Domain General Face Forgery Detection by Learning to Weight

Ke Sun[1], **Hong Liu**[2], **Qixiang Ye**[3], **Yue Gao**[4],
Jianzhuang Liu[5], **Ling Shao**[6], **†Rongrong Ji**[1,7]*
[1]Media Analytics and Computing Lab, Department of Artificial Intelligence,
School of Informatics, Xiamen University, 361005, China
[2]National Institute of Informatics, Japan [3]University of Chinese Academy of Sciences, China
[4]Tsinghua University, China [5]Noah's Ark Lab, Huawei Technologies, China
[6]Inception Institute of Artificial Intelligence, Abu Dhabi, UAE [7] Institute of Artificial Intelligence, Xiamen University

Abstract

In this paper, we propose a domain-general model, termed learning-to-weight (LTW), that guarantees face detection performance across multiple domains, particularly the target domains that are never seen before. However, various face forgery methods cause complex and biased data distributions, making it challenging to detect fake faces in unseen domains. We argue that different faces contribute differently to a detection model trained on multiple domains, making the model likely to fit domain-specific biases. As such, we propose the LTW approach based on the meta-weight learning algorithm, which configures different weights for face images from different domains. The LTW network can balance the model's generalizability across multiple domains. Then, the meta-optimization calibrates the source domain's gradient enabling more discriminative features to be learned. The detection ability of the network is further improved by introducing an intra-class compact loss. Extensive experiments on several commonly used deepfake datasets to demonstrate the effectiveness of our method in detecting synthetic faces. Code and supplemental material are available at *https://github.com/skJack/LTW*.

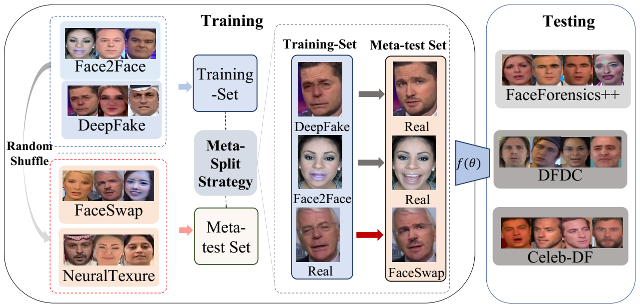

Figure 1. An overview of the proposed method for the general face forgery problem. The left part shows the training process, which is based on a meta-learning framework. This framework randomly splits and shuffles a multi-source domain dataset. During training, the meta-test set is generated based on the meta-split strategy. Then, the meta-optimization tries to learn a domain-invariant model, which can be used in detecting unseen domains.

Introduction

Recent years have witnessed significant advances in face recognition. In particular, technologies such as deep learning have greatly improved the performance of this task. However, through sophisticated manipulation of face images, existing facial recognition systems are at risk of being crippled. For instance, using recent GAN methods (Karras, Laine, and Aila 2019; Goodfellow et al. 2014; Brock, Donahue, and Simonyan 2018), it is possible to generate fake faces that can fool the existing face recognition systems. Therefore, it is crucial to develop approaches that can distinguish between real and fake faces, which has received a lot of attention from the research community.

The current research on face forgery detection mainly consists of face manipulation and detection. The face manipulation focuses on generating fake faces for attacking the face detection system. There are four main facial manipulations including entire face synthesis, identity swapping, attribute manipulation, and expression swapping (Tolosana et al. 2020; Mirsky and Lee 2020). Among these, identity swapping (also known as deepfake) is currently most focused on by forgery methods. As such, several relevant public benchmarks, such as FaceForensics++ (Rossler et al. 2019), Celeb-DF (Li et al. 2019), and DFDC (Dolhansky et al. 2020), have been released to help develop and verify reliable fake face detection algorithms.

However, the fake faces in the above mentioned datasets were crafted in a relatively homogeneous manner. Thus, the data distributions and face identities are roughly the same for the training and test sets. In contrast, in real-world applications, a model trained on a given training set (*source domain*) is always used on a very different test set (*target domain*), which is often previously unseen.

While domain adaptation may be a solution, adapting the knowledge learned from a source domain to target ones, it is still unable to handle unseen target domains due to lack of unseen training data. Therefore, generalized face forgery detection is less studied and more challenging because attackers can use unseen face manipulations to attack the existing face recognition system. In this paper, we argue that domain

*Corresponding author: rrji@xmu.edu.cn.

generalization needs to be considered. Similar to (Guo et al. 2020), we call this scenario *general face forgery detection.*

In this paper, we focus on building a more effective model for general face forgery detection. Before briefly describing our method, we first analyze some of the properties of this task. First, the fake facial images are always synthesized through a generative adversarial network. These face images are crafted by different methods and contain multiple biases because of the generator's inherent bias. Such virtual and biased properties make the data distribution of the source domain more complex, leading to large semantic gaps between source domain data.

Secondly, due to the different generators, the quality of each fake face varies greatly. Some samples may have domain-invariant features, while others have domain-biased features. Therefore, directly training on samples with equal weights will corrupt the generalization ability of a detection model.

