Supplementary Material of Self-supervised Learning of Adversarial Example: Towards Good Generalizations for Deepfake Detection

Liang Chen[1] Yong Zhang[2*] Yibing Song[2] Lingqiao Liu[1*] Jue Wang[2]
[1] The University of Adelaide [2] Tencent AI Lab
{liangchen527, zhangyong201303, yibingsong.cv, arphid}@gmail.com
lingqiao.liu@adelaide.edu.au

In this supplementary material, we provide, 1. Details to generate forgery regions in Section 1.

2. Visualizations of the embedded representations in Section 2;

3. Visualization of the estimated forgery regions of our model in Section 3;

4. Additional experiments regarding the generalizability comparisons under different datasets in Section 4;

5. Additional experiments regarding the generalizability comparisons under different image compression levels in Section 6.

6. Ablation studies regarding the effectiveness of the facial dividing strategy of our method in Section 5.

7. Analyses of the selected weight hyper-parameters (*i.e.* α, μ, and γ in Eq. (4) in the manuscript) in Section 7;

8. Pseudo codes and details in Section 8.

1. Details to generate forgery regions

The forgery regions are generated based on the facial landmarks as stated in the manuscript. Taking the nose forgery region for an example, we first draw a line in an all-zero image with the endpoints that lie at the top and bottom indicated by the landmarks, and the forgery region can be obtained by dilating the line. We will include those details in the revised version.

2. 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 [9]). In particular, we show the t-SNE visualization of features extracted from different classifiers on FF++ test set in Figure 1. We can observe that both two models can well separate real and fake data, while the distributions of fake data are quite different. Even if all forgeries in the original FF++ dataset are considered as one class, they still reveal unique artifacts of each face manipulation algorithm as shown in Figure 1 (a). This may explain the pool generalizability of the baseline model. In contrast, the fake representations extracted from our detector are more mixed together, which indicates our model tends to explore more common features among the forgeries for classification. These observations verify that the proposed adversarial augmentation and self-supervised tasks can indeed boost the generalizability of a deepfake detector.

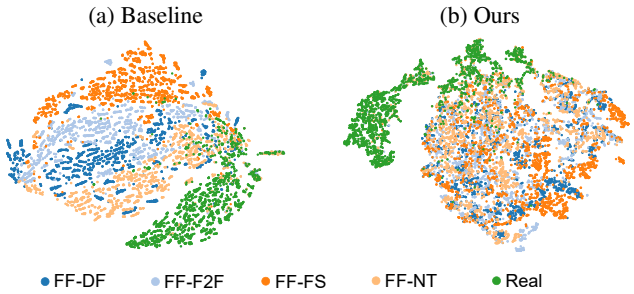

Figure 1. T-SNE visualization of features from different models. Each dot represents the feature of an image on FF++ test set. Representations of forgeries are more mixed in our detector than that in the baseline model (*i.e.* Xception [9]), indicating that the proposed adversarial augmentation and self-supervised tasks enable the model to learn more generalizable features.

3. Visualizations of the Estimated Forgery Regions

In this section, we show the estimated forgery regions of our detector. The visualizations are obtained by upsampling the output forgery map (*i.e.* $\mathbf{M}_e$ in our manuscript with the size of $H/16 \times W/16$) to the size of $H \times W$ to the input size. Some examples are shown in Figure 2. The forgery regions in (a) are not provided in the original dataset and are roughly estimated by comparing the forgeries to the pristines, and

*Corresponding authors. This work is done when L. Chen is an intern in Tencent AI Lab.

