FDFtNet: Facing Off Fake Images using Fake Detection Fine-tuning Network

Hyeonseong Jeon, Youngoh Bang, and Simon S. Woo

Department of Artificial Intelligence
SungKyunKwan Univesity, Suwon, S. Korea
{cutz, byo, swoo}@g.skku.edu

Abstract. Creating fake images and videos such as "Deepfake" has become much easier these days due to the advancement in Generative Adversarial Networks (GANs). Moreover, recent research such as the few-shot learning can create highly realistic personalized fake images with only a few images. Therefore, the threat of Deepfake to be used for a variety of malicious intents such as propagating fake images and videos becomes prevalent. And detecting these machine-generated fake images has been quite challenging than ever.
In this work, we propose a light-weight robust fine-tuning neural network-based classifier architecture called *Fake Detection Fine-tuning Network* (*FDFtNet*), which is capable of detecting many of the new fake face image generation models, and can be easily combined with existing image classification networks and fine-tuned on a few datasets. In contrast to many existing methods, our approach aims to reuse popular pre-trained models with only a few images for fine-tuning to effectively detect fake images. The core of our approach is to introduce an image-based self-attention module called *Fine-Tune Transformer* that uses only the attention module and the down-sampling layer. This module is added to the pre-trained model and fine-tuned on a few data to search for new sets of feature space to detect fake images. We experiment with our FDFtNet on the GANs-based dataset (*Progressive Growing GAN*) and Deepfake-based dataset (*Deepfake* and *Face2Face*) with a small input image resolution of 64×64 that complicates detection. Our FDFtNet achieves an overall accuracy of 90.29% in detecting fake images generated from the GANs-based dataset, outperforming the state-of-the-art.

Keywords: Fake Image Detection · Neural Networks · Fine-tuning

1 Introduction

The emergence of Generative Adversarial Networks (GANs) [7], which produces high-quality images through a generator and a discriminator that are trained adversely and competitively, enables the generated outputs to be highly realistic and sophisticated [18, 40, 19, 37]. However, such high-quality images and videos generated by machines have been abused (e.g., DeepNude [22]) and harmed the general public (e.g., DeepFake [35]). Furthermore, a recent study using the few-shot learning technique [30] in GAN allows Deep Learning models to produce high-quality outputs with only a small amount of training data. Zakharov et al. [40] demonstrated that models capable of generating highly realistic personalized talking head faces can be constructed using few-shot learning techniques, where the training inputs provide attention to the generator as a compressed

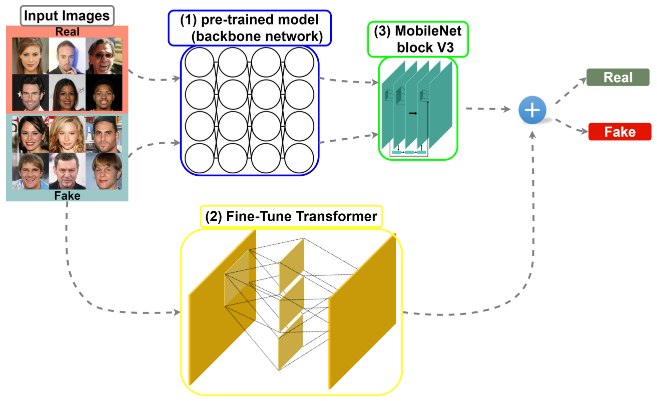
Input Images
Real
Fake
(1) pre-trained model (backbone network)
(3) MobileNet block V3
(2) Fine-Tune Transformer
Real
Fake

Fig. 1: Overview of our FDFtNet. FDFtNet modules are shown in yellow and green: (2) Fine-Tune Transformer to an input image and, (3) MobileNet block V3 is attached to (1) pre-trained model (backbone network), where details of each block is shown in Sec. 3.

form of feature landmarks, extracted through embedding layers. Leveraging this method, DeepFake can easily be generated even with only a small amount of training data. Recently reported incidents [39] related to DeepFake [35] and DeepNude [22] show that these technologies are an imminent threat to the public.

Most of the previous approaches have focused on exploiting metadata information or handcrafted characteristics of images to detect fake images. However, these approaches fail to detect GAN-based fake images, because they are created from scratch and metadata can be also forged; handcrafted features are no longer useful for detection. Recent models, such as *ShallowNet* [32] and *FakeTalkerDetect* [17], used neural networks to detect GANs-generated fake images, and *FaceForensics* [25] showed various forgery detection techniques. However, they lack generalization and will thus have difficulties coping with newly developed DeepFake generation techniques.

