Patch Diffusion: A General Module for Face Manipulation Detection

Baogen Zhang\textsuperscript{1}, Sheng Li\textsuperscript{1*}, Guorui Feng\textsuperscript{2}, Zhenxing Qian\textsuperscript{1}, Xinpeng Zhang\textsuperscript{1}

\textsuperscript{1}Fudan University  
\textsuperscript{2}Shanghai University  
\{bgzhang19, lisheng, zxqian, zhangxinpeng\}@fudan.edu.cn, grfeng@shu.edu.cn

Abstract

Detection of manipulated face images has attracted a lot of interest recently. Various schemes have been proposed to tackle this challenging problem, where the patch-based approaches are shown to be promising. However, the existing patch-based approaches tend to treat different patches equally, which do not fully exploit the patch discrepancy for effective feature learning. In this paper, we propose a Patch Diffusion (PD) module which can be integrated into the existing face manipulation detection networks to boost the performance. The PD consists of Discrepancy Patch Feature Learning (DPFL) and Attention-Aware Message Passing (AMP). The DPFL effectively learns the patch features by a newly designed Pairwise Patch Loss (PPLoss), which takes both the patch importance and correlations into consideration. The AMP diffuses the patches through attention-aware message passing in a graph network, where the attentions are explicitly computed based on the patch features learnt in DPFL. We integrate our PD module into four recent face manipulation detection networks, and carry out the experiments on four popular datasets. The results demonstrate that our PD module is able to boost the performance of the existing networks for face manipulation detection.

Introduction

In recent years, with the tremendous progress made in computer vision and deep learning, it becomes easy to generate real-look alike fake face images. People can manipulate the face images by softwares such as ZAO, FaceApp, FakeApp, etc. Along with the fun offered by these softwares, malicious face manipulations would mislead the public, which may result in unforeseeable consequences, especially when relevant techniques are applied on celebrities. Therefore, we urgently need advanced and effective methods for manipulated face detection.

The schemes for face manipulation detection can be mainly divided into two categories, the video-based methods and the image-based methods. The video-based methods focus on the differences among different frames to find temporal manipulation traces for classification. Several interesting ideas such as eye blinking (Li, Chang, and Lyu 2018), physiological measurement (Fernandes et al. 2019; Ciftci, Demir, and Yin 2020) and emotions (Mittal et al. 2020) are proposed to achieve satisfactory performance. However, when the input is only a single image, the video-based methods may fail due to the loss of sequential information. This can be addressed by using the image-based method, whose main idea is to detect the manipulated traces inside a single image. Researchers have devoted efforts in developing effective image-based face manipulation techniques. Various manipulation traces have been investigated to facilitate the detection, including the anomaly in color (Li et al. 2020a), the blending boundary (Li et al. 2020b; Tarasiou and Zafeiriou 2020; Wang et al. 2020b), the anomaly in frequency (Qian et al. 2020), and etc. These schemes explore the inconsistency of the traces from the whole image for face manipulation detection, which may not work well when the manipulation traces are subtle.

Some studies (Chai et al. 2020; Mayer and Stamm 2019) have shown that, the subtle traces located inside small local patches such as manipulation contours or anomaly in texture (Zhang, Karaman, and Chang 2019), would be helpful for differentiating the genuine and fake area. Most recently, Chen et al. (Chen et al. 2021) and Zhao et al. (Zhao et al. 2021b) consider the similarity among different patches to boost the performance. Despite the advantage, these approaches treat the patches equally, which neglect the diverse impact of different patches for the task of face manipulation detection. More efforts are required to fully exploit the discrepancy among different patches in face manipulation detection.

In this paper, we propose a module named Patch Diffu-
sion (PD) for face manipulation detection. Unlike the existing patch-based schemes which either process the patches independently or simply compute the similarity among different patches, we propose to diffuse the patches using graph networks such that the manipulated region can be gradually merged to prompt the detection, as shown in Fig. 1. In particular, our PD module is composed of two parts: Discrepancy Patch Feature Learning (DPFL) and Attention-Aware Message Passing (AMP). The DPFL focuses on learning effective patch feature representations by taking both the patch importance and correlations into consideration, where a Pairwise Patch Loss (PPLoss) is newly designed for patch feature learning. The AMP constructs a graph from the patches and gradually diffuses the patches through message passing. We propose an attention-aware adjacent matrix for message passing in AMP, which is explicitly computed from the patch features learnt in DPFL.

