APSIPA Transactions on Signal and Information Processing, 2022, 11, e41

Original Paper

DefakeHop++: An Enhanced Lightweight Deepfake Detector

Hong-Shuo Chen[1]*, Shuowen Hu[2], Suya You[2] and C.-C. Jay Kuo[1]

[1] *University of Southern California, Los Angeles, CA, USA*
[2] *DEVCOM Army Research Laboratory, Adelphi, MD, USA*

ABSTRACT

On the basis of DefakeHop, an enhanced lightweight Deepfake detector called DefakeHop++ is proposed in this work. The improvements lie in two areas. First, DefakeHop examines three facial regions (i.e., two eyes and mouth) while DefakeHop++ includes eight more landmarks for broader coverage. Second, for discriminant features selection, DefakeHop uses an unsupervised approach while DefakeHop++ adopts a more effective approach with supervision, called the Discriminant Feature Test (DFT). In DefakeHop++, rich spatial and spectral features are first derived from facial regions and landmarks automatically. Then, DFT is used to select a subset of discriminant features for classifier training. As compared with MobileNet v3 (a lightweight CNN model of 1.5M parameters targeting at mobile applications), DefakeHop++ has a model of 238K parameters, which is 16% of MobileNet v3. Furthermore, DefakeHop++ outperforms MobileNet v3 in Deepfake image detection performance in a weakly-supervised setting.

Keywords: Deepfake detection, green learning, green AI, weakly-supervised learning.

*Corresponding author: Hong-Shuo Chen, hongshuo@usc.edu.

Received 29 April 2022; Revised 25 August 2022
ISSN 2048-7703; DOI 10.1561/116.00000126

1 Introduction

It is common to see fake videos appearing on social media platforms nowadays due to the popularity of Deepfake techniques. Several mobile apps can help people create fake content without any special editing skills. Generally speaking, deepfake programs can change the identity of one person in real videos to another realistically and easily. The number of Deepfake videos surges rapidly in recent years. Fake videos may result in serious damage to our society since people can be fooled by videos on the Internet, and some misinformation may make the public panic and anxious. To address this emerging threat, it is essential to develop lightweight Deepfake detectors that can be deployed on mobile phones.

With the fast growing Generative Adversarial Network (GAN) technology, image forgery techniques keep evolving in recent years. They are effective in reducing manipulation traces detectable by human eyes. It becomes very challenging to distinguish Deepfake images from real ones with human eyes against new generations of Deepfake technologies. Furthermore, adding different kinds of perturbation (e.g., blur, noise and compression) can hurt the detection performance of Deepfake detectors since manipulation traces are mixed with perturbations. A robust Deepfake detector should be able to tell the difference between real images and fake images generated by the GAN techniques although both of them experience such modifications.

There have been three generations of fake video datasets created for research and development purposes. They demonstrate the evolution of Deepfake techniques. The UADFV dataset [20] belongs to the first generation. It has only 50 video clips generated by one Deepfake method. Its real and fake videos can be easily detected by humans. FaceForensics++ [27] and Celeb-DF-v2 [21] are examples of the second generation. They contain more video clips with more identities. It is difficult for humans to distinguish real and fake faces for them. DFDC [9] is the third generation dataset. It contains more than 100 K fake videos which are generated by 8 Deepfake techniques and perturbed by 19 kinds of distortions as shown in Figure 1. The size of the third generation dataset is very large. It is designed to test the performance of various Deepfake detectors in an environment close to real world applications.

State-of-the-art Deepfake detectors usually use the deep neural networks (DNNs) to solve the Deepfake detection problem. Although they offer high detection performance, their model sizes are so large that they cannot be deployed on mobile phones. For example, the winning team of the DFDC contest [31] used seven pre-trained EfficientNets that contain 432 million parameters. In this research, our goal is to develop a lightweight model with its size less than 256K. For a machine learning model with its size less than 256K, it can be run in any terminal device with a limited amount of memory such as Raspberry Pi.

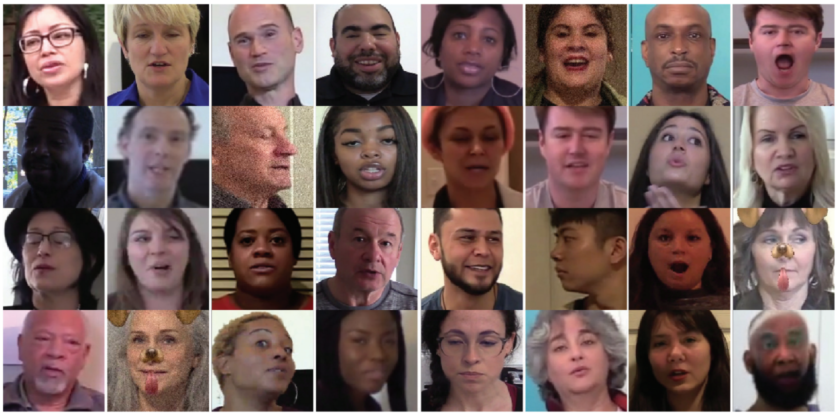

Figure 1: Visualization of real and fake faces extracted from the third generation DFDC dataset [9]. The left and right four columns depict real and fake faces, respectively. Eight Deepfake techniques are used to generate fake videos. Furthermore, 19 perturbations are added to real and fake videos. Exemplary perturbations include compression, additive noise, blur, change of brightness, contrast and resolution, and overlay with flower and dog patterns, random faces and images. A good Deepfake detector should be able to distinguish real and fake videos with or without perturbation.

It is easy to generate fake video content and disseminate over the Internet and social media nowadays, e.g., video sharing on Facebook, Instagram, WeChat, YouTube, etc. They are not monitored or censored by cloud servers. As a result, fake videos can be transmitted among different communities. The motivation for developing a deepfake detector for mobile development is to ensure that every mobile device can detect deepfake videos without the help from cloud servers. One application scenario is to develop an app that can be installed on a browser to flag detected fake videos generated by Deepfake algorithms.