Method	DFDC [2]			CelebDF [6]			DF1.0 [4]		
	AUC ↑	AP ↑	ERR ↓	AUC ↑	AP ↑	ERR ↓	AUC ↑	AP ↑	ERR ↓
Xception [9]	0.679	0.716	0.380	0.594	0.715	0.460	0.698	0.807	0.329
Face X-ray [5]	0.700	0.737	0.350	0.742	0.823	0.336	0.723	0.819	0.302
F3Net [8]	0.780	0.808	0.291	0.751	0.829	0.321	0.832	0.847	0.292
RFM [12]	0.801	0.811	0.257	0.743	0.828	0.314	0.815	0.839	0.279
SRM [7]	0.797	0.819	0.299	0.794	**0.861**	0.276	0.738	0.816	0.300
Ours	**0.818**	**0.831**	**0.250**	**0.797**	0.860	**0.275**	**0.918**	**0.930**	**0.252**

Table 1. Generalizability Comparisons Under Different Datasets in terms of AUC (Area Under Curve), AP (Average Precision), and ERR (Equal Error Rate). All methods are trained on the FF++ dataset [9]. Results for [9, 5, 7] are directly cited from [7].

Figure 2. Visualizations of the estimated forgery regions (*i.e.* $\mathbf{M_e}$) of our method.

the forgery regions in (b) are selected in our region selecting space. We can observe that our estimated region maps are close to the ground truth forgery regions in all three cases.

4. Additional Generalizability Comparisons Under Different Datasets

In this section, we provide more experimental results regarding the generalizability comparisons under different datasets. All the models are trained on the FF++ data [9] and tested on three challenging datasets including CelebDF [6], DFDC [2], and DF1.0 [4]. Results are shown in Table 2. We observe that the proposed method performs favorably than others in terms of AUC (Area Under Curve), AP (Average Precision), and ERR (Equal Error Rate).

5. Ablation Studies Regarding the Face Dividing Strategy

As we finely divide a facial image into 10 regions for selecting, questions may be raised whether the selection space is large enough to cover all scenarios or our strategy is better than the other face dividing scheme [5, 13]. To answer these questions, we conduct the following two ablation experiments: (1) We use newly synthesized samples, whose forgery regions are randomly located in the face region, and the forgery regions are also with random sizes (*i.e*. Ours w/ ran. face), to repalce the augmented samples in our original implementation. This setting can cover all facial regions ideally; (2) We replace our facial dividing strategy with that in [5, 13] where the whole inner face is used instead of dividing it to small regions (*i.e*. Ours w/ all face). Note these two settings avoid the selectings of region number but still require selecting the blending types and the mixup blending ratios, and the self-supervised tasks corresponding to them are the same as those in our original implementation.

We also use the four methods in FF++ (*i.e*. DF [1], F2F [11], FS [3], and NT [10]) for training and test the models in three benchmark datasets (*i.e*. CelebDF [6], DFDC [2], and DF1.0 [4]). Comparison results in the term of AUC is reported in Table 2 where our face dividing strategy performs better than random selecting and that from [5, 13]. The main reason is that most deepfake techniques focus on facial features such as eyes, nose, and mouth, and these features are well covered in our selecting space. While other facial regions may not be used as broadly as these facial features in current deepfake methods. Meanwhile, with the help of adversarial training, the samples can select the most challenging region that enables the classifier to learn more generalizable features. From the experiments, we can conclude that the proposed facial region selecting space is large enough for our training process, and it performs than the other face selecting strategy [5, 13].

