OST: Improving Generalization of DeepFake Detection via One-Shot Test-Time Training

Anonymous Author(s)
Affiliation
Address
email

Abstract

State-of-the-art deepfake detectors perform well in identifying forgeries when they are evaluated on a test set similar to the training set, but struggle to maintain good performance when the test forgeries exhibit different characteristics from the training images, e.g., forgeries are created by unseen deepfake methods. Such a weak generalization capability hinders the applicability of current deepfake detectors. In this paper, we introduce a new learning paradigm specially designed for the generalizable deepfake detection task. Our key idea is to construct a test-sample-specific auxiliary task to update the model before applying it to the sample. Specifically, we synthesize pseudo-training samples from each test image and create a test-time training objective to update the model. Moreover, we propose to leverage meta-learning to ensure that a fast single-step test-time gradient descent, dubbed one-shot test-time training (OST), can be sufficient for good deepfake detection performance. Extensive results across several benchmark datasets demonstrate that our approach performs favorably against existing arts in terms of generalization to unseen data and robustness to different post-processing steps.

1 Introduction

Deep neural network models have brought remarkable advances to image editing and generation techniques. While such progress is revolutionizing the multimedia production industry, it also creates negative social impacts since it has never been easier to create highly deceivable forgery images. Among those new technologies, deepfake, which uses deep learning models to substitute the identity of one person with another or alter the facial features in a portrait, is particularly harmful since it can lead to severe digital crime and undermine the social trust system. To counteract such a negative impact, deepfake detection technique is developed to automatically recognize pristine or forgery and is receiving increasing attention in the research community [23, 30, 52, 18, 37, 4, 34, 53, 12, 19, 17].

So far, existing deepfake detection methods achieve favorable performances when training and test forgeries are from the same dataset and generated by the same deepfake method. In practice, the test forgeries are usually generated by unknown methods or applied with different image postprocessing approaches. This discrepancy will inevitably create a distribution drift between training and test data. Unfortunately, existing deepfake detectors do not generalize well, and their performance tends to decrease significantly when evaluated across datasets. This phenomenon inspires recent studies [23, 37, 30, 52, 34] on improving model generalizations to recognize face forgeries generated from unseen methods. For example, a two heads network is proposed in [23] to discriminate face forgeries by amplifying blending artifacts. In [37], a face forgery frequency network is developed to mine forgery patterns in frequency domains for better generalization. Detail discrepancies are investigated in [30], and facial source feature inconsistencies are discovered in [52] to detect face forgeries, respectively. These methods try to explore common features among the training forgeries for better

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

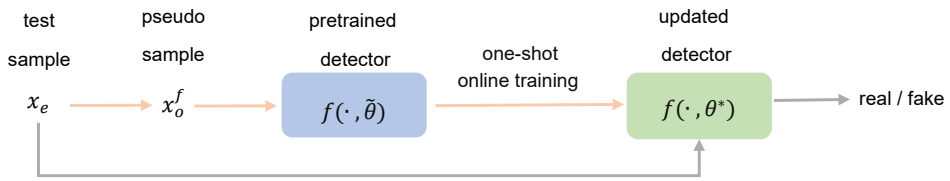


Figure 1: A glimpse of the proposed one-shot test-time training framework during online predictions. For every test sample x_e, we first synthesize a pseudo-training sample x_o^f based on x_e. Then the pretrained detector can be updated via a supervised learning step with x_o^f, *i.e.* x_o^f is with a known label as fake. The final result is obtained by applying the updated detector to the test sample.

classification. But the test data often exhibit different characteristics, and the learned common features may not be shared by them. Since the test samples are not seen in the training stages, obtaining good generalization seems unreachable for current detectors.

This work introduces a new learning paradigm specially-designed for the generalizable deepfake detection task. Specifically, we allow the detector to "see" the test samples before making the final prediction by conducting an additional "training step" at the test time. One challenge of this idea is that the label of the test image is unavailable for the training objective. We overcome this problem by synthesizing a pseudo-training sample based on the test image and using it to update the deepfake detection model online. In a common deepfake detection setting [23, 52], no matter real or forgery for the evaluated test sample, we are confident that the synthesized pseudo sample is a forgery. This unique property enables our detector to train on the synthesized sample that has similar content to the test sample, thus better adapting to the test characteristic.

Moreover, we propose to use only one-step gradient descent, dubbed one-shot test-time training (OST), to update the model online for better computational efficiency. A glimpse of OST is shown in Figure 1. To ensure that such an OST scheme can always lead to a decent model without overfitting the pseudo-training sample, we employ meta-learning to train a good initial model in a similar style as MAML [14]. Note that although test-time training (TTT) has been previously proposed for the general image classification task [41], our approach adopts a different TTT objective that is more specific for deepfake detection. Meanwhile, as both suggested by a recent work [29] and our experimental study, the general-purposed self-supervised TTT algorithm in [41] may fail to bring any improvements but could even deteriorate the detection accuracy. On the contrary, our OST method can significantly improve the generalization performance of the deepfake detector and could attain superior performances over the existing solutions in multiple benchmark datasets.

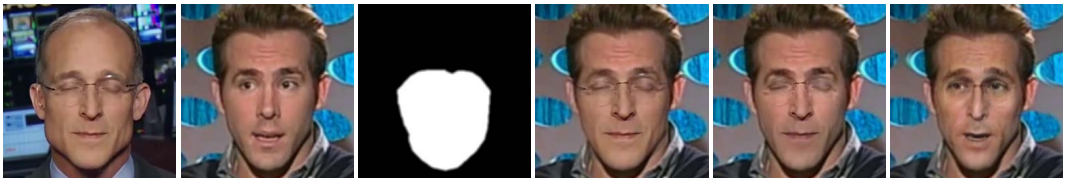
2 Related Works

In this section, we conduct a brief survey on the most relevant arts, including existing deepfake detection methods and test-time training (TTT)-based works.

2.1 Deepfake Detection

Since the deepfake forgeries have led to great threats to societal security, it is of paramount importance to develop effective detectors against it. By formulating the detecting as a vanilla binary classification problem (*i.e.* pristine or forgery), current end-to-end trained detectors [38] with a simple Xception baseline [7] can obtain a high detection accuracy. Besides, with more powerful network structures and more informative image features embedded in the network inputs, existing methods [3, 2, 31, 37, 34] are able to achieve even more remarkable success when the training and test forgeries are synthesized by the same deepfake algorithms. Thus, the real challenge in this task lies in how to generalize a learned detector to forgeries created by unseen methods.

