DDA4210 Project Report by Group 34

Cheng, Chaofan; Wen, Keen; Zhao, Xueling; Zheng, Keru
CUHK(SZ)
Group 34

Abstract

The rapid development of deep learning has promoted highly realistic fake face videos and images, posing significant social risks. This study addresses the limitations of current face forgery detection methods and proposes a modified detection model that integrates a Gram Block for global texture information and a channel-wise attention module, enhancing cross-method detection capabilities. Experiments conducted using the FaceForensics++ database show that our model outperforms traditional methods in terms of convergence and cross-dataset generalizability, though it is sometimes at the expense of within-method accuracy. Despite some limitations, such as insufficient training data and inaccuracies in data labeling, our model highlights the potential for broader application on various AI face forgery techniques.

1 Introduction

1.1 Significance

The advancement of Deep Learning enables the creation of highly realistic fake face videos and images, which can easily mislead people. This prevalence of synthetic faces poses significant social risks, such as the spread of fake news and unreliable evidence.

Current face forgery methods could be roughly divided into 3 categories: face synthesis, deepfake and face swap. Face synthesis methods involve using deep learning models, particularly VAEs and GANs, to generate entirely new facial images from scratch. Deepfake utilizes GANs to generate or alter faces in videos. This technology can create highly realistic videos. Face Swap involves digitally replacing the face in one image with the face from another image using facial detection, feature matching, and image blending techniques.

1.2 Novelty

Currently, most of the detection methods use CNN to extract features for detection. Among them, textures is an important cue for fake face detection. Although texture-based CNN detection methods worked well on GAN-generated fake face image, techniques like Deepfake and Face Swap, which involve swapping faces on real images, might not be as easily detected by solely texture-based approaches. It is observed that models trained on specific datasets might struggle with generalization when exposed to different fake image generation technologies.

Therefore, in this paper, we aim to produce a type of face forgery detection model that can enhance the cross-database accuracy.

Preprint. Under review.

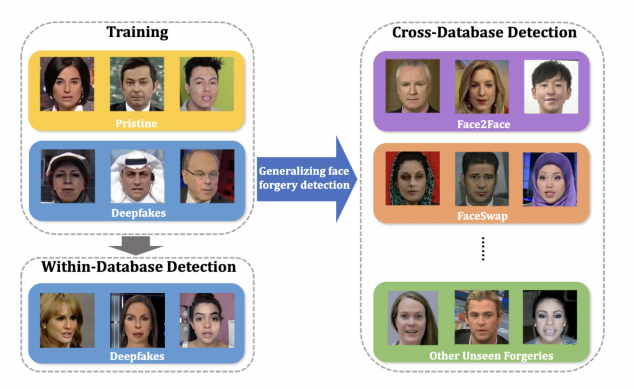


Figure 1: Enhancing cross-database detection capabilities

2 Related work

Texture feature Liu *et al.*(2020) showed CNNs take textures as an important cue for fake face detection[2].

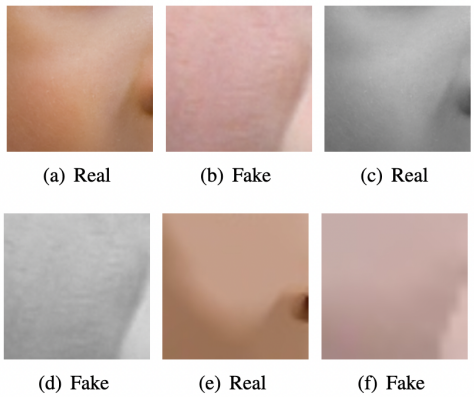


Figure 2: Comparison of textures in real and GAN-generated face images

Deepfake and Face Swap Methods Although texture-based CNN detection methods worked well on GAN-generated fake face image, techniques like Deepfake and Face Swap, which involve swapping faces on real images, might not be as easily detected by solely texture-based approaches.

Biased Detection It is observed that models trained on specific datasets might struggle with generalization when exposed to different fake image generation technologies.

