T-GD: Transferable GAN-generated Images Detection Framework

Hyeonseong Jeon [1] **Youngoh Bang** [1] **Junyaup Kim** [2] **Simon S. Woo** [2 3]

Abstract

Recent advancements in Generative Adversarial Networks (GANs) enable the generation of highly realistic images, raising concerns about their misuse for malicious purposes. Detecting these GAN-generated images (GAN-images) becomes increasingly challenging due to the significant reduction of underlying artifacts and specific patterns. The absence of such traces can hinder detection algorithms from identifying GAN-images and transferring knowledge to identify other types of GAN-images as well. In this work, we present the Transferable GAN-images Detection framework (T-GD), a robust transferable framework for an effective detection of GAN-images. T-GD is composed of a teacher and a student model that can iteratively teach and evaluate each other to improve the detection performance. First, we train the teacher model on the source dataset and use it as a starting point for learning the target dataset. To train the student model, we inject noise by mixing up the source and target datasets, while constraining the weight variation to preserve the starting point. Our approach is a self-training method, but distinguishes itself from prior approaches by focusing on improving the transferability of GAN-image detection. T-GD achieves high performance on the source dataset by overcoming *catastrophic forgetting* and effectively detecting state-of-the-art GAN-images with only a small volume of data without any metadata information.

1. Introduction

Recent advancements in Generative Adversarial Networks (GANs) (Choi et al., 2019; Zakharov et al., 2019; Karras et al., 2019b; Shaham et al., 2019) enable the generation of realistic images, which has now become feasible through few-shot or single-shot learning. Some GANs manage to further reduce visible artifacts and patterns, such as blurred object shape, checkerboard artifacts, semantically strange objects, and unnatural backgrounds. For these reasons, even high-resolution images produced by the latest GANs are hardly distinguishable from real images or by human inspection.

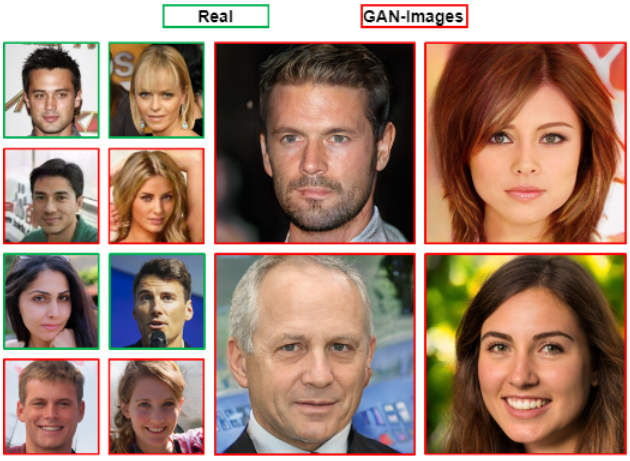

Figure 1. Sample data used for our experiment. Images inside the green border are real images, while those inside the red border are GAN-images. The left-hand side images in row order are from CelebA, StarGAN, FFHQ, and StyleGAN. The right-hand side images in row order are from PGGAN and StyleGAN2.

A typical way of detecting GAN-images is to train Convolutional Neural Networks (CNNs) and a binary classifier with a large number of images generated from GANs. Some researchers (Marra et al., 2019a; Zhang et al., 2019; Yu et al., 2019) have shown that the detection performance can be improved by analyzing artifacts and patterns in GAN-images. Many of the existing methods has achieved high performance in detecting GAN-images when the model tests on the same dataset used during the training phase (Tariq et al., 2019; 2018; Jeon et al., 2019). Moreover, this binary classifier can be realized by the use of existing and well-structured CNN architectures (Tan & Le, 2019; Jeon et al., 2020).

[1]Department of Artificial Intelligence, Sungkyunkwan University, Suwon, S. Korea [2]Computer Science and Engineering Department, Sungkyunkwan University, Suwon, S. Korea [3]Department of Applied Data Science, Sungkyunkwan University, Suwon, S. Korea. Correspondence to: Simon S. Woo <swoo@g.skku.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

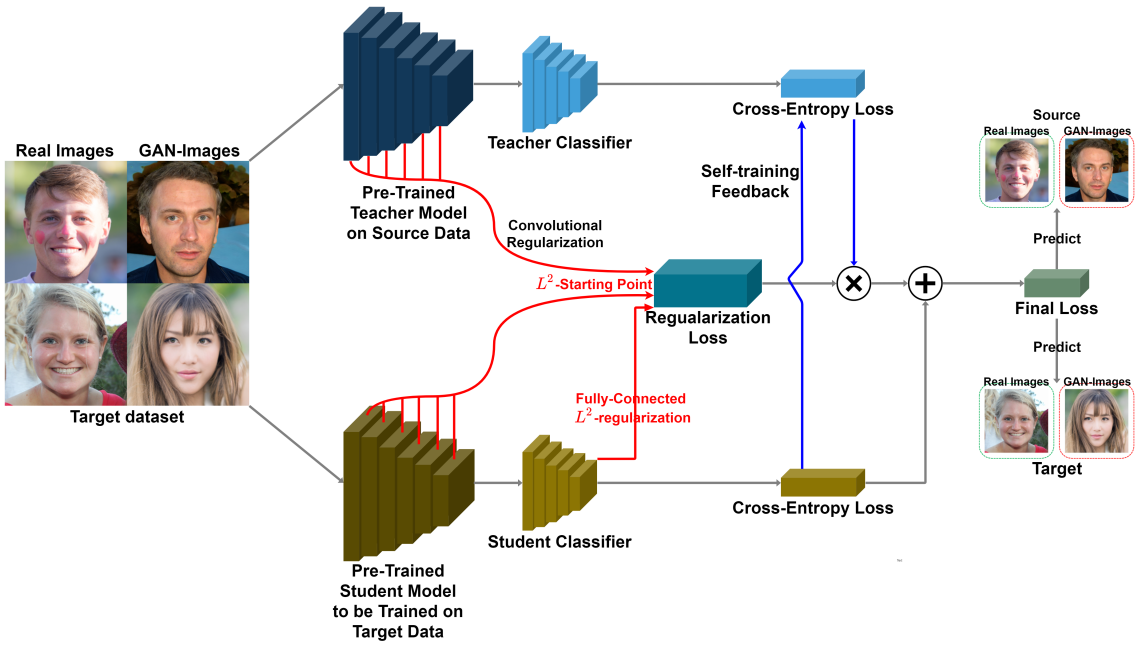

Figure 2. Overview of our T-GD network. For efficient transfer learning, our network uses L^2-*Starting Point* (red) and *Self-training* (blue).

However, above methods are ineffective for improving transfer learning performance. That is, when the CNN classifier is trained on one dataset, it shows poor performance on other datasets. ForensicTransfer (Cozzolino et al., 2018) introduced an autoencoder for the GAN-image detection. They apply autoencoder and detect GAN-images through reconstruction error. This learning method has advantages regarding lower data usage, when the model is well trained. Although the ForensicTransfer showed promise for model transferability, its performance remains mediocre. In previous research, either artifacts, patterns, or augmentations were utilized individually for successful transfer learning, yet it is possible to combine them to transfer knowledge of GAN-image detection.