A lightweight Deepfake detector called DefakeHop was developed in [2]. An enhanced version of DefakeHop, called DefakeHop++, is proposed in this work. The improvements lie in two areas. First, DefakeHop examines three facial regions (i.e., two eyes and mouth) only. DefakeHop++ includes eight more landmark regions to offer more information on human faces. Second, DefakeHop uses an unsupervised energy criterion to select discriminant features. It does not exploit the correlation between features and labels. DefakeHop++ adopts a supervised tool, called the Discriminant Feature Test (DFT), in feature selection. The latter is more effective than the former due to supervision. In DefakeHop++, spatial and spectral features from multiple facial regions and landmarks are generated automatically and, then, DFT is used to select a subset of discriminant features to train a classifier. As compared with MobileNet v3, which is a lightweight CNN model of 1.5M parameters targeting mobile applications, DefakeHop++ has an even smaller model size of 238K parameters (i.e., 16% of MibileNet v3). In terms of Deepfake image detection performance, DefakeHop++ outperforms MobileNet v3 without data augmentation and leverage of pre-trained models.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. DefakeHop++ is described in detail in Section 3. Experimental results are shown in Section 4. Finally, concluding remarks are given in Section 5.

2 Review of Related Work

2.1 Deepfake Detection

Most state-of-the-art Deepfake detection methods use DNNs to extract features from faces. Their models are trained with heavy augmentation (e.g., deleting part of faces) to increase the performance. Despite the high performance of these models, their model sizes are usually very large. They have to be pre-trained by other datasets in order to converge. Several examples are given below. The model of the winning team of the DFDC Kaggle challenges [31] has 432M parameters. Heo *et al.* [14] improved this model by concatenating it with a Vision Transformer(VIT), which has 86M parameters. Zhao *et al.* [43] proposed a model that exploits multiple spatial attention heads to learn various local parts of a face and the textural enhancement block to learn subtle facial artifacts. To reduce the model size and improve the efficiency, Sun *et al.* [32] proposed a robust method based on the change of landmark positions in a video. These landmarks are calibrated by the neighbor frames. Afterwards, a two-stream RNN is trained to learn the temporal information of the landmark position. Since they did not consider the image information and only consider the position information, the model size is 0.18M which is relatively small. Tran *et al.* [35] applied MobileNet to different facial regions and InceptionV3 to the entire faces. Although this model is smaller, it still demands 26M parameters.

2.2 DefakeHop

To address the challenge of the need of huge model sizes and training data, a new machine learning paradigm called green learning has been developed in the last 6 years [5, 6, 17, 18]. Its main goal is to reduce the model size and training time while keeping high performance. Green learning has been applied to different applications, e.g., [15, 22, 24, 28, 29, 39–42]. Based on green learning, DefakeHop was developed in [2] for the Deepfake detection task. It first extracts a large number of features from three facial regions using PixelHop++ [5] and then refines them using feature distillation modules.

Finally, it feeds distilled features to the XGBboost classifier [4]. The DefakeHop model has only 42.8 K parameters, yet it outperforms many DNN solutions in detection accuracy against the first and second generataion Deepfake datasets. Recently, DefakeHop has been used to detect fake satellite images in [3].

3 DefakeHop++

DefakeHop++ is an improved version of DefakeHop. An overview of the DefakeHop++ system is shown in Figure 2. Facial blocks of two sizes are first extracted from frames in a video sequence in the pre-processing step. These blocks are then passed to DefakeHop++ for processing. DefakeHop++ consists of four modules: (1) one-stage PixelHop, (2) spatial PCA, (3) discriminant feature test (DFT), and (4) Classifier. The pre-processing step and the four modules are elaborated below.

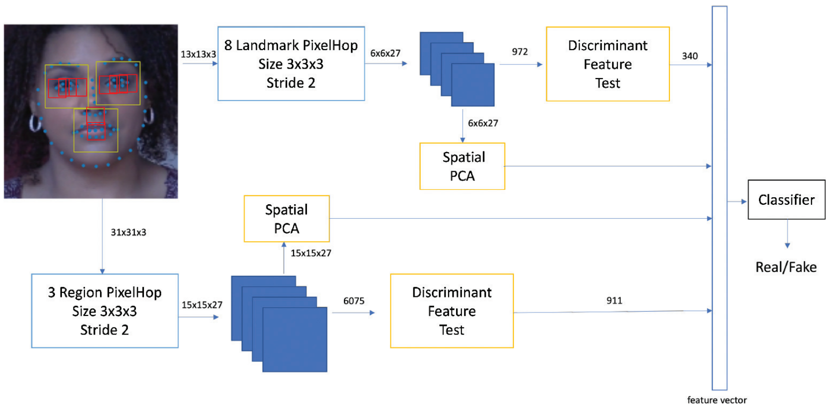
13x13x3
8 Landmark PixelHop Size 3x3x3 Stride 2
6x6x27
972
Discriminant Feature Test
340
6x6x27
Spatial PCA
Spatial PCA
15x15x27
31x31x3
3 Region PixelHop Size 3x3x3 Stride 2
15x15x27
6075
Discriminant Feature Test
911
Classifier
Real/Fake
feature vector

Figure 2: An overview of DefakeHop++.

3.1 Pre-processing Step

Frames are extracted from video sequences. For training videos, we extract three frames per second. Since the video length is typically around 10 s, we obtain around 30 frames per video. On the other hand, the length and the frame number per second (FPS) are not fixed in test videos. We uniformly sample 100 frames from each test video.

Facial landmarks are obtained by OpenFace2. With 68 facial landmarks, we crop faces with 30% of margin and resize them to 128×128 with zero padding (if needed). We do not align the face since face alignment may distort the original face image and the side face is difficult to align. We conduct experiments on the discriminant power of each landmark and find that two eyes and the mouth are the most discriminant regions. Thus, we crop out 8 smaller blocks that cover 6 representative landmarks from two eyes, one from the nose and one from the mouth. Furthermore, we crop out three larger blocks to cover the left eye, right eye and mouth. We make the block size a hyper-parameter for user to choose. For the experiments reported in Section 4, we adopt smaller blocks of size 13×13 centered at landmarks and larger blocks of size 31×31 for three facial regions (i.e., two eyes and the mouth) since they give the best results. The extracted small and large blocks are illustrated in Figure 3.

