(z|n=0) \sim N(\mu_0, \sigma_0^2 I_{d \times d})$, $P(z|n=1) \sim N(\mu_1, \sigma_1^2 I_{d \times d})$, respectively. And we use another variable $Z$ to replace $z=\mu_0$ and $z=\mu_1$, and then $P(Z|n=0) \sim N(0, \sigma_0^2 I_{d \times d})$, $P(Z|n=1) \sim N(0, \sigma_1^2 I_{d \times d})$. In this case, we can
bound $I(z; n)$ as follows:

$$
I(z; n) \geq -0.5\mathbb{E}_{z \sim N(0, \sigma^2 I_{d \times d})} \log \left( 0.5 + \frac{\sigma_0}{2\sigma_1} I_{d \times d} \cdot e^{-\frac{\|z\|_{\mu_0 - \mu_1}^2 + \|z\|^2}{2\sigma_1^2 I_{d \times d}}} \right) - 0.5\mathbb{E}_{z \sim N(0, \sigma^2 I_{d \times d})} \log \left( 0.5 + \frac{\sigma_1}{2\sigma_0} I_{d \times d} \cdot e^{-\frac{\|z\|_{\mu_0 - \mu_1}^2 + \|z\|^2}{2\sigma_0^2 I_{d \times d}}} \right)
$$

(10)

**Proof:** According to the definition of Shannon Mutual Information, we have

$$
I(z; n) = \mathbb{E}_{n \sim P(n)} KL(P(z|n)\|P(z)) = \mathbb{E}_{n \sim P(n)} \mathbb{E}_{z \sim P(z|n)} \log \frac{P(z|n)}{P(z)}
$$

(11)

And because $n$ conforms $P(n) \sim \{pr(0) = 0.5, pr(1) = 0.5\}$, Eq. (11) can re-written as

$$
I(z; n) = 0.5\mathbb{E}_{z \sim P(z|n=0)} \log \frac{P(z|n = 0)}{P(z)} + 0.5\mathbb{E}_{z \sim P(z|n=1)} \log \frac{P(z|n = 1)}{P(z)}
$$

(12)

For a particular Gaussian distribution $x \sim N(\mu, \sigma^2)$, its probability density function is $P(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x-\mu)^2/2\sigma^2}$. Next, we have the marginal distribution of $z$ that $P(z) = \sum_n P(z, n) = \sum_n P(z|n) \cdot P(n) = 0.5P(z|n = 0) + 0.5P(z|n = 1) = \frac{1}{\sqrt{2\pi\sigma_i I_{d \times d}}} e^{-(z-\mu_0)^2/2\sigma_0^2 I_{d \times d}} + \frac{1}{\sqrt{2\pi\sigma_1 I_{d \times d}}} e^{-(z-\mu_1)^2/2\sigma_1^2 I_{d \times d}}$. With the above formulas, we can write $I(z; n)$ as follows:

$$
I(z; n) = -0.5\mathbb{E}_{z \sim P(z|n=0)} \log \left( 0.5 + \frac{\sigma_0}{2\sigma_1} I_{d \times d} \cdot e^{-(z-\mu_0)^2/2\sigma_1^2 I_{d \times d} + (z-\mu_1)^2/2\sigma_0^2 I_{d \times d}} \right) - 0.5\mathbb{E}_{z \sim P(z|n=1)} \log \left( 0.5 + \frac{\sigma_1}{2\sigma_0} I_{d \times d} \cdot e^{-(z-\mu_0)^2/2\sigma_0^2 I_{d \times d} + (z-\mu_1)^2/2\sigma_1^2 I_{d \times d}} \right)
$$

(13)