In this paper, our objective is to maintain a high detection performance during transfer learning on the source and target datasets, without suffering *catastrophic forgetting*. While many studies on transfer learning have already shown impressive performance, they have not applied for GAN-image detection. Therefore, in this work, we propose a novel regularization method with self-training for transfer learning by combining and transforming regularization, augmentation, self-training, and learning strategies to improve transferability of GAN-image detection. In particular, our approach is inspired by starting point as the reference (SPAR), L^2-SP (Li et al., 2018), which regularizes the weight variation of a target model by referring to the weights pre-trained on the source dataset. The limitation of the latter method is that it cannot provide an optimal solution for transfer learning. As the regularization strength changes, the control of the regularization can easily be lost. Our approach overcomes this issue through self-training, where the teacher model automatically helps control the strength of regularization when the student model learns from the target dataset.

In addition, we introduce a novel augmentation method to solve the over-fitting problem by transforming Cutmix (Yun et al., 2019), which randomly mixes up a rectangle patch of training images. Note that the original Cutmix mixes up inter-class images; our experiment showed that this renders the learning process highly unstable for a binary classifier, and thus we transform the inter-class Cutmix to an intra-class Cutmix to increase the stability of the learning process. Also, our method combines Gaussian blur (Xuan et al., 2019) and Joint Photograph Experts Group (JPEG) compression (Wang et al., 2019), which were previously studied for GAN-image detection.

For transfer learning, we apply learning strategies, such as Weight Standardization (Qiao et al., 2019) (WS), Group Normalization (Wu & He, 2018) (GN), and the tuning of learning rate and momentum rate (Li et al., 2020). WS and GN achieved comparable performance on image classification, not depending on batch size statistics. Also, we implement transfer learning using low learning and momentum rates for stochastic gradient descent (SGD) inspired by (Li et al., 2020). These have been experimented on object detection transfer, but we demonstrate that these strategies also work well for transfer learning across different domains. Fi-

nally, we integrate these approaches into one framework, the Transferable GAN-generated image Detection framework (T-GD). Compared to general methods of transfer learning and recent GAN-image detection, we show that T-GD is equipped with robust transferability and achieves high performance.

2. Related Work

GAN-image detection. Widely used approaches for the detection of GAN-images include the addition of a learning method, the transformation of GAN-images, the application of data augmentation, and the use of metadata. Another technique, based on multi-task incremental learning (Marra et al., 2019b), shows great promise for transferability within different types of GAN-images, changing the existing learning method (autoencoder) and loss function (incremental learning). However, a lingering issue is the need for a large amount of data; our work directly alleviates this problem, maintaining the model performance with a smaller amount of data. (Nataraj et al., 2019) proposed using a combination of co-occurrence matrices and JPEG compression to transform GAN-images for data augmentation. Co-occurrence matrices are extracted from three color channels in the pixel domain following JPEG compression and are used to train the CNN. In their approach, the JPEG compression contributed to the improvement of performance, but the transformation of the input data into co-occurrence matrices caused over-fitting and reduced generalizability.

Some methods identify a unique artifact spectrum caused by the up-sampling component (Zhang et al., 2019), while others use the photo-response non-uniformity (PRNU) pattern as the input of CNN classifiers (Marra et al., 2019a). Augmentation techniques requiring domain knowledge of GAN-image detection, such as Gaussian blur and Gaussian noise (Xuan et al., 2019), were also studied and the combination of Gaussian blur and JPEG compression was shown to achieve high performance (Wang et al., 2019). However, employing a single data augmentation method achieved limited transfer learning performance. Therefore, we use the combination of JPEG compression and Gaussian blur to achieve better transfer learning results. Similar to prior digital fingerprint techniques, GAN fingerprints (i.e., image and model fingerprints) are used to differentiate real and GAN-images using metadata (Yu et al., 2019). They assume *white-box attack scenarios*, where detectors possess knowledge of the data and the model (metadata information) of attackers. Our approach differs from theirs, where ours is constrained to *a non-adaptive black-box approach scenario*; detectors only possess knowledge of the training data.

Transfer learning. Several transfer learning methods have been explored in terms of meta-learning, self-supervised learning, domain adaptation (Zamir et al., 2018), knowledge distillation, and continual learning. In particular, starting point as the reference (SPAR) (Li et al., 2018) and Deep Learning Transfer using Feature Map with Attention (DELTA) (Li et al., 2019) use L^2 regularization as a starting point to maintain the source dataset as the inductive learning method during domain adaptation. The difference from ours is that we use self-training to control the regularization effect, which has the advantage of preventing either excessive or minor regularization.

(Yim et al., 2017) proposed two additional layers to calculate the flow of solution procedure matrix for knowledge distillation, but our T-GD shows robust performance in transfer learning without these layers and expensive computation. Learning with continual tasks (Zenke et al., 2017), where the node weight is regularized based on the importance of previous tasks, is similar to our method. However, we provide the following differences: First, the source and target datasets differ in size (see Table 2), resulting in relatively low computational cost in L^2-SP. Second, in our task, achieving generalizability using a small amount of target data is as essential as the prevention of forgetting; to address this trade-off, we chose L^2-SP using all weights. Third, unlike continual learning on each independent task, our T-GD focuses on transfer learning within the GAN-image domains.

Self-training. Self-training methods (Yalniz et al., 2019; Xie et al., 2019) were used to increase the state-of-the-art top-1 accuracy of ImageNet (Russakovsky et al., 2015). The difference from our work is that our objective is to increase transferability and not the single model performance. We use a teacher-student structure, and inject noise to the student model and the input data to prevent over-fitting by effective techniques, such as dropout (Srivastava et al., 2014), stochastic depth (Huang et al., 2016), and intra-class Cutmix (data augmentation)). Also, self-training was used for domain adaptation by (RoyChowdhury et al., 2019). Their self-training model is video-specific, applying a teacher model to the target domain, which is different from our image classification task.

3. Our Method

The first step of T-GD is to train the pre-trained models, namely the binary classifiers predicting whether an image is GAN-generated or not.

CNN binary classifier. We chose CNN binary classifiers as classifiers for the source dataset. This choice has three advantages: (1) it is easy to reuse pre-trained models, (2) many pre-studied CNN architectures can be utilized, and (3) it shows a more stable performance in binary classification than other methods such as autoencoders. We pre-train the CNN binary classifier on the source dataset and trans-

fer (fine-tune) this pre-trained model to the target dataset. For instance, EfficientNet (the CNN classifier) is trained on the PGGAN-dataset (the source) and fine-tuned on the StyleGAN-dataset (the target).

EfficientNet. We implemented EfficientNet-B0 (Tan & Le, 2019) and used it as the CNN classifier. Although EfficientNet-B0 has the lowest number of parameters (about four million) among the EfficientNets, it performs well in GAN-image detection in our experiment, compared to Inception-V3 (Szegedy et al., 2017) and Xception (Chollet, 2017). Another change we make to the model is the use of WS (Qiao et al., 2019) and GN (Wu & He, 2018), instead of batch normalization (BN) (Ioffe & Szegedy, 2015), due to their superior efficiency regarding transfer learning to that of batch statistics.

ResNext. We implemented ResNext32×4d (Xie et al., 2017) and used it as the CNN classifier, where ResNext32×4d has more parameters (about twenty million) than EfficientNet-B0. We also replace BN with WS and GN.