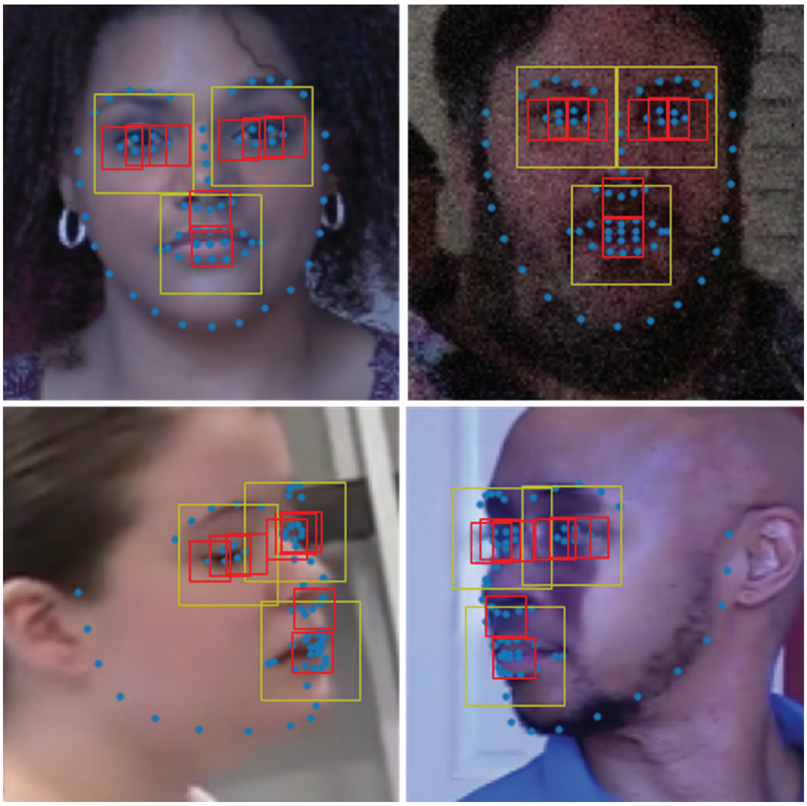

Figure 3: Illustration of the prepossessing step. It first extracts 68 landmarks (in blue) from the face. Then, it crops out blocks of size 31×31 from three regions (in yellow) and blocks of size 13×13 from eight landmarks (in red). The step can extract consistent blocks despite different head poses and perturbations.

3.2 Module 1: One-Stage PixelHop

A PixelHop unit [5, 6] contains a set of filters used to extract features from training blocks. While the filter weights of traditional filter banks (e.g., the Gabor or Laws filters) are fixed, those of the PixelHop are data dependent. Its filter banks are defined by the Saab transform, which decomposes a local neighborhood (i.e., a patch) into one DC (direct current) and multiple AC (alternated current) components. The DC component is the mean of each patch. The AC components are obtained through the principal component analysis (PCA) of patch residuals.

To give an example, we set the patch size to 3×3 in the experiment. For the color face image input, each patch contains $(3 \times 3) \times 3$ degrees of freedom, including nine spatial pixel locations and three spectral components. By collecting a large number of patches, we can conduct PCA to obtain AC filters. The eigenvectors and the eigenvalues of the covariance matrix correspond to AC filters and their mean energy values, respectively. Since most natural images are smooth, the leading AC filters extract low-frequency features of higher energy. When the frequency index goes higher, the energy of higher frequency components decays quickly. The Saab filter bank decouples the 27 correlated input dimensions into 27 decorrelated output dimensions. Besides decorrelating input components, the Saab transform allows to discard some high frequency channels due to their very small variances (or energy values).

The horizontal output size of a block can be calculated as

$$\frac{\text{horizontal block size} - \text{horizontal filter size}}{s} + 1, \tag{1}$$

where s is the stride parameter along the horizontal direction. For example, if the horizontal block size is 13, the horizontal filter size is 3 and the horizontal stride is 2, then the horizontal output size is equal to 6. The same computation applies to the vertical output size.

3.3 Module 2: Spatial PCA

The output of the same frequency component may still have correlations in the spatial domain. This is especially true for the DC component and low AC components. The correlation can be removed by spatial PCA. The idea of spatial PCA is inspired by the eigenface method in [36]. For each channel, we train a spatial PCA and keep the leading components that have cumulative energy up to 80% of the total energy. This helps reduce the output feature dimension.

Modules 1 and 2 describe the feature generation procedure for DefakeHop++. It is worthwhile to point out the differences in feature generation of DefakeHop and DefakeHop++.

- DefakeHop only focuses on two eyes and mouth three regions. Besides these three regions, DefakeHop++ zooms into the neighborhood of 8 landmarks called a block to gain more detailed information. The justification of these 8 landmarks is given in the last paragraph of Section 4.1
- DefakeHop conducts three-stage PixelHop units and applies spatial PCA to the response output of all three stages. The pipeline is simplfied to a one-stage PixelHop in DefakeHop++. Yet, the simplification does not hurt the performance since the spatial PCA applied to each block (or region) still offer the global information of the corresponding block (or region). Note that such simplification is needed as DefakeHop++ covers more spatial patches and regions.

3.4 Module 3: Discriminant Feature Test

DefakeHop selects responses from channels of larger energy as features into a classifier under the assumption that features of higher variance are more discriminant. This is an unsupervised feature selection method. Recently, a supervised feature selection method called the discriminant feature test (DFT) was proposed in [38]. DFT provides a powerful method in selecting discriminant features using training labels. The DFT process can be simply stated below. For each feature dimension, we define an interval using its minimum and maximum values across all training samples. Then, we partition the interval into two sub-intervals at all possible split positions which are equally spaced in the interval. For example, if we may select 31 split positions uniformly distributed over the interval. For each partitioning, we can use the maximum likelihood principle to assign a predicted label to each training sample and compute the cross entropy of all training samples accordingly. The split position that gives the lowest cross entropy is chosen to be the optimal split of the feature and the associated cross entropy is used as the cost function of this feature. The lower the cross entropy, the most discriminant the feature dimension. We refer to [38] for more details.