For the first row of Eq. (13), if we replace $z$ with another variable $Z = z - \mu_0$, it is easy to prove that $Z \sim N(0, \sigma_0^2 I_{d \times d})$. In addition, we conduct the same operation on the second term of Eq. (13). With the replacement of $z$, Eq. (13) becomes:

$$
I(z; n) = -0.5\mathbb{E}_{z \sim N(0, \sigma_0^2 I_{d \times d})} \log \left( 0.5 + \frac{\sigma_0}{2\sigma_1} I_{d \times d} \cdot e^{-(z+\mu_0-\mu_1)^2/2\sigma_1^2 I_{d \times d} + z^2/2\sigma_0^2 I_{d \times d}} \right) - 0.5\mathbb{E}_{z \sim N(0, \sigma_0^2 I_{d \times d})} \log \left( 0.5 + \frac{\sigma_1}{2\sigma_0} I_{d \times d} \cdot e^{-(z+\mu_1-\mu_0)^2/2\sigma_0^2 I_{d \times d} + z^2/2\sigma_1^2 I_{d \times d}} \right)
$$

(14)

In Eq. (14), both $Z$ and $\mu_0 - \mu_1$ are $d$ dimensional vectors, and thus according to triangle inequality, $\|Z \pm (\mu_0 - \mu_1)\| \geq \|Z\| - \|\mu_0 - \mu_1\|$. In this case, we can get the following relation:

$$
e^{-\frac{\|z+\mu_0-\mu_1\|^2}{2\sigma_1^2 I_{d \times d}}} + \frac{z^2}{2\sigma_0^2 I_{d \times d}} \leq e^{-\frac{\|z\|^2}{2\sigma_1^2 I_{d \times d}}} + \frac{\|z\|^2}{2\sigma_0^2 I_{d \times d}}
$$

(15)

With the relation of Eq. (15), Eq. (14) becomes the follows, proving the theorem:

$$
I(z; n) \geq -0.5\mathbb{E}_{z \sim N(0, \sigma_0^2 I_{d \times d})} \log \left( 0.5 + \frac{\sigma_0}{2\sigma_1} I_{d \times d} \cdot e^{-\frac{\|z\|_{\mu_0 - \mu_1}^2 + \|z\|^2}{2\sigma_1^2 I_{d \times d}}} \right) - 0.5\mathbb{E}_{z \sim N(0, \sigma_0^2 I_{d \times d})} \log \left( 0.5 + \frac{\sigma_1}{2\sigma_0} I_{d \times d} \cdot e^{-\frac{\|z\|_{\mu_1 - \mu_0}^2 + \|z\|^2}{2\sigma_0^2 I_{d \times d}}} \right)
$$

(16)

**B** Implementation Settings

**B.1 Network Architecture**

To build the classification models, we use several popular architectures as the bottom feature extractor and attach them with fully-connected layers as the top classifier, which are shown in Table 5.
Specifically, the backbone network of digits is VGG-11, that of CIFAR10 & STL10 is VGG-13, and we use both ResNet50 and VGG-19 for VisDA. The classifiers of all models are the same, i.e., 3 linear layers with ReLU and dropout. As for the GAN in the augmentation framework, the generator $G$ is made up of 4 ConvTranspose blocks and 2 Residual blocks, and the discriminator $D$ consists of a feature extractor with 4 convolution layers, a binary classifier and a multi-class classifier. These two classifiers are composed of sequential fully-connected layers and share the same representations extracted from the front extractor. The detailed architecture is shown in Table 6 and 7.

Table 5: The architecture of classification models. ‘img’ is the dimension of representations extracted from the feature extractor.

| Classifier | Feature Extractor |
|------------|-------------------|
| Linear(256, K). | Backbone Network (VGG-11/VGG-13/VGG-19/ResNet50) [10:]. |
| Linear(256, 256), ReLU, Dropout; Linear(512*img*img, 256), ReLU, Dropout; | |

Table 6: The architecture of the generator $G$. ‘dim’ is the dimension sum of the latent space and input label.

| Out | Tanh(). |
| Conv4 | BatchNorm(3), ReLU. ConvTranspose2d(128, 3, 4, 2, 1); |
| Conv3 | BatchNorm(128), ReLU. ConvTranspose2d(256, 128, 4, 2, 1); |
| ResBlocks | ResidualBlock(256) * 2. |
| Conv2 | BatchNorm(256), ReLU. ConvTranspose2d(512, 256, 4, 2, 1); |
| Conv1 | BatchNorm(512), ReLU. ConvTranspose2d(1024, 512, 4, 2, 1); |
| Input | Linear(dim, 1024). |

Table 7: The architecture of the discriminator $D$.