In this paper, we propose *Fake Detection Fine-tuning Network* (*FDFtNet*), a new robust fine-tuning neural network-based architecture for fake image detection. FDFtNet combines *Fine-Tune Transformer* (FTT), with a pre-trained Convolutional Neural Network (CNN) as a backbone, and *MobileNet block V3* (MBblockV3). Figure 1 shows an overview of our approach, where we utilize well-known, existing CNN architectures [29, 31, 8, 14, 13, 12] for fake image detection. Our FTT is designed to use different feature extraction from images using the self-attention, and MBblockV3 extracts the feature using different convolution and structure techniques. MBblockV3 is added to the pre-trained model as a backbone network after removing the classification layers. We apply data augmentation by implementing the Cutout method to overcome the limitation of using a small fine-tuning dataset and improve the performance. Our approach provides a reusable fine-tuning network, improving the existing backbone CNN architectures, which were not designed to detect fake images effectively.

We experiment with three datasets, i.e., the GAN-based dataset (CelebFaces Attributes Dataset (CelebA) [21] + Progressive Growing GAN (PGGAN) [18]) and the

Deepfake-based dataset (FaceForensics [1] + Deepfake [35] and Face2Face [33]), and four baseline models, i.e., SqueezeNet [15], ShallowNetV3 [32], ResNetV2 [9], and Xception [2]. Our result shows that FDFtNet, combined with Xception, achieves an accuracy of 97.02%, which is higher than that of the state-of-the-art methods. Our main contributions are as follows:

– We propose FDFtNet, a novel neural network-based fake image detector, showing superior performance on detecting fake images compared to previous approaches by achieving 97.02% accuracy.
– We provide a robust fine-tuning neural network-based classifier, which requires only small amount data for fine-tuning and can be easily integrated with popular CNN architectures.
– We improve the baseline model accuracy from 4% to 45% through our FTT and MBblockV3, where FTT is not explored in other image classification and fine-tuning research.

2 RELATED WORK

Traditional image forgery detection methods. Many researchers [5, 20, 38, 23] have investigated various digital forensics algorithms to detect forged images. One way to detect forged images is to analyze them in the frequency domain. However, it is difficult to analyze images with refined smooth edges, thus giving rise to a different method, such as JPEG Ghost [5]. In JPEG Ghost, the forged part is regularly copied from different real images. The normalized pixel distance of the reproduced image differs from the original image, causing a difference in JPEG quality. However, this method will not work if the original image and the forged image have the same quality level. Another approach is Error Level Analysis (ELA) [20], which checks the error level of the images. However, with GANs-generated fake images, ELA cannot classify the error level between the real and generated images. Another algorithm called the Copy-move Forgery detection [23] is based on Pixel Based approach. Firstly, dyadic wavelet transform (DWT) is applied to the input image. This transforms the original image to an image of a reduced dimension representation, i.e., the LL1 sub-band. Then this LL1 sub-band is divided into sub-images. To compute the spatial offset between the Copy-move regions, the phase correlation is adopted. The Copy-Move regions can easily be located by pixel matching, which shifts the input image according to the offset, and calculate the difference between its shifted version and the original image. In the final step, the Mathematical Morphological Operations (MMO) are used to remove isolated points to improve the location. Traditional digital forensic tools fail to detect GANs-generated images, because they are generated as a single image. For this reason, these approaches are not effective.
Image forgery detection with neural networks. Various CNN-based models have been used to detect forged images. ShallowNet [32] outperformed previous architectures in detecting real vs. PGGAN with a shallow layer architecture. However, their approach showed limitations when detecting other types of DeepFake images. FaceForensic++ [26] proposed a forgery detection method tailored to facial manipulations and provided an extensive evaluation in a supervised manner. They also generated a large facial manipulations dataset based on computer graphics-based methods, such as *Face2Face* [33] and

FaceSwap [36]. In addition, they introduced an automatic metric that takes into account the four forms of distortion in realistic scenarios (i.e., random encoding and random dimensions). Using these benchmarks, they analyzed various methods of forgery detection pipeline. However, transfer learning or fine-tuning capabilities were not explored.
Self-attention and Transformer. To achieve long-term dependencies on image data, CNN needs to increase the amount of computation via deeper layers, because one-time convolution computation sees only the convolution kernel size. In contrast, self-attention solves this long-term dependency issue by using the softmax outputs of the entire sequence that provide attention to CNN. Zhang et al. [41] used self-attention modules to generate images with GANs. Our FTT is different in that we build only self-attention modules, such as Transformer, during the feature extraction in the classification tasks. We apply FTT for the image feature extractor and not for generator. This approach is similar to the *Multi-head Attention Module* [34] (Query, key, and Value), but the difference is that FTT is suitable for the image to be applied to the 1×1 convolution.