We use Figures 4 and 5 to explain the DFT idea. Two features are compared in Figure 4. The feature in the left subfigure has a lower cross entropy value than the one in the right subfigure. The left one is more discriminant than the right one, which is intuitive. Figure 5 is used to describe the feature selection process for a given landmark block. The feature dimension of one landmark block is 972. We perform DFT on each feature and obtain 972 cross entropy values. The y-axis in both subfigures is the cross entropy value while the x-axis is the channel index. In the left subfigure, a smaller channel index indicates a lower frequency channel. We see that discriminant channels of lower cross entropy actually spread out in both low and high frequency channels. In the right plot, we sort channels based on their cross entropy values, which have an elbow point at 350. As a result, we can reduce the feature number from 972 to 350 (35%) by selecting 350 features with the lowest cross entropy.

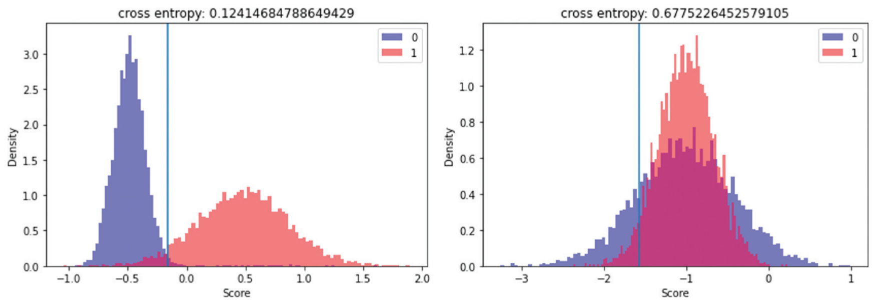

Figure 4: Illustration of the feature selection idea in the discriminant feature test (DFT). For the feature dimension associated with the left subfigure, samples of class 0 and class 1 can be easily separated by the blue partition line. Its cross entropy is lower. For the feature dimension associated with the right subfigure, samples of class 0 and class 1 overlap with each other significantly. It is more difficult to separate them and its cross entropy is higher. Thus, the feature dimension in the left subfigure is preferred.

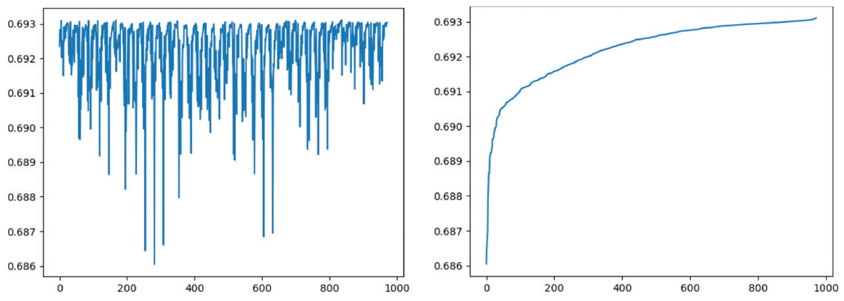

Figure 5: Illustration of the feature selection process for a single landmark, where the y-axis is the cross entropy of each dimension and the x-axis is the channel index. The left and right subfigures show unsorted and sorted feature dimensions.

3.5 Module 4: Classification

In DefakeHop, soft decisions from different regions are used to train several XGBoost classifiers. Then, another ensemble XGBoost classifier is trained to make the final decision. This is a two-stage decision process. We find that a lot of detailed information is lost in the first-stage soft decisions and, as a result, the ensemble classification idea is not effective. To address this issue, we simply collect all feature vectors from different regions and landmarks, apply

DFT for discriminant feature selection and train a LightGBM classifier in the last stage in DefakeHop++. LightGBM and XGBoost play a similar role. The main difference between DefakeHop and DefakeHop++ is that the former is a two-stage decision process while the latter is a one-stage decision process. The one-stage decision process can consider the complicated relations of features in landmarks and regions across all frequency bands. Once all frame-level predictions for a video are obtained, we simply use their mean as the final prediction for the video.

4 Experiments

We compare the detection performance of DefakeHop++, state-of-the-art deep learning and non-deep learning methods on several datasets as well as their model sizes and training time in this section to demonstrate the effectiveness of DefakeHop++.

4.1 Experimental Setup

4.1.1 Datasets

Deepfake video datasets can be categorized into three generations based on the dataset size and Deepfake methods used for fake image generation. The first-generation datasets includes UADFV and FF++. The second-generation datasets include Celeb-DF version 1 and version 2. The third-generation dataset is the DFDC dataset. The datasets of later generations have more identities of different races, utilize more deepfake algorithms to generate fake videos, and add more perturbation types to test videos. Apparently, the later generation is more challenging than the earlier generation. The datasets used in our experiments are described below.

- **UADFV** [20]
  UADFV is the first Deepfake detection dataset. It consists of 49 real videos and 49 fake videos. Real videos are collected from YouTube while fake ones are generated by the FakeApp mobile App [11].

- **FaceForensics++ (FF++)** [27]
  It contains 1000 real videos collected from YouTube. Two popular methods, FaceSwap [10] and Deepfakes [8], are used in fake video generation. Each of them generated 1000 fake videos of different quality levels, e.g., RAW, HQ (high quality) and LQ (low quality). In the experiment, we focus on HQ compressed videos since they are more challenging to detect by Deepfake detection algorithms.

- **Celeb-DF** [21]
  Celeb-DF has two versions. Celeb-DF v1 contains 408 real videos from YouTube and 795 fake videos. Celeb-DF v2 consists of 890 real and 5639 fake videos. Fake videos are created by an advanced version of DeepFake. These videos contain subjects of different ages, ethnicities and sex. Celeb-DF v2 is a superset of Celeb-DF v1. Celeb-DF v1 and v2 have been widely tested on many Deepfake detection methods.

- **DFDC** [9]
  DFDC is the third generation dataset. It contains more than 100K videos generated by 8 different Deepfake algorithms. The test videos are perturbed by 19 distractors and augmenters such as change of brightness/contrast, logo overlay, dog filter, dots overlay, faces overlay, flower crown filter, grayscale, horizontal flip, noise, images overlay, shapes overlay, change of coding quality level, rotation, text overlay, etc. The dataset was generated to mimic the real-world application scenario. It is the most challenging one among the four benchmark datasets.