Several works have been devoted to addressing the generalizing problem recently. For example, [23] suggests that the blending operation is ubiquitous in the current deepfake synthesizing process. As a consequence, they propose to detect the blending boundaries hidden in the forgeries and use them as classification clues. Moreover, [28] shows that the up-sampling step in synthesizing models can bring artifacts to the synthesized forgeries, and they use the phase spectrums of the forgeries to capture these artifacts. As existing forgery synthesizing steps often involve two images from different identities and



(a) Image A (b) Image B (c) Deformed mask (d) Forgery 1 (e) Forgery 2 (f) Forgery 3

Figure 2: Generation of a pseudo-training sample. Two images shown in (a) and (b) can be blended in the region defined by the deformed blending mask in (c) to synthesize forgeries. Forgeries in (d)-(f) are produced by the alpha, Poisson [35], and learning-based blending [6] methods, respectively.

different sources, [30] suggest using high-pass filters from SRM [15] to reveal detail discrepancies of the forgeries. A similar idea is adopted in [52], where they use the cue of the source feature inconsistency within the forged images for detection. Although these methods are effective in many cases, the low-level artifacts they rely on are sensitive to post-processing steps that vary in different datasets, thus jeopardizing their generalization. Some other works propose to borrow features from other tasks, such as lips reading [18], facial image decomposition [55], and landmark geometric [42], to imply the abnormity of forgeries. Although these features can bring certain improvements, there is a great chance that future deepfake algorithms will be designed based on these detectors to synthesize more natural forgeries, causing even bigger threats to societal security.

Compared to existing detectors, the advantages of our method are as follows: (1) we adopt a MAML-based OST framework to enable the fast adaptation of the learned detector to the test data, which improves generalization regardless of the varying post-processing steps; (2) OST does not rely on hand-crafted or borrowed features, which leaves fewer traces for the deepfake algorithms to attack.

2.2 Test-time Training

The concept of TTT was firstly introduced in [41] for generalization to out-of-distribution test data, where a self-supervised rotation prediction task is utilized with the main classification task during training, and only the self-supervised task is adopted to help improve the visual representation during inference, which indirectly improves semantic classification. This framework is theoretically proved to be effective [41] and is further used in other related areas [25, 46, 36, 5, 50]. For example, [25] proposes a reconstruction task within the main pose estimation framework, which can be trained by comparing the reconstructed image with the ground truth borrowed from other frames. [46] demonstrate that the predictions with lower entropy have lower error rates, and they use entropy to provide finetuning signals when given a test image. Instead of only minimizing the entropy of the predicted posterior, [36] also suggests maximizing the noise robustness of the feature representation during test. In some recent works [5, 50], the TTT framework has also been utilized within a model agnostic meta-learning (MAML) paradigm [14] which allows the trained model to be optimized in a way such that it can quickly adapt to any test images. To enable fast adaptation, these works use contrastive loss [5] or smooth loss [50] to finetune the models during the meta-test phase.

However, despite some encouraging results, current TTT methods aim to select empirical self-supervised tasks, which is at high risk of deteriorating the performance when the tasks are not properly selected [29]. This work introduces a new learning paradigm specially designed for the deepfake detection task. Recall that a forgery can be easily synthesized by blending two different images [23, 52]. We thus use a newly synthesized forgery as a pseudo-training sample to finetune the pretrained detector during inference. Our method is easy to implement and can avoid the tedious work of selecting an effective self-supervised task.

3 Proposed Method

Our method consists of an offline meta-training step and an online test-time training step. The online test-time-training step will first generate a pseudo-training sample for each test image and then perform a single gradient descent update to obtain the sample-specific model parameters. The offline meta-training mimics the online test-time-training operation by sampling episodes from training data. In the following, we first describe the test-time-training step, involving both pseudo-training sample creation and one-shot update. Then we elaborate on the meta-learning process.

Algorithm 1 One-shot online training

Require: Pretrained detector $f(\cdot, \tilde{\theta})$, learning rate γ, test sample x_e, training dataset $\mathbf{D}$.

1: Ramdomly select x_r from $\mathbf{D}$, and synthesize forgery x_o^f by blending x_e and x_r

2: Evaluate the one-shot loss $\sum_{x_s \in \{x_o^f, x_r\}} L(f(x_s, \tilde{\theta}), y_s)$

3: Update parameters with gradient descent: $\theta^* = \tilde{\theta} - \gamma \sum_{x_s \in \{x_o^f, x_r\}} \nabla_{\tilde{\theta}} L(f(x_s, \tilde{\theta}), y_s)$

4: Get the final classification result by evaluating the updated detector $f(\cdot, \theta^*)$ in the test sample x_e

3.1 Online Test-Time Training

In the following, we assume that a deepfake detection model $\tilde{\theta}$ has already been learned. The OST process will create an updated model parameter θ^* that is adaptive to each individual test image. This is achieved by first generating a pseudo-training sample and then using it to form a mini-training set and update model parameters via one-shot training.

Generating pseudo-training samples: As shown in Figure 2, we first create a synthesized training image x_o^f by blending the test image x_e (shown in Figure 2 (b)) with an image (template image) x_r sampled from the training set $\mathbf{D}$ (shown in Figure 2 (a)). Specifically, we spatially align these two images via facial landmarks [39] and color transfer. Then, we generate a facial mask based on landmarks and refine this mask via deformation and blurring as shown in (c). The masked facial region in one image can be blended with another to formulate the pseudo-training face forgeries. In practice, we randomly utilize alpha, Poisson [26], and a learning-based method [6] for blending as shown in (d)-(f). These pseudo-training samples are labeled as negative (i.e., fake). Note that although we are unknown about whether the test image is pristine or forgery (i.e., real or fake), we are confident that the synthesized pseudo image is a forgery.

One-shot online training: In practice, we use the images with known labels (i.e., the randomly sampled template image x_r and the synthesized image x_o^f) to form a mini-training set. The label of x_r is provided by the training set, e.g., it can be real or fake dependent on the sampled image, while x_o^f is always fake since it is a synthesized image. Then we can perform gradient descent to update the model. In our design, we only perform a single step gradient descent, dubbed one-shot online training. Formally, the initial model parameter is updated via

$$\theta^* = \tilde{\theta} - \gamma \sum\nolimits_{x_s \in \{x_o^f, x_r\}} \nabla_{\tilde{\theta}} \mathcal{L}(f(x_s, \tilde{\theta}), y_s), \tag{1}$$

where y_s is the labels to x_s; $\mathcal{L}(f(x_s, \tilde{\theta}), y_s)$ is the one-shot loss; $\mathcal{L}(\cdot, \cdot)$ is the loss function for the deepfake detector, i.e., the AM-Softmax loss [47] in this paper inspired by the previous work [30]; γ is the step size. After the updating step, we can obtain the final result by applying the updated detector $f(\cdot, \theta^*)$ to the test sample x_e. The algorithm of the OST scheme is shown in Algorithm 1.