3.1. L^2-SP

The next step is transfer learning. The weight of the pre-trained model from the source dataset is used as the SPAR. In particular, we use L^2-SP for transfer learning. Regularization can lead to a better optimization by preventing over-fitting when learning from scratch; L^2-SP differs in that the starting point from a well pre-trained source dataset guides the learning process by referring to the information of the pre-trained source dataset. This method does not require freezing the weights of the pre-trained model nor using weight decay. Our method regularizes convolution layers and fully-connected (FC) layers independently.

General form of regularization. Let w be the weight parameters, and $J(\hat{y}_i, y_i)$ be the loss function of the neural networks, where $\hat{y}_i$ is the i^{th} score predicted by the models and y_i is the i^{th} label. And $\Omega(w)$ is the p-norm function of the weight w as a general form of regularization loss, $f_w(x_i)$ is the neural network function with the i^{th} data x_i, and n is the dataset size. Equation 1 indicates the general form of the loss function with a weight regularization component:

$$\begin{aligned} &\min_w \frac{1}{n}\sum_{i=1}^{n} J(\hat{y}_i, y_i) + \lambda \cdot \Omega(w), \\ &\hat{y}_i = f_w(x_i), \end{aligned} \tag{1}$$

where λ balances the regularization and the loss function, J is the cross-entropy function, and Ω is the L^1 or L^2-norm of the parameter w.

L^2 regularization. L^2 regularization is used in transfer learning to avoid over-fitting and to overcome the forgetting of the learned information or *catastrophic forgetting*.

$$\Omega_{l2}(w) = \|w\|_2^2. \tag{2}$$

Equation 2 is the Ω function, namely the L^2-norm of w.

$$\min_w \frac{1}{n}\sum_{i=1}^{n} J(\hat{y}_i, y_i) + \beta \cdot \Omega_{l2}(w_{fc}). \tag{3}$$

In Eq. 3, the first term is the same as in Eq. 1, representing the cross-entropy loss function. The second term is the Ω_{l2} function or the L^2 regularization term (Eq. 2) of w_{fc}, the weights of the FC layers, scaled by β, which is equivalent to λ in Eq. 1. Note that the L^2 regularization is applied solely to the FC layers since over-fitting and forgetting are delayed, but not completely prevented in the course of learning.

L^2-SP. Let w' be the pre-trained weights from the source dataset, as shown in Section 3.1, serving as the starting point (SP) as the reference, as well as a regularization point that provides guidance for transfer learning when fine-tuning. Using L2-norm, we define L^2-SP as follows:

$$\Omega_{sp}(w, w') = \|w_{conv} - w'_{conv}\|_2^2, \tag{4}$$

where w_{conv} denotes the weights of the convolution layers, excluding those of the FC layers. Equation 4 indicates that L^2-SP is a one-to-one mapping between the convolution layers of the source and target datasets, e.g., the PGGAN-classifier (the source) to the StyleGAN-classifier (the target).

Loss function. We combine Eq. 4, sharing the architecture of the source and target models, with the second term of Eq. 3, accounting for the FC layer (final layer) as follows:

$$\begin{aligned} &\min_w \frac{1}{n}\sum_{i=1}^{n} J(\hat{y}_i, y_i) + \alpha \cdot \Omega_{sp}(w, w') + \beta \cdot \Omega_{l2}(w_{fc}), \\ &J(\hat{y}_i, y_i) = -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i), \end{aligned} \tag{5}$$

where J is the negative log-likelihood loss function, and α and β are tunable hyperparameters of which (Li et al., 2019) use values in the range from 0.1 to 0.01. The difference from L^2-SP is that we transform α and β into γ, a parameter which adjusts itself according to the learning situation. More details about the transformed parameters are provided in Section 3.3.

3.2. Self-training for L^2-SP

We transform the transfer learning framework into a self-training framework. In other words, the source/target model is changed into a teacher/student model. In addition to the role of a typical source model, which serves as SPAR and a regularizer to guide the learning process, the teacher model has the role of adjusting the parameters based on the learned

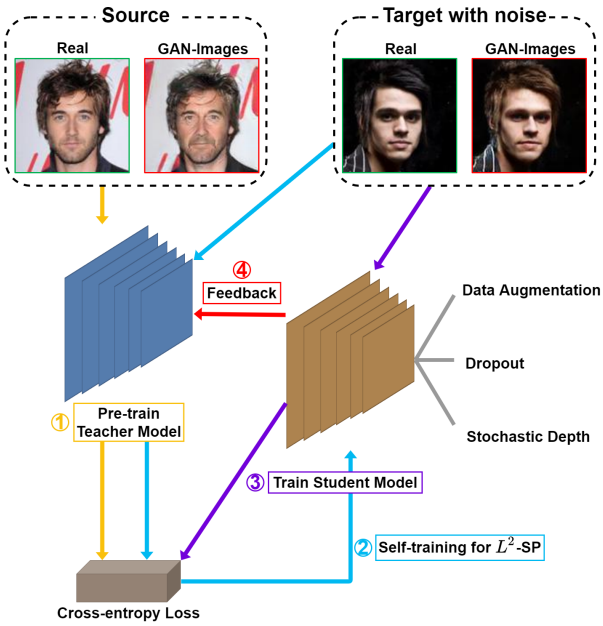

Figure 3. Overview of our self-training method. Note that the number of processes shown in this figure equals to that of the ordered processes demonstrated in Alg. 1.

target dataset (training loss). That is, the teacher model directly controls α and β, as shown in Eq. 5.

Let $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ be the labeled source dataset and $\{(\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), ..., (\tilde{x}_m, \tilde{y}_m)\}$ the labeled target dataset. In a typical self-training process, unlabeled data is used to increase the generalizability for single dataset performance, e.g., ImageNet, by learning from extra training data. However, we assume the usage of additional data is highly limited. Hence, we only use labeled data for transfer learning.

$$J(\hat{y}_i, \tilde{y}_i) = -\tilde{y}_i \log(\hat{y}_i) - (1 - \tilde{y}_i) \log(1 - \hat{y}_i), \\ \hat{y}_i = f_{w'}^{noised}(\tilde{x}_i^{noised}). \tag{6}$$

In Eq. 6, w' denotes the weights of the teacher model, and $f_{w'}$ denotes the pre-trained models from the source dataset. $J(\hat{y}_i, \tilde{y}_i)$ denotes the binary cross-entropy loss, but the input data, $\tilde{x}_i^{noised}$, is from the target dataset with noise injection.

$$\gamma := s\sigma\left(-\frac{1}{m}\sum_{i=1}^{m} J(\hat{y}_i, \tilde{y}_i)\right), \\ \sigma(x) = 1/(1 + e^{-x}). \tag{7}$$

In Eq. 7, s is a hyperparameter taking values from 0.1 to 2.0, and the γ score is obtained through the sigmoid function σ to help stabilize training, whose input is the negative mean