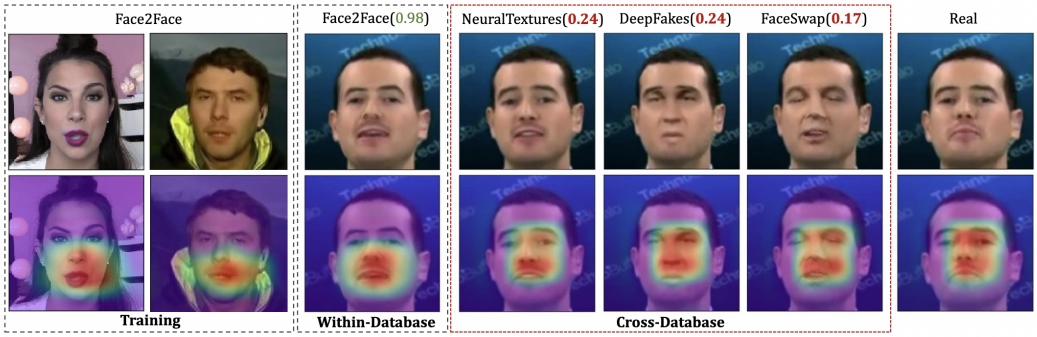


Figure 3: Overfitting FTF(Face2Face) unique fake textures by Luo *et al.* (2021)

Two-stream Architecture Zhou *et al.* (2018) proposed a two stream-architecture that extract high frequency noise using SRM filter (Selective Retinex Modulation Filter).

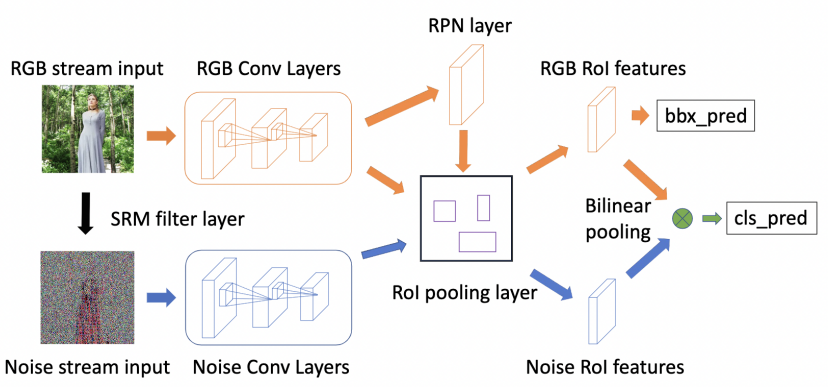


Figure 4: Two-stream Architecture proposed by Zhou *et al.* (2018)

3 Methodology

Our Modification

- Adding Gram Block to calculate global texture information
- Adding Channel-wise attention module in Fusion part

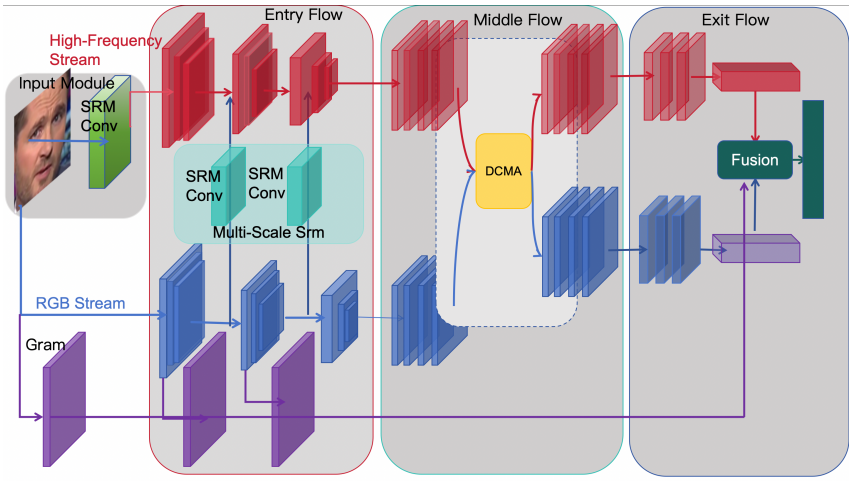


Figure 5: Flow Chart of Our Model

3.1 Gram Block

Gram Matrix The Gram matrix is calculated as follows.