Algorithm 1 Pseudo codes of our model in a pytorch style.

```
#  I_p, I_f: prisitine and forgery from the original training dataset
#  I_a: newly synthesized adversarial forgery
#  G(·,θ), D(·,w): our synthesizer and detector networks
#  R_g: index for facial regions
#  T_g, T_gt: reference number for blending type and its GT
#  A_g, A_gt: a continuous mixup blending ratio and its GT
#  M_d, M_gt: deformed final mask and its GT
#  α, μ, γ: weight hyper-parameters. They are fixed as 0.1, 0.05, 0.1

for x in loader:  #  load a minibatch x with N samples
    if x is forgery:
        #  the original forgery will skip the synthesizing process
        I_a = x; labels=1;  T_gt = 4  #  M_gt is provided

    else:
        #  randomly select a forgery from the dataset if the
        #  input is a pristine from the original training dataset
        I_p = x;  I_f = randompick(loader)

        #  generate two probability distributions and a scalar
        p(R_g), p(T_g), A_g = G(I_p, I_f, θ)  #  N×10, N×4, N×1

        #  sampling according to the probabilities
        R_g, T_g = sample(p(R_g), p(T_g))  #  N×1, N×1

        #  obtain the final mask based on the selected region by applying
        #  random deformation and blurring with a random sized kernel
        M_d = process(R_g)

        #  defining the labels
        labels=1; M_gt = M_d; T_gt = T_g; A_gt = A_g;
        τ = 0  #  binary weight for computing mixup ratio loss

        #  forgery synthesizing process
        if T_g == 0:  #  alpha blending
            I_a=alpha_blending(I_p, I_f, M_d)
        elif T_g == 1:  #  Poisson blending
            I_a=Poisson_blending(I_p, I_f, M_d)
        elif T_g == 2:  #  mixup blending
            I_a=mixup_blending(I_p, I_f, M_d, A_g);  τ = 1
        else:  #  do nothing when T_g == 3, the input is still pristine
            I_a=I_p;  labels=0;  M_gt=0

    #  four outputs of the detector network
    logits, M_e, T_e, A_e = D(I_a, w)

    #  compute the mainstream loss three self-supervised losses
    loss = AMSoftmaxLoss(logits, labels)+α× MSELoss(M_e, M_gt) + \
           μ×AMSoftmaxLoss(T_e, T_gt) + τ × γ× MSELoss(A_e, A_gt)

    #  SGD updates for the detector network
    loss.backward()
    update(w)

    if T_gt != 4:  #  the synthesizer does not update when not used
        #  compute the discrete loss for the synthesizer network
        logpm = log(p(R_g)).gather(1, R_g.data) + \
                log(p(T_g)).gather(1, T_g.data)

        #  REINFORCE updates for the synthesizer network
        REINFORCE_loss = -loss.detach()×logpm
        REINFORCE_loss.backward()
        update(θ)
```

Method	DF			F2F			FS			NT			Avg.
	DFDC	CelebDF	DF1.0	DFDC	CelebDF	DF1.0	DFDC	CelebDF	DF1.0	DFDC	CelebDF	DF1.0	
Ours w/ ran. face	0.764	0.697	0.663	0.765	0.683	0.761	0.754	0.719	0.687	0.717	0.643	0.806	0.722
Ours w/ all face	0.755	0.713	0.709	0.729	0.694	0.740	0.723	0.730	**0.715**	0.702	0.712	0.862	0.732
Ours	**0.772**	**0.730**	**0.742**	**0.787**	**0.781**	**0.786**	**0.742**	**0.800**	0.695	**0.741**	**0.759**	**0.889**	**0.769**

Table 2. Ablation studies regarding the effectiveness of the face dividing strategy. The metric is AUC. Please see Sec. 5 faced for detailed experimental settings.

Training set	Method	Test set			
		LQ		HQ	
		DF	FS	DF	FS
F2F (LQ)	Xception [9]	0.666	0.504	0.698	0.559
	Face X-ray [5]	0.675	0.511	0.690	0.529
	F3Net [8]	0.698	0.560	0.719	0.578
	RFM [12]	0.699	**0.580**	0.732	0.627
	SRM [7]	0.727	0.552	0.753	0.550
	Ours	**0.746**	0.578	**0.810**	**0.694**
F2F (HQ)	Xception [9]	0.546	0.511	0.749	0.753
	Face X-ray [5]	0.578	0.525	0.662	**0.859**
	F3Net [8]	0.571	0.521	0.798	0.677
	RFM [12]	0.556	0.515	0.794	0.646
	SRM [7]	0.592	0.533	0.792	0.754
	Ours	**0.610**	**0.585**	**0.825**	0.762

Table 3. Additional generalizability comparisons across different compression levels in the term of AUC. Our method achieves comparable performance against existing methods.