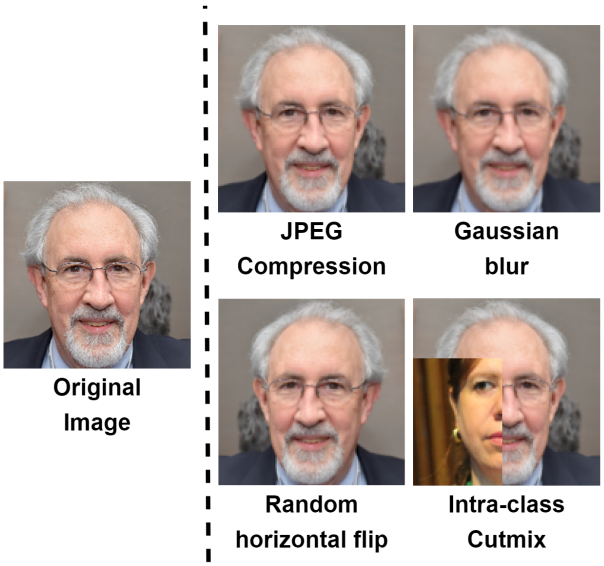

Figure 4. Examples of different noisy data augmentation techniques applied to the original input image.

loss function described in Eq. 6. In the transfer phase, we use the same noised target data for both the teacher and student models. The teacher is evaluated on the data (Eq. 6), and the negative value of the result is taken and transformed by the sigmoid function γ in Eq. 7, where γ regulates the intensities of both L^2-SP (Eq. 4) and the L2-norm of FC layer (Eq. 3). An analysis of γ and the error amplification of self-training is presented in Supp. A.

Final loss function. We replace α and β with γ in Eq. 5 to act as a changeable balancing parameter for regularization as follows:

$$\min_{w} \frac{1}{n}\sum_{i=1}^{n} J(\hat{y}_i, y_i) + \gamma \cdot \Omega_{sp}(w, w') + \gamma \cdot \Omega_{l2}(w_{fc}). \tag{8}$$

The final loss function, as shown in Eq. 8, is composed of a cross-entropy term and an L^2-SP term for the self-training of the student model. Figure 2 shows an overview of this entire pipeline, and Supp. Alg. 1 presents the detailed algorithm.

Augmentation and noised model. Noise is injected to both the data and the model. For data augmentation, we use JPEG compression (Wang et al., 2019), and Gaussian blur (Xuan et al., 2019), random horizontal flip, and a transformed version of Cutmix (Yun et al., 2019), called intra-class Cutmix. We apply dropout (Srivastava et al., 2014) to the FC layer at a stronger rate than for the pre-trained model. In addition, we apply stochastic depth (Huang et al., 2016), randomly dropping the paths of residual layers, also at a stronger rate than for the pre-trained model.

Method	Category	Zero-shot (Pre-trained model)				Transfer Learning			
	Dataset	PGGAN	StarGAN	StyleGAN	StyleGAN2	PGGAN	StarGAN	StyleGAN	StyleGAN2
GeneralTransfer	PGGAN	99.91%	56.81%	49.47%	49.32%	99.86%	87.06%	54.17%	54.18%
EfficientNet-B0	StarGAN	66.47%	99.88%	52.01%	52.10%	95.90%	89.87%	99.03%	99.04%
(Base model)	StyleGAN	49.80%	50.04%	99.96%	99.97%	66.89%	51.12%	99.94%	99.95%
	StyleGAN2	45.23%	49.00%	99.99%	99.99%	91.33%	88.16%	45.26%	47.37%
ForensicTransfer†	PGGAN	97.15%	50.27%	53.57%	53.27%	69.35%	72.40%	76.50%	76.50%
	StarGAN	47.09%	85.34%	49.51%	49.48%	90.14%	51.32%	53.14%	53.14%
	StyleGAN	49.23%	49.66%	99.12%	99.97%	76.57%	58.93%	65.83%	65.85%
	StyleGAN2	49.22%	49.66%	99.12%	99.12%	76.58%	58.94%	65.84%	65.84%
T-GD	PGGAN	99.91%	56.81%	49.47%	49.32%	**95.87%**	91.61%	**98.12%**	**98.13%**
EfficientNet-B0	StarGAN	66.47%	99.88%	52.01%	52.10%	94.94%	**97.32%**	97.29%	93.34%
(Base model)	StyleGAN	49.80%	50.04%	99.96%	99.97%	84.92%	90.00%	**97.83%**	97.71%
	StyleGAN2	45.23%	49.00%	99.99%	99.99%	84.91%	90.01%	97.83%	**97.71%**
T-GD	PGGAN	99.81%	61.25%	49.76%	49.91%	94.91%	93.21%	87.37%	87.58%
ResNext32×4d	StarGAN	41.43%	99.78%	48.37%	48.50%	98.88%	96.15%	91.48%	91.26%
(Base model)	StyleGAN	41.05%	49.16%	99.99%	99.99%	85.93%	79.69%	94.31%	94.31%
	StyleGAN2	38.90%	50.31%	99.90%	99.88%	87.20%	80.19%	**98.39%**	95.38%

Table 1. Performance results. "†" indicates our implementation. All 4 GAN datasets are evaluated with 4 models as well as our models. The Dataset column indicates pre-trained model from a source dataset, and the Dataset row indicates the target test set for transfer learning. The evaluation metric is AUROC (%). The underlined results are the source dataset performance after transfer learning. The best results are highlighted in bold. The Zero-shot category represents the performance of a pre-trained model without any additional training and the Transfer learning category represents each pre-trained model transferred from the source to target dataset.

	Source Data			Target Data
Dataset	Train	Validation	Test	Transfer
PGGAN	64,202	16,051	18,799	2,000
StarGAN	137,239	15,260	50,000	2,000
StyleGAN	33,739	3,900	30,000	2,000
StyleGAN2	42,356	3,900	30,000	2,000

Table 2. GAN-generated datasets used in our experiment, where train, validation, test, as well as transfer dataset are shown. We only use 2,000 images for transfer learning.

Intra-class Cutmix algorithm. The pseudo-code of the intra-class Cutmix algorithm is shown in Supp. Algorithm 2. First, the mini-batch data is shuffled and the index at which the target label Y_m equals to the shuffled target label Y'_m, i.e., real to real and GAN-image to GAN-image, is denoted as same_index in Algorithm 2. If a random variable ρ drawn from a uniform distribution between 0 and 1 is greater than the fixed Cutmix parameter (0.2 when pre-training, and 0.5 when transfer learning), then the input X_m and the shuffled input X'_m are mixed by replacing a randomly cropped region of the input X_m to the region of the shuffled input X'_m. Cutmix (Yun et al., 2019) mixes the target label Y_m through interpolation. Intra-class Cutmix does not mix the label, because the input data X_m still belongs to the same class following replacement.

4. Experimental Results

The description of our dataset and training details are presented in Section D and E, respectively.

4.1. Baselines

General transfer learning method. It is common practice to freeze some weights of pre-trained model from source dataset, and fine-tune the model with weight decay to the target dataset. We also experiment with them for GAN-image detection and call the method GeneralTransfer. We freeze all layers except for the top block layers and FC layers, and the base model is EfficientNet-B0, while GeneralTransfer is trained for 500 epochs with low learning (0.001) and momentum rates (0.1). Our method differs in that we train all weights and regularize them. The rest of the process is the same, e.g., noise injection to the model and the input data.

ForensicTransfer. We implement ForensicTransfer as the baseline in our experiment, where we trained it for 30 epochs in the pre-training stage and for 10 epochs in the transfer stage. The data usage is identical as in our method, but the difference appears in data augmentation and learning strategies. In our experiment, we follow the same data augmentation and learning strategy as ForensicTransfer.