4.1.2 Evaluation Metrics

Each Deepfake detector assigns a probability score of being a fake one to all test images. Then, these scores can be used to plot the Receiver Operating Characteristic (ROC) curve. Then, the area under curve (AUC) score can be used to compare the performance of different detectors. We report the AUC scores at the frame level as well as the video level.

4.1.3 Discriminability Analysis of Landmarks

As mentioned in Section 3.1, there are 68 landmarks. We use the AUC scores to analyze the discriminability of the 68 landmark regions in Figure 6. We see from the figure that the performance of different landmarks varies a lot. Landmarks in two eye regions are most discriminant. This is attributed to the fact that eyes have rich details and complex movement and, as a result, they cannot be well synthesized by any Deepfake algorithms. The mouth region is the next discriminant one because it is difficult to synthesize lip motion and teeth. It is worthwhile to point out that the cheek and nose regions are not very useful in separating real and fake images. This could be explained that the cheek or nose regions do not contain complex texture information and Deepfake algorithms can reproduce them realistically. This justifies our choice of six landmarks from eyes, one landmark from the nose and one landmark from the mouth as described in Section 3.1.

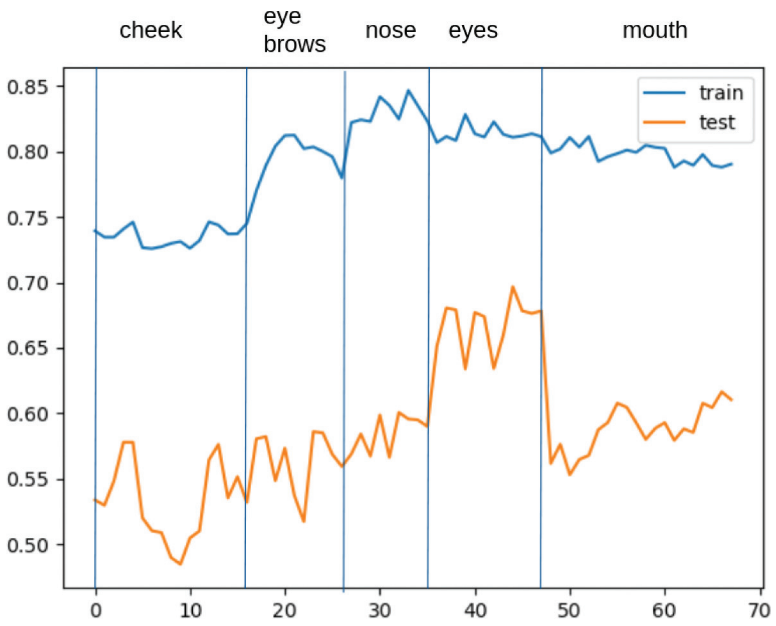

Figure 6: Analysis of landmark discriminability, where the x-axis is the landmark index anad the y-axis is the AUC score. Landmarks in the two eye regions are most discriminant in both training and test datasets.

4.2 Detection Performance Comparison

4.2.1 First Generation Datasets

The performance of a few Deepfake detectors on two first generation datasets, UADFV and FF++, is compared in Table 1. Both DefakeHop and DefakeHop++ achieve a perfect AUC score of 100% against UADFV. DSP-FWA and FWA are two closer ones with AUC scores of 97.7% and 97.4%, respectively. Actually, UADFV is an easy dataset where the visual artifacts are visible to human eyes. It is interesting to see that the extremely large models do not reach perfect detection results. This could be explained by the small size of UADFV (i.e., 49 real videos and 49 fake videos). The model sizes of DefakeHop and DefakeHop++ can be sufficiently trained by a small dataset size. For FF++, Multi-attentional achieves the best AUC score (i.e., 99.8%) while Xception-raw and Xception-c23 achieves the second best (i.e., 99.7%). DefakeHop++ with its AUC computation at the video level has the next highest AUC score (i.e., 99.3%). The performance gap among them is small. Furthermore, the performance of Xception-raw and Xception-c23 is boosted by a larger dataset size of FF++, which has 1000 real videos of different quality levels.

Table 1: Comparison of detection performance of several methods on the first genreation datasets with AUC as the performance metric. The AUC results of DefakeHop++ in both frame-level and video-level are given. The best and the second-best results are shown in boldface and underbared, respectively. The AUC results of benchmarking methods are taken from [21] and the number of parameters are from https://keras.io/api/applications. Also, we use ^{a} to denote deep learning methods and ^{b} to denote non-deep-learning methods.

		1st Generation		
Method	Model	UADFV	FF++	#param
Two-stream [44]	InceptionV3^a[33]	85.1%	70.1%	23.9M
Meso4 [1]	Designed CNN^a	84.3%	84.7%	28.0K
MesoInception4 [1]	Designed CNN^a	82.1%	83.0%	28.6K
HeadPose [37]	SVM^b	89.0%	47.3%	–
FWA [20]	ResNet-50^a[12]	97.4%	80.1%	25.6M
VA-MLP [23]	Designed CNN^a	70.2%	66.4%	–
VA-LogReg [23]	Logistic Regression^b	54.0%	78.0%	–
Xception-raw [27]	XceptionNet^a[7]	80.4%	<u>99.7%</u>	22.9M
Xception-c23 [27]	XceptionNet^a[7]	91.2%	<u>99.7%</u>	22.9M
Xception-c40 [27]	XceptionNet^a[7]	83.6%	95.5%	22.9M
Multi-task [25]	Designed CNN^a	65.8%	76.3%	–
Capsule [26]	CapsuleNet^a[30]	61.3%	96.6%	3.9M
DSP-FWA [19]	SPPNet^a[13]	<u>97.7%</u>	93.0%	–
Multi-attentional [43]	Efficient-B4^a[34]	–	**99.8%**	19.5M
DefakeHop [2]	DefakeHop^b	**100%**	96.0%	42.8K
Ours (Frame Level)	DefakeHop++^{b}	**100%**	98.4%	238K
Ours (Video Level)	DefakeHop++^{b}	**100%**	99.3%	238K

4.2.2 Second Generation Datasets with Cross-Domain Training

The performance of several Deepfake detectors, which are trained on the FF++ dataset, against two second generation datasets is compared in Table 2. We see that video-level DefakeHop++ gives the best AUC score while frame-level DefakeHop++ gives the second best AUC score for Celeb-DF-v1. As to Celeb-DF-v2, Multi-attentional yield the best AUC score while Xception-raw and Xception-c23 offer the second best scores. DefakeHop++ is slightly inferior to them. Furthermore, we show the performance of DefakeHop and DefakeHop++ under the same domain training in the last four rows of Table 2. Their performance has improved significantly. Video-level DefakeHop++ outperforms video-level DefakeHop by 2.5% in Celeb-DF v1 and 6.1% in Celeb-DF v2.