| Classifiers | Binary Classifier | Multiple Classifier |
|-------------|-------------------|---------------------|
| Linear(128, 1), Linear(256, 128), ReLU, Dropout; Linear(512, 256), ReLU, Dropout; | Linear(128, K), Linear(256, 128), ReLU, Dropout; Linear(512, 256), ReLU, Dropout; |
| Conv2d(256, 512, 3, 2, 1), LeakyReLU, Dropout. Conv2d(128, 256, 3, 2, 1), LeakyReLU, Dropout; Conv2d(64, 128, 3, 2, 1), LeakyReLU, Dropout; Conv2d(3, 64, 3, 2, 1), LeakyReLU, Dropout; | |

B.2 Hyper Parameters

Scaling factors and upper bounds. As introduced in Section 3.2 of the main paper, there are two scaling factors ($\alpha$, $\alpha'$) that control the trade-off between the maximization of $I(z; n)$ and the sufficiency property that $I(z; y | n = 0) = I(z; y | n = 1)$. Here, we conduct experiments using different values ($\alpha = 0.01, 0.05, 0.10, 0.20, 0.50$ and $\alpha' = 0.01, 0.05, 0.10, 0.20, 0.50$), and evaluate their impact to the performance of NTL. For Target-Specified NTL, we select the combination of MNIST→USPS, STL10→CIFAR10 and VisDA-T→VisDA-V as the representatives to carry out experiments. The results are presented in Tables 8 and 9. It is easy to conclude that NTL can work effectively with different scaling factors. As for the upper bounds ($\beta$, $\beta'$), we set them for the sake
of preventing the auxiliary domain loss and the MMD distance from dominating the optimization objective, affecting the convergence of training.

Table 8: The experiments on different values of the scaling factor $\alpha$.

| Scaling Factor $\alpha$ | Source/Target(%) |
|-------------------------|-----------------|
|                         | 0.01 0.05 0.10 0.20 0.50 |
| MT/US                   | 98.6/14.3 98.2/14.2 97.9/14.5 97.7/14.2 97.6/14.3 |
| STL10/CIFAR10           | 88.0/11.1 87.6/12.2 85.4/09.4 82.9/08.7 83.0/12.1 |
| VisDA-T/VisDA-V         | 93.8/08.8 94.9/08.9 94.4/11.3 93.5/08.7 93.2/09.8 |

Table 9: The experiments on different values of the scaling factor $\alpha'$.

| Scaling Factor $\alpha'$ | Source/Target(%) |
|--------------------------|-----------------|
|                         | 0.01 0.05 0.10 0.20 0.50 |
| MT/US                    | 99.3/14.2 98.8/14.3 97.9/14.5 99.2/14.2 99.3/14.2 |
| STL10/CIFAR10            | 87.2/08.8 82.0/10.3 85.4/09.4 85.8/10.8 85.4/11.4 |
| VisDA-T/VisDA-V          | 96.6/11.0 95.3/09.0 94.4/11.3 92.5/08.9 96.4/10.3 |

Training parameters. For the optimization of NTL, we utilize Adam as the optimizer, with learning $\gamma = 0.0001$ and batch size of 32. For all datasets, we randomly select 8,000 samples from their own training sets as the source data, and 1,000 samples from their own testing sets as the test data (if a dataset does not have test set, we select its test data from the training set without overlapping with the chosen 8,000 source samples). And the sample quantities of the source and auxiliary domain are always the same. In the training of adversarial augmentation, the optimizer is also Adam, and we set the learning rate to $\gamma = 0.0002$ with two decay momentums 0.5 and 0.999. The batch size is 64, and the dimension of the latent space fed to the generator is 256.