3 Fake Detection Fine-tuning Network (FDFtNet)

In this section, we describe the architecture design of our FDFtNet. The main difference from other fake detection methods is that we utilize well-known, reusable pre-trained models and fine-tune the backbone networks with only a few data to improve the fake detection performance. Figure 1 shows an overview of our model, which is composed of 1) a pre-trained model, 2) Fine-Tune Transformer (FTT), and 3) MobileNet V3 blocks (MBblockV3). First, we describe the three datasets that are used in our experiment. Second, we explain why we use pre-trained models. Third, we describe our two modules, *Fine-Tune Transformer* and *MobileNet blocks V3*.

3.1 Dataset Description

CelebA. CelebFaces Attributes Dataset (CelebA) [21] is a large-scale face attributes dataset with more than 200,000 celebrity images. It is widely used for benchmarking and as inputs for generating training and test datasets for various GAN and VAE approaches. We use CelebA as an input to generate PGGAN fake images.
PGGAN. For the GAN-generated image, we used Progressive Growing GANs Dataset (PGGAN), consisting of 100,000 GAN-generated fake celebrity images at 1024×1024 resolution using the CelebA dataset. The key idea in PGGAN is to grow both the generator and discriminator progressively. The training starts with both the generator and discriminator having a low resolution. New layers are added as the training process advances, thus increasing the resolution of the generated images.
FaceForensics. FaceForensics [1] is a video dataset comprised of more than 500,000 frames, containing faces from 1004 videos that can be used to study image or video forgeries. All videos are cut down to short continuous clips that contain mostly frontal faces. We use these datasets as the source for the Deepfakes [35] and Face2Face [33] fake image generation.

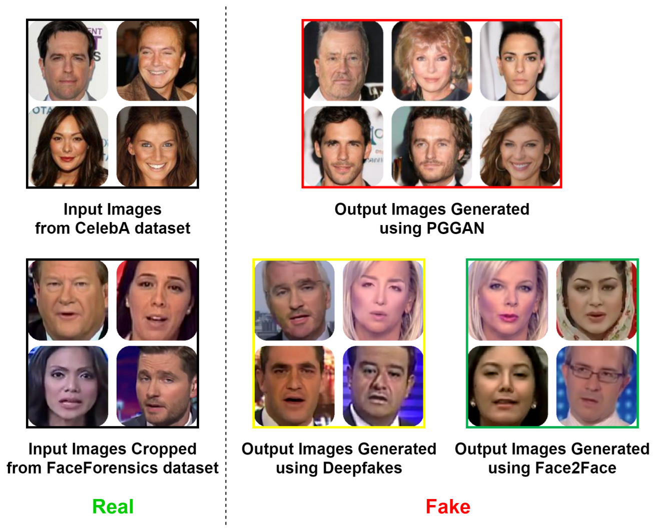

Fig. 2: Illustration of our datasets. CelebA [21] images are used as inputs for PGGAN [18] fake image generation. Images from the FaceForensics [25] dataset are cropped and used as input images for Deepfakes [35] and Face2Face [33] fake image generation.

Deepfakes. Deepfakes [35] was the first publicly available method, which anyone can download and use to produce fake images and videos. The code is based on two autoencoders with a shared encoder. To produce a forged image, the trained encoder and decoder of the source are applied to the target image face. The output of the autoencoder is then blended with the target image. For our experiment, we used the *FaceSwap* [36] implementation.
Face2Face. The goal of Face2Face [33] is to animate the facial expressions of the target video by a source actor and re-render the manipulated output video in a photo-realistic fashion. Face2Face first addresses the under-constrained problem of facial identity recovery from monocular video by non-rigid model-based bundling. At run time, the model tracks facial expressions of both the source and target videos using a dense photometric consistency measure. Reenactment is then achieved by fast and efficient deformation transfer between the source and the target. The mouth interior that best matches the re-targeted expression is retrieved from the target sequence and warped to produce an accurate fit. Finally, Face2Face re-renders the synthesized target face on top of the corresponding video stream such that it seamlessly blends with the real-world illumination. Since our goal is to detect fake images, we use each frame from the generated Face2Face output.

3.2 Description of Pre-trained Backbone CNN networks

We use the following pre-trained CNN networks as our backbone networks as shown in Fig. 1, as well as our baselines: SqueezeNet, ShallowNetV3, ResNetV2, and Xception.

They are the backbone networks to be used for fine-tuning in our architecture. Therefore, our goal is to compare these baselines with our approaches that use these backbone pre-trained networks.

SqueezeNet. SqueezeNet [15] has an AlexNet-level accuracy with fewer parameters and would generally have poor performance in fake detection tasks, because SqueezeNet is not designed for fake detection. We chose SqueezeNet as the baseline, because our FDFtNet can provide a huge improvement.