To demonstrate the generalization ability of our PD module, we integrate it into four recent face manipulation detection networks including CNNDetect (Wang et al. 2020a), Xception (Rossler et al. 2019a), DSP-FWA (Li and Lyu 2019), and WM-Wsdan-Effb (Zhao, Cui, and Zhou 2020) to see how the performance could be boosted. We conduct experiments on four popular datasets. The results demonstrate that, by integrating our PD module into the existing networks, the performance of face manipulation detection can be further boosted in both the intra and inter-database scenarios. The main contributions of this paper are summarized as follows.

- We propose a patch diffusion module for the task of face manipulation detection, which can be integrated into the existing networks for performance boosting.
- We propose a novel patch feature learning scheme by incorporating the patch importance and correlation, which is achieved by a PPLoss newly designed.
- We propose an attention-aware adjacent matrix for patch diffusion, which is computed explicitly from the patch features.

Related Works

Face Manipulation Detection. Existing face manipulation detection schemes can be mainly categorized into the video-based methods and the image-based methods.

The video-based methods perform face manipulation detection based on the inconsistencies between frames. Works on heart rate estimation (Fernandes et al. 2019; Ciftci, Demir, and Yin 2020; Hernandez-Ortega et al. 2020; Qi et al. 2020) detect the manipulation through biological signals collected from videos. Li et al. (Li, Chang, and Lyu 2018) and Mittal et al. (Mittal et al. 2020) determine whether the frame has been altered based on the facial action and expression. Li et al. (Li et al. 2020c) introduces partial face attack in deepfake videos and propose a multiple instance learning framework for face manipulation detection. Several other works (Sabir et al. 2019; Masi et al. 2020; Chen et al. 2020) create new model structures based on the recurrent network.

The image-based methods detect inconsistencies between fake and genuine regions from a single image. Liu et al. (Li-
The Proposed Method

The architecture of our proposed patch diffusion (PD) module is given in Fig. 2. The input of the PD module is the feature map of a convolutional layer of an existing network. Its output is a feature map with the same dimension as the input. Each spatial location in the feature map represents a local patch. Our PD model processes the input feature map by, 1) discrepancy patch feature learning (DPFL) and 2) attention-aware message passing (AMP). We propose a Pairwise Patch Loss (PPLoss) to learn effective patch feature representations in DPFL. These are used to compute an attention-aware adjacent matrix which explicitly serves as an attention mechanism for patch diffusion in AMP. In what follows, we explain the DPFL and AMP in detail, as well as the module integration and training.

Discrepancy Patch Feature Learning (DPFL)

Let’s denote input feature map as \( F \in \mathbb{R}^{H \times W \times C} \), where \( H, W, C \) are the height, width and number of channels, respectively. Each spatial location in \( F \) corresponds to a local patch represented as a \( C \) dimensional vector, with a total of \( N_p = H \times W \) patches.

Given a feature map \( F \), we first transform it into \( X = \psi(F) \in \mathbb{R}^{H \times W \times C'} \), where \( \psi(\cdot) \) is a 1 \( \times \) 1 convolution layer with \( C' \) kernels. This is to make the feature dimension of each local patch invariant among different networks. For the \( i \)th patch, let’s denote \( x_i \) as the corresponding feature vector in \( X \). Therefore, we have \( X = \{x_i\}_{i=1}^{N_p} \) in terms of the \( N_p \) patches in the image.