B.3 Triggering and Authorization Patch

As mentioned in Section 4.2 and 4.3 of our main paper, we attach a patch on the data to utilize NTL for ownership verification and usage authorization. We create the patch in a simple way. Specifically, for the pixel of $i$-th row and $j$-th column in an RGB image, if either $i$ or $j$ is even, then a value of $v$ is added to the R channel of this pixel (the channel value cannot exceed 255). Intuitively, the patch is dependent on pixel values of each image. Thus the changes of feature space brought by the attachment of these patches for various images are not the same. In our experiments, if the content of the image is simple, e.g., MNIST, USPS and SVHN, the $v$ with a small value can shift the feature space sufficiently, but for more complicated images, we have to increase $v$ to enable source images attached with and without the patch differentiable. Specifically, we pick the value as follows: MNIST, USPS, SVHN ($v = 20$); MNIST-M, SYN-D, CIFAR10, STL10 ($v = 80$); VisDA ($v = 100$). As mentioned in the main paper, we will explore the unforgeability and uniqueness of patch generation in the further work.

B.4 Implementation of Watermark Removal Approaches

In the Section 4.2 of the main paper, we implement four model watermark removal approaches to verify the effectiveness of NTL-based ownership verification. Here, we introduce how to implement these approaches. FTAL [18] is an approach that fine-tunes the entire watermarked model using the original training data. To implement it, we use 30% of training set that has been learned by NTL to fine-tune the entire model. When using RTAL [18], the top classifier is randomly initialized before fine-tuning. In our experiments, we load the feature extractor of the model trained with NTL and randomly initialize a classifier to attach on the extractor, and then use 30% of the training set to fine-tune this combined model. As for EWC [19], we use the code of [19] to compute the fisher
information of network parameters and adjust the learning rate of fine-tuning. The data used by EWC is also 30% of the training set. Finally, AU [19] utilizes the watermarked model to pseudo label additional unlabeled samples from other similar domains, and these samples will be used to fine-tune the model together with the original training set. Following this principle, we use 30% of the training set and the same quantity of unlabeled samples from other domains (the proportion ratio between these two parties is 1:1) to fine-tune the model trained with our NTL. We conduct all fine-tuning methods for 200 epochs.

Table 10: The results (%) of authorizing model usage on CIFAR10 & STL10 and VisDA.

| Test          | CIFAR10            | STL10            |
|---------------|--------------------|------------------|
| Source        | with Patch         | without Patch    | with Patch         | without Patch                  |
| CIFAR10       | 85.9 (±0.97)       | 11.5 (±1.29)     | 42.3 (±1.57)       | 13.5 (±2.01)                   |
| STL10         | 20.7 (±1.01)       | 11.7 (±0.98)     | 85.0 (±1.36)       | 12.1 (±1.75)                   |

Table 11: The experiment results (%) of VisDA on VGG-19.

| Test          | VisDA-T            | VisDA-V           |
|---------------|--------------------|-------------------|
| Cases         | with Patch         | without Patch     | with Patch         | without Patch                  |
| Supervised Learning | -                  | 93.4 (±1.83)     | -                  | 45.2 (±2.24)                   |
| Target-Specified NTL | -                  | 92.8 (±1.04)     | -                  | 08.9 (±2.01)                   |
| Source-Only NTL  | -                  | 92.4 (±1.17)     | -                  | 11.3 (±1.00)                   |
| Ownership Verification | 10.1 (±0.90)  | 93.0 (±0.98)     | -                  | -                              |
| Model Authorization | 92.9 (±1.31)  | 12.7 (±2.22)     | 21.4 (±1.01)       | 11.0 (±0.79)                   |

C Additional Experimental Results

C.1 Augmentation Data of Other Datasets

In the main paper, we present the augmentation data of MNIST, and in this section, we include the augmentation data of other datasets as follows: Figure 5 for USPS, Figure 6 for SVHN, Figure 7 for MNIST-M, Figure 8 for SYN-D, Figure 9 for CIFAR10, Figure 10, Figure 11 for VisDA-T.

C.2 Model Usage Authorization on CIFAR10 & STL10 and VisDA

Here we present the experiment of authorizing the model usage on CIFAR10 & STL10 and VisDA, shown in Table 10. According to the results, the model performs well on the data attached with the authorized patch and has bad performance on all other samples.