To handle the above mentioned problems, this paper proposes a novel algorithm for general face forgery detection, which is based on a newly defined learning-to-weight (LTW) framework. The basic idea of LTW containing two branches is simple but effective. The first branch is the basic binary detection model, which extracts the features of each image and determines whether the input image is real or fake. The second branch is a weight-aware network that predicts the domain-adaptive weight score of each image in the training batch. Then, we use the meta-learning framework to combine these two branches together, which not only learns the parameters of the weight-aware network but also calibrates the source/target domain shift through the meta-optimization. Since the face forgery detection is a binary classification problem, we further propose a new regularization loss, termed Intra-Class Compact (ICC) Loss . This regularization aims to aggregate all real samples into a subset and then push such fake subset away from the real one, which helps to improve the performance of detection. Finally, we construct five benchmarks to validate our model and compare it with related methods (Nguyen et al. 2019; Cozzolino et al. 2018; Li et al. 2018). Our experiments show that the proposed LTW outperforms the state-of-the-arts, with a 2% improvement on average over the best compared method.

Related Work

Face Forgery Detection

In early research, the change in a face image after JPEG compression was used to determine its authenticity (Agarwal and Farid 2017). McCloskey and Albright (2018) extracted simple RGB features and then used the SVM classifier to determine whether a face is true or false. To further improve the performance, Matern, Riess, and Stamminger (2019) studied the properties of the visual artifacts and proposed a simple but effective pipeline that achieves surprisingly good results.

However, these methods are based on hand-crafted features and are not very robust or sensitive to the quality of training data. To overcome these issues, recent research has focused on deeply supervised learning methods. The simplest approach is to use a convolutional neural network (CNN) for binary classification (Tan and Le 2019; Chollet 2017). This kind of method is very effective and is widely used in DFDC competitions (Dolhansky et al. 2020). Besides, several other characteristic methods have been proposed. For instance, a two-stream tampered face detection model was proposed in (Zhou et al. 2017), where one stream detect low-level inconsistencies between image patches and the other explicitly detects tampered faces. Afchar et al. (2018) proposed MesoNet, which uses a small number of layers to focus on the mesoscopic properties of face images. Moreover, some works (Nguyen et al. 2019; Cozzolino et al. 2018; Li et al. 2020) use generative models to reconstruct the face images, thereby reducing the noise introduced by face manipulation and improving detection performance.

Domain Generalization

Recent deep models face a common problem that there is a large gap between the distributions of training and test data, which greatly affects model performance. Therefore, domain generalization has been widely studied, and has become an important problem.

Muandet, Balduzzi, and Schölkopf (2013) proposed a domain-invariant component analysis method that learns an invariant transformation by minimizing the dissimilarity across different domains. Another direction is to use a generative model (*i.e.,* auto-encoder) to learn a latent embedding and map both the source domain and the target domain to the same space (Glorot, Bordes, and Bengio 2011; Chen et al. 2012). MLDG (Li et al. 2018) was the first to use meta learning for domain generalization while Meta-face (Guo et al. 2020) later applied it to the open set of face recognition. Different from these, our LTW method not only considers the domain shift but also the sample diversity.

Meta Learning

Meta learning, also known as *learning to learn*, aims to learn certain parameters or strategies for few-shot learning.

The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new tasks using only a small number of training samples (Finn, Abbeel, and Levine 2017; Nichol, Achiam, and Schulman 2018). Recently, some works have used a meta-learning framework to solve domain generalization (Vinyals et al. 2016; Qiao et al. 2018). Our main difference from them is that we use the meta-learning to resolve the new problem of general face forgery detection with our LTW framework.

Proposed Method

Overall Framework

We first introduce the overall framework of the proposed learning to weight (LTW) algorithm, based on meta learning. It aims to improve the generalization of the detector so that fake faces in unseen domains can also be recognized. Therefore, we define the basic detection model as $f(\theta)$, where θ is the parameter set of a neural network. In the training stage, we select N source domain D^s_{train} =