We annotate the face image into three regions including the fake region, real region and the background region, as shown in Fig. 3. For the \( i \)th patch \( x_i \), let’s denote \( \mathcal{T}^f_i \) and \( \mathcal{T}^r_i \) as the ratios of the fake and real region that occupy the patch, respectively. The ground truth of the \( i \)th patch is computed as

\[
y_i = \begin{cases} 0, & \mathcal{T}^f_i > \delta, \\ 1, & \mathcal{T}^f_i > \delta \text{ and } \mathcal{T}^r_i < \delta, \\ 2, & \text{otherwise} \end{cases}
\] (1)

where \( \delta \) is a threshold for the tolerance of annotation errors. \( y_i = 0, 1 \) or 2 refer to the case that the patch is annotated as fake, real or background, respectively. We set a relatively small \( \delta \) with \( \delta = 0.1 \) for robust patch annotation.

As indicated in (Li et al. 2020b), the traces around the edges of the fake area are more distinct than that in the other areas, which plays an important role for manipulated face detection. This is to say, the importance of each patch should be paid on the patches which contain edges of different regions. We evaluate the importance of the \( i \)th patch by a score \( s_i \) which is formulated below.

\[
s_i = \begin{cases} 2 - \mathcal{T}^f_i, & \text{if } y_i = 0 \\ 2 - \mathcal{T}^r_i, & \text{if } y_i = 1 \\ 2 - \mathcal{T}^b, & \text{if } y_i = 2 \end{cases}
\] (2)

where \( \mathcal{T}^b \) is the ratio of the background region in the patch and \( s_i \in [1, 2] \). The larger the \( s_i \), the more important the patch will be.

With \( y_i \) and \( s_i \) available, we propose here a Pairwise Patch Loss (PPLoss) by considering the discrepancy between a pair of two patches (say \( x_i \) and \( x_j \)). The discrepancy lies in the following two aspects, 1) the distance between \( x_i \) and \( x_j \), which is computed by the L2-norm denoted as \( d_{ij} = ||x_i - x_j|| \), and 2) the difference between the patch ground truth (i.e., \( y_i \) and \( y_j \)), which is indicated by a hyper-parameter \( \epsilon_{ij} \) given below.

\[
\epsilon_{ij} = \begin{cases} 0, & \text{if } y_i = y_j \\ \alpha, & \text{if } y_i \neq y_j \end{cases}
\] (3)

where \( \alpha \) is a hyperparameter depending on the ground truth of the two patches. The PPLoss is formulated by

\[
\mathcal{L}_p = \frac{1}{N_c} \sum_{i=1}^{N_p} \sum_{j \in N_i} \{f(s_i, s_j)|d_{i,j} - \epsilon_{ij}|\}
\] (4)

Figure 2: Patch diffusion for face manipulation detection.
Figure 3: Illustrations of different regions in a face image, (a) a fake face image with a fake region and a background region, (b) a fake face image with a fake region, a real region and a background region, (c) a genuine face image with a real face region and a background region. The fake region and real region are shown in red masks and white masks, with the rest being the background region. Better viewed in color.

where

\[ f(s_i, s_j) = \begin{cases} 1, & s_i = 1 \text{ and } s_j = 1 \\ s_i s_j \omega, & \text{otherwise} \end{cases} \]  

(5)

is a function evaluating the importance of a pair of two patches with \( \omega > 1 \) as a hyperparameter, \( |z| \) computes the absolute value of \( z \), and \( N_i \) denotes the indexes of a local neighborhood (within two hops) of the \( i \)th patch, and \( N_e \) is the total patch pairs that are used to compute the loss.

Our PPLoss contains two parts: \( |d_{ij} - \epsilon_{ij}| \) and \( f(s_i, s_j) \). Here, we refer them as \( D_{ij} \) and \( W_{ij} \) for short. The \( D_{ij} \) tries to maximize the inter-class distance and minimize the intra-class distance (\( \epsilon_{ij} = 0 \) when \( i = j \)). Since we have three classes of patches, it would be difficult to learn patch features that can maximize the inter-class distance for different pairs of classes. Instead, we regard such distance as hyperparameters (i.e., \( \alpha \)) whose optimal values can be determined in validation. Intuitively, \( \alpha \) should be large enough for classes that are difficult to be differentiated (e.g., real and fake). This is in accordance with the optimal values of \( \alpha \) found in validation (see Section 4 for details). The \( W_{ij} \) additionally weights \( D_{ij} \) by taking the patch importance into consideration. It makes the patches with edges contribute more during the training, which effectively exploits the distinct traces left on the edges for learning representative patch features.