C.3 Additional Results of VisDA on VGG-19

To demonstrate the effectiveness of NTL on different network architectures, we also carry out experiments of VisDA on VGG-19. All other settings are the same as before, and the results are shown in Table 11. We can easily see that the performance is consistent with the aforementioned other experiments, which shows the wide applicability of NTL.
Table 13: The error range (%) of experiment results for Supervised Learning and Source-Only NTL respectively (the left of '/' is Supervised Learning, and the right is Source-Only NTL).

| Source/Non-S | MT  | US  | SN  | MM  | SD  |
|--------------|-----|-----|-----|-----|-----|
| MT           | ±0.37/±0.13 | ±1.74/±1.23 | ±1.89/±1.22 | ±0.58/±0.21 | ±1.97/±1.04 |
| US           | ±1.27/±1.09  | ±0.10/±0.34  | ±1.21/±0.77  | ±1.64/±0.78  | ±1.17/±0.79  |
| SN           | ±1.23/±1.04  | ±1.67/±1.31  | ±0.97/±0.79  | ±0.86/±1.11  | ±2.17/±1.03  |
| MM           | ±1.28/±0.31  | ±1.78/±1.67  | ±1.66/±1.75  | ±1.78/±0.54  | ±2.10/±1.12  |
| SD           | ±0.91/±1.05  | ±1.61/±1.24  | ±1.27/±1.01  | ±1.19/±1.47  | ±0.97/±0.88  |

Table 14: The error range (%) of experiment results for authorizing usage of models trained with Source-Only NTL on digits.

| Source with Patch | Test | Test without Patch |
|-------------------|------|--------------------|
|                   | MT   | US    | SN    | MM    | SD    | MT   | US    | SN    | MM    | SD    |
| MT                | ±0.43 | ±1.11 | ±1.89 | ±1.09 | ±1.37 | ±2.44 | ±1.22 | ±0.90 | ±1.74 | ±1.29 |
| US                | ±1.73 | ±0.78 | ±0.88 | ±2.21 | ±2.60 | ±1.11 | ±0.73 | ±1.12 | ±1.71 | ±0.45 |
| SN                | ±1.01 | ±1.04 | ±0.36 | ±0.97 | ±0.98 | ±1.17 | ±1.01 | ±0.92 | ±1.39 | ±0.51 |
| MM                | ±1.29 | ±1.19 | ±0.34 | ±0.90 | ±1.20 | ±2.25 | ±0.87 | ±1.13 | ±1.23 | ±1.28 |
| SD                | ±0.67 | ±1.29 | ±1.21 | ±1.82 | ±0.47 | ±1.39 | ±0.97 | ±1.38 | ±1.27 | ±2.61 |

C.4 Error Bar

We conduct all experiments with three random seeds (2021, 2022, 2023), and present the error range in this section. Table 12 is the error range of Target-Specified NTL corresponding to Table 1 of the main paper; Table 13 presents the error of experiments on Source-Only NTL corresponding to Table 3 of the main paper; Table 14 shows the error of model authorization which is presented as Table 4 in our main paper.

D Possible Attacks based on NTL

Although we propose the Non-Transferable Learning for protecting the Intellectual Property in AIaaS, if the model owner is malicious, they can also utilize NTL to poison or implant backdoor triggers to their model evasively and release the model to the public. In the setting of applying Target-Specified NTL to verify the model ownership, the patch we used can also be regarded as a trigger for certain misclassification backdoor. From the results of ownership verification in the main paper, we can see the possibility of launching NTL-based target backdoor attacks. As for the case of Source-Only NTL, our objective is shaped like an universal poison attack by restricting the generalization ability of models. The results in our main paper demonstrate the feasibility of this poison attack. In addition, recently, there are more domain adaptation (DA) works about adapting the domain-shared knowledge within the source model to the target one without the access to the source data [46, 47, 49]. However, if the source model is trained with Source-Only NTL, we believe that these DA works will be ineffective. In other words, our NTL can be regarded as a type of attack to these source-free DA works.
References

[1] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. Mlaas: Machine learning as a service. In 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), pages 896–902. IEEE, 2015. 1

[2] Ralf Herbrich. Machine learning at amazon. In WSDM, page 535, 2017. 1

[3] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. arXiv preprint arXiv:1703.10847, 2017. 1

[4] Jean-Pierre Briot, Gaëtan Hadjeres, and François Pachet. Deep learning techniques for music generation. Springer, 2020. 1

[5] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. Journal of Field Robotics, 37(3):362–386, 2020. 1