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l} \tag{1}$$

where F^l represents the l-th feature map whose spatial dimension is vectorized, and F_{ik}^{l} represents the kth element in the ith feature map of layer l.

Liu *et al.* (2020) showed Gram matrix is a good descriptor for global or long-range texture as follows.

Objective of Gram Matrix

- **Aggregation of Global Information**

  Each element of the Gram matrix is a result of an inner product between feature map vectors. This operation aggregates information across the entire spatial extent of the feature maps, which implies that it considers the global arrangement of features, not just local or nearby pixel values.

- **Independence from Spatial Layout**

  Since the Gram matrix is essentially a set of dot products, it is invariant to the spatial arrangement of features; it focuses on the presence of features rather than their position. This quality allows it to capture the overall textural information without being limited by the specific layout that the receptive fields of the CNN would impose.

In summary, G^l serves as a descriptor that captures long-range interactions across the whole feature map.

3.2 Channel-wise attention module in Fusion part

Pseudocode for Fusion Model Each CNN filter performs as a pattern detector, and each channel of a feature map is a response activation of the corresponding convolutional filter. Applying an attention mechanism in channel wise manner can be viewed as a process of selecting semantic attributes. At the end of the flow, we apply the fusion model to fuse the high-level features of the two modalities.

Algorithm Feature fusion Module and Prediction (channel-wise attention)

Input: Feature maps from two separate streams
Output: Fused feature map with channel-wise attention applied

```
Class FeatureFusionModule inherits nn.Module
    self. convblk ← Sequential container with:
        Conv2d()
        BatchNorm2d(out_chan)
        ReLU()
    self.ca ← ChannelAttention(out_chan, ratio=16)
    self.init_weight()
function INIT_WEIGHT(self)
    for each layer in self do
        if layer is instance of Conv2d then
            Initialize weights with kaiming normal
            Initialize biases to zero (if any)
        end if
    end for
end function
```

Channel-wise Attention Module

The channel-wise attention is applied to the concatenated features with a new architectural unit, which we term the "Squeeze-and Excitation" (SE) block. The SE block proposes a mechanism that allows the network to perform feature re-calibration, through which it can learn to use global information to selectively emphasise informative features and suppress less useful ones. (Hu, J. , 2018).

The Squeeze-and-Excitation block is a computational unit which can be constructed for any given transformation (function 1). The basic structure of the SE building block is illustrated in the figure which consists of transform, squeeze, excitation, and re-scaling. The features U are first passed through a squeeze operation, which aggregates the feature maps across spatial dimensions H × W to produce a channel descriptor. This descriptor embeds the global distribution of channel-wise feature responses, enabling information from the global receptive field of the network to be leveraged by its lower layers. This is followed by an excitation operation, in which sample-specific activation, learned for each channel by a self-gating mechanism based on channel dependence, governs the excitation of each channel. The feature maps U are then reweighed to generate the output of the SE block which can then be fed directly into subsequent layers (Hu, J., 2018).

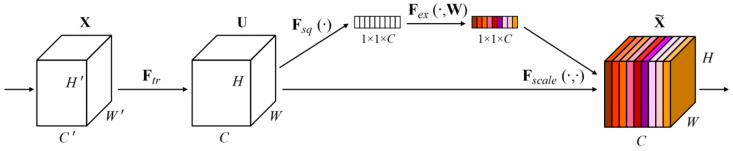
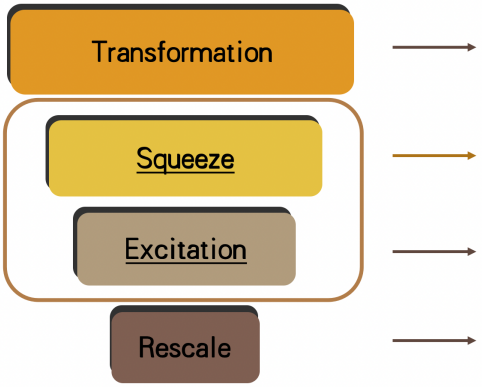


Figure 1: A Squeeze-and-Excitation block.