Table 2: Comparison of detection performance of several Deepfake detectors on the second genreation datasets under cross-domain training and with AUC as the performance metric. The AUC results of DefakeHop anad DefakeHop++ in both frame-level and video-level are given. The best and the second-best results are shown in boldface and underbared, respectively. Furthermore, we include results of DefakeHop and DefakeHop++ under the same-domain training in the last 4 rows. The AUC results of benchmarking methods are taken from [21] and the number of parameters are from https://keras.io/api/applications. Also, we use ^{a} to denote deep learning methods and ^{b} to denote non-deep-learning methods.

		2nd Generation		
Method	Model	Celeb-DF v1	Celeb-DF v2	#param
Two-stream [44]	InceptionV3^a	55.7%	53.8%	23.9M
Meso4 [1]	Designed CNN^a	53.6%	54.8%	28.0K
MesoInception4 [1]	Designed CNN^a	49.6%	53.6%	28.6K
HeadPose [37]	SVM^b	54.8%	54.6%	–
FWA [20]	ResNet-50^a	53.8%	56.9%	25.6M
VA-MLP [23]	Designed CNN^a	48.8%	55.0%	–
VA-LogReg [23]	Logistic Regression^b	46.9%	55.1%	–
Xception-raw [27]	XceptionNet^a	38.7%	48.2%	22.9M
Xception-c23 [27]	XceptionNet^a	–	65.3%	22.9M
Xception-c40 [27]	XceptionNet^a	–	65.5%	22.9M
Multi-task [25]	Designed CNN^a	36.5%	54.3%	–
Capsule [26]	CapsuleNet^a	–	57.5%	3.9M
DSP-FWA [19]	SPPNet^a	–	<u>64.6%</u>	-
Multi-attentional [43]	Efficient-B4^a	–	**67.4%**	19.5M
Ours (Frame Level)	DefakeHop++^{b}	<u>56.30%</u>	60.5%	238K
Ours (Video Level)	DefakeHop++^{b}	**58.15%**	62.4%	238K
Ours (Trained on Celeb-DF, Frame Level)	DefakeHop^b	93.1%	87.7%	42.8K
Ours (Trained on Celeb-DF, Video Level)	DefakeHop^b	95.0%	90.6%	42.8K
Ours (Trained on Celeb-DF, Frame Level)	DefakeHop++^{b}	<u>95.4%</u>	<u>94.3%</u>	238K
Ours (Trained on Celeb-DF, Video Level)	DefakeHop++^{b}	**97.5%**	**96.7%**	238K

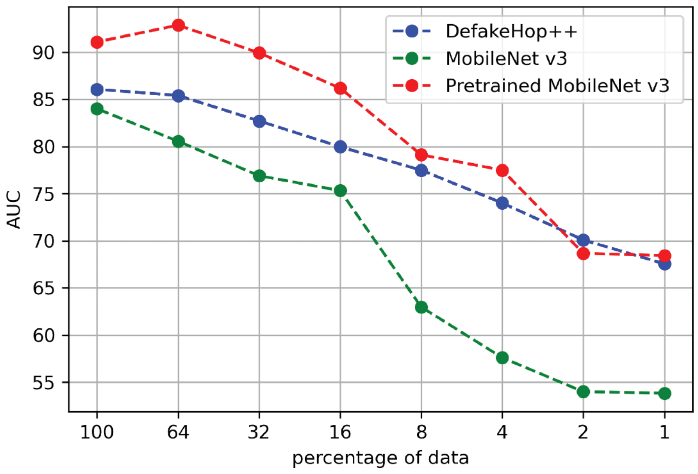

Figure 7: Detection performance comparison of DefakeHop++, MobileNet v3 with pre-training by ImageNet and MobileNet v3 without pre-training as a function of training data percentages of the DFDC dataset.

4.2.3 Third Generation Dataset

The size of the third generation dataset, DFDC, is huge. It demands a lot of computational resources (including training hardware, time and large models) to achieve high performance. Since our main interest is on lightweight detection algorithms, we focus on the comparison of DefakeHop++ and MobileNet v3, which has 1.5M parameters and targets at mobile applications. We train both models with parts and/or all of DFDC training data and report the detection performance on the test dataset in Figure 7. We have the following three observations from the figure. First, pre-trained MobileNet v3 gives the best result, DefakeHop++ the second, and MobileNet v3 without pre-training the worst. It shows that, if there are sufficient training data in training, the detection performance of a larger model can be boosted. For the same reason, the detection performance of all three models decreases as the training data of DFDC becomes less. Second, with 1–8% of DFDC training data, the performance of DefakeHop++ and pre-trained MobileNet v3 is actually very close. The performance gap between DefakeHop++ and MobileNet v3 without pre-training is significant in all training data ranges. For example, with only 1% of the DFDC training data, the AUC score of DefakeHop++ reaches 68% while that of MobileNet V3 can only reach 54%. Third, with 100% of the DFDC training data but without any data augmentation, DefakeHop++ still can achieve an AUC score of 86%, which is 5% lower than pre-trained MobileNet v3.