[6] Guofa Li, Yifan Yang, Xingda Qu, Dongpu Cao, and Keqiang Li. A deep learning based image enhancement approach for autonomous driving at night. Knowledge-Based Systems, 213: 106617, 2021. 1

[7] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep reinforcement learning for building hvac control. In Proceedings of the 54th annual design automation conference 2017, pages 1–6, 2017. 1

[8] Shichao Xu, Yixuan Wang, Yanzhi Wang, Zheng O’Neill, and Qi Zhu. One for many: Transfer learning for building hvac control. In Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, pages 230–239, 2020. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 1, 7

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 1, 7
Figure 7: Augmentation results of MNIST-M generated by the generative adversarial augmentation framework.

Figure 8: Augmentation results of SYN-D generated by the generative adversarial augmentation framework.

[11] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[12] Xiang Zhang, Xiaocong Chen, Lina Yao, Chang Ge, and Manqing Dong. Deep neural network hyperparameter optimization with orthogonal array tuning. In *International Conference on Neural Information Processing*, pages 287–295. Springer, 2019.

[13] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, pages 587–601, 2017.

[14] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin’ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 269–277, 2017.

[15] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. 2019.

[16] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.

[17] Jingjing Zhao, Qingyue Hu, Gaoyang Liu, Xiaoqiang Ma, Fei Chen, and Mohammad Mehedi Hassan. Afaf: Adversarial fingerprinting authentication for deep neural networks. *Computer Communications*, 150:488–497, 2020.

[18] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In 27th {USENIX} Security Symposium ({USENIX} Security 18), pages 1615–1631, 2018.

[19] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. Refit: a unified watermark removal framework for deep learning systems with limited data. *arXiv preprint arXiv:1911.07205*, 2019.
Figure 9: Augmentation results of CIFAR10 generated by the generative adversarial augmentation framework.

Figure 10: Augmentation results of STL10 generated by the generative adversarial augmentation framework.

[20] Manaar Alam, Sayandeep Saha, Debdeep Mukhopadhyay, and Sandip Kundu. Deep-lock: Secure authorization for deep neural networks. *arXiv preprint arXiv:2008.05966*, 2020. 2, 4

[21] Abhishek Chakraborty, Ankit Mondai, and Ankur Srivastava. Hardware-assisted intellectual property protection of deep learning models. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. 2, 4

[22] Ziqi Wang, Marco Loog, and Jan van Gemert. Respecting domain relations: Hypothesis invariance for domain generalization. *arXiv preprint arXiv:2010.07591*, 2020. 2, 3

[23] Haoliang Li, YuFei Wang, Renjie Wan, Shiqi Wang, Tie-Qiang Li, and Alex C Kot. Domain generalization for medical imaging classification with linear-dependency regularization. *arXiv preprint arXiv:2009.12829*, 2020. 3

[24] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *European Conference on Computer Vision*, pages 561–578. Springer, 2020. 2, 3

[25] Hui Tang and Kui Jia. Discriminative adversarial domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5940–5947, 2020. 2, 3

[26] Jianfei Yang, Han Zou, Yuxun Zhou, Zhaoyang Zeng, and Lihua Xie. Mind the discriminability: Asymmetric adversarial domain adaptation. In *European Conference on Computer Vision*, pages 589–606. Springer, 2020. 2, 3

[27] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2

[28] Gilles Blanchard, Gunmin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. *Advances in neural information processing systems*, 24: 2178–2186, 2011. 3
Figure 11: Augmentation results of VisDA-T generated by the generative adversarial augmentation framework.

[29] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5400–5409, 2018.

[30] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. Correlation-aware adversarial domain adaptation and generalization. Pattern Recognition, 100:107124, 2020.

[31] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. Advances in Neural Information Processing Systems, 33, 2020.

[32] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In International Conference on Machine Learning, pages 7728–7738. PMLR, 2020.

[33] Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees GM Snoek, and Ling Shao. Learning to learn with variational information bottleneck for domain generalization. In European Conference on Computer Vision, pages 200–216. Springer, 2020.

[34] Yingjun Du, Xiantong Zhen, Ling Shao, and Cees GM Snoek. Metanorm: Learning to normalize few-shot batches across domains. 2020.