ShallowNetV3. ShallowNetV3 [32] has the highest area under receiver operating characteristic (AUROC) (93.99%) on 64×64 resolution images from the CelebA and PGGAN datasets. However, ShallowNetV3 has burdensome fully-connected layers (FC layer) for binary classification. Convolution layers have 115,490 parameters, while FC layers have 4,725,762 parameters. In addition, since this approach has not been tested on deepfakes other than those generated by PGGAN, we aim to investigate the performance.

ResNetV2. ResNetV2 has been widely adopted in many image classification tasks. We chose ResNetV2 [9] as one of the baselines, because ResNetV2 has an opposing characteristic to ShallowNetV3 in terms of the model depth, i.e., ResNetV2 has 50 layers, while ShallowNetV3 has only 8 layers. We believe that these two architectures would show complementary results and we plan to see the effect of our approach on such deep and shallow CNN architectures.

Xception. Xception [2] has been served as the baseline for fake image detection in [32, 26]. For FaceForenscis++, Xception showed the highest accuracy, i.e., 96.36% in Deepfake and 86.86% in Face2Face, justifying our choice of it as a baseline. In addition, Xception has no FC layers, but extracts various image feature spaces thanks to *depthwise separable convolutions*, compared to the burdensome FC layers in the ShallowNetV3. We cut the classification layers in a pre-trained model, and add our FTT and MBblockV3 modules.

3.3 Fine-Tune Transformer (FTT)

Fine-Tune Transformer (FTT) consists of several self-attention modules, as shown in Fig. 3, where each attention module has $f(x)$, $g(x)$, and $h(x)$ using a 1×1 convolution filter. We iterate M times from the image inputs. M is a hyper-parameter and we empirically determined that $M = 3$ yields the highest performance.

$$\begin{aligned} &f(x) = W_f x, \qquad g(x) = W_g x, \qquad h(x) = W_h x, \\ &\beta_{j,i} = Softmax\left(f(x_i)^T g(x_j)\right). \end{aligned} \tag{1}$$

In Fig. 3, the input x of the previous layers or the input image is divided into three feature spaces $f(x)$, $g(x)$, and $h(x)$. As shown in Eq. 1, all of them are obtained through the 1×1 convolution, where W_f, W_g, and W_h are the respective filter weights of each space. $f(x)$ and $g(x)$ have b channel bottleneck ratio parameter, $\frac{C}{b}$, where C is the number of channels. In this study, we choose $b = 8$ as suggested by Zhang et al. [41]. In particular, we use the dot-product attention to produce the attention map β in Fig. 3, synthesizing the i^{th} and j^{th} locations after the $Softmax$ operation as shown in the above equation.

Input size	Operation	Num. Parameters	Output dim.	Stride
$W \times H \times C$	1×1 Conv	16	C / b	1
$W \times H \times C$	1×1 Conv	16	C / b	1
$W \times H \times C$	1×1 Conv	16	C	1
$W \times H \times C / b$	Matmul	0	$W \times H$	-
$WH \times WH$	Softmax	0	$W \times H$	-
$WH \times WH$	Batchdot	0	C	-
$W \times H \times C$	Multiply	1	C	-
$W \times H \times C$	Add	0	C	-

Table 1: Specification for the self-attention module. Conv denotes convolution, W, H, and C define the input size for the previous layer, and b denotes the bottleneck ratio in the block. The number of parameters are simulated with the following hyperparameters: $W = H = C = 64$ and $b = 8$.

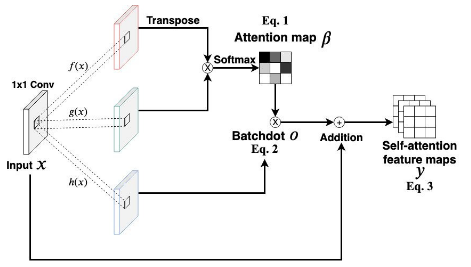

Fig. 3: Self-attention module in the Fine-Tune Transformer. The input x (the image or the output from the previous layer) is divided by a 1×1 convolution into f, g, and h. The attention map β is the softmax output from f and g. The batchdot o multiplies h and the attention map β. The input image x is added to o. The final output y is the self-attention feature maps.

$$o_j = Batchdot\left(\beta_{j,i}\ , h\left(x\right)\right) \tag{2}$$

$$y_i = \gamma o_j + x_i. \tag{3}$$

After obtaining the attention map β, we apply the *Batchdot* operation to multiply the attention map $\beta_{j,i}$ with $h(x)$, as shown in Eq. 2, and produce output o_j. After the *Batchdot* multiplication, o_j is added to the input x_i. Finally, the self-attention feature map, y_i, is obtained via multiplying γ and adding the input x_i, as shown in Eq. 3. In particular, γ is a learnable parameter initialized as 0 at the early stage of learning. This is favorable, since the softmax function equally provides attention to all the feature spaces at the early stage of learning.