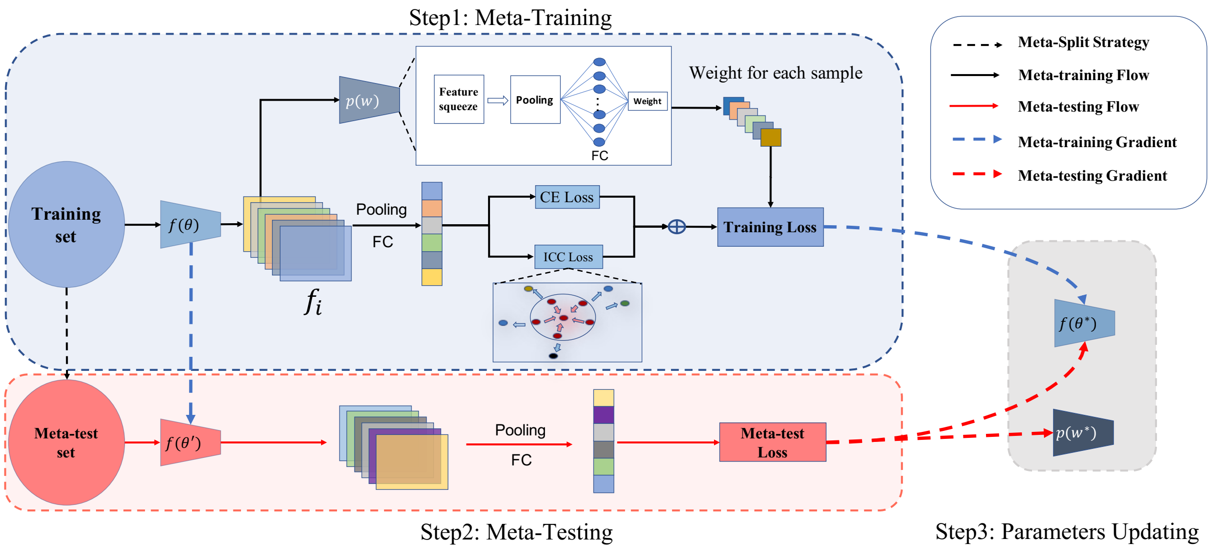

Figure 2. Overview of our proposed method. The black arrow means the meta-training flow, while the orange arrow represents the meta-testing flow. The blue dotted arrow means the gradient of the training loss, which updated for the θ and the red dotted arrow represents the gradient of meta-test loss which to update w and θ. Note that the training loss consists of the CE loss and ICC loss.

$\{d^s_{d1}, d^s_{d2}, ...d^s_{dN}\}$ for training, where d^s_{di} represent the ith subset based on the attack method. Then, the trained model $f(\theta)$ is evaluated on the M unseen target domain test set $D^t_{test} = \{d^t_{d1}, d^t_{d2}, ...d^t_{dM}\}$, as shown in Fig. 1. As a result, our goal is to train the detection model $f(\theta)$ on the source domain D^s_{train}, which can obtain better performance on the unseen test domain D^t_{test} without any model updating.

Specifically, the LTW module has two branches. The first is the binary classification neural network $f(\theta)$, which aims to extract features and determine the authenticity of each face. The other branch is a weight-aware network $p(w)$ that is dependent on the latent feature in $f(\theta)$. The whole module is shown in the upper box of Fig. 2. This module can assign domain-adaptive weights to each sample, and it helps the basic model $f(\theta)$ mine for more domain-general features. Note that during inference, only trained $f(\theta)$ is needed. Therefore, the key is to design the weight-aware network, the architecture of which is shown in Fig. 2.

Different from (Shu et al. 2019), our weight-aware network takes a feature map f_i, outputed by the last convolutional layer in $f(\theta)$, as input. We define f_i as the latent feature extracted from a given input image x_i. To avoid adding too many parameters and reducing the inference speed, $p(w)$ is designed to be very small, containing only 1.024M parameters. In detail, we use two depth-wise separable convolutional layers to squeeze the number of feature map channels. Then, this is followed by a fully connected layer and the predicted scores are outputted for the input samples. To achieve better results, we use the Sigmoid function to normalize scores to $[0, 1]$. In what follows, we use $p(f_i; w)$ to denote the weight of sample x_i, with w being the parameters of the weight-aware network.

In order to update the parameters of $p(w)$ and calibrate the domain gradient of $f(\theta)$, we embed the LTW framework into a meta-learning strategy (Finn, Abbeel, and Levine 2017; Shu et al. 2019). This framework contains two advantages. 1) The basic model $f(\theta)$ can avoid overfitting to each specific domain. 2) The weight-aware network $p(w)$ can be optimized by meta-gradient to measure each sample's domain generalization.

Meta-Split Strategy To simulate the domain shift, we divide the source domains D^s_{train} into training domains D^s_{nt} and meta-domains D^s_{meta} during each training epoch. To avoid overfitting to particular data distribution, we use a random selection scheme to shuffle the source domains. Specifically, we randomly split N source domains D^s_{train} into two subsets D^s_{nt} and D^s_{meta}, where D^s_{nt} has N/2 domain sets used to train the detector, and D^s_{meta} contains the rest to assist in the model training. Note that our split scheme guarantees that there is no domain overlap between them. During training, for each epoch, a sample batch is sampled from D^s_{nt} named training set. Then a meta-test set is formed in this way: For a fake (real) face in the batch, find its corresponding real (fake) face from D^s_{meta}. This strategy enables the model to learn pair information between different domains. Fig. 1 shows this meta-split strategy.

Learning Process

We divide our whole learning process into three steps: Meta-Training, Meta-Testing and Parameters Updating, which are detailed in the next subsections and also shown in Fig 2.

Step 1: Meta-Training This stage is used to calculate the training set's loss based on the binary detection model $f(\theta)$,

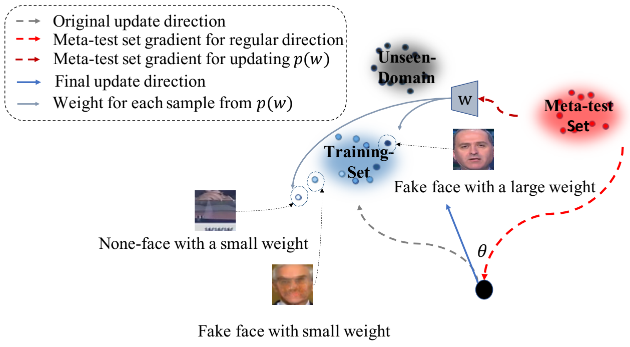
Original update direction
Meta-test set gradient for regular direction
Meta-test set gradient for updating $p(w)$
Final update direction
Weight for each sample from $p(w)$
Unseen-Domain
w
Meta-test Set
Training-Set
Fake face with a large weight
None-face with a small weight
θ
Fake face with small weight