Effect of α	Test set			Avg
	DFDC	CelebDF	DF1.0	
$\alpha = 0$	0.794	0.754	0.842	0.797
$\alpha = 0.1$	**0.818**	**0.797**	**0.918**	**0.844**
$\alpha = 1$	0.798	0.789	0.886	0.824
$\alpha = 10$	0.801	0.734	0.872	0.802

Table 4. Effect of α to our model in the term of AUC while training on the FF++ dataset. Other hyper-parameters μ and γ are set to be 0.05 and 0.1 in this setting.

Effect of μ	Test set			Avg
	DFDC	CelebDF	DF1.0	
$\mu = 0$	0.805	0.751	0.859	0.805
$\mu = 0.05$	**0.818**	**0.797**	**0.918**	**0.844**
$\mu = 0.5$	0.817	0.728	0.901	0.815
$\mu = 5$	0.801	0.733	0.892	0.809

Table 5. Effect of μ to our model in the term of AUC while training on the FF++ dataset. Other hyper-parameters α and γ are both set to be 0.1 in this setting.

Effect of γ	Test set			Avg
	DFDC	CelebDF	DF1.0	
$\gamma = 0$	0.803	0.740	0.873	0.805
$\gamma = 0.1$	0.818	**0.797**	**0.918**	**0.844**
$\gamma = 1$	0.804	0.762	0.878	0.815
$\gamma = 10$	**0.827**	0.744	0.871	0.814

Table 6. Effect of γ to our model in the term of AUC while training on the FF++ dataset. Other hyper-parameters α and μ are set to be 0.1 and 0.05 in this setting.

6. Additional Generalizability Comparisons Under Different Compression Levels

Some results regarding the generalizability comparisons under different image compression levels are given in Sec. 4.2 in our manuscript. In this section, we provide more experimental results conducted in the FF++ datasets. Evaluation results are shown in Tabel 3. Our method performs favorably against current state-of-the-art methods.

7. Hyper-Parameter Analyses

The weight hyper-parameters in Eq. (4) in the manuscript are empirically fixed as $\alpha = 0.1$, $\mu = 0.05$, and $\gamma = 0.1$. In this section, we conduct ablation studies to analyze the sensitivity of our model to different settings of these hyper-parameters. We follow the cross-dataset setting by training our model on the FF++ dataset and testing it on CelebDF [6], DFDC [2], and DF1.0 [4]. The hyper-parameters are set by varing one while fixing the other two. Evaluation results in the term of AUC are reported on Table 4, 5, and 6. We observe our model performs the best when adopting the proposed hyper-parameters (*i.e.* by setting $\alpha = 0.1$, $\mu = 0.05$, and $\gamma = 0.1$).

8. Pseudo Codes

Algorithm 1 presents the pseudo codes of the prosed method. Please refer to the comments for detailed steps.

References

[1] DeepFakes. www.github.com/deepfakes/faceswap Accessed 2021-04-24. 3

[2] Deepfake detection challenge. https://www.kaggle.com/c/deepfake-detection-challenge Accessed 2021-04-24. 2, 3, 4

[3] FaceSwap. www.github.com/MarekKowalski/FaceSwap Accessed 2021-04-24. 3

[4] 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. 2, 3, 4

[5] 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. 2, 3, 4

[6] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A new dataset for deepfake forensics. In *CVPR*, 2020. 2, 3, 4

[7] Yuchen Luo, Yong Zhang, Junchi Yan, and Wei Liu. Generalizing face forgery detection with high-frequency features. In *CVPR*, 2021. 2, 4

[8] 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. 2, 4

[9] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *ICCV*, 2019. 1, 2, 3, 4

[10] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *TOG*, 38(4):1–12, 2019. 3

[11] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016. 3

[12] Chengrui Wang and Weihong Deng. Representative forgery mining for fake face detection. In *cvpr*, 2021. 2, 4

[13] Tianchen Zhao, Xiang Xu, Mingze Xu, Hui Ding, Yuanjun Xiong, and Wei Xia. Learning self-consistency for deepfake detection. In *ICCV*, 2021. 3