Understand the effect of one-shot test-time training: The proposed test-time-training algorithm can also be understood as a domain adaptation method. Specifically, in OST, each image is viewed as a specific domain defined by its content, and the unseen test image will have a domain gap to the training data. The pseudo-training sample created by using the above procedure is more relevant to the test image than the training samples since x_o^f is synthesized based on the test image. Thus a fast training on the sample could make the detector adapt better to the test image. More evidence on the above analysis is presented in Sec. 5.2.

3.2 Offline Meta-training

The proposed offline meta-training step is to ensure that a good initialization can be learned for the online OST step. To this end, a MAML [14] style meta-training scheme is utilized. Specifically, we construct a training episode by first randomly sampling two images x_r^1 and x_r^2 from $\mathbf{D}$, and then create two synthesized forgery images x_1^f and x_2^f with different blending operations, say, one with alpha blending and one with Poisson blending. Then we will use x_r^1 and x_1^f to perform one-shot

Algorithm 2 Offline meta-training

Require: Meta-batch size T, learning rates λ and γ, training dataset $\mathbf{D}$.

while not done **do**
  for each t in T **do**
    Sample x_r^1 and x_r^2 from $\mathbf{D}$, and synthesize forgeries x_1^f, x_2^f by blending x_r^1, x_r^2
    Obtain support set (x_s, y_s) and query set (x_q, y_q) using our split strategy
    Evaluate the inner loss $L_t(f(x_s, \theta), y_s)$
    Compute adapted parameters with gradient descent: $\theta_t' = \theta - \gamma \nabla_\theta L_t(f(x_s, \theta), y_s)$
  end for
  Evaluate the meta loss $\sum_t L_t(f(x_q, \theta_t'), y_q)$
  Update the models with gradient descent: $\theta \leftarrow \theta - \lambda \nabla_\theta \frac{1}{T} \sum_t L_t(f(x_q, \theta_t'), y_q)$
end while

Return: $f(\cdot, \tilde{\theta})$

Method	DF			F2F			FS			NT			Avg.
	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	
Xception [38]	0.654	0.808	0.617	0.708	0.695	0.745	0.708	0.657	0.605	0.646	0.724	0.838	0.700
Face X-ray [23]	0.609	0.811	0.668	0.633	0.679	0.766	0.646	0.625	0.795	0.613	0.645	0.866	0.696
F3Net [37]	0.682	0.812	0.658	0.679	0.729	0.761	0.679	0.641	0.651	0.672	0.826	0.932	0.727
RFM [45]	**0.758**	0.833	0.717	0.736	0.711	0.732	0.714	0.593	0.714	0.726	0.816	0.846	0.741
SRM [30]	0.679	0.855	0.720	0.687	0.801	0.775	0.671	0.705	0.771	0.656	0.791	**0.936**	0.754
Ours	0.757	**0.869**	**0.938**	**0.798**	**0.880**	**0.947**	**0.802**	**0.824**	**0.909**	**0.752**	**0.841**	0.929	**0.854**

Table 1: Generalizability comparisons with state-of-the-art methods in the term of AUC. The best results are in bold. The first row denotes the training data, and the second row shows the corresponding test dataset. Our method performs favorably among the models compared.

training. The updated model is evaluated on x_r^2 and x_2^f, and the incurred loss is used to update the model parameters with step size λ. In the above design, x_r^1 and x_1^f are akin to the support set, and x_2^f and x_r^2 are akin to the query set in standard meta-learning. Note that we include both x_r^2 and x_2^f in the query set. They represent two different scenarios: x_r^2 is an image from the original training domain while x_2^f represents an image from a new (synthesized randomly) domain. Including both the two images encourages the model works well for the existing domain and be able to generalize to a new domain. Formally, the updating process in one episode can be written as:

$$\theta \leftarrow \theta - \lambda \sum_{x_q \in \{x_r^2, x_2^f\}} \nabla_\theta \mathcal{L}(f(x_q, \theta'), y_q), \quad s.t. \quad \theta' = \theta - \gamma \sum_{x_s \in \{x_r^1, x_1^f\}} \nabla_\theta \mathcal{L}(f(x_s, \theta), y_s). \quad (2)$$

Eq. 2 is also referred to as meta update and inner update in a standard MAML-like paradigm, where $\mathcal{L}(f(x_q, \theta'), y_q)$ and $\mathcal{L}(f(x_s, \theta), y_s)$ are referred to as the meta and inner losses. The overall algorithm for the offline meta-training is illustrated in Algorithm 2.

4 Experiments

This section first presents the setups and then shows extensive experimental results to demonstrate the superiority of our approach. Please refer to the supplementary material for more experimental results.

4.1 Settings

Training and test datasets. Following the protocols in existing deepfake detection methods [23, 30, 52], we use the data in the Faceforencis++ (FF++) dataset [38] for training. This dataset contains 1000 videos, in which 720 videos are used for training, 140 videos are reserved for verification, and the rest are used for test. Each real video in this dataset are manipulated with four deepfake methods, including DeepFake (DF) [9], Face2Face (F2F) [44], FaceSwap (FS) [13], and NeuralTexture (NT) [43], to generate four fake videos. Each video in FF++ is processed to have three video qualities, namely raw, lightly compressed (HQ), and heavily compressed (LQ), and the HQ version is adopted by default unless otherwise stated.

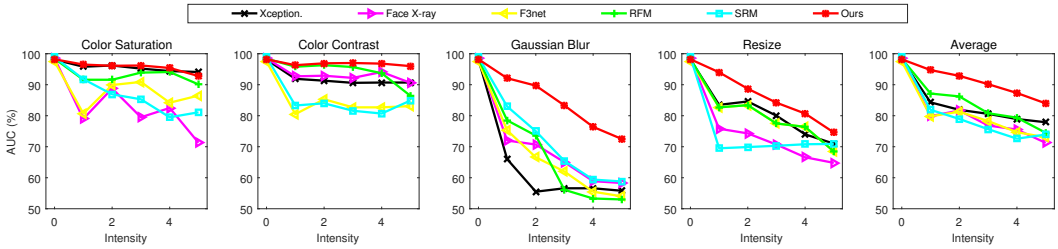


Figure 3: Generalizability comparisons under different post-processing operations in the term of AUC. The compared methods undergo five different levels of four particular types of post-processing steps. "Average" denotes the mean across all post-processing steps at each intensity level. Our method performs competitively against existing arts.

Method	Training dataset	DFDC			CelebDF		
		AUC ↑	ACC ↑	ERR ↓	AUC ↑	ACC ↑	ERR ↓
MLDG [22]	FF++	0.682	0.607	0.370	0.609	0.595	0.418
LTW [40]		0.690	0.631	0.368	0.641	0.634	0.397
Ours		**0.833**	**0.714**	**0.250**	**0.748**	**0.673**	**0.312**

Table 2: Comparisons with models based on meta learning in terms of AUC, ACC, and ERR.