Figure 3. Visualization of our method. The grey arrow represents the original direction to update θ, which lacks generalization for the unseen domains. The red arrows indicate the meta gradient flow used for updating the w and correcting the original direction. The weight-aware network is responsible for giving each sample a weight . For example, it gives a wrongly detected face or low quality face a small weight and gives a relatively high quality sample a large weight. The blue arrow denotes the final direction of updating θ. In the training set, a deeper blue dot means a face with a larger weight.

with the help of the weight-aware module. Specially, with the meta-split strategy, we sample K training data from D^s_{nt} denoted as $X_s = \{x_i, y_i\}^K_{i=1}$. Generally, the model parameters θ can be obtained by minimizing a loss function L. In contrast, we add a domain-adapted weight $p(f_i; w)$ when calculating the final loss. Specifically, the training loss $T(\theta, w)$ is formulated as:

$$T(\theta, w) = \frac{1}{K}\sum_{i=1}^{K} L((x_i, y_i); \theta) * p(f_i; w) \qquad (1)$$

where L is our loss function defined later.

Step 2: Meta-Testing After meta-training, we obtain the weighted loss T on the training-set, the next challenge is how to update the parameters W of the feature weight net and make full use of the current unseen domain data D^s_{meta} to improve the model's generalization. Inspired by (Li et al. 2018), we use the second derivative of the model parameters. In order to do that, first we need to virtually update θ using gradient descent:

$$\theta^{'} = \theta - \alpha \nabla_\theta T(\theta, w) \qquad (2)$$

where α is the learning rate of meta-training stage. Then, the model with its virtual parameters $\theta^{'}$ is tested on the $X_m = \{x^{'}_i, y^{'}_i\}^K_{i=1}$ generated by the meta-split strategy. By evaluating on it, which is the relatively unseen domain for the current epoch, model $f(\theta)$ can learn general features across different domains. The meta-test loss is defined as follows:

$$M(\theta^{'}, w) = \frac{1}{K}\sum_{i=1}^{K} L((x^{'}_i, y^{'}_i); \theta^{'}, w), \qquad (3)$$

Step 3: Parameters Updating Since the meta-training stage calculates the weighted loss T on the training-set and the meta-testing stage calculates the loss M on the meta-test set, we introduce our parameter updating method in this stage. We define the final objective as:

$$\arg\min_{\theta, w} T(\theta, w) + \beta M(\theta^{'}, w), \qquad (4)$$

where β weights the importance of the meta-loss. By minimizing the Eq.(4), the parameters θ and w can be updated as follows, using gradient descent.

$$\theta^{*} = \underbrace{\theta - \alpha\nabla_\theta T(\theta, w)}_{meta-train\ update} + \underbrace{\beta(\theta^{'} - \gamma\nabla_{\theta^{'}} M(\theta^{'}, w))}_{meta-test\ update} \qquad (5)$$

$$w^{*} = w - \phi\nabla_w M(\theta^{'}, w) \qquad (6)$$

Hyperparameters γ and ϕ are the learning rates for the update of θ and w, respectiely.

Analysis The meta-testing stage is equivalent to calculating the second-order derivative of the meta-test set. As shown in Fig. 3, the gradient of meta-testing is used to update the parameters of w and calibrate the gradient of training loss. On the one hand, if the learning gradient of a training sample contributes to the decline of the meta-test loss $M(\theta^{'}, w)$, it is considered as beneficial for the domain generalization and the weight-aware network tend to give it a higher weight. On the other hand, the second-order derivative of the meta-testing step guides and modifies the direction of the parameters θ update, so that the model can address unseen domains.

As noted in (Guo et al. 2020), optimizing the parameters by Eqs.(5) and (6) means that we force the $f(\theta)$ and $p(w)$ to perform well on training set and meta-test set.

Loss Function

Because of the variety of attacks, the model may not be able to cover all types of forged faces, but the distribution of the real faces is relatively stable. To take advantage of this, we first consider this problem as a form of one-class classification. Inspired by (Perera and Patel 2019; Kuang et al. 2019), we propose a novel loss function called the Intra-Class Compact (ICC) Loss. The basic idea is to make positive samples gather and push the negative samples away from the positive center. Specifically, define $O = \{o_1, o_2, ..., o_n\} \in R^{n\times 1}$ to be the output of the model $f(\theta)$ after pooling and a fully-connected layer, as shown in Fig. 2, with label $Y = \{y_1, y_2, ...y_n\}, y_i \in \{(0, 1)\}$, where 0 indicates a real sample and 1 is a fake sample. We choose positive and negative sample sets O^{real} and O^{fake} according to Y.