Method	Dataset	PGGAN	StarGAN	StyleGAN	StyleGAN2	Bedroom	Bird
T-GD (*EfficientNet-B0*)	Bedroom	86.25%	88.08%	90.47%	90.25%	<u>94.25%</u>	90.15%
	Bird	87.80%	78.32%	98.49%	98.49%	97.63%	<u>88.82%</u>

Table 3. Transfer learning results of non-face GAN-images. Dataset column indicates pre-trained model from source dataset, and Dataset row indicates test set of target dataset for transfer learning. Evaluation metric is AUROC (%).

Method	Base model	Source dataset	Target dataset	Source AUROC	Target AUROC
w self-training	EfficientNet-B0	StarGAN	PGGAN	**<u>99.15</u>%**	**94.94**%
w/o self-training	EfficientNet-B0	StarGAN	PGGAN	<u>98.96%</u>	92.54%
w augmentation	EfficientNet-B0	PGGAN	StyleGAN2	**<u>95.08</u>%**	98.13%
w/o augmentation	EfficientNet-B0	PGGAN	StyleGAN2	<u>85.04%</u>	99.38%

Table 4. Ablation study for self-training and data augmentation. The augmentation includes intra-class Cutmix, JPEG compression, Gaussian blur, and random horizontal flip. Our model with self-training shows a 2.40% higher target AUROC than those without self-training, increasing the target AUROC from 92.54% to 94.94%. Our model with augmentation shows a 10.04% higher source AUROC than those without augmentation. The underlined results are the source dataset performance after transfer learning. The best results are highlighted in bold.

4.2. Performance Evaluation

AUROC metric. We use Area Under Receiver Operating characteristic Curve (AUROC) to evaluate the model. A broader area under the AUROC indicates a stronger model whose prediction is well classified by a decision boundary. In our work, the AUROC is more suitable for evaluating models than accuracy, since the AUROC does not require a threshold.

Pre-trained model performance. We present our overall performance results in Table 1, where the same test datasets are used as in earlier section. In the zero-shot category, all diagonal terms of results (underlined results) represent single dataset performance of pre-trained models, as shown in the 3rd column of Table 1. The baselines and T-GD have strong GAN-image detection ability, most of which achieving over 99% AUROC.

GeneralTransfer vs. T-GD. GeneralTransfer freezes the pre-trained weights and fine-tunes the model, learning from the target dataset. In the Transfer Learning category, GeneralTransfer shows a trade-off between the source and target performance: after transfer learning, we observe high AUROC for the source dataset (99.86% from pre-trained PGGAN), but low performance for transfer learning (54.17%, and 54.18% AUROC to StyleGAN and StyleGAN2, respectively), or forgetting of the learned source dataset (47.37% AUROC from pre-trained StyleGAN2), but high AUROC for the target dataset (91.33%, and 88.16% for StyleGAN and StarGAN, respectively). On the contrary, T-GD (*EfficientNet-B0*) shows consistent source dataset results: PGGAN (95.87%), StarGAN (97.32%), StyleGAN (97.85%), and StyleGAN2 (97.71%). T-GD is well transferred on the target dataset. In particular, it achieves 98.12%, and 98.13% AUROC for StyleGAN and StyleGAN2, respectively, maintaining 95.87% AUROC from the pre-trained PGGAN.

ForensicTransfer vs. T-GD. ForensicTransfer shows some generalizability, but yields low performance; With pre-trained PGGAN, StyleGAN, and StyleGAN2, ForensicTransfer achieves 69.35%, 65.83%, and 65.84%, respectively. It also clearly shows a trade-off between performance of the source and target datasets. Using pre-trained StarGAN for transfer learning on PGGAN, PGGAN performance shows 90.14%, but forgets the learned information from StarGAN (51.32%).

ResNext vs. EfficientNet. In comparison to T-GD from different base models, the results show subtle differences. EfficientNet-B0 has generally stronger performance in GAN-image detection with fewer parameters. Although the number of parameters affects the classification performance, the performance of EfficientNet (3M) was superior to that of ResNext (20M) in transfer tasks. Therefore, T-GD performance is not directly related to the number of parameters.

Non-face GAN-image detection. T-GD is effective not only for GAN-generated face detection, but also for non-face tasks. We experimented with transfer learning from

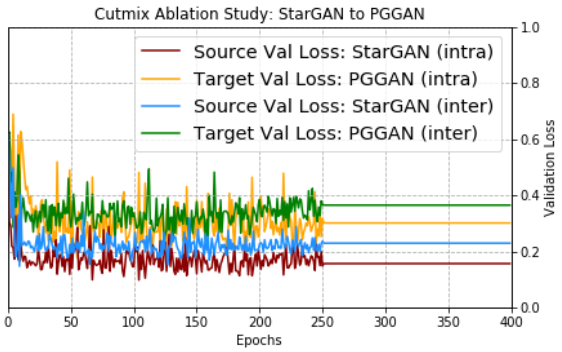

Figure 5. Validation loss in transfer learning between Cutmix and intra-class Cutmix. We can observe that after 50 epochs, the validation loss for intra-class Cutmix (yellow and red) is considerably lower and more stable than that for Cutmix (green and blue).

non-face GAN-images as the source (PGGAN-images from LSUN-bedroom and LSUN-bird) to face GAN-images as the target. We achieved stable AUROC on both detection tasks as shown in Table 3.

4.3. Ablation Study

Self-training effect. In Section 3.2, we explained why self-training is used for L^2-SP. In this ablation study, we validate this method through an ablation study. As shown in Table 4, we experiment with T-GD, which is pre-trained on the StarGAN dataset and transferred to the PGGAN dataset, to compare the performance of our model with and without self-training, while keeping all other settings the same. For the target dataset with self-training, we achieved an AUROC that is 2.40% higher than that of the target dataset without self-training (from 92.54% to 94.94%). The AUROC of the source dataset also increased from 98.96% to 99.15%, as shown in Table 4.

Data augmentation effect. We utilized the following data augmentation methods to avoid over-fitting in transfer learning: intra-class Cutmix, JPEG compression, Gaussian blur, and random horizontal flip. We experiment with these augmentation methods through an ablation study. The performance of our model pre-trained on the PGGAN dataset and transferred to the StyleGAN2 dataset, with and without augmentation, is shown in Table 4. For the target dataset with augmentation, the AUROC of T-GD dropped from 99.38% to 98.13% (1.25%), but we achieved a 10.04% higher AUROC for the source dataset than that of the same dataset without augmentation (from 85.04% to 95.08%). Despite the small reduction in the target AUROC, the drastic increase in the source AUROC implies that over-fitting can be avoided through these augmentation methods in transfer learning, while preventing *catastrophic forgetting*.

Inter-Cutmix vs. Intra-class Cutmix. In Fig. 5, the X-axis and the Y-axis represent the training epochs and the validation loss, respectively. The validation loss performance of Cutmix is shown on green and blue, while that of intra-class Cutmix is presented on yellow and red. Red and blue lines represent the respective validation loss for the source dataset (StarGAN) and, yellow and green represent the target dataset (PGGAN) with respect to epochs. The model pre-trained on the StarGAN dataset is transferred to the PGGAN dataset. The base model is EfficientNet-B0, and all other settings remain the same except for Cutmix. Our experiment shows that intra-class Cutmix has a lower validation loss, and thus a higher performance for both the source and target datasets. We conclude that the improved performance of intra-class Cutmix attributes to the fact that GAN-image detection is a binary classification problem, where the two classes are real and GAN-generated. Mixing only two labels can be harmful and degrade the classification performance.