To comprehensively evaluate the generalizability of our method, we use four more benchmark datasets for test, including the DeepfakeDetection (DFD) [8] which consists of 363 real and 3068 fake videos created by the improved DF technique; Deepfake Detection Challenge (DFDC) [11] that contains over 1,000 real and over 4,000 fake videos manipulated by multiple Deepfake, GANbased, and non-learned methods; DeeperForensics-1.0 (DF1.0) [20] that is composed of over 11,000 fake videos generated by their DFVAE method, and the CelebDF [27] dataset that contains 408 real videos and 795 synthesized videos generated by the improved DF method. The forgeries in the training and test datasets do not overlap both in their contents and creating methods.

Implementation details. We use Xception [7] as the backbone network, and the parameters are initialized by the weights pretrained on the ImageNet [10]. We use DLIB [39] for face extraction and alignment, and we resize the aligned faces to 256×256 for all the sampled frames in the training and test datasets. We use the Adam optimizer [21] for optimizing the network with $\beta_1 = 0.9$ and $\beta_1 = 0.999$, and the meta-batch size is set to be 20. The learning rates for the inner update (*i.e.* γ in Eq. (1) and (2)) and meta update (*i.e.* λ in Eq. (2)) are fixed as 0.0005 and 0.0002 for both the offline and online training phases.

4.2 Generalizability Comparisons

We compare our method with several state-of-the-arts, including Xception [38], Face X-ray [23], F3net [37], RFM [45], and SRM [30], to demonstrate its superior generalizability. To ensure fair comparisons, we use the provided codes of Xception, RFM, and SRM from the authors, and we reimplement Face X-ray and F3Net rigorously following the companion paper's instructions and train these models under the same settings.

Generalizing to different datasets. Forgeries in different datasets are often created by different deepfake methods. In a real-world scenario, suspicious images are likely to be created by unseen methods from unseen sources, thus the generalizability to different datasets would be crucial. To evaluate the generalizability, we conduct experiments by training the compared methods on each of the four data in FF++ (*i.e.* DF, F2F, FS, and NT) and test them on the benchmark datasets, including DF1.0, DFD, DFDC. The experimental setting is not trivial since all the deepfake methods that synthesize the test forgeries are unseen in the training dataset.

Experimental results in the term of Area Under Curve (AUC) are listed in Table 1. Existing arts [38, 23, 37, 30] rely on imperceptible artifacts for generalizing, but these artifacts may appear with different patterns in forgeries from different datasets, thus limiting their generalizing. RFM [45] suggests mining more regions in the detected images for classifying. However, this approach still cannot avoid the domain gap between the training and test samples. In comparison, our method

	[54]	[1]	[26]	[32]	[49]	[33]	[24]	[31]	[28]	[51]	Ours
FF++	0.701	0.847	0.930	0.664	0.473	0.966	0.968	0.932	0.969	**0.998**	0.982
CelebDF	0.538	0.548	0.646	0.550	0.546	0.575	0.563	0.734	0.724	0.674	**0.748**

Table 3: Extensive evaluations with other state-of-the-arts (*i.e.* Two-stream [54], MesoNet [1], FWA [26], VA-MLP [32], Headpose [49], Capsule [33], SMIL [24], Two-branch [31], SPSL [28], and MADD [51]) in the term of AUC. The compared models are trained on the FF++ dataset [38] while testing on both FF++ [38] and CelebDF [27]. Our method performs favorably against existing arts.

Method	DF			F2F			FS			NT			Avg.
	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	
Baseline	0.742	0.841	0.910	0.693	0.816	0.924	0.735	0.750	**0.913**	0.736	**0.848**	0.895	0.817
SSL-TTT	0.717	0.862	0.859	0.666	0.820	0.891	0.733	0.684	0.833	0.677	0.825	0.863	0.786
Ours	**0.757**	**0.869**	**0.938**	**0.798**	**0.880**	**0.947**	**0.802**	**0.824**	0.909	**0.752**	0.841	**0.929**	**0.854**

Table 4: Ablation studies regarding the effectiveness of the proposed OST. The metric is AUC. The baseline model is our method without the online training step (but using synthesized images at the training stage). "SSL-TTT" is the existing TTT scheme that with a self-supervised task [41]. We implement it with our MAML-based framework using the same self-supervised task from [41].

suggests finetuning the pretrained detector on the test data, and it enables adaptation to the new data before evaluation, thus improving generalizing. Meanwhile, we can observe that our method outperforms other models in most cases, and it surpasses the second best [30] by 10% in the term of overall performance, which clearly demonstrates the advantages of the proposed OST strategy for generalizing to new forgeries.

Generalizing to different post-processing steps. For real-world deepfake detection requirements, it is of paramount importance that a detector can generalize to different post-processing steps. We conduct experiments with different post-process steps to evaluate the generalizability of our method. Specifically, we train the compared models on the original FF++ data and test them on FF++ samples that were exposed to various unseen post-processing steps.

Following [20], we consider four popular post-processing steps, including changes in color saturation, changes in color contrast, Gaussian blur, and resize: downsample the image by a factor then upsample it to the original resolution. Note that all the models are trained without these augmentations, so that these post-processing steps are unseen during training. The results are plotted in Figure 3. It is evident that our method can better generalize to unseen post-process operations than other models [38, 23, 37, 30, 45]. In particular, the changes in color saturation and color contrast can hardly affect our model, while state-of-the-arts suffer from performance declines to different extents. Moreover, OST maintains higher performances than all the compared models when the post-processing steps affect the high frequency content of the images (*i.e.* blurring and resizing), where existing arts undergo significant deterioration. We attribute this to their reliance on the imperceptible low-level clues, which will be damaged by these steps. We also include comparisons regarding another post-processing step (*i.e.* compression), please refer to the supplementary material for details.

4.3 State-of-the-art Comparisons

Comparisons with other models based on meta learning. The idea of using meta learning to boost the generalization of the deepfake detector has been explored in the previous work, such as that in LTW [40]. Inspired by MLDG [22], which uses meta-learning to solve the domain generalization problem, LTW assumes forgeries from different deepfake methods as different domains, so the generalizable deepfake detection task can be regarded as a domain generalizing problem. In this experiment, we compare our method with LTW and MLDG. Note that both of these models require the training dataset to contain more than one forgery creating methods. We thus train the models on the FF++ dataset and test them on CelebDF and DFDC datasets.

The comparison results are shown in Table 2, where the figures are directly cited from the reported statistics in [40]. We observe that the proposed method outperforms others in terms of ACC, AUC, and EER. The main reason is that although LTW and MLDG can learn good initializations via meta learning, their detectors cannot finetune on the unknown test domains, thus limiting the overall performances. In comparison, our pretrained detector can adapt to the test samples via OST, thus being more effective than LTW and MLDG for the generalizable deepfake detection task.