The distance between the positive samples and their real center is defined as

$$L_{positive} = \frac{1}{|O^{real}|}\sum_{j=1}^{|O^{real}|}(o^{real}_j - C_{real})^2, \qquad (7)$$

where

$$C_{real} = \frac{1}{N}\sum_{j=1}^{N} o^{real}_j. \qquad (8)$$

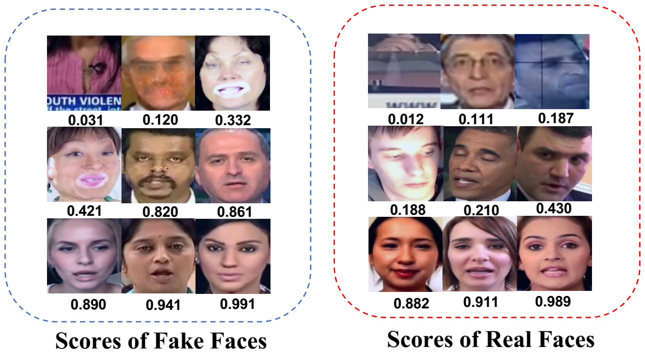

Figure 4. Samples with their weights predicted by the weight-aware network. The left shows the fake face weights, while the right shows the real face.

The distance between the negative samples and C_{real} is:

$$L_{negative} = \frac{1}{|O^{fake}|} \sum_{j=1}^{|O^{fake}|} (o_j^{fake} - C_{real})^2. \quad (9)$$

Then the ICC loss L_{icc} is

$$L_{icc} = L_{positive} - L_{negative}. \quad (10)$$

Note that C_{real} is updated in each mini-batch. The intra-class compact loss gathers the positive samples together while keeping all types of negative samples away from the positive center. By doing this, the model is not only forced to dig more discriminative features but also improved in generalization because the unseen attack will be pushed away from the real center. Finally the overall loss during the meta-training period is:

$$L = L_{ce} + \lambda L_{icc}, \quad (11)$$

where L_{ce} is the binary cross entropy loss, and λ is the weighting parameter to balance the importance of L_{icc}.

Experiments

Evaluation Settings

To evaluate the capability of our proposed method, we build different benchmarks based on three popular deepfake databases FaceForensics++ (Rossler et al. 2019), Celeb-DF (Li et al. 2019), and DFDC (Li et al. 2019).

Evaluation Benchmarks There are four different approaches in the FaceForensics++ database, including two computer graphics methods (Face2Face and FaceSwap) and two learning-based approaches (DeepFakes and NeuralTextures). The results obtained by these attack methods are quite different. As such, we use these different generation methods as the basis for splitting the source and target domains. Besides, to further evaluate our method, we also consider the quality of videos, testing on both higher quality (quantization parameter equal to 23) and lower quality images (quantization 40). The specific content of each benchmark is shown in Tab. 1.

Name	Compression	Source domains	Target domain(s)
GID-DF23/40	C23/40	Face2Face FaceSwap NeuralTextures	DeepFake
GID-F2F23/40	C23/40	DeepFake FaceSwap NeuralTextures	Face2Face
GID-FS23/40	C23/40	DeepFake Face2Face NeuralTextures	FaceSwap
GID-NT23/40	C23/40	DeepFake Face2Face FaceSwap	NeuralTextures
GCD	C23	DeepFake Face2Face FaceSwap NeuralTextures	Celeb-DF DFDC DeepFake Face2Face FaceSwap NeuralTextures

Table 1. Summary of the five benchmarks we design. The number 23 or 40 under "compression" means the quantization parameter. Source domains are used for training and target domains for evaluation. GID: general intra-datasets. GCD: general cross datasets.

Evaluation Metrics To fully evaluate the quality of a model, we apply accuracy score (ACC), log loss, area under the receiver operating characteristic curve (AUC) and equal error rate (EER) as our evaluation metrics. Specifically, Log loss is used in the DFDC competition, and is formulated by:

$$LogLoss = -\frac{1}{n} \sum_{i=1}^{n} y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i}), \quad (12)$$

where n is the number of images being predicted, $\hat{y_i}$ is the predicted probability of the ith image being fake, y_i is the label of the sample, and $log()$ is the natural (base e) logarithm.

Training Settings For each benchmark, we use FaceForensics++ for training and use the multitask cascaded CNNs to extract faces, where the margin is set to 16 (**?**). We follow the official division of the dataset, in which 720 videos are used for training, 140 videos for validation and 140 videos for testing. As for the testing of GCD-23, we choose 518 test videos from Celeb-DF and use the last 200 real videos of the DFDC data set with its corresponding fake videos to form the Celeb-DF and DFDC test sets.

In order to simulate the lack of data in the practical scenario, different from (Rossler et al. 2019; Qian et al. 2020) that used 270 frames, we only use 10 frames for training and testing.

Implementation Details

Each input face is resized to 224×224. In order to clearly reflect the capability of the model, we remove all additional tricks that may improve generalization, such as data augmentation (great impact on model generalization), early stop, and so on. To reduce the randomness of a mini-batch and initial values of the parameters, all the methods share the same random seeds.