Next, in our FTT, we apply the self-attention module three times (M = 3) with an input size of 64×64×3, as shown in Table 2. The first layer is a 3×3 separable convolution with 32 filters and 2 strides followed by Batch Normalization (BN) [16] and ReLU. The dimension of the output feature map from the self-attention module is 32, 64, and 128, respectively; the width (number of channels) is doubled when the resolution

Input size	Operation	Num. Parameters	Output dim.	Stride
$64 \times 64 \times 3$	3×3 DConv	123	32	2
$32 \times 32 \times 32$	BN	128	32	-
$32 \times 32 \times 32$	ReLU	0	32	-
$32 \times 32 \times 32$	**1st Stage self-attention**	1,321	**32**	-
$32 \times 32 \times 32$	3×3 DConv	2,336	64	2
$16 \times 16 \times 64$	BN	256	64	-
$16 \times 16 \times 64$	ReLU	0	64	-
$16 \times 16 \times 64$	**2nd Stage self-attention**	5,201	**64**	-
$16 \times 16 \times 64$	3×3 DConv	8,768	128	2
$8 \times 8 \times 128$	BN	512	128	-
$8 \times 8 \times 128$	ReLU	0	128	-
$8 \times 8 \times 128$	**3rd Stage self-attention**	20,641	**128**	-
$8 \times 8 \times 128$	1x1 Conv	73,728	576	1
$8 \times 8 \times 576$	BN	2,304	576	-
$8 \times 8 \times 576$	ReLU	0	576	-
$8 \times 8 \times 576$	GAP	0	576	-

Table 2: Specification for Fine-Tune Transformer (FTT). Conv, BN, DConv, and GAP denote convolution, batch normalization, depth-wise separable convolution, and global average pooling operation, respectively. The "Attention" operation in bold indicates the end of one transformer block. We repeat FTT three times ($M = 3$) to maximize the performance.

is down-sampled, as shown in Table 2. After that, self-attention is performed three times ($M = 3$), followed by SeparableConv3×3, BN, and ReLU. The main reason we apply self-attention modules in FTT is to overcome the limitations of CNN in achieving long-term dependencies, caused by the use of numerous Conv filters with a small size. On the other hand, only a one-time use of the FTT is necessary to achieve the long-term dependencies, avoiding the construction of deep CNN layers. Also, a three-time application of self-attention modules allows us to explore and learn diverse deep features of the input images via fine-tuning.

3.4 MobileNet block V3

We chose MobileNet block V3 (MBblockV3) to explore the image feature space through inverted residual structure and linear bottleneck [27]. Depthwise separable convolutions, as in Xception and MobileNetV1 [12], are also included in MBblockV3. Comprehensively, MobileNet is an architecture that has already proven its efficiency by using a small number of parameters, drastically increasing computational efficiency. We chose MBblockV3, because it is a suitable module for the efficient extraction of the feature space over the pre-trained feature space. FTT and MBblockV3 are repeatedly used M and N times, respectively. Each of them is added before the final classification layer. MBblockV3 has the parameter N after the pre-trained model. In our experiment, we use $N = 4$, determined empirically, yielding the best performance for fine-tuning. In particular, we use the modified *h-swish* [11] and the ReLU6 as activation functions. This non-linearity [24, 4, 10] significantly improves the performance of neural networks and is defined as follows:

$$\text{h-swish[x]} = x\frac{\text{ReLU6}\left(x+3\right)}{6}, \text{ where ReLU6[x]} = min\left(max\left(0,x\right),6\right). \tag{4}$$

Input size	Operation	Num. Parameters	Output dim.	Stride
$W \times H \times C$	1×1 Conv	294,912	576	1
$W \times H \times 576$	BN	2,304	576	-
$W \times H \times 576$	h-swish	0	576	-
$W \times H \times 576$	3×3 DConv	5,184	576	1 or 2
$W \times H \times 576$	BN	2,304	576	-
$W \times H \times 576$	**GAP**	0	576	-
$1 \times 1 \times 576$	**1×1 Conv**	82,944	144	1
$1 \times 1 \times 144$	**ReLU**	0	144	-
$1 \times 1 \times 144$	**1×1 Conv**	82,944	576	1
$1 \times 1 \times 576$	**hard-sigmoid**	0	576	-
$1 \times 1 \times 576$	**Multiply**	0	576	-
$W \times H \times 576$	h-swish	0	576	-
$W \times H \times 576$	1×1 Conv	73,728	128	1
$W \times H \times 128$	Linear	0	128	-
$W \times H \times 128$	BN	2,304	128	-
$W \times H \times 128$	Add	0	128	-

Table 3: Specification for MBblockV3 with $W = H = 8$ and $C = 256$. Conv, BN, DConv and GAP denote convolution, batch normalization, depth-wise separable convolution, and global average pooling. W, H and C indicate input size. If the stride of 3x3 DConv is 2, the addition operation is skipped, and W and H are divided by 2. Bold operations represent the Squeeze-and-Excitation block.