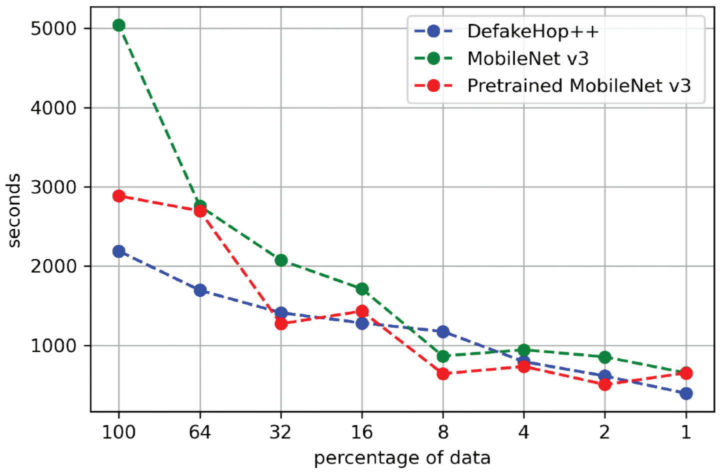

Figure 8: Training time comparison of DefakeHop++, MobileNet v3 with pre-training by ImageNet and MobileNet v3 without pre-training as a function of training data percentages of the DFDC dataset, where the training time is in the unit of seconds. The training time does not include that used in the pre-processing step.

Furthermore, we compare the training time on the three models as a function of the training data percentage in Figure 8. The model is trained on AMD Ryzen 9 5950X with Nvidia GPU 3090 24G. If a CNN is not pre-trained, it generally needs more time to converge. The training time of DefakeHop++ is lowest for 64% and 100% of the total training data. Its training time is about the same as the pre-trained MobileNet v3 for the cases of 1-32% training data.

4.3 Model Size of DefakeHop++

DefakeHop++ consists of one-stage PixelHop, Spatial PCA, DFT and Classifier modules. The size of each component can be computed as follows.

- **PixelHop.** The parameters are the filter weights. Each filter has a size of $(3 \times 3) \times 3 = 27$. Since there are 27 filters, the total number of parameters of PixelHop is $27 \times 27 = 729$ parmaeters.
- **Spatial PCA.** For each channel, we flatten 2D spatial responses to a 1D vector and train a PCA. We conduct spatial PCA on channels of higher energy (i.e. those with cumulative energy up to 80% total energy). Furthermore, we set an upper limit of 10 channels to avoid a large number of filters for a particular channel. Based on this design guideline, the averaged channel numbers for a landmark and a spatial region are 35 and 40, respectively.

- **DFT.** DFT is used to select a subset of discriminant features. Its parameters are indices of selected channels. For 8 landmark blocks, we keep features in top 35%. For 3 spatial regions, we keep features in top 15%. Then, the number of parameters of DFT is $(8 \times 340) + (3 \times 911) = 5,453$ as shown in Table 3.
- **Classifier.** We use LightGBM [16] as the classifier and set the maximum number of leaves to 64. As a result, the maximum intermediate node number of a tree is bounded by 63. We store two parameters (i.e., the selected dimension and its threshold) at each intermediate node and one parameter (i.e., the predicted soft decision score) at each leaf node. The number of parameters for one tree is bounded by 190. Furthermore, we set the maximum number of tree to 1000. Thus, the number of parameters for LightGBM is bounded by 190K.

Table 3: The number of parameters for various parts.

	Subsystem	Number	Parameters	Total
Pixelhop	Landmarks	8	$3 \times 3 \times 3 = 729$	5832
	Regions	3	$3 \times 3 \times 3 = 729$	2187
Spatial PCA	Landmarks	8	$6 \times 6 \times 35 = 1,260$	10,080
	Regions	3	$15 \times 15 \times 40 = 9,000$	27,000
DFT	Landmarks	8	$6 \times 6 \times 27 \times 0.35 = 340$	2720
	Regions	3	$15 \times 15 \times 27 \times 0.15 = 911$	2733
LightGBM	–	1	190,000	190,000
Total				**237,832**

4.4 Inference time of DefakeHop and DefakeHop++

Since the data preprocessing step of DefakeHop++ and DefakeHop as described in Section 3.1 is essentially the same, we focus on the comparison of the inference time in Modules 1–4 as stated in Sections 3.2–3.5 and report the results in Table 4. As shown in the table, the average inference time per video for DefakeHop++ and DefakeHop is 46.5 msec and 92.2 msec, respectively.

Table 4: Comparison of inference time of DefakeHop and DefakeHop++.

Method	Inference time per video (ms)	Speed up
DefakeHop	92.2	1×
DefakeHop++	46.5	2×

The inference time of DefakeHop++ is significantly less because it uses only one PixelHop stage while DefakeHop uses three PixelHop stages.

5 Conclusion and Future Work

A lightweight Deepfake detection method, called DefakeHop++, was proposed in this work. It is an enhanced version of our previous solution called DefakeHop. Its model size is significantly smaller than that of state-of-the-art DNN-based solutions, including MobileNet v3, while keeping reasonably high detection performance. It is most suitable for Deepfake detection in mobile/edge devices.

Fake image/video detection is an important topic. The fake content is not restricted to talking head videos. There are many other application scenarios. Examples include fake satellite images, image splicing, image forgery in publications, etc. Heavyweight fake image detection solutions are not practical. Furthermore, fake images can appear in many forms. On one hand, it is unrealistic to include all possible perturbations in the training dataset under the setting of heavy supervision. On the other hand, the performance could be quite poor with little supervision. It is essential to find a midground and look for a lightweight weakly-supervised solution with reasonable performance. This paper shows our research effort along this direction. We will continue to explore and generalize the methodology to other challenging Deepfake problems.

References

[1] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: A Compact Facial Video Forgery Detection Network," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2018, 1–7.

[2] H.-S. Chen, M. Rouhsedaghat, H. Ghani, S. Hu, S. You, and C.-C. J. Kuo, "DefakeHop: A Light-Weight High-Performance Deepfake Detector," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2021, 1–6.

[3] H.-S. Chen, K. Zhang, S. Hu, S. You, and C.-C. J. Kuo, "Geo-DefakeHop: High-Performance Geographic Fake Image Detection," *arXiv preprint arXiv:2110.09795*, 2021.

[4] T. Chen and C. Guestrin, "Xgboost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, 785–94.

[5] Y. Chen and C.-C. J. Kuo, "PixelHop: A Successive Subspace Learning (SSL) Method for Object Classification," *arXiv preprint arXiv:1909.08190*, 2019.

[6] Y. Chen, M. Rouhsedaghat, S. You, R. Rao, and C.-C. J. Kuo, "PixelHop++: A Small Successive-Subspace-Learning-Based (SSL-based) Model for Image Classification," *arXiv preprint arXiv:2002.03141*, 2020.