Method	GID-DF23				GID-DF40				GID-F2F23				GID-F2F40			
Metric	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER
Basemodel	0.495	0.697	0.485	0.512	0.485	0.699	0.451	0.531	0.523	0.691	0.556	0.461	0.505	0.694	0.537	0.471
Alltrain-Basemodel	0.824	0.862	0.911	0.170	0.676	1.995	0.753	0.32	0.633	1.755	0.801	0.268	0.614	1.812	0.674	0.384
FocalLoss-Basemodel	0.813	**0.466**	0.903	0.177	0.674	0.876	0.749	0.306	0.608	**0.760**	0.798	0.273	0.610	**0.761**	0.672	0.383
ForensicTransfer	0.720	-	-	0.331	0.682	-	-	0.333	0.645	-	-	0.385	0.550	-	-	0.452
Multi-task	0.703	-	-	0.374	0.667	-	-	0.351	0.587	-	-	0.401	0.565	-	-	0.440
MLDG	0.842	0.760	0.918	0.152	0.671	0.952	0.730	0.329	0.634	1.907	0.771	0.304	0.581	2.434	0.617	0.419
Ours	**0.856**	0.792	**0.927**	**0.145**	**0.691**	**0.715**	**0.756**	**0.305**	**0.656**	1.422	**0.802**	**0.271**	**0.657**	1.025	**0.724**	**0.331**

Table 2. Performance on the GID-DF23/40 and GID-F2F23/40 benchmarks. The highest results are highlighted in bold.

Method	GID-FS23				GID-FS40				GID-NT23				GID-NT40			
Metric	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER
Basemodel	0.517	0.694	0.517	0.488	0.503	0.695	0.512	0.489	0.500	0.696	0.493	0.505	0.497	0.696	0.499	0.499
Alltrain-Basemodel	0.500	4.156	0.543	0.470	0.580	3.257	0.614	0.425	0.608	1.926	0.774	0.291	0.564	2.748	0.600	0.434
FocalLoss-Basemodel	0.484	2.241	0.503	0.498	0.575	1.465	0.596	0.429	0.604	**1.125**	0.759	0.310	0.566	0.748	0.605	0.426
ForensicTransfer	0.460	-	-	0.523	0.530	-	-	0.471	0.569	-	-	0.457	0.550	-	-	0.453
Multi-task	0.497	-	-	0.495	0.517	-	-	0.484	0.603	-	-	0.402	0.560	-	-	0.446
MLDG	0.527	1.562	0.609	0.431	0.581	2.434	0.617	0.419	0.621	1.716	0.78	**0.290**	0.569	2.733	0.607	0.423
Ours	**0.549**	**1.233**	**0.64**	**0.397**	**0.625**	**1.179**	**0.681**	**0.364**	**0.653**	1.561	**0.773**	0.294	**0.585**	**1.763**	**0.608**	**0.415**

Table 3. Comparative results on GID-FS23/40 and GID-NT23/40.

Unless otherwise stated, we use the following setting. An EfficientNet-b0 is used as our backbone with only 5.3M and 0.39B FLOPS, which was proven effective in the Deepfake Detection Challenge. The model is pre-trained on the ImageNet (Deng et al. 2009). The learning rate α for meta-training and γ for meta-testing is both 0.001 with Adam optimizer. We use a stepLR scheduler, where the step-size is 5 and gamma is set to 0.1. The weight-aware network update learning rate ϕ is 0.001. The hyperparameter β which balances the meta-training and meta-testing is set to 1. And the hyperparameter λ to balance the CE loss and the ICC loss is set to 0.01. The batch size is 25. To evaluate the effectiveness of the model, we compare it with six baselines: **(1) Basemodel**: The model pre-trained on ImageNet without any fine-tuning on a forged face dataset. This is the simplest baseline without any discrimination ability. **(2) Alltrain-Basemodel**: The Basemodel trained on all source domains. This method can be used as the fairest and powerful baseline. **(3) FocalLoss-Basemodel**: To compare the same weighting method, we use the focal loss in Alltrain-Basemodel. **(4) ForensicTransfer** (Cozzolino et al. 2018): This method is the first to highlight the generalized forgery face detection problem. We reproduce it and run it on our benchmarks. **(5) Multi-task** (Nguyen et al. 2019): We run its official code. Note that when training ForensicTransfer and Multi-task, their hyperparameters are adjusted according to their papers. **(6) MLDG** (Li et al. 2018): This method use meta-learning to solve the domain generalization problem. We adapt it for the generalized forged face detection problem.

Evaluation Results

Results on GID In Tab. 2 and Tab. 3, our method is compared to the baselines on the GID benchmarks with images of different qualities. We can observe that overall our method achieves the best results in all four metrics. Specifically on GID-F2F40, GID-FS40, GID-NT23, our method achieves an average improvement of 5% compared to the baselines, without increasing any model parameters. FocalLoss-Basemodel achieves good results in reducing log loss, but its accuracy is not as good as our method. In most cases, MLDG is better than Alltrain-Basemodel, which also shows that meta-learning does improve the model's generalization ability. And our model is on average about 3% higher than MLDG on ACC.

Results on GCD Tab. 4 shows the results on the GCD benchmark. On this benchmark, we not only aim to achieve good results on the test set corresponding to the target domains but also focus on the performance across the datasets. Tab. 4 provides the results, where GCD-OthersAVG means the average performance on the test sets of DeepFake, Face2Face, FaceSwap, and NeuralTextures. We can conclude that our method performs well on both cross datasets and intra-datasets compared with the baselines. We achieve great performance improvements from 60.9% to 63.4% on the CeleDF testset and also get 1.5% higher on ACC than the best baseline on DFDC. These observations show that our method performs well not only on the source domains but also on the target domains, indicating its generalization ability.