	DFDC	DFD	DF1.0	CelebDF
$\|f(x_r, \tilde{\theta})) - f(x_e, \tilde{\theta}))\|$	2.06	1.06	0.94	1.43
$\|f(x_o^f, \tilde{\theta})) - f(x_e, \tilde{\theta}))\|$	1.63	0.82	0.61	1.03

Table 5: Average feature distances of the test samples (*i.e.* x_e) from different test sets to the training sample (*i.e.* x_r) and their corresponding pseudo-training forgeries (*i.e.* x_o^f). Distance between x_e and x_o^f is closer than that between x_e and x_r in all cases.

Method	DF			F2F			FS			NT			Avg.
	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	
NN	0.790	0.857	0.944	0.806	0.841	0.938	0.791	0.831	0.914	0.760	0.858	0.931	0.855
Avg	0.773	0.898	0.956	0.768	0.820	0.929	0.773	0.809	0.924	0.751	0.847	0.940	0.849
X w/o OST	0.761	0.836	0.928	0.729	0.774	0.912	0.755	0.784	0.894	0.739	0.825	0.906	0.820
X w/ OST	0.763	0.845	0.962	0.733	0.785	0.943	0.766	0.812	0.946	0.775	0.889	0.929	0.846
Ours	0.757	0.869	0.938	0.798	0.880	0.947	0.802	0.824	0.909	0.752	0.841	0.929	0.854

Table 6: Effects of different sampling and learning process. We select the training sample to be blended by using nearest neighbor sampling, and average sampling. The detection results are shown on the first and second rows, respectively. We use the Xception network without and with our OST scheme (*i.e.* X w/o OST and X w/ OST) to show if our method can still thrive in a regular framework. The results are shown on the third and fourth rows, respectively.

Comparisons with other state-of-the-art detectors. We further evaluate our method against several more state-of-the-art algorithms, including the Two-stream [54], MesoNet [1], Headpose [49], FWA [26], VA-MLP [32], Capsule [33], SMIL [24], Two-branch [31], SPSL [28], and MADD [51]. All the models are trained on the FF++ dataset and tested on FF++ and CelebDF datasets. We cite the figures from [28] directly for some compared models. Results in the term of AUC are listed in Table 3. Our method achieves favorable performance when generalizing to CelebDF while still maintaining a good result on FF++ compared with all the other methods.

5 Algorithm Analysis

In this section, we first conduct ablation studies to validate the effectiveness of our OST algorithm. Then, we illustrate why the pseudo samples can lead to better generalization. Besides, we discuss how sample selection affects our OST algorithm and the adaption of our method to a regular framework.

5.1 Ablation Studies

Our ablation studies on evaluating the proposed OST method consist of the comparisons of our model to the following two configurations. The first one is the baseline configuration that does not perform online training but include all the other operations, e.g., augmentation of domains via synthesized images. The second one is a self-supervised test-time-training (i.e., SSL-TTT algorithm in [41]). In SSL-TTT, we use MAML as offline meta-learning and a rotation prediction task [16] for training and test, which is the same as that in [41]. The data collection, split, and other implementation configurations remain the same to ensure fair comparisons.

We train the baseline method, SSL-TTT, and our method on four types of data in the FF++ dataset. Then, we evaluate these learned detector models on the DFDC, DFD, and DF1.0 datasets. Table 4 shows the evaluation results by using the AUC metric. These results indicate that our OST method improves baseline by 4%. This improvement shows the effectiveness of our online test-time training method. On the other hand, we observe that SSL-TTT does not improve the baseline performance. This corresponds to the observation in [29] that an inappropriate self-supervised task (i.e., rotation prediction task) in the TTT scheme may jeopardize the main deepfake detection performance. Overall, the experiments show that our OST method is more generalizable than the existing TTT method [41] for the deepfake detection task.

5.2 Explanation for OST: Pseudo Samples Can Mitigate the Diverged Data Distributions

In section 3.1, we explain that the effect of OST is by training the detector on a synthesized pseudo-training sample that is closer to the test images than the original training data. This is due to the

synthesized image being created based on the test image. In this subsection, we further provide some experimental evidence for this argument.

We experimentally validate the above analysis by computing the feature distance of test samples (i.e., x_e) to the training samples (*i.e.* x_r), and their corresponding pseudo-training samples (*i.e.* x_o^f). We use test samples from the four test sets (*i.e.* DFDC, DFD, DF1.0, CelebDF) for evaluation, and the euclidean distance is adopted as the evaluation metric. Table 5 show the results. We observe that in all test sets, the average distance between x_o^f and x_e is smaller than that between x_r and x_e. These results consistently correspond to our aforementioned analysis. As a result, we can conclude that the generated pseudo-training samples are with features that reside in the intermediate state of the training and test data. By finetuning the detector with pseudo-training samples that are close to test samples, we can make the model better adapt to the domain defined by the test sample.

5.3 Further Analysis

Sample selections. During forgery synthesis, we randomly select one training sample to blend with the test sample. To investigate whether the random selection is an effective approach, we conduct ablation studies where our random selection is compared to the other two configurations. The first configuration is the nearest neighbor sampling (*i.e.* NN sampling). For each test sample, we select the training sample whose feature distance is closest for blending. The second configuration is the average sampling (*i.e.* Avg sampling), where we average the training sample performance for evaluation. We use these configurations on four types of data on FF++ datasets, and evaluate their performance on DFDC, DFD, and DF1.0 datasets. The first and second rows of Table 6 show the results. We observe that the differences of the average performance among these three sampling strategies (*i.e.* NN sampling, Avg sampling, and random sampling) are small (less than 0.01). The small difference indicates that the sampling strategies during inference are not critical in our model. To this end, we use the random sampling strategy for efficiency considerations.

OST adaptions. Our learning framework is based on MAML which can adapt to new tasks efficiently. We analyze how our OST method functions when we substitute MAML with a regular training scheme. The training data is kept the same, and the backbone is still Xception. Results are shown in the third and fourth rows of Table 6. We observe that OST with regular end-to-end network training performs on par with the MAML framework. On the other hand, the detection performance decreases when the model does not employ our OST process. These results indicate that our method suites regular training configurations and improves prevalent network backbones. However, the MAML framework enables a faster adaption during online inference (i.e., 10 times faster for each test sample). In practice, we utilize the MAML framework and integrate our OST method.

6 Conclusion and Discussions

In this paper, we propose a new learning paradigm specially designed for the generalizable deepfake detection task. To recap briefly, for each test sample, we suggest finetuning the pretrained detector with a pseudo-training sample, which is synthesized by blending the test samples and a randomly selected template image, before the classification step. We empirically find that the proposed online training scheme enables the pretrained model to adapt to the sample-specific statistics, thus facilitating the generalizability. We implement our method in a MAML-based scheme to enable fast adaptation to different test samples, which shows superior performance against state-of-the-art algorithms for generalization to unseen forgeries and various post-processing operations.