Since clipping the input values at the bottom layers may have a side effect of distorting the data distribution [28], we apply these activation functions at the top layers to reduce distortion and extract different signals from ReLU. Next, the *Squeeze-and-Excitation blocks* (SE block) in Squeeze and Excitation networks [13] are applied in the bottleneck layer. Global information on the image resolution is embedded in the squeeze stage, and information aggregation is used to capture channel dependencies and is re-calibrated through the gated computation (element-wise multiplication), similar to the attention mechanism in the excitation stage. Details of the SE block parameters are summarized in Table 3.

4 Experimental Results

4.1 Training details

Dataset	Train	Validation	Test	**Fine-tune**
PGGAN	128,404	32,100	37,566	**1,000 (real), 1,000 (fake)**
Deepfake	60,000	18,000	20,000	**1,000 (real), 1,000 (fake)**
Face2Face	60,000	18,000	20,000	**1,000 (real), 1,000 (fake)**

Table 4: The respective size of the train, validation, test, and fine-tune sets. We use only 1,000 real and fake images, respectively, for fine-tuning.

All datasets have train, validation, test, and fine-tune sets. The size of each dataset is shown in Table 4. Our training set is only fine-tuned with 1,000 samples for real and

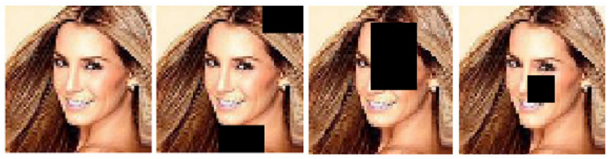

Fig. 4: Example of a Cutout data augmentation. Random regions of the original image (left) are masked out by black rectangles. Every epoch, the rectangular mask changes in form and all images are resized to 64×64 resolution.

fake images, respectively, and we use the validation set to check the training strategy. Our FDFtNet is trained with Stochastic Gradient Descent (SGD) with momentum for 300 epochs on all 3 datasets. The learning rate is initialized at 0.3 and annealed using a cosine function. The momentum rate is set to 0.9 and the mini-batch size is set to 128. Early stopping is applied, when the validation loss ceases to decrease for 20 epochs. All other weight parameters are initialized based on the research by Glorot and Bengio [6]. To reenact the most challenging scenario in detecting fake images, all input images are resized to 64×64 resolution.

Data augmentation. Because we just use 1,000 samples for each class, we apply data augmentation. Input images are translated into a width and height range of [-2, 2] with the nearest-padding on empty pixels generated after translation. Zoom and rotation are also applied to a degree range of [-0.2, 0.2]. We also perform random horizontal flipping. These data augmentations are applied to all fine-tune sets. For validation and test sets, only a 1 / 255 scaling augmentation to the input image is applied.

Cutout. Cutout method applies squared zero masks on a random location of each input image. Fig. 4 presents an example of a Cutout data augmentation. DeVries et al. [3] used random zero masks of 16 pixels for CIFAR-10 (32×32 pixels images), 5 random iteration parameters α for cutting, and 16 random size multipliers β for the cutting masks. We use 4×4 pixels mask, 3 iterations, and 5-size multipliers for cutting masks for 64×64 images (α = 3 and β = 5). Since we use random translation, we do not use random center cropping, which was used in the original paper. When we conducted with the original setting, we faced severe underfitting with no convergence of losses. We observed higher performance with a setting of low Cutout parameters (α = 3 and β = 5) as compared to the implementation without Cutout, which showed strong overfitting. Because we fine-tune with a small amount of data, we apply this non-aggressive parameter setting.