Visualization of the Weights

In this section, we visualize images with their weights generated by the weight-aware network. We sample some representative images from the FaceForensics++ dataset. The results are shown in Fig. 4. We can clearly see that, for fake images, smaller weights are given to the obviously failed generated faces, wrongly detected faces and inferior quality faces, while higher quality faces are given larger weights. For real faces, very noisy or blurry and those under extreme

Method	GCD-CeleDF				GCD-DFDC				GCD-OthersAVG			
Metric	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER	ACC	LOSS	AUC	EER
Basemodel	0.590	0.679	0.501	0.488	0.514	0.694	0.525	0.483	0.509	0.695	0.513	0.492
Alltrain-Basemodel	0.609	3.176	0.624	0.407	0.615	2.006	0.685	0.374	0.914	0.339	0.981	0.060
FocalLoss-Basemodel	0.606	1.162	0.616	0.413	0.617	**0.787**	0.669	0.378	0.905	**0.240**	0.982	0.057
ForensicTransfer	0.620	-	-	0.204	0.540	-	-	0.464	0.766	-	-	0.297
Multi-task	0.584	-	-	0.511	0.511	-	-	0.494	0.767	-	-	0.285
MLDG	0.595	**1.691**	0.609	0.418	0.607	1.334	0.682	0.370	0.918	0.247	0.978	0.070
Ours	**0.634**	2.506	**0.641**	**0.397**	**0.631**	1.807	**0.690**	**0.368**	**0.938**	0.246	**0.985**	**0.048**

Table 4. Performance comparison on the GCD benchmarks. GCD-OthersAVG represents the average performance on the test sets of the four source domains: DeepFake, FaceSwap, Face2Face, and NeuralTextures.

GID-DF23 Ablation Study				
	ACC	LOSS	AUC	EER
Alltrain-Basemodel	0.824	0.862	0.911	0.17
w/o ICC Loss	0.844	1.032	0.914	0.154
w/o Weight-aware	0.837	0.817	0.917	0.160
w/o Meta-testing	0.840	0.830	0.919	0.158
Ours-full	**0.856**	**0.792**	**0.927**	**0.145**

Table 5. Ablation study on the GID-DF23 benchmark.

GCD-AVG				
BackBone	ACC	LOSS	AUC	EER
Efficientnet-b2	0.811	1.179	0.872	0.166
Efficientnet-b2+Ours	**0.826**	**0.876**	**0.875**	**0.163**
Efficientnet-b4	0.823	1.030	0.874	0.168
Efficientnet-b4+Ours	**0.830**	**0.871**	**0.873**	**0.166**
Xception	0.797	1.504	0.857	0.179
Xception+Ours	**0.820**	**0.925**	**0.879**	**0.171**
Resnet50SE	0.783	**0.954**	0.845	0.203
Resnet50SE+Ours	**0.802**	0.999	**0.863**	**0.181**
VGG19BN	0.705	0.728	0.783	0.283
VGG19BN+Ours	**0.731**	**0.716**	**0.807**	**0.260**

Table 6. Performance comparison of five backbone architectures with and without our method.

light conditions images tend to obtain smaller weights than normal face images. We argue that high quality faces usually contain general features that have a great contribution to the model generalization.

Ablation Study

Effectiveness of Each Component To evaluate the contributions of different components, we compare our full method with its three separate ablated versions on the GID-DF23 benchmark. The quantitative results are shown in Tab. 5. Removing the weight-aware network is equivalent to setting all $p(f_i, w) = 1$ in Eq.(1). Our model without meta-testing is such one where first we train the whole LTW framework, and then retrain it with $\beta = 0$ and fix the pre-trained weight-aware network. Obviously, it can be seen that the ICC loss, weight-aware network, and meta-testing are all effective, because the performance drops when any of them are removed. Specifically, the weight-aware network and meta-testing is more important. The performance drops nearly 2% and 1.6% when removing those two components,respectively.

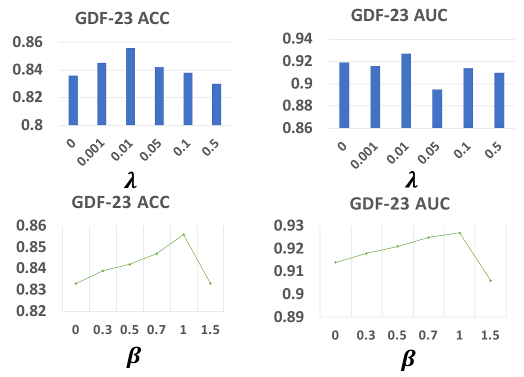

Figure 5. Results with different λ and β on GDF-23.

Dependency on Backbone We hope that our method is model-independent, which means it can be used in any related deep models. As such, we verify the effect of using different backbones. We test on GCD and use the average of all test set results as the evaluation criterion. We simply adjust the input dimension of the weight-aware network to be consistent with the backbone's feature extract module. We can observe from Tab. 6 that our method does improves the network performance regardless of the types of backbones.