Limitation. As the pseudo-training samples are synthesized following existing deepfake pipelines, our method cannot be applied to the scenario when the fake images are created following other protocols, such as the case when fake images are totally synthesized by GAN-based methods. It is our future work to develop methods that aim for both deepfakes and GAN-synthesized fake images.

Ethic statement. This work is designed to help people better fight against the abuse of deepfake technology. It does not involve any human or animal subjects, and there is no violation of personal privacy while conducting experiments. We do not anticipate any potentially harmful consequences to our work. Through our study and releasing our code, we hope to raise stronger research and societal awareness towards the problem of generalizable deepfake detection.

References

[1] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Mesonet: a compact facial video forgery detection network. In *WIFS*, 2018.

[2] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *CVPR Workshops*, 2019.

[3] Irene Amerini, Leonardo Galteri, Roberto Caldelli, and Alberto Del Bimbo. Deepfake video detection through optical flow based cnn. In *ICCV Workshops*, 2019.

[4] Shivangi Aneja and Matthias Nießner. Generalized zero and few-shot transfer for facial forgery detection. *arXiv preprint arXiv:2006.11863*, 2020.

[5] Alexander Bartler, Andre Bühler, Felix Wiewel, Mario Döbler, and Bin Yang. Mt3: Meta test-time training for self-supervised test-time adaption. *arXiv preprint arXiv:2103.16201*, 2021.

[6] Renwang Chen, Xuanhong Chen, Bingbing Ni, and Yanhao Ge. Simswap: An efficient framework for high fidelity face swapping. In *ACM MM*, 2020.

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[8] Deepfakedetection. https://ai.googleblog.com/2019/09/contributing-data-to-deepfakedetection.html Accessed 2021-04-24.

[9] DeepFakes. www.github.com/deepfakes/faceswap Accessed 2021-04-24.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[11] Deepfake detection challenge. https://www.kaggle.com/c/deepfake-detection-challenge Accessed 2021-04-24.

[12] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Ting Zhang, Weiming Zhang, Nenghai Yu, Dong Chen, Fang Wen, and Baining Guo. Protecting celebrities with identity consistency transformer. In *CVPR*, 2022.

[13] FaceSwap. www.github.com/MarekKowalski/FaceSwap Accessed 2021-04-24.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[15] Jessica Fridrich and Jan Kodovsky. Rich models for steganalysis of digital images. *IEEE TIFS*, 7(3):868–882, 2012.

[16] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[17] Alexandros Haliassos, Rodrigo Mira, Stavros Petridis, and Maja Pantic. Leveraging real talking faces via self-supervision for robust forgery detection. In *CVPR*, 2022.

[18] Alexandros Haliassos, Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. Lips don't lie: A generalisable and robust approach to face forgery detection. In *CVPR*, 2021.

[19] Shuai Jia, Chao Ma, Taiping Yao, Bangjie Yin, Shouhong Ding, and Xiaokang Yang. Exploring frequency adversarial attacks for face forgery detection. In *CVPR*, 2022.

[20] Liming Jiang, Ren Li, Wayne Wu, Chen Qian, and Chen Change Loy. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection. In *CVPR*, 2020.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018.

[23] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and Baining Guo. Face x-ray for more general face forgery detection. In *CVPR*, 2020.

[24] Xiaodan Li, Yining Lang, Yuefeng Chen, Xiaofeng Mao, Yuan He, Shuhui Wang, Hui Xue, and Quan Lu. Sharp multiple instance learning for deepfake video detection. In *ACMMM*, 2020.

[25] Yizhuo Li, Miao Hao, Zonglin Di, Nitesh Bharadwaj Gundavarapu, and Xiaolong Wang. Test-time personalization with a transformer for human pose estimation. In *Neurips*, 2021.

[26] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. In *CVPR Workshops*, 2018.

[27] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A new dataset for deepfake forensics. In *CVPR*, 2020.

[28] Honggu Liu, Xiaodan Li, Wenbo Zhou, Yuefeng Chen, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In *CVPR*, 2021.

[29] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *Neurips*, 2021.

[30] Yuchen Luo, Yong Zhang, Junchi Yan, and Wei Liu. Generalizing face forgery detection with high-frequency features. In *CVPR*, 2021.

[31] Iacopo Masi, Aditya Killekar, Royston Marian Mascarenhas, Shenoy Pratik Gurudatt, and Wael AbdAl-mageed. Two-branch recurrent network for isolating deepfakes in videos. In *ECCV*, 2020.

[32] Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *WACV Workshops*, 2019.

[33] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. In *ICASSP*, 2019.

[34] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. Deepfake detection based on discrepancies between faces and their context. *IEEE TPAMI*, 2021.

[35] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *SIGGRAPH*. 2003.

[36] S Prabhu Teja and François Fleuret. Uncertainty reduction for model adaptation in semantic segmentation. In *CVPR*, 2021.

[37] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *ECCV*, 2020.

[38] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *ICCV*, 2019.

[39] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: database and results. *IVC*, 47:3–18, 2016.

[40] Ke Sun, Hong Liu, Qixiang Ye, Jianzhuang Liu, Yue Gao, Ling Shao, and Rongrong Ji. Domain general face forgery detection by learning to weight. In *AAAI*, 2021.

[41] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[42] Zekun Sun, Yujie Han, Zeyu Hua, Na Ruan, and Weijia Jia. Improving the efficiency and robustness of deepfakes detection through precise geometric features. In *CVPR*, 2021.

[43] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *TOG*, 38(4):1–12, 2019.

[44] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[45] Chengrui Wang and Weihong Deng. Representative forgery mining for fake face detection. In *CVPR*, 2021.

[46] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

[47] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE SPL*, 25(7):926–930, 2018.

[48] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020.

[49] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses. In *ICASSP*, 2019.

[50] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. In *Neurips*, 2021.

[51] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. Multi-attentional deepfake detection. In *CVPR*, 2021.

[52] Tianchen Zhao, Xiang Xu, Mingze Xu, Hui Ding, Yuanjun Xiong, and Wei Xia. Learning self-consistency for deepfake detection. In *ICCV*, 2021.

[53] Yinglin Zheng, Jianmin Bao, Dong Chen, Ming Zeng, and Fang Wen. Exploring temporal coherence for more general video face forgery detection. In *ICCV*, 2021.

[54] Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S. Davis. Two-stream neural networks for tampered face detection. In *CVPR Workshop*, 2017.

[55] Xiangyu Zhu, Hao Wang, Hongyan Fei, Zhen Lei, and Stan Z Li. Face forgery detection by 3d decomposition. In *CVPR*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A] Code will be released upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section 6.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

In this appendix, we provide,

1. Visualized pipeline of the pseudo training sample generating process regarding Figure 2 in the manuscript.
2. Time consumption and computational complexity comparisons against other models.
3. More ablation studies regarding multiple gradient descents in the online adapting step and evaluation on images with different resolutions.
4. Evaluations with other metrics including true positive, true negative, false positive, and false negative.
5. Discuss generalization settings introduced in [48] by calibrating the classification threshold during inference.
6. Revision of Table 2 and 4 by including more results from recent meta-learning-based arts and results from TENT [46].
7. Visualizations of the embedded representations.