4.2 Performance evaluation

We present our overall performance results in Table 5, where the same test datasets (see Table 4) are used for evaluating the baselines and our model. In Table 5, the baselines and the backbones are interchangeably used, where the backbone is the pre-trained CNN network used in our FDFtNet. We use the accuracy (ACC) and area under the receiver operating characteristic (AUROC) as evaluation metrics for our experiments. We experimented with all four baseline models on each dataset with similar training strategies for each dataset. As shown in Table 5, the experimental results show that

Model	Dataset	PGGAN		Deepfake		Face2Face	
	Backbone	ACC (%)	AUROC	ACC (%)	AUROC	ACC (%)	AUROC
SqueezeNet	baseline	50.00	50.00	50.00	50.00	50.00	50.00
FDFtNet (Ours)	SqueezeNet	88.89	92.76	92.82	97.61	87.73	94.20
ShallowNetV3†	baseline	85.73	92.90	89.77	92.81	83.35	88.49
FDFtNet (Ours)	ShallowNetV3	88.03	94.53	94.29	97.83	84.55	93.28
ResNetV2	baseline	84.80	88.58	81.52	89.72	58.83	62.47
FDFtNet (Ours)	ResNetV2	84.83	94.05	91.03	96.08	85.15	92.91
Xception	baseline	87.12	94.96	95.10	98.92	85.78	93.67
FDFtNet (Ours)	Xception	**90.29**	**95.98**	**97.02**	**99.37**	**96.67**	**98.23**

Table 5: Overall performance evaluation results. The evaluation metrics used are accuracy (ACC / %) and area under receiver operating characteristic (AUROC / %). The underlined results are improved performance compared to the baseline and the best detection results among all are highlighted in bold.

our FDFtNet has superior detection performance in both ACC and AUROC, compared to all the baselines. In terms of training data size, our model shows high performance using 1,000 images for real and fake, respectively. We will now explain the detailed performance improvement for each dataset.

PGGAN. To yield the best detection performance, we freeze the weight parameters of all layers of the pre-trained models. FTT with parameter $M = 3$ is used, and MBblockV3 with parameter $N = 2$ is added; the same data augmentation is applied. Table 5 shows the results of our models compared with the baseline models. Our results show that Xception, among all baseline models, achieved the highest performance (87.12% ACC and 94.96% AUROC). Our model showed a performance of 90.29% ACC and 95.98% AUROC, which is higher than that of ShallowNetV3 with ensemble [32]. ShallowNetV3 is improved from 85.73% and 92.90% ACC to 88.03% and 94.53% AUROC, respectively, similar to the ensemble version. SqueezeNet baseline shows the lowest baseline performance, but it is significantly improved to a similar level to that of ShallowNetV3, from 50.00% to 92.76%, by applying our model.

Deepfake. Here also, the same data augmentation techniques are applied. For FTT, we use $M = 3$ and $N = 4$ for MBblockV3. Cutout has $\alpha = 3$ iteration parameters and $\beta = 10$ multiplier parameters. The results show that all models achieve significant improvement in performance. Table 5 indicates that Xception has the highest performance of 95.10% ACC and 98.92% AUROC. Using our approach, this baseline model is also improved to 97.02% ACC and 99.37% AUROC. ShallowNetV3 has 89.77% ACC and 92.81% AUROC. They increased to 94.29% ACC and 97.83% AUROC, respectively. ResNetV2 is also improved from 81.52% ACC and 89.72% AUROC to 91.03% ACC and 96.08% AUROC. SqueezeNet baseline shows the lowest performance, 50.00% ACC and AUROC, but is improved to 92.82% ACC and 97.61% AUROC.

Face2Face. The training strategies for Face2Face are very similar to those of the Deepfake dataset. Data augmentation is also applied. M, N, α, and β are set to 3, 4, 3, and 10, respectively. The interesting point is that ResNetV2 baseline performed poorly (58.83% ACC and 62.47% AUROC), but significant improvements are made using our methods

(85.15% ACC and 92.91% AUROC). Our results demonstrate the generalization ability of our approach, improving the poorly performing baseline above 90% across all models and datasets. Compared to *FaceForensics Benchmark Results* [1], the highest state-of-the-art method is Xception, which shows 96.4% ACC in Deepfake and 86.9% ACC in Face2Face. Our FDFtNet achieves higher performance (97.02% and 96.67%) than the current state-of-the-art method for the same dataset.

5 Ablation study, discussions, and limitations

Method	backbone	Dataset	Acc	AUROC
With FTT	Xception	Deepfake	**97.02%**	**99.37%**
Without FTT	Xception	Deepfake	94.56%	98.89%

Table 6: Ablation study for Fine-Tune Transformer (FTT). Our models with FTT has 2.46% higher accuracy (ACC) than those without FTT, increasing the ACC from 94.56% to 97.02%.

In Section 3.3, we explained the reason for using FTT. In this section, we validate each module and technique through an ablation study. In Table 6, we choose the Xception model and the Deepfake dataset to compare our model with and without the FTT, while all other settings remain the same. With FTT, we can achieve about 2.5% higher performance than without FTT as shown in Table 6. Our current work has the following limitations: First, we used both real and fake data for training and fine-tuning, but we have constrained resources in practice. In FakeTalkerDetect [17] for fake detection, researchers used Siamese networks for training only on real data. However, in our implementation, few-shot learning and unbalanced learning are major obstacles to achieving high performance. Second, transfer learning is required to improve the performance. We trained each model on each dataset, e.g., PGGAN, Deepfake, and Face2Face. For future work, we plan to research the transfer learning ability to further generalize our model.