Next, we combine our PPLoss with the classification loss of the network to be integrated (say \( \mathcal{L}_{cls} \)) for feature learning, which is given as

\[ \mathcal{L} = \lambda \mathcal{L}_p + \mathcal{L}_{cls}, \]

(6)

where \( \lambda \) is the weight balancing \( \mathcal{L}_p \) and \( \mathcal{L}_{cls} \). The PPLoss helps us to learn representative patch features, where patches belong to the same region (fake, real or background) tend to be similar. For the patches belong to different regions, their difference will be close to \( \alpha \). The PPLoss also makes sure that the patches containing edges of different regions contribute more during the learning, which effectively exploits the distinct traces left on the edges of the manipulation area.

Attention-aware Message Passing (AMP)

Initialization: First of all, we transform the feature map \( \mathbf{F} \) by \( \mathbf{V} = \rho(\mathbf{F}) \in \mathbb{R}^{H \times W \times C} \), where \( \rho(\cdot) \) is a 1 × 1 convolution layer with \( C \) kernels. We denote each patch in \( \mathbf{V} \) as a node. Thus, the feature map \( \mathbf{V} \) contains \( N_p = H \times W \) nodes with \( \mathbf{V} = \{v_i\}_{i=1}^{N_p} \). Then, we construct an undirected graph \( \mathbf{G} = (\mathbf{V}, \mathbf{E}) \) with \( \mathbf{V} \) as the nodes and \( \mathbf{E} \) as the edges. Initially, each node is fully connected to all the other nodes in the graph to form the edges.

Message Passing: In order to perform the patch diffusion, we gradually updates \( \mathbf{V} \) through message passing using a graph convolutional network (GCN) (Kipf and Welling 2017), which is formulated as

\[ \mathbf{V}^{(l+1)} = \mathbf{A} \mathbf{V}^{(l)} \mathbf{W}^{(l)} \]

(7)

where \( \mathbf{V}^{(l)} \) and \( \mathbf{V}^{(l+1)} \) refer to the nodes at the \( l \)th and \( (l + 1) \)th iteration with \( \mathbf{V}^{(0)} = \mathbf{V} \). \( \mathbf{W}^{(l)} \) is a layer-specific trainable weight matrix for intra-node feature fusion and \( \mathbf{A} \) is the adjacent matrix for message passing with a dimension of \( N_p \times N_p \).

The adjacent matrix defines how two nodes exchange information, which is important for message passing. Most of the existing works incorporate a binary adjacent matrix (Kipf and Welling 2017). In such a case, \( \mathbf{A}(i, j) = 1 \) means there is an edge between \( v_i \) and \( v_j \) (i.e., \( v_i \) and \( v_j \) belong to the same region), \( \mathbf{A}(i, j) = 0 \) means \( v_i \) and \( v_j \) are not connected (i.e., \( v_i \) and \( v_j \) belong to different regions). Some works (Chen et al. 2019; Te et al. 2020) implicitly define the adjacent matrix by a learnable 1 × 1 convolutional layer, which serves as an attention mechanism for message passing.

Instead of using a convolutional layer to learn the adjacent matrix, we propose here an explicit way to compute an attention-aware adjacent matrix. To be more specific, we explicitly compute a weight \( \mathbf{A}(i, j) \) for the message passing between \( v_i \) and \( v_j \), where

\[ \mathbf{A}(i, j) = \frac{\mathbf{x}_i \mathbf{x}_j^T}{||\mathbf{x}_i|| \cdot ||\mathbf{x}_j|| + \xi}, \]

(8)

where \( \xi \) is a scalar to prevent division by 0. As such, the correlations between any two nodes (i.e., patches) are considered for the message passing, which would be helpful to lead a smooth patch diffusion.

After \( l \) iterations, we can get a more representative feature map \( \mathbf{V}^{(l)} \). The output feature map is computed as

\[ \mathbf{F}' = \text{BN}(\mathbf{V}^{(l)}) + \mathbf{F}, \]

(9)

where BN is the batch normalization.