Grad-CAM. We perform a qualitative analysis of each GAN-image output by using Gradient Class Activation Map (Grad-CAM). The results are presented in Supp. Section C. We describe the results obtained from Grad-CAM, which visualizes the essential regions from an input image required for the prediction of its class.

5. Conclusion

We present T-GD network, a method to maintain high performance on both the source and target datasets for the GAN-image detection during transfer learning. We propose the novel regularization and augmentation techniques, the L^2-SP self-training and intra-class Cutmix, building upon well-known CNN backbone models. While previous research focused on leveraging the metadata information from different GAN models, our method outperforms over other approaches on both source and target datasets without using any metadata from the GAN models. In particular, when PGGAN-images are used as the source data for transfer learning, we observe the best transfer learning performance. Therefore, we recommend PGGAN as the guided dataset for the source data. As the GAN detection classifier evolves, the new generation methods will also appear in the future. Hence, the lack of training data from new GAN generators will be a significant problem. To cope with this issue, we plan to work on classifying GAN-images with few-shot or zero-shot learning. We also hope that future work will continue to challenge and improve existing transfer learning strategies. Our code is available here.[1]

[1] https://github.com/cutz-j/T-GD

Acknowledgements

We thank Siho Han for providing his expertise to greatly improve this work. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00421, AI Graduate School Support Program (Sungkyunkwan University)), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2017R1C1B5076474, and 2020R1C1C1006004). Also, this research was results of a study on the "HPC Support" Project, supported by the 'Ministry of Science and ICT' and NIPA. Additionally, this work was partly supported by Institute for Information & communication Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01343, Regional strategic industry convergence security core talent training business).

References

Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains. *arXiv preprint arXiv:1912.01865*, 2019.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Cozzolino, D., Thies, J., Rössler, A., Riess, C., Nießner, M., and Verdoliva, L. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Jeon, H., Bang, Y., and Woo, S. S. Faketalkerdetect: Effective and practical realistic neural talking head detection with a highly unbalanced dataset. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Jeon, H., Bang, Y., and Woo, S. S. Fdftnet: Facing off fake images using fake detection fine-tuning network. *arXiv preprint arXiv:2001.01265*, 2020.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., and Soatto, S. Rethinking the hyperparameters for fine-tuning. *arXiv preprint arXiv:2002.11770*, 2020.

Li, X., Grandvalet, Y., and Davoine, F. Explicit inductive bias for transfer learning with convolutional networks. *arXiv preprint arXiv:1802.01483*, 2018.

Li, X., Xiong, H., Wang, H., Rao, Y., Liu, L., and Huan, J. Delta: Deep learning transfer using feature map with attention for convolutional networks. *arXiv preprint arXiv:1901.09229*, 2019.

Marra, F., Gragnaniello, D., Verdoliva, L., and Poggi, G. Do gans leave artificial fingerprints? In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 506–511. IEEE, 2019a.

Marra, F., Saltori, C., Boato, G., and Verdoliva, L. Incremental learning for the detection and classification of gan-generated images. *arXiv preprint arXiv:1910.01568*, 2019b.

Nataraj, L., Mohammed, T. M., Manjunath, B., Chandrasekaran, S., Flenner, A., Bappy, J. H., and Roy-Chowdhury, A. K. Detecting gan generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019 (5):532–1, 2019.

Qiao, S., Wang, H., Liu, C., Shen, W., and Yuille, A. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

RoyChowdhury, A., Chakrabarty, P., Singh, A., Jin, S., Jiang, H., Cao, L., and Learned-Miller, E. Automatic adaptation of object detectors to new domains using self-training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 780–790, 2019.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Shaham, T. R., Dekel, T., and Michaeli, T. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4570–4580, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Tariq, S., Lee, S., Kim, H., Shin, Y., and Woo, S. S. Detecting both machine and human created fake face images in the wild. In *Proceedings of the 2nd international workshop on multimedia privacy and security*, pp. 81–87, 2018.

Tariq, S., Lee, S., Kim, H., Shin, Y., and Woo, S. S. Gan is a friend or foe? a framework to detect various fake face images. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 1296–1303, 2019.

Wang, S.-Y., Wang, O., Zhang, R., Owens, A., and Efros, A. A. Cnn-generated images are surprisingly easy to spot... for now. *arXiv preprint arXiv:1912.11035*, 2019.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Xie, Q., Hovy, E., Luong, M.-T., and Le, Q. V. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Xuan, X., Peng, B., Wang, W., and Dong, J. On the generalization of gan image forensics. In *Chinese Conference on Biometric Recognition*, pp. 134–141. Springer, 2019.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

Yim, J., Joo, D., Bae, J., and Kim, J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4133–4141, 2017.

Yu, N., Davis, L. S., and Fritz, M. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7556–7566, 2019.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6023–6032, 2019.

Zakharov, E., Shysheya, A., Burkov, E., and Lempitsky, V. Few-shot adversarial learning of realistic neural talking head models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9459–9468, 2019.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722, 2018.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017.

Zhang, X., Karaman, S., and Chang, S.-F. Detecting and simulating artifacts in gan fake images. *arXiv preprint arXiv:1907.06515*, 2019.

A. Pipeline of Self-training for L^2-SP

Algorithm 1 Self-training for L^2-SP

Require: a source data $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ and a target data $\{(\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), ..., (\tilde{x}_m, \tilde{y}_m)\}$.

1: Pre-train teacher model w' and minimizes the cross-entropy loss on the source dataset.
Input: Data (x_i, y_i), size n, output $\hat{y}_i$
Objective: $\min_w \sum_{i=1}^{n} J(\hat{y}_i, y_i) + \lambda \cdot \Omega_{l2}(w)$

2: w' is used as a starting point for learning the student model.
Input: Data $(\tilde{x}_i^{noised}, \tilde{y}_i^{noised})$, size m, output $\hat{y}_i'$
γ **score:** $\sigma(-\frac{1}{n}\sum_{i=1}^{n} J(\hat{y}_i', \tilde{y}_i))$

3: Learn student model w and minimizes the cross-entropy loss and the regularization terms with γ.
Input: Data $(\tilde{x}_i^{noised}, \tilde{y}_i^{noised})$, size m, output $\hat{y}_i^{\star}$
Objective: $\min_w \sum_{i=1}^{n} J(\hat{y}_i^{\star}, y_i) + \gamma \cdot \Omega_{sp}(w, w') + \gamma \cdot \Omega_{l2}(w_{fc})$

4: Feedback: Use the student model as the teacher model and go back to step **2**

Pipeline. Figure 3 depicts the overall self-training process, and Algorithm 1 describes the detailed process. The inputs are labeled data from the target dataset. We separated the learning of the teacher model (pre-training classifiers on the source dataset) from that of the student model (fine-tuning the transferred model). The algorithm shows a self-training method, in which the teacher and the student learn by exchanging feedback. Feedback refers to the process in which we copy the weight of the student onto that of the teacher at a pre-defined cycle, which we chose to be 200. When the student model is trained (the stage of transfer learning), augmented input data and the noise-injected model are used. It renders the transfer learning more robust during the training process.