6 Conclusion

We propose FDFtNet, which is a robust fine-tuning neural network-based architecture, to detect fake images and significantly improve the baseline CNN architectures. Our model achieves the state-of-the-art accuracy in fake image detection on the GAN-based dataset and the Deepfake-based dataset. Our experimental results with the use of a limited amount of data show the exploration and exploitation of image feature space beyond the pre-trained models. Our results show that FDFtNet is a promising method for detecting fake images generated by powerful deep learning methods, requiring only a small amount of images for re-training. Therefore, FDFtNet can be a viable option even for detecting new fake images in a real-world scenario, where available datasets are extremely limited. Further, we offer open source versions of our work for it to be widely leveraged by the research community [1].

[1] `https://anonymous.4open.science/r/FDFtNet/`

References

1. This table lists the benchmark results for the binary classification scenario. `http://kaldir.vc.in.tum.de/faceforensics_benchmark/` (2019)
2. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1251–1258 (2017)
3. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
4. Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Networks **107**, 3–11 (2018)
5. Farid, H.: Exposing digital forgeries from jpeg ghosts. IEEE transactions on information forensics and security **4**(1), 154–160 (2009)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016)
10. Hendrycks, D., Gimpel, K.: Bridging nonlinearities and stochastic regularizers with gaussian error linear units (2016)
11. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. arXiv preprint arXiv:1905.02244 (2019)
12. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
14. Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., Keutzer, K.: Densenet: Implementing efficient convnet descriptor pyramids. arXiv preprint arXiv:1404.1869 (2014)
15. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
16. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
17. Jeon, H., Bang, Y., Woo, S.S.: Faketalkerdetect: Effective and practical realistic neural talking head detection with a highly unbalanced dataset. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 0–0 (2019)
18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
19. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
20. Krawetz, N., Solutions, H.F.: A picture's worth... Hacker Factor Solutions **6**, 2 (2007)
21. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of the IEEE international conference on computer vision. pp. 3730–3738 (2015)

22. lwlodo: Official deepnude algorithm source code. https://github.com/lwlodo/deep_nude (2019)
23. Mankar, S.K., Gurjar, A.A.: Image forgery types and their detection: A review. International Journal of Advanced Research in Computer Science and Software Engineering. **5**(4) (2015)
24. Ramachandran, P., Zoph, B., Le, Q.V.: Swish: a self-gated activation function. arXiv preprint arXiv:1710.05941 **7** (2017)
25. Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., Nießner, M.: Faceforensics: A large-scale video dataset for forgery detection in human faces. arXiv preprint arXiv:1803.09179 (2018)
26. Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., Nießner, M.: Faceforensics++: Learning to detect manipulated facial images. arXiv preprint arXiv:1901.08971 (2019)
27. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
28. Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., Aleksic, M.: A quantization-friendly separable convolution for mobilenets. In: 2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2). pp. 14–18. IEEE (2018)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
30. Sun, Q., Liu, Y., Chua, T.S., Schiele, B.: Meta-transfer learning for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 403–412 (2019)
31. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
32. Tariq, S., Lee, S., Kim, H., Shin, Y., Woo, S.S.: Gan is a friend or foe?: a framework to detect various fake face images. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 1296–1303. ACM (2019)
33. Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., Nießner, M.: Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2016)
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
35. Wikipedia: Deepfake. https://en.wikipedia.org/wiki/Deepfake (2019), [Online; accessed 15-July-2019]
36. Wikipedia: Deepfakes github. https://github.com/deepfakes/faceswap (2019), [Online; accessed 15-October-2019]
37. Wu, J., Huang, Z., Acharya, D., Li, W., Thoma, J., Paudel, D.P., Van Gool, L.: Sliced wasserstein generative models. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019), https://arxiv.org/pdf/1706.02631.pdf
38. Yang, X.: Estimating distribution costs with the e aton–k ortum model. Review of Development Economics **19**(3), 653–665 (2015)
39. Yin, C.: Altering faces via ai deepfake may be outlawed. China Daily (Apr 2019), http://global.chinadaily.com.cn/a/201904/22/WS5cbd15c4a3104842260b76c8.html
40. Zakharov, E., Shysheya, A., Burkov, E., Lempitsky, V.: Few-shot adversarial learning of realistic neural talking head models. arXiv preprint arXiv:1905.08233 (2019)
41. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. arXiv preprint arXiv:1805.08318 (2018)