Module Integration and Training

As a module, our PD can be integrated into the existing convolutional neural networks. In order to achieve satisfactory performance, the dimension of the input feature map for our PD should not be too small or too large. Because each spatial element in the feature map corresponds to a local image patch. Feature maps with large dimensions refer to small image patches which do not contain sufficient information for
patch diffusion. On the contrary, feature maps with small dimensions indicate that the image patches are large, which are not good for the detection of the subtle manipulation traces.

In the training phase, both the images and the corresponding patch ground truth will be fed into the integrated network, which will then be trained according to the loss given in Eq. (6). After the training, the integrated network can be used to predict whether an input face image is genuine or fake according to the output of the classification layer.

Experiments

Setup

Training dataset. We select FaceForensics++ (Rossler et al. 2019b) as the training dataset. It is a large-scale face manipulation dataset containing 1000 real videos, in which 720 videos are used for training, 140 videos are reserved for validation and 140 videos are for testing. Each real video has four manipulated versions which are generated by DeepFakes (DF) (Tora 2018), Face2Face (F2F) (Thies et al. 2016), FaceSwap (FS) (Kowalski 2018) and NeuralTextures (NT) (Thies, Zollhöfer, and Nießner 2019). In addition, each video has three compression levels: RAW, High Quality (HQ) and Low Quality (LQ), respectively. We evenly select 16 frames from each video. For each frame, we extract 81 facial landmarks \(^1\) to crop the facial area.

For the fake face images generated by DF and F2F, we obtain the fake and background regions according to the facial mask given in the database, while the real regions are automatically annotated according to the facial landmarks. For those generated by NT and FS, we also annotate different regions according to the facial landmarks. With the ground truth, we calculate the performance based on Eq. (1). Therefore, there is no human labor involved for the annotations in our experiment.

Test datasets. To have a thorough demonstration for the generality of our propose scheme, we conduct the testing on the following databases: (1) FaceForensics++ (Rossler et al. 2019b): all the videos in the test set are used for testing. (2) DeepfakeDetection \(^2\) (DFD): DFD is a large dataset for synthetic video detection, we use all the RAW videos for testing. (3) Celeb-DF (Li et al. 2020d): Celeb-DF is a large-scale challenging DeepFake video dataset with two versions: Celeb-DFv1 (CD1) and Celeb-DFv2 (CD2), and we test on both of them. (4) DeeperForensics (DFR) (Jiang et al. 2020): DFR is a recently released face swap video dataset with 60,000 videos, it is a representative dataset for face manipulation detection, and we test on all the videos.

Implementation details. We choose four recent face manipulation detection networks, including CNNDetect (Wang et al. 2020a), Xception (Rossler et al. 2019b), DSP-FWA (Li and Lyu 2019) and WM-Wsdan-Effb (Zhao, Cui, and Zhou 2020) for the integration of our PD module. The CNNDetect (Wang et al. 2020a) is shown to have good performance for deepfake detection. The Xception (Rossler et al. 2019b) is a baseline model in FaceForensics++, which is a popular method in the area of face manipulation detection. The DSP-FWA (Li and Lyu 2019) achieves satisfactory performance on Celeb-DF. The WM-Wsdan-Effb (Zhao, Cui, and Zhou 2020) is the runner up in the facebook Deepfake Detection Challenge (Dolhansky et al. 2019). Note that the winner of this challenge \(^3\) drops out part of the image for data augmentation, which is not appropriate for our module.

For each image, we adopt the method proposed in (Rossler et al. 2019b) to automatically and conservatively crop the facial area into a square, which is then resized to 224 \(\times\) 224 for training and testing. Our module is integrated after a convolutional block with a feature map of \(14 \times 14 \times C\) as the output. In other words, we set \(H = W = 14\) and \(N_p = 196\) fixed during the experiment, while \(C\) varies among different models and we set \(C' = C/2\). We set \(\alpha = 2, 15, 17\) for a pair of patches whose ground truth are real and background, fake and background, real and fake, respectively. Other hyperparameters are set by \(\lambda = 1, \omega = 3, l = 1, \xi = 10^{-6}\). The code is available at https://github.com/starxchina/Patch-Diffusion.