$F_{tr}: \mathbf{X} \to \mathbf{U}, \mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}, \mathbf{U} \in \mathbb{R}^{H \times W \times C}$
$V = [V_1, V_2, \ldots, V_C]$ denote the learned set of filter kernels
Take F_{tr} to be a convolutional operator. $*$ denotes convolution

$$u_c = V_c * \mathbf{X} = \sum_{s=1}^{C'} V_c^s * X^s. \tag{1}$$

A statistic $z \in \mathbb{R}^C$ is generated by shrinking U through spatial dimensions $H \times W$, where the c-th element of z is calculated by:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j). \tag{2}$$

Employ a simple gating mechanism with a sigmoid activation, where δ refers to the ReLU function, $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$. :

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z)), \tag{3}$$

Final output of the block is obtained by rescaling the transformation output U with the activations where $\tilde{\mathbf{X}} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_C]$ and $F_{scale}(u_c, s_c)$ refers to channel-wise multiplication between the feature map $u_c \in \mathbb{R}^{H \times W}$ and the scalar s_c :

$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c, \tag{4}$$

In conclusion, the SE-block helps improves the representational capacity of a network by enabling it to perform dynamic channel-wise feature recalibration. It also improve the representational power of a network by explicitly modeling the interdependencies between the channels of its convolutional features.

3.3 Loss Function

Original Loss Function

- **The original softmax loss is given by**

$$L_S = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{e^{\mathbf{W}_{y_i}^T \mathbf{f}_i}}{\sum_{j=1}^{c} e^{\mathbf{W}_j^T \mathbf{f}_i}} = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{e^{\|\mathbf{W}_{y_i}\| \|\mathbf{f}_i\| \cos(\theta_{y_i})}}{\sum_{j=1}^{c} e^{\|\mathbf{W}_j\| \|\mathbf{f}_i\| \cos(\theta_j)}}, \tag{2}$$

- **The A-softmax loss is given by**

$$\mathcal{L}_{AS} = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{e^{\|\mathbf{f}_i\| \psi(\theta_{y_i})}}{e^{\|\mathbf{f}_i\| \psi(\theta_{y_i})} + \sum_{j=1, j \neq y_i}^{c} e^{\|\mathbf{f}_i\| \cos(\theta_j)}}, \tag{3}$$

Normalize the weight vectors (making $\|\mathbf{W}_i\|$ to be 1) and generalize the target logit from $\|\mathbf{f}_i\| \cos(\theta_{y_i})$ to $\|\mathbf{f}_i\| \psi(\theta_{y_i})$, where the $\psi(\theta)$ is usually a piece-wise function defined as

$$\psi(\theta) = \begin{cases} (-1)^k \cos(m\theta) - 2k, & \text{if } \theta \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right) \\ 1 + \lambda, & \text{otherwise} \end{cases} \tag{4}$$

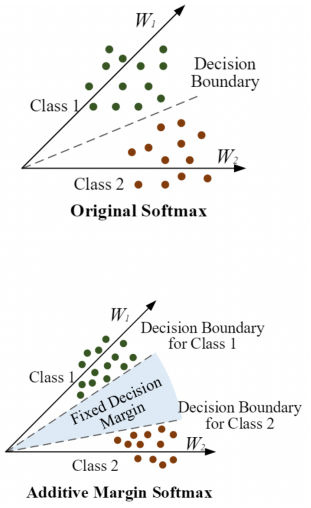
The Additive Margin Softmax loss

The A-softmax loss defines a general function $\psi(\theta)$ to introduce the large margin property. Motivated by that, we further propose a specific $\psi(\theta)$ that introduces an additive margin to the softmax loss function given by:

$$\psi(\theta) = \cos\theta - m. \tag{5}$$

Use cosine as the similarity to compare two face features, apply both feature normalization and weight normalization to the inner product layer in order to build a cosine layer. Then we scale the cosine values using a hyper-parameter s. Finally, the loss function becomes

$$\begin{aligned}\mathcal{L}_{AMS} &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot(\cos\theta_{y_i}-m)}}{e^{s\cdot(\cos\theta_{y_i}-m)}+\sum_{j=1,j\neq y_i}^{c}e^{s\cdot\cos\theta_j}} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot(\mathbf{W}_{y_i}^{T}\mathbf{f}_i-m)}}{e^{s\cdot(\mathbf{W}_{y_i}^{T}\mathbf{f}_i-m)}+\sum_{j=1,j\neq y_i}^{c}e^{s\mathbf{W}_{j}^{T}\mathbf{f}_i}}.\end{aligned} \tag{6}$$



The AM softmax is more simple and intuitive compare to the $\psi(\theta)$ in angular softmax. The angular softmax can only impose an unfixed angular margin, while the additive margin softmax incorporates the fixed hard angular margin. Adopting the AM-Softmax Loss as the objective function leads to smaller intra-class variations and larger inter-class differences than the regular cross-entropy loss.

4 Experimental Results

4.1 Data Preparation

To evaluate the performance of the our model, we utilized four datasets from the FaceForensics++ (FF++)[4] database, which comprises real videos sourced from YouTube as well as videos manipulated using various face-forging techniques. Specifically, the Deepfakes (DF) and NeuralTextures (NT) datasets employ deep learning-based methods, while the Face2Face (F2F) and FaceSwap (FS) datasets utilize face-swap techniques.

First, we download 510 videos from the authentic dataset and the corresponding 4*510 AI-manipulated videos. Then, we utilize face_recognition library in Python to extract faces from 30 frames of each video, obtaining 15,300 face images from the original videos. We also obtain 15,300 AI-manipulated face images for each type (DFD, F2F, FS, DF). Through the processes of extracting frames and faces from the videos, we transform a complex AI video detection problem into an AI image detection problem.

4.2 Experimental Setup

We adopt AM-softmax as loss function and use AUC as the metric to evaluate the testing results. The input images are resized to 256*256. To reduce prediction bias and enhance model performance, we construct the training set with equal amount of real and fake face images.

4.3 Training and Testing

Training process.

- **Experimental Process:** We use 15,300 original video face images and 15,300 images generated using one AI face-swapping technique (e.g., DF) as our dataset. First, we clean and shuffle this data, then split it into a training set and a validation set in a 3:1 ratio. Each image is randomly horizontally flipped and randomly cropped. Finally, the images are converted into tensors and normalized.
- **Model Training:** We selected different models for training, primarily focusing on the ResNet model, the original paper's Two-stream Model, and our modified Two-stream Gram Model. Using the trained models, we predicted a mix of FS images and real images, and saved the prediction results in the corresponding files.
- **Different Models:** We used different fake face datasets (DF, FS, DFD, F2F) as the training set. For each fake face dataset, we created a mixed dataset (DF and original videos, FS and original videos, NT and original videos, F2F and original videos) to use as the training set. We trained three models on each combination, resulting in a total of 12 models (4*3). Each model was trained for 15 epochs.

Figure 6 visualizes the training loss throughout the training process of the models, observing a smoother convergence and a faster convergence speed of the Two-stream Gram Matrix Model than the other two models.