Impact of λ and β λ is a hyperparameter weighting the CE loss and ICC loss, while β weights the training loss and meta-test loss. The results are shown in Fig. 5, we test it on the GID-DF23 and show the ACC and AUC with different hyperparameters values. The best value of λ is 0.01. A proper value 1 for β gives the best result, which means that the meta-training and the meta-testing are equally updated.

Conclusion

In this work, we consider the generalization of the forgery face detection problem and propose a novel framework, named Learning to Weight (LTW), to address it. We build our method based on the meta-learning strategy. To the best of our knowledge, we are the first to consider generalization from the perspective of sample differences in this problem.

Acknowledgments

This work is supported by the National Science Fund for Distinguished Young (No.62025603), the National Natural Science Foundation of China (No.62025603, No. U1705262, No. 62072386, No. 62072387, No. 62072389, No. 62002305, No.61772443, No. 61802324 and No. 61702136) and Guangdong Basic and Applied Basic Research Foundation (No.2019B1515120049).

References

Afchar, D.; Nozick, V.; Yamagishi, J.; and Echizen, I. 2018. Mesonet: a compact facial video forgery detection network. In *WIFS*, 1–7. IEEE.

Agarwal, S.; and Farid, H. 2017. Photo forensics from JPEG dimples. In *WIFS*, 1–6. IEEE.

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* .

Chen, M.; Xu, Z.; Weinberger, K.; and Sha, F. 2012. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683* .

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 1251–1258.

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

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255. Ieee.

Dolhansky, B.; Bitton, J.; Pflaum, B.; Lu, J.; Howes, R.; Wang, M.; and Ferrer, C. C. 2020. The DeepFake Detection Challenge Dataset. *arXiv preprint arXiv:2006.07397* .

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NeurlPS*, 2672–2680.

Guo, J.; Zhu, X.; Zhao, C.; Cao, D.; Lei, Z.; and Li, S. Z. 2020. Learning meta face recognition in unseen domains. In *CVPR*, 6163–6172.

Karras, T.; Laine, S.; and Aila, T. 2019. A style-based generator architecture for generative adversarial networks. In *CVPR*, 4401–4410.

Kuang, H.; Ji, R.; Liu, H.; Zhang, S.; Sun, X.; Huang, F.; and Zhang, B. 2019. Multi-modal Multi-layer Fusion Network with Average Binary Center Loss for Face Anti-spoofing. In *ACM MM*, 48–56.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2018. Learning to generalize: Meta-learning for domain generalization. In *AAAI*.

Li, L.; Bao, J.; Zhang, T.; Yang, H.; Chen, D.; Wen, F.; and Guo, B. 2020. Face x-ray for more general face forgery detection. In *CVPR*, 5001–5010.

Li, Y.; Yang, X.; Sun, P.; Qi, H.; and Lyu, S. 2019. Celeb-df: A new dataset for deepfake forensics. *arXiv preprint arXiv:1909.12962* .

Matern, F.; Riess, C.; and Stamminger, M. 2019. Exploiting visual artifacts to expose deepfakes and face manipulations. In *WACVW*, 83–92. IEEE.

McCloskey, S.; and Albright, M. 2018. Detecting gan-generated imagery using color cues. *arXiv preprint arXiv:1812.08247* .

Mirsky, Y.; and Lee, W. 2020. The Creation and Detection of Deepfakes: A Survey. *arXiv preprint arXiv:2004.11138* .

Muandet, K.; Balduzzi, D.; and Schölkopf, B. 2013. Domain generalization via invariant feature representation. In *ICML*, 10–18.

Nguyen, H. H.; Fang, F.; Yamagishi, J.; and Echizen, I. 2019. Multi-task learning for detecting and segmenting manipulated facial images and videos. *arXiv preprint arXiv:1906.06876* .

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* .

Perera, P.; and Patel, V. M. 2019. Learning deep features for one-class classification. *IEEE Transactions on Image Processing* 28(11): 5450–5463.

Qian, Y.; Yin, G.; Sheng, L.; Chen, Z.; and Shao, J. 2020. Thinking in Frequency: Face Forgery Detection by Mining Frequency-aware Clues. In *ECCV*, 86–103. Springer.

Qiao, S.; Liu, C.; Shen, W.; and Yuille, A. L. 2018. Few-shot image recognition by predicting parameters from activations. In *CVPR*, 7229–7238.

Rossler, A.; Cozzolino, D.; Verdoliva, L.; Riess, C.; Thies, J.; and Nießner, M. 2019. Faceforensics++: Learning to detect manipulated facial images. In *ICCV*, 1–11.

Shu, J.; Xie, Q.; Yi, L.; Zhao, Q.; Zhou, S.; Xu, Z.; and Meng, D. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurlPS*, 1919–1930.

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

Tolosana, R.; Vera-Rodriguez, R.; Fierrez, J.; Morales, A.; and Ortega-Garcia, J. 2020. Deepfakes and beyond: A survey of face manipulation and fake detection. *arXiv preprint arXiv:2001.00179* .

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *NeurIPS*, 3630–3638.

Zhou, P.; Han, X.; Morariu, V. I.; and Davis, L. S. 2017. Two-stream neural networks for tampered face detection. In *CVPRW*, 1831–1839. IEEE.