Intra-database Evaluation for the PD Module

In this section, we evaluate the performance of the four existing networks on FaceForensics++ before and after the integration of our PD module. Three indicators are used to evaluate the network performance including: ACC (correctly classified images/total images), AUC (area under the Receiver Operating Characteristic curve) and EER (Equal Error Rate). In the following discussions, all the three indicators are reported in terms of percentages.

We use all the subsets of the FaceForensics++ (DF, F2F, FS, NT, Real) to train the models. Each training image has three different compression levels, and we conduct the training and testing on the same compression level. Table 1 gives the performance on FaceForensics++ for different compression levels, where RAW means no compression, HQ and LQ indicate high and low compression quality, respectively. It can be seen that our PD module is able to boost the performance for majority of the cases regardless the networks. The only two cases that we are not able to achieve better results (by using the PD module) is the ACC and EER on RAW for WM-Wsdan-Effb. This is mainly because the performance of WM-Wsdan-Effb is already very high on RAW. On the other two compression levels (i.e., HQ and LQ), our PD module is able to improve the performance consistently on all the four existing networks. For most of the existing networks, more performance gain can be achieved using our PD module on LQ, where the performance of the four existing networks are not satisfactory. After integrating the PD module for the detection of LQ, the improvement of the ACC is 4.73%, 1.96%, 0.46% and 0.88% for the CNNDetect, Xception, DSP-FWA, and WM-Wsdan-Effb, respectively. Such improvements are reduced to 2.6%, 1.1% and 0.54% for the CNNDetect, Xception and WM-Wsdan-Effb on HQ. For the DSP-FWA, however, we are able to gain more improvement of the ACC (i.e., 5%) on the HQ than on the LQ.

---

\(^1\)https://github.com/codeniko/shape-predictor_81_face_landmarks
\(^3\)https://github.com/selimsef/dfdc_deepfake_challenge
Cross-database Evaluation for the PD Module

On different manipulation types. By following the same protocol as the work in (Chai et al. 2020; Dang et al. 2020), we conduct cross-database evaluation between different subsets in the FaceForensics++, where the AUC is served as the indicator. In particular, we select one subset for training, and then test the performance on the other three subsets. Both the training and testing are carried out on the RAW level, and we choose Xception as a representative network. The results are given in Table 2. It can be seen that integrating our PD module significantly improves the performance. For example, when trained on FS training and test on the rest three subsets, the improvements of the AUC are 15.64%, 17.46% and 22.66% for DF, F2F and NT, respectively. This indicates that our PD works well for the detection of unseen manipulation types.

On different databases. Next, we conduct the cross-database evaluation on different databases. In particular, we carry out the training on the FaceForensic++ and the testing on DFD, Celeb-df (CD1), Celeb-df (CD2) and DFR, the results of which are shown in Table 3. We find that our PD module can consistently improve the AUC for different networks. But the improvement here is less when compared with the cross-database evaluation for different manipulation types.

Ablation Studies

In this section, we conduct ablation studies on different components of our PD module. All the ablation studies are carried out on the LQ of FaceForensics++ with Xception as the basic network.

DPFL and AMP. We gradually switch off the DPFL and AMP in our PD module to see how the performance changes. As shown in Table 4, we can see that using both of them achieves the best performance, with 1.96% of higher ACC than the original Xception, 1.27% higher ACC than using DPFL only, and 2.95% higher ACC than using the AMP only. We observe that using the AMP along will degrade the performance of the Xception, which indicates the effective-
Table 6: Ablation studies on patch importance.