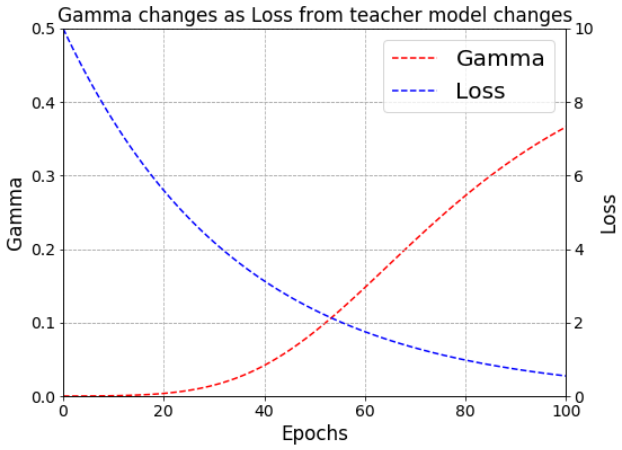

Figure 6. Self-training parameter γ variation. Each red line and blue line represents cross-entropy loss $J(\hat{y}_i, \tilde{y}_i)$ from the teacher model and γ which varies accordingly. The left-Y-axis, right-Y-axis, and X-axis represent Gamma value, right-Y-axis, loss $J(\hat{y}_i, \tilde{y}_i)$

Gamma. Figure 6 shows how γ changes as $J(\hat{y}_i, \tilde{y}_i)$ changes. γ depends on the loss of a large dataset. For a large loss of the target dataset, the target model requires more training to achieve smaller loss and cause γ to reduce. On the other hand, for a small loss, learning from the target dataset requires less training and cause γ to increase. In an early training phase, the teacher loss is very high, because the model was not trained. As a result, γ is close to zero, and the regularization of the $L2$-SP and L^2-norm is weakened. As the target is trained, the loss decreases, while γ increases, meaning that the regularization is stronger; figure 6 indicates this variation. This allows us to avoid an extreme regularization effect, either excessive or insufficient, which is a critical issue when using the fixed hyperparameter. T-GD avoids the error amplification for two reasons: first, we used a small volume of labeled data, unlike the typical self-training method using a large volume of unlabeled data, and second, γ is scaled within a range of 0 to 0.5 by the sigmoid function. Consequently, extremely high/low loss values of the teacher due to false information do not have a significant impact on the learning.

B. Intra-class Cutmix

Algorithm 2 Intra-class Cutmix algorithm

Require: a target data $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, where the input data is composed of tensors of size $m \times c \times w \times h$, where m is the mini-batch size, c is the channel (3), w is the width (128), and h is the height (128).
for each epoch **do**
  $X_m, Y_m = \{(x_i, y_i), ..., (x_m, y_m)\}$
  if training **then**
    X'_m, Y'_m = random_shuffle(X_m, Y_m)
    same_index = [Y'_m==Y_m]
    ρ = Uniform(0, 1)
    if $\rho \geq$ cutmix_prob **then**
      b_x, b_y = Uniform(0, w), Uniform(0, h)
      b_w, b_h = Sqrt(1 - λ), Sqrt(1 - λ)
      $x1, x2$ = Round(Clip($b_x - b_w/2$, min=0)), Round(Clip($b_x + b_w/2$, max=w))
      $y1, y2$ = Round(Clip($b_y - b_h/2$, min=0)), Round(Clip($b_y + b_h/2$, max=h))
      X_m[same_index,:,$x1$:$x2$,$y1$:$y2$] = X'_m[same_index,:,$x1$:$x2$,$y1$:$y2$]
    end if
  end if
  $\hat{Y}_m$ = feed_forward(X_m)
  J = loss_function($\hat{Y}_m$, Y_m)
  update()
end for

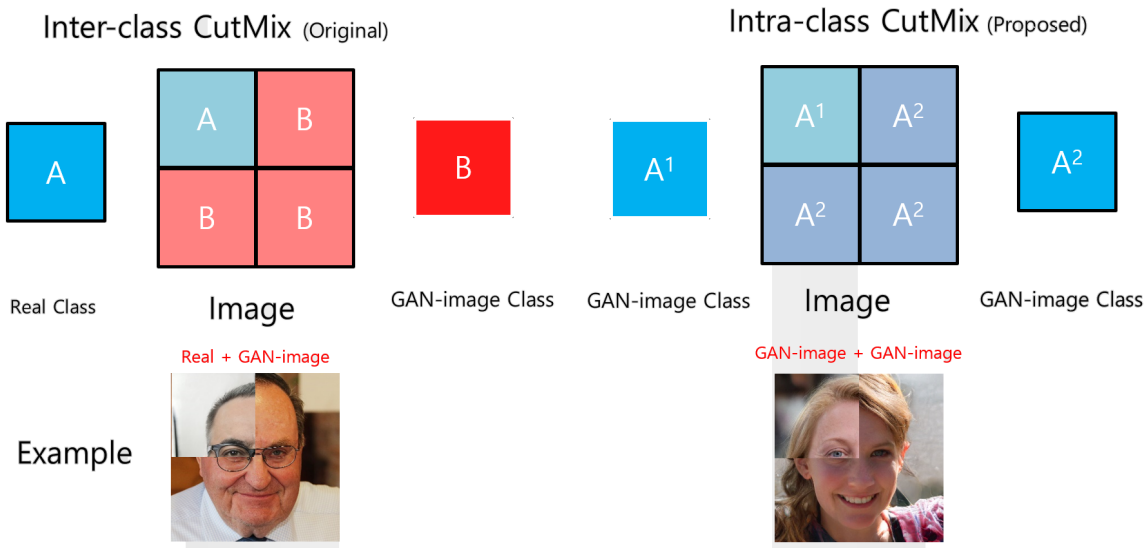

Figure 7. Comparison between Cutmix and proposed Intra-class Cutmix.

Algorithm 2 presents the pseudo-code for the intra-class Cutmix. Figure 7 describes the difference between Cutmix and Intra-class Cutmix. On the left side, the original CutMix, is referred to as Inter-class CutMix. Inter-class Cutmix replaces the chosen patch with another image patch in the same location. The ground truth labels are also mixed proportionally to the area of the patches. On the right side is shown our proposed Intra-class CutMix. The ground truth is not used in the mixing region. In our experiment, we found that the inter-class CutMix for a binary classification causes highly unstable training.