[7] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, 1251–8.

[8] "Deepfakes github," https://github.com/deepfakes/faceswap, 2018.

[9] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. C. Ferrer, "The Deepfake Detection Challenge (DFDC) Dataset," *arXiv preprint arXiv:2006.07397*, 2020.

[10] "FaceSwap github," https://github.com/MarekKowalski/FaceSwap, 2018.

[11] "Fakeapp," https://www.fakeapp.com, 2018.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 770–8.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 2015, 1904–16.

[14] Y.-J. Heo, Y.-J. Choi, Y.-W. Lee, and B.-G. Kim, "Deepfake Detection Scheme based on Vision Transformer and Distillation," *arXiv preprint arXiv:2104.01353*, 2021.

[15] P. Kadam, M. Zhang, S. Liu, and C.-C. J. Kuo, "R-PointHop: A Green, Accurate and Unsupervised Point Cloud Registration Method," *arXiv preprint arXiv:2103.08129*, 2021.

[16] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A Highly Efficient Gradient Boosting Decision Tree," *Advances in Neural Information Processing Systems*, 30, 2017.

[17] C.-C. J. Kuo, "Understanding Convolutional Neural Networks with a Mathematical Model," *Journal of Visual Communication and Image Representation*, 41, 2016, 406–13.

[18] C.-C. J. Kuo, M. Zhang, S. Li, J. Duan, and Y. Chen, "Interpretable Convolutional Neural Networks via Feedforward Design," *Journal of Visual Communication and Image Representation*, 60, 2019, 346–59.

[19] Y. Li and S. Lyu, "Exposing DeepFake Videos By Detecting Face Warping Artifacts," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[20] Y. Li and S. Lyu, "Exposing Deepfake Videos by Detecting Face Warping Artifacts," *arXiv preprint arXiv:1811.00656*, 2018.

[21] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, “Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, 3207–16.

[22] X. Liu, F. Xing, C. Yang, C.-C. J. Kuo, S. Babu, G. E. Fakhri, T. Jenkins, and J. Woo, “VoxelHop: Successive Subspace Learning for ALS Disease Classification Using Structural MRI,” *arXiv preprint arXiv:2101.05131*, 2021.

[23] F. Matern, C. Riess, and M. Stamminger, “Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations,” in *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, IEEE, 2019, 83–92.

[24] M. Monajatipoor, M. Rouhsedaghat, L. H. Li, A. Chien, C.-C. J. Kuo, F. Scalzo, and K.-W. Chang, “BERTHop: An Effective Vision-and-Language Model for Chest X-ray Disease Diagnosis,” *arXiv preprint arXiv:2108.04938*, 2021.

[25] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, “Multi-task Learning for Detecting and Segmenting Manipulated Facial Images and Videos,” *arXiv preprint arXiv:1906.06876*, 2019.

[26] H. H. Nguyen, J. Yamagishi, and I. Echizen, “Use of a Capsule Network to Detect Fake Images and Videos,” *arXiv preprint arXiv:1910.12467*, 2019.

[27] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “Faceforensics++: Learning to Detect Manipulated Facial Images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, 1–11.

[28] M. Rouhsedaghat, Y. Wang, X. Ge, S. Hu, S. You, and C.-C. J. Kuo, “FaceHop: A Light-Weight Low-resolution Face Gender Classification Method,” in *International Conference on Pattern Recognition*, Springer, 2021, 169–83.

[29] M. Rouhsedaghat, Y. Wang, S. Hu, S. You, and C.-C. J. Kuo, “Low Resolution Face Recognition in Resource-constrained Environments,” *Pattern Recognition Letters*, 149, 2021, 193–9.

[30] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules,” *Advances in Neural Information Processing Systems*, 30, 2017.

[31] S. Seferbekov, “A Prize Winning Solution for DFDC Challenge,” https://github.com/selimsef/dfdc_deepfake_challenge, 2020.

[32] Z. Sun, Y. Han, Z. Hua, N. Ruan, and W. Jia, “Improving the Efficiency and Robustness of Deepfakes Detection through Precise Geometric Features,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 3609–18.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, 2818–26.

[34] M. Tan and Q. Le, “Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks,” in *International Conference on Machine Learning*, PMLR, 2019, 6105–14.

[35] V.-N. Tran, S.-H. Lee, H.-S. Le, and K.-R. Kwon, “High Performance Deepfake Video Detection on CNN-based with Attention Target-specific Regions and Manual Distillation Extraction,” *Applied Sciences*, 11(16), 2021, 7678.

[36] M. Turk and A. Pentland, “Eigenfaces for Recognition,” *Journal of Cognitive Neuroscience*, 3(1), 1991, 71–86.

[37] X. Yang, Y. Li, and S. Lyu, “Exposing Deep Fakes using Inconsistent Head Poses,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, 8261–5.

[38] Y. Yang, W. Wang, H. Fu, and C.-C. J. Kuo, “On Supervised Feature Selection from High Dimensional Feature Spaces,” *arXiv preprint arXiv:2203.11924*, 2022.

[39] K. Zhang, B. Wang, W. Wang, F. Sohrab, M. Gabbouj, and C.-C. J. Kuo, “AnomalyHop: An SSL-based Image Anomaly Localization Method,” *arXiv preprint arXiv:2105.03797*, 2021.

[40] M. Zhang, P. Kadam, S. Liu, and C.-C. J. Kuo, “Unsupervised Feedforward Feature (UFF) Learning for Point Cloud Classification and Segmentation,” in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, IEEE, 2020, 144–7.

[41] M. Zhang, Y. Wang, P. Kadam, S. Liu, and C.-C. J. Kuo, “PointHop++: A Lightweight Learning Model on Point Sets for 3D Classification,” in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, 3319–23.

[42] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, “PointHop: An Explainable Machine Learning Method for Point Cloud Classification,” *IEEE Transactions on Multimedia*, 22(7), 2020, 1744–55.

[43] H. Zhao, W. Zhou, D. Chen, T. Wei, W. Zhang, and N. Yu, “Multi-attentional Deepfake Detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, 2185–94.

[44] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, “Two-stream Neural Networks for Tampered Face Detection,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, 2017, 1831–9.