<table>
<thead>
<tr>
<th>Method</th>
<th>ACC (HQ)</th>
<th>AUC (HQ)</th>
<th>EER (HQ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face X-ray</td>
<td>99.87</td>
<td>97.59</td>
<td>99.46</td>
</tr>
<tr>
<td>LRL</td>
<td>-</td>
<td>98.80</td>
<td>-</td>
</tr>
<tr>
<td>PCL</td>
<td>-</td>
<td>99.92</td>
<td>-</td>
</tr>
<tr>
<td>WM-Wsdan-Effb</td>
<td>99.62</td>
<td>96.39</td>
<td>99.49</td>
</tr>
<tr>
<td>WM-Wsdan-Effb+PD</td>
<td>99.62</td>
<td>98.48</td>
<td>99.92</td>
</tr>
</tbody>
</table>

Table 7: Comparisons of intra-database evaluation with the state-of-the-art.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Set</th>
<th>Test Set (AUC (%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face X-ray</td>
<td>DF</td>
<td>99.41 74.99 71.74 58.74</td>
</tr>
<tr>
<td>LRL</td>
<td>F2F</td>
<td>84.39 80.88 97.66 63.21</td>
</tr>
<tr>
<td>PCL</td>
<td>FS</td>
<td>61.77 53.44 62.00 97.13</td>
</tr>
<tr>
<td>WM-Wsdan-Effb</td>
<td>NT</td>
<td>70.32 92.23 65.04 52.79</td>
</tr>
<tr>
<td>Xception+PD</td>
<td>F2F</td>
<td>85.09 82.65 94.48 100.00</td>
</tr>
<tr>
<td></td>
<td>FS</td>
<td>99.87 99.14 92.79 61.60</td>
</tr>
</tbody>
</table>

Table 9: Comparisons of cross-database evaluation on different manipulation types in FaceForensics++.

In this section, we compare the networks integrated with the PD module against the state-of-the-art work including the Face X-ray (Li et al. 2020b), LRL (Chen et al. 2021), PCL (Zhao et al. 2021b) and Patch Forensics (Chai et al. 2020). All these works as well as our PD module require to annotate the patches for training, and the work in (Li et al. 2020b) and (Zhao et al. 2021b) generate additional data from the training set for augmentation. For fair comparison, we follow the same protocols as suggested in these works to evaluate our method, where the results of the existing works are duplicated from the literature.

First of all, we conduct the intra-database comparison on FaceForensics++ by following the protocol given in (Li et al. 2020b; Chen et al. 2021; Zhao et al. 2021b). In particular, we use the training sets of RAW and HQ in FaceForensics++ for training, and the corresponding test set for testing. Table 7 gives the comparisons among different schemes. It can be seen that, by integrating PD with WM-Wsdan-Effb, we can achieve better performance than the existing scheme in most of the cases.

We then conduct the cross-database comparison on DFD and CD2 as suggested in (Li et al. 2020b; Chen et al. 2021; Zhao et al. 2021b), where all the models are trained in FaceForensics++. As shown in Table 8, our WM-Wsdan-Effb+PD performs consistently better than the existing schemes. Our scheme performs significantly better than the LRL here, with over 8% higher AUC on DFD and around 4% higher AUC on CD2. When compared with (Li et al. 2020b) and (Zhao et al. 2021b), the improvement is not significant because both of these schemes use additional data for data augmentation. It might be a good choice to combine such strategy with our proposed PD module.

Next, we compare our method with the Patch Forensics (Chai et al. 2020) in Table 9, where the average precision (AP) of Patch-Forensics are the best results on different Xception blocks reported in (Chai et al. 2020). It can be seen that our method performs better than the Patch Forensics in majority of the cases with over 10% higher AP.

Conclusion

In this paper, we propose a face manipulation detection module by taking advantage of the graph network. This module is termed as the Patch Diffusion (PD) module, which consists of Discrepancy Patch Feature Learning (DPFL) and Attention-Aware Message Passing (AMP). The DPFL learns the patch features according to a newly designed Pairwise Patch Loss. While the AMP performs the message passing to diffuse the patches according to an adjacent matrix explicitly computed from the patch features. Various experiments demonstrate the effectiveness of our PD module in the existing networks for face manipulation detection.
Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62072114, U20A20178, 62072295, U20B2051 and U1936214, in part by Alibaba Group through Alibaba Innovative Research (AIR) Program, in part by the Shanghai Science and Technology Committee under Grant 21010500200.

References