C. Gradient Class Activation Map (Grad-CAM)

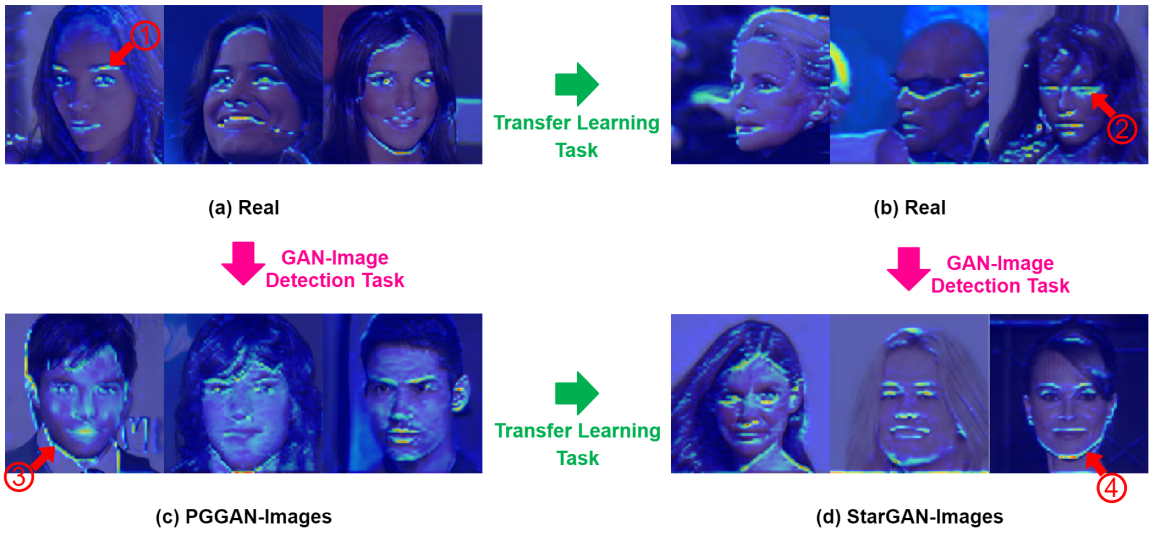

Figure 8. Class activation map output of the images. (a) and (b) show the CelebA-HQ real images for PGGAN and StarGAN, respectively; (c) and (d) show the GAN-images from PGGAN and StarGAN, respectively. In the images tinged with blue, we can observe that activated facial regions are highlighted. Moreover, different levels of intensity are represented via varying highlight colors, where regions highlighted in yellow show the strongest activation, while those highlighted in green and sky blue show weaker activations. Red circled numbers from 1 to 4 point to edge representations, such as eyebrows, glasses, and the jawline.

In this section, we describe the results obtained from Grad-CAM, which visualizes the essential regions from an input image required for the prediction of its class.

Grad-CAM. Grad-CAM is generated based on the gradient between the input image and the predicted class. Using this heat map, we can measure how the layer output affects the prediction by evaluating the pixel values: positive pixels resulting from the convolution and ReLU layers translate to activated regions in Grad-CAM represented by fluorescent color, while negative pixels show no activation in blue color. For this experiment, we utilized EfficientNet-B0 as the base model, pre-trained on the PGGAN dataset, and then transferred to the StarGAN dataset.

Figure 9. Grad-CAM output of a real image with sharp representations. Although both characterized as edge representations, the lips are activated (circled number 2), while the word ”VOGUE” in the foreground and the words in the background are not (circled numbers 1 and 3).

Edge representation in T-GD. Figure 8 (a) and (b) show the Grad-CAMs for the real class, and (c) and (d) show those of the GAN-image class. For the task of GAN-image detection (magenta arrow in Fig. 8), we observe that both Grad-CAMs focus on the edges of the face, indicated by circled numbers. For the task of transfer learning (green arrow in Fig. 8), we observe that similar regions of the face are highlighted, showing distinct activations on the jawline for both the PGGAN and StarGAN datasets as indicated by the circled numbers 3 and 4 in Fig. 8. This implies that the pre-trained model on the source dataset has been successfully transferred to the target dataset.

Facial edge representation with sharp representations. Our experiment shows that T-GD can effectively distinguish facial representations from letters present in the foreground and the background, as indicated by the circled numbers 1 and 3 in Fig. 9, respectively.

D. Dataset Description

We describe three real and four GAN-generated image dataset we used in our experiment.

CelebA. CelebFaces Attributes Dataset (CelebA) (Liu et al., 2015) is a large-scale face attributes dataset with more than 200,000 celebrity images.

CelebA-HQ. CelebA-High-quality (CelebA-HQ) consists of 30,000 images (Liu et al., 2015). This applied various image processing to center the images on the facial region.

FFHQ. Provided by StyleGAN (Karras et al., 2019a), Flickr-Faces-HQ (FFHQ) consists of 70,000 high-quality images crawled from Flickr at a 1024×1024 resolution. The images represent individuals of different ages and ethnicities, contain various backgrounds, and have much better coverage of accessories, such as eyeglasses, sunglasses, and hats, compared to CelebA-HQ.

PGGAN. The key idea of PGGAN (Karras et al., 2017) is to grow the generator and the discriminator progressively. Model training starts at a low resolution, with the addition of layers to increase the spatial resolution of the generated images. For PGGAN-images, we used the official implementation dataset[2] provided by the author, consisting of 100,000 GAN-generated fake celebrity images at a 1024×1024 resolution generated from the CelebA-HQ dataset. For our experiment, we resized each image to a 128×128 resolution.

StarGAN. StarGAN (Choi et al., 2018) is capable of learning mappings among multiple domains using only a single model that can generate image-to-image translated high quality images. For the image generation, we used the official implementation source code and CelebA dataset (Liu et al., 2015) to generated 128×128 resolution GAN-images. We generated StarGAN-images from this model and insure that we follow their official implementation by using their pre-trined model[3]. We generated five attributes GAN-images from one CelebA image: black-hair, blond-hair, brown-hair, male, and young attributes. Then, we randomly chose one of five images as the source dataset.

StyleGAN. StyleGAN (Karras et al., 2019a) architecture leads to an automatically learned, unsupervised separation of high-level attributes and stochastic variation in the generated images, enabling an intuitive and scale-specific control of the synthesis process. For StyleGAN-images, we used the official implementation dataset[4] provided by the author, consisting of 100,000 GAN-generated celebrity images at a 1024×1024 resolution generated from the FFHQ (Karras et al., 2019a) dataset. For our experiment, we resized the image to a 256×256 resolution.

StyleGAN2. StyleGAN2 (Karras et al., 2019b) redesigns the generator normalization, revisits the progressive growing, and regularizes the generator to encourage a good conditioning when mapping latent vectors to images. For StyleGAN2-images, we used the official implementation dataset[5] provided by the author, under the same condition as in StyleGAN (Karras et al., 2019a).

E. Training Details

We implement EfficientNet-B0 (Tan & Le, 2019) and ResNext32×4d (Xie et al., 2017). We change BN to GN and WS for better transferability. For both pre-training teacher models, we use a batch size of 512, stochastic gradient descent (SGD) optimizer with a momentum 0.9, and gradual warm-up start by 4 times for 20 epochs with cosine-annealing. The initial learning rate is 0.04 and epochs are 300. Different data augmentation techniques are applied: JPEG compression (0.2 rate), Gaussian Blur (0.2), intra-class Cutmix (0.2), random horizontal flip (0.2), dropout (0.2), and stochastic depth (0.2). In the stage of transfer learning, we use a batch size of 200, SGD optimizer with a momentum 0.1, and an initial learning rate of 0.01. All augmentation rates are set to 0.5, except for dropout and stochastic depth: JEPG compression (0.5), Gaussian Blur (0.5), intra-class Cutmix (0.5), random horizontal flip (0.5), dropout (0.2), and stochastic depth (0.2). The training is completed at 1000 iterations.

[2] https://github.com/tkarras/progressive_growing_of_gans
[3] https://github.com/yunjey/stargan
[4] https://github.com/NVlabs/stylegan
[5] https://github.com/NVlabs/stylegan2

References

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8789–8797, 2018.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019a.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*, 2019b.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.