A Visualization of the Pseudo Sample Generation Pipeline

The pipeline for generating a pseudo training sample x_o^f is shown in Figure 4. For every test sample x_e, we first randomly select a template image x_r from the training dataset and align these two images in geometry based on their landmarks. Then, we can extract a convex hull using the landmarks of x_e and obtain the deformed final mask by applying random deformation and blurring on the convex hull. With the deformed mask, we can crop the corresponding color corrected face content from x_r. Finally, with x_e, the deformed mask, and the cropped face, we can obtain the pseudo training sample x_o^f by using alpha or Poisson [35] blending methods. Note that when using the leaning-based blending [6] method, we can directly use x_r and x_e as input and output x_o^f, which does not require the alignment, cropping, or correction steps. In the following OST step, we randomly select a forgery as x_o^f. **We will replace Figure 2 with Figure 4 in the future version**.

B Time Consumption and Computational Complexity Comparisons

To comprehensively evaluate the proposed method, we provide the time consumption and computational complexity comparisons in Table 7. All the compared methods are evaluated on the same device using a 256×256 image. All the models are implemented with the Xception baseline [7], thus the computational complexities are nearly the same for most models except for SRM [30] which uses a dual branch network architecture. Because our model includes two forwards and one backward operations during inference, thus the corresponding computational complexity is more than others. Meanwhile, our method involves the generation of pseudo training samples which uses CPU for the task mostly (except for the learning-based generating method [6]). Thus, it requires much more running time than others. However, it is noteworthy to point out that in this relatively early stage, the deepfake detection community considers little on time consumption but more on the accuracy. We will explore possible speedup techniques such as model compression and distillation in future work.

C More Ablation Studies

C.1 Evaluations with Multiple Gradient Descent Steps

To study if more updating steps can improve the detection, we conduct ablation studies by using different gradient descent steps during evaluation. The model is trained on the FF++ dataset, and it

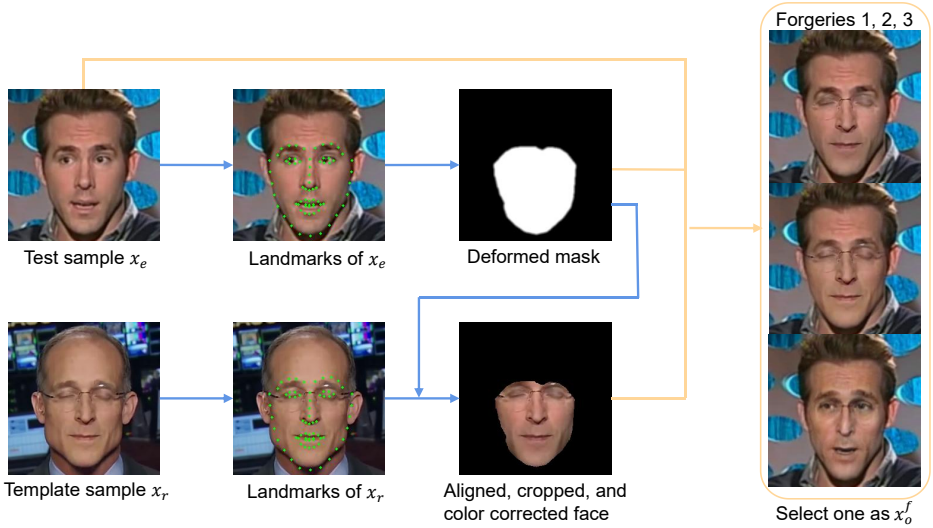


Figure 4: Pipeline for generating pseudo training samples. Forgeries 1, 2, 3 (from top to bottom) in the right are produced by the alpha, Poisson [35], and learning-based blending [6] methods, respectively.

	Xceptiion [38]	Face X-ray [23]	F3Net [37]	RFM [45]	SRM [30]	Ours
TC (s)	0.015	0.017	0.019	0.015	0.037	0.062 + 0.074
CC (MACs(G))	6.01	6.01	6.05	6.01	13.81	18.03

Table 7: Time consumption (TC) and computational complexity (CC) comparisons. All methods are evaluated on the same device with a 256×256 image. The inference time of our method takes 0.062 seconds, and our pseudo sample generating process takes 0.074 seconds. Our method takes two forwards and one backward, thus requiring more flops.

is evaluated on the DFDC, DFD, and DF1.0 datasets. As shown in Table 8, the accuracies of using more gradient descent steps are not improved greatly while the computational times are proportional to the gradient descent steps. Meanwhile, using more gradient descent steps also requires much more memory costs in an MAML-based framework. We thus use only one gradient descent step in our method as a compromise between efficiency and accuracy.

C.2 Evaluations Using Images with Different Resolutions

Following the setting in previous works [23, 30], faces from the training and test datasets are resized to 256×256. To evaluate if the resolution can also influence the generalizability of the detector, we conduct ablation studies by using samples with different resolutions during training and test. Specifically, we evaluate our model on images with resolutions of 200×200 and 320×320. Results are listed in Table 9. We observe that the differences between the three different resolutions are rather subtle, indicating that image resolution is not an influential factor for generalization.

D Evaluations of Different Methods with Other Metrics

To evaluate how the compared methods perform on pristine and deepfake separately, we also report true positive (TP), true negative (TN), false negative (FN), false positive (FP), and true negative rate (TNR) for them. The methods are trained with the FF++ dataset and evaluated using DFDC, DFD, and DF1.0 datasets. Results are shown in Table 10. We observe that the TNR of our method is much larger than other methods, indicating that our method is more likely to detect a given deepfake image. In comparison, TNRs for other methods are around 0.5, which shows that these methods can hardly identify a fake image.

Steps	DF			F2F			FS			NT			Avg.	TC(s)
	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0		
1 step	0.757	0.869	0.938	0.798	0.880	0.947	0.802	0.824	0.909	0.752	0.841	0.929	0.854	0.065
2 step	0.759	0.902	0.927	0.794	0.892	0.942	0.797	0.830	0.922	0.758	0.836	0.947	0.859	0.121
3 step	0.783	0.860	0.942	0.804	0.906	0.934	0.801	0.840	0.929	0.747	0.843	0.948	0.861	0.181

Table 8: Performances of using different gradient descent steps. Improvements brought by more updating steps are subtle, while the time consumptions (TC) are proportional to the updating steps.