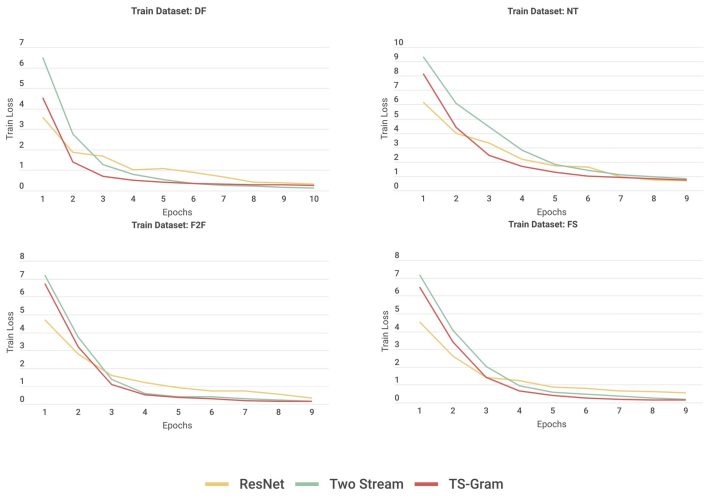


Figure 6: Training loss of models trained on different types of fake images

Within-dataset prediction. We first evaluate the performance of the models on distinguishing fake data generated by the same manipulation technique as it trained on. The results in Table 1 indicate that the two stream based models generally outperform resnet.

Cross-dataset prediction. We subsequently evaluate the cross-dataset generalization ability of the models. Results in Figure 7 show that models trained on datasets manipulated by one deep learning-based face-faking method exhibits better generalization to datasets manipulated by other deep learning-based methods. However, these models demonstrated lower performance when tested on datasets manipulated by face swap techniques.

Notably, our model achieves a slight improvement in cross-method prediction scenarios, although this sometimes came at the expense of a reduction in within-method AUC. Similar observations were made for models trained on face swap data.

	ResNet	Two-stream	TS-Gram
DF	0.921	0.988	**0.994**
NT	**0.904**	0.884	0.901
F2F	0.913	**0.961**	0.956
FS	0.909	0.962	**0.981**

Table 1: Within-dataset test AUC

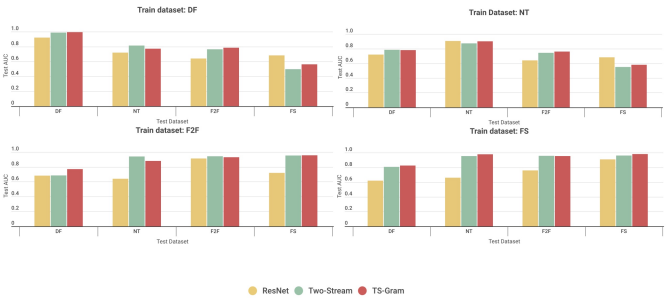


Figure 7: Model performance on detecting different forgery techniques

5 Conclusion

In conclusion, our model demonstrates superior convergence capabilities than the models used for comparison. This enhanced performance can be attributed to the global information aggregation of the gram matrix in each epoch. Additionally, our model displays promise in enhancing cross-methods generalizability, suggesting its potential for broader applicability across various AI face forgery methods.

However, it is also important to be aware of several limitations in our study. Firstly, the insufficiency of training data may lead to underestimated model performance, potentially limiting the model's ability to generalize effectively. Secondly, the utilized face detection algorithm does not distinguish real and fake faces, introducing mislabeled real images into the fake face data samples, potentially compromising model performance, particularly when multiple faces appear in a single frame. While such occurrences are not frequent in the datasets, they nevertheless raise a concern, highlighting the importance of improving the face detection algorithm to ensure accurate labeling and improve model robustness. Addressing these limitations will be critical for further enhancing the efficacy and reliability of our model in real-world applications.

References

[1] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 2018.

[2] Zhengzhe Liu, Xiaojuan Qi, and Philip H.S. Torr. Global texture enhancement for fake face detection in the wild. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[3] Yuchen Luo, Yong Zhang, Junchi Yan, and Wei Liu. Generalizing face forgery detection with high-frequency features. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2021.

[4] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Niessner. Faceforensics++: Learning to detect manipulated facial images. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.