[35] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12556–12565, 2020.
[36] Long Zhao, Ting Liu, Xi Peng, and Dimitris Metaxas. Maximum-entropy adversarial data augmentation for improved generalization and robustness. *arXiv preprint arXiv:2010.08001*, 2020. 3

[37] Lei Li, Ke Gao, Juan Cao, Ziyao Huang, Yepeng Weng, Xiaoyue Mi, Zhengze Yu, Xiaoya Li, et al. Progressive domain expansion network for single domain generalization. *arXiv preprint arXiv:2103.16050*, 2021. 3

[38] Zhenlin Xu, Deyi Liu, Junlin Yang, and Marc Niethammer. Robust and generalizable visual representation learning via random convolutions. *arXiv preprint arXiv:2007.13003*, 2020. 3

[39] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 3, 4

[40] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2017–2025, 2015. 3

[41] Masato Ishii and Masashi Sugiyama. Source-free domain adaptation via distributional alignment by matching batch normalization statistics. *arXiv preprint arXiv:2101.10842*, 2021. 3

[42] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 136–144, 2016.

[43] Shichao Xu, Lixu Wang, Yixuan Wang, and Qi Zhu. Weak adaptation learning—addressing cross-domain data insufficiency with weak annotator. *arXiv preprint arXiv:2102.07358*, 2021. 3

[44] Myeongjin Kim and Hyeran Byun. Learning texture invariant representation for domain adaptation of semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12975–12984, 2020. 3

[45] Huafeng Li, Yiwen Chen, Dapeng Tao, Zhengtao Yu, and Guanqiu Qi. Attribute-aligned domain-invariant feature learning for unsupervised domain adaptation person re-identification. *IEEE Transactions on Information Forensics and Security*, 16:1480–1494, 2020. 3

[46] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020. 3, 16

[47] Sk Miraj Ahmed, Dripta S Raychaudhuri, Sujoy Paul, Samet Oymak, and Amit K Roy-Chowdhury. Unsupervised multi-source domain adaptation without access to source data. *arXiv preprint arXiv:2104.01845*, 2021. 16

[48] Lixu Wang, Songtao Liang, and Feng Gao. Providing domain specific model via universal no data exchange domain adaptation. In *Journal of Physics: Conference Series*, volume 1827, page 012154. IOP Publishing, 2021. 3

[49] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4544–4553, 2020. 3, 16

[50] Yushi Cheng, Xiaoyu Ji, Lixu Wang, Qi Pang, Yi-Chao Chen, and Wenyuan Xu. mid: Tracing screen photos via moiré patterns. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021. 3

[51] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018. 3

[52] Zheng Li, Chengyu Hu, Yang Zhang, and Shangqing Guo. 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*, pages 126–137, 2019. 3
[53] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. The Journal of Machine Learning Research, 19(1):1947–1980, 2018. 4, 11

[54] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics, 22(14):e49–e57, 2006. 5

[55] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: towards deeper understanding of moment matching network. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pages 2200–2210, 2017. 6

[56] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014. 6

[57] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In Neural Information Processing Systems (NIPS), 2016. 6

[58] J Michael Cherry, Caroline Adler, Catherine Ball, Stephen A Chervitz, Selina S Dwight, Erich T Hester, Yankai Jia, Gail Juvik, TaiYun Roe, Mark Schroeder, et al. Sgd: Saccharomyces genome database. Nucleic acids research, 26(1):73–79, 1998. 7

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 7

[60] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine, 29(6):141–142, 2012. 7

[61] Jonathan J. Hull. A database for handwritten text recognition research. IEEE Transactions on pattern analysis and machine intelligence, 16(5):550–554, 1994. 7

[62] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 7

[63] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. The journal of machine learning research, 17(1):2096–2030, 2016. 7

[64] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. Effects of degradations on deep neural network architectures. arXiv preprint arXiv:1807.10108, 2018. 7

[65] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 215–223. JMLR Workshop and Conference Proceedings, 2011. 7

[66] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for domain adaptation. arXiv preprint arXiv:1706.05208, 2017. 7

[67] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017. 7

[68] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 7