Resolution	DF			F2F			FS			NT			Avg.
	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	
200×200	0.741	0.855	0.962	0.782	0.862	0.959	0.790	0.794	0.947	0.713	0.812	0.937	0.846
320×320	0.755	0.916	0.937	0.721	0.858	0.948	0.843	0.801	0.939	0.760	0.823	0.931	0.853
256×256	0.757	0.869	0.938	0.798	0.880	0.947	0.802	0.824	0.909	0.752	0.841	0.929	0.854

Table 9: Ablation studies regarding using images with different resolutions.

E Calibration for Better Generalizing

Following the setting in [48], we calibrate the model with a randomly selected pristine and deepfake pair from the test dataset. We first augment the image pair for 128 times to obtain a small calibration set. Then the small calibration set is passed into our detection model to get the logits, which are then fitted by a logistic regression. Similarly, we take the weight and bias learned from the logistic regression to adjust the output of our model. After applying the bias and a sigmoid function to the output logit, we can finally obtain the calibrated probability. Comparisons of accuracies from different methods before and after calibration are listed in Table 11. We observe that our method still outperforms existing methods for both calibrated and uncalibrated results. Meanwhile, we note the calibration does not improve the accuracy in all cases. This phenomenon is consistent with the observations in [48].

F Comparisons with More Methods

We first revise Table 2 by comparing it with a recent meta learning-based method MT3 [5] which uses a contrastive loss between the original sample and its augmentation to update the parameters during inference. We reimplement the method using our framework and train the model using the FF++ dataset. Similarly, it is tested on two benchmarks under three different metrics. The results are listed in Table 12. We observe that our method performs favorably against MT3 in both benchmarks.

Besides TTT[41], we also compare our method with another test time adaptation method TENT [46] which updates the parameters from the batch layer by minimizing the output logits of the detector during inference. We include the results in Table 13. We note that our method obtains more effective performances than the other two in almost all cases. For TTT [41], the main reason for its ineffectiveness is that an inappropriate self-supervised task (i.e., rotation prediction task) may jeopardize the main deepfake detection performance. This corresponds to the observation in [29] . Meanwhile, the small entropy regularization in TENT [46] may not be as effective in the binary classification task as it in the multi class classification task, explaining why its improvements over the baseline model is subtle.

G Visualizations of the Embedded Representations

This section presents the plots of the 2D orthogonal projection of the extracted patterns from our model and that from the baseline model (i.e. Xception [38]). Specifically, we show the t-SNE visualization of the extracted features from the FF++ dataset with different detectors in Figure 5. We observe that both the two models can well separate pristine and deepfake, but the distributions of real and fake representations are rather different. Although the different forgeries are all considered one class, the baseline detector seems to reveal unique artifacts of each face manipulation algorithm as shown in Figure 5 (a). In contrast, the fake representations extracted from our OST model are more mixed together, which indicates our model tends to explore more common features for classification, which can boost generalization when encountering forgeries created by unknown methods.

Dataset (P/N)	DFDC (1533/1613)				DFD (921/7963)				DF1.0 (10050/10050)				Avg.
Metric	TP	TN	FN	FP	TP	TN	FN	FP	TP	TN	FN	FP	TNR
Xception [38]	1410	700	123	913	783	5375	138	2588	8857	2064	1193	7986	0.415
Face X-ray [23]	1288	784	245	829	814	4984	107	2979	9435	3118	615	6932	0.453
F3Net [37]	1387	679	146	934	902	4465	19	3498	9338	4880	712	5170	0.511
RFM [45]	1109	1192	424	421	763	6191	158	1772	9731	2835	319	7215	0.521
SRM [30]	1348	841	185	772	874	5302	47	2661	10037	3712	13	6338	0.502
Ours	1032	1214	501	399	781	6602	140	1361	8336	9816	1714	234	0.898

Table 10: Performances on pristine and deepfake separately. Here TP, TN, FP, and FN represent true positive, true negative, false positive, and false negative, respectively. P/N are the numbers of the positive (pristine) and negative (deepfake) samples in the corresponding dataset.

Method	DFDC		DFD		DF1.0	
	Original	Calibrated	Original	Calibrated	Original	Calibrated
Xception [38]	0.671	0.704	0.693	0.688	0.543	0.622
Face X-ray [23]	0.659	0.674	0.653	0.649	0.625	0.684
F3Net [37]	0.657	0.691	0.604	0.659	0.707	0.693
RFM [45]	0.731	0.702	0.783	0.796	0.625	0.649
SRM [30]	0.696	0.688	0.695	0.714	0.684	0.677
Ours	0.714	0.703	0.831	0.858	0.903	0.881

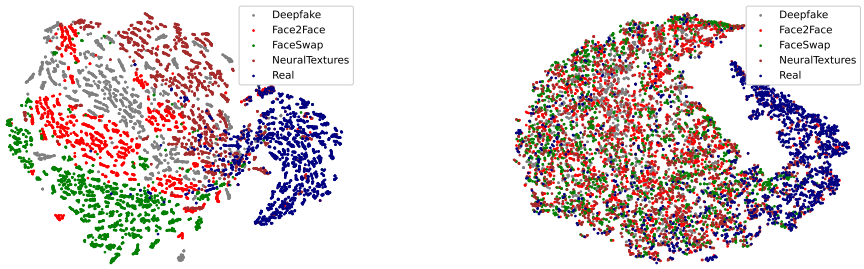
Table 11: Performances comparisons for detectors before and after calibration. We show the accuracy of recent detectors trained on the FF++ dataset and test them on different benchmarks. The calibration refers to the two-shot classifier calibration in [48], and the original results are uncalibrated.

Method	Training dataset	DFDC			CelebDF		
		AUC ↑	ACC ↑	ERR ↓	AUC ↑	ACC ↑	ERR ↓
MLDG [22]	FF++	0.682	0.607	0.370	0.609	0.595	0.418
LTW [40]		0.690	0.631	0.368	0.641	0.634	0.397
MT3 [5]		0.775	0.667	0.307	0.701	0.664	0.319
Ours		**0.833**	**0.714**	**0.250**	**0.748**	**0.673**	**0.312**

Table 12: Revision of Table 2 by adding comparison with MT3 [5].

Method	DF			F2F			FS			NT			Avg.
	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	DFDC	DFD	DF1.0	
Baseline	0.742	0.841	0.910	0.693	0.816	0.924	0.735	0.750	0.913	0.736	**0.848**	0.895	0.817
TTT [41]	0.717	0.862	0.859	0.666	0.820	0.891	0.733	0.684	0.833	0.677	0.825	0.863	0.786
TENT [46]	0.749	0.851	0.915	0.707	0.826	0.916	0.725	0.752	**0.915**	0.748	0.840	0.886	0.819
Ours	**0.757**	**0.869**	**0.938**	**0.798**	**0.880**	**0.947**	**0.802**	**0.824**	0.909	**0.752**	0.841	**0.929**	**0.854**

Table 13: Revision of Table 4 by including comparison with TENT [46].



(a) Baseline (b) Ours

Figure 5: T-SNE visualization of features from different models.