Deepfake Detection: Attention-based Suspicious Region In-consistency Detector (ASRID)

Yicheng Wang Qixuan Guo Yujun Zhang
Hong Kong University of Science and Technology
{ywangmy, qguoai, yzhanglo}@connect.ust.hk

Abstract

Deepfakes refer to videos where the person's face has been convincingly manipulated by AI in different forms, such as identity swap, expression swap, attribute manipulation, and even entire face synthesis. In this paper, we try to explore new approaches and opportunities in the detection of Deepfakes and proposed an Attention-based Suspicious Region In-consistency Detector. Our work builds on related works and derives the score from both static and dynamic perspective, achieving a respectively satisfying accuracy of 85% in prediction.

1. Introduction

Due to easy access to vast public databases and the quick development of deep learning techniques, specifically Generative Adversarial Networks, incredibly realistic false material with concomitant social repercussions has been produced in this era of fake news [12]. There are significant risks because these technologies may be exploited maliciously. We can minimize the adverse societal effects of spoofing video content and unethical conduct with efficient Deepfake detection.

2. Related Work

2.1. Deepfake Detection Challenge

The Deepfake Detection Challenge (DFDC) [4] initiated in 2019 aims to encourage academics to develop new tools that can assist in identifying Deepfakes and altered media. Up to the closing of the submission, the Deepfake Detection Challenge has received 8,581 entries from 2,265 teams. Winner teams on the leaderboard have their code made public and can be referenced as previous work.

2.2. Prize Winning Models

All the prize winning models take advantage of various augmentation approaches, as well as the ensemble of multiple independently trained models. See Table 1 for the results.

Model	Public (train & val)	Private (test)
1st	≈ 0.2	0.4279
2nd	0.287	0.4284
3rd	$\sim$	0.4352

Table 1. Results of Top winner

1st Place solution The first place solution [10] suggested that approaches other than frame-by-frame classification did not work well on the dataset. It also mentioned that the model relies heavily on data augmetation, particularly the dropout of part of the face like nose, eyes, or mouth.

2nd Place solution The second place team started with the CNN-LSTM model that captures sequential information, but switched to frame-by-frame model due to the bad outcome of the former one. They found that the Weakly Supervised Data Augmentation Network model [7] very useful for training this binary classification task with minor supervision.

3rd Place solution The third place team [9] pointed out that the main difficulties of Deepfake detection is the severe overfitting. To resolve this issue, they came up with a novel augmentation strategy that mixed the fake image and the real image which contorlled by a weight between zero and one, to enable a fractional label that is equal to the weight of the real image. To a certain extent, this approach also solve the issue of little supervision provided by the binary label.

2.3. The Multi-attention Model

The Multi-attentional Deepfake Detection model [13] includes several spatial attention heads to direct the network's to attend on various local parts, and adds textural feature enhancement block to magnify the subtle artifacts in shallow features. They also proposed an auxiliary loss that measures the low-level textural feature and regional independence to significantly improve the accuracy of prediction.

3. Data

3.1. DFDC Dataset

The DFDC dataset is by far the largest currently and publicly-available face swap video dataset, with over 124,000 total clips sourced from 3,426 paid actors, featuring eight facial modification algorithms [2]. The outstanding forgery quality of the fake videos in this dataset makes it the most challenging dataset for Deepfake detection tasks at the moment. The whole dataset is of size of around 470GB, and we will randomly sample a small part of them for training, while part of the remaining ones can be used to test and evaluate.

3.2. Data Preprocessing

Data Augmentation From our survey, we found that many models suffer from severe overfit Our conjecture is that the videos in the dataset are all taken from a limited number of people, and multiple fake videos are produced from a single original video, which may induce the model to memorize these faces and lose generalization ability. Therefore, proper data augmentations are subtle.

The data augmentation [1] is performed every time when the data loader sample the images. In order to improve the model's performance by protecting it from overfitting, the images go through a series of operations, with proper compression, noise, resizing, and color jittering. Since we add some randomness to the process, this improves the data even further while reducing the amount of time needed for preprocessing. Some examples of the augmented images are shown in Figure 1

Face Alignment The actual Deepfake manipulations are only applied on faces, but some noises are added to the surroundings, which may distract the model. In addition, on account of training efficiency, processing the whole frame is quite time-consuming. Hence, we utilize FaceNet [3] face detector package implemented in PyTorch for face alignment which can recognize multiple faces in a video

Figure 1. Data Augmentation

automatically centralize and resize the face. https://github.com/timesler/facenet-pytorch

Extracting Frames and Face Regions Note that the fake videos have exactly the same geometry with the original one, so we can just apply above face detector to the original video, compute the box coordinates of the face, and use them to also crop the corresponding frames in the fake videos.

According to our survey, the sequential information is not as important (though we still propose a dynamic model for that), thus we extract one out of every ten frames, and at most 30 frames per video, ignoring videos that are too short.

Together with the augmentation utility mentioned above, the cropped face images are resized to 380×380 pixels.

Splitting Dataset DFDC officials has divided each video into training and validation sets, however, we re-assign the videos randomly to training, validation and testing sets, with proportion $7 : 2 : 1$, respectively.

4. Methods

4.1. Overview

Our Method utilizes the attention technique to focus on specific features of faces, where it evaluates the criteria by combining two perspectives: static images and sequential images. The main idea is to concurrently learn discriminative areas at multiple scales or picture sections and drive the fusion of these features from different regions and contributes to the prediction with the integration of static and dynamic score. See Figure 2

4.2. Model: Attention-based Suspicious Region Inconsistency Detector(ASRID)

For the details about the components of our model, please refer to the Ablation Study section below.

Feature Extraction, Baseline and Attention The baseline model is a pretrained feature extractor leveraging the MobileNetV3-Small [6] with a simple static fully-connected classifier. Given how MobileNetV3-Small outperforms the MobileNetV3-Large with multiplier scaled to match the performance by nearly 3% (see Figure 4, we adopt MobileNetV3-Small as the backbone network for the consideration of its excellent performance given its low memory consumption.

According to the Multi-attentional Deepfake Detection paper [13], multihead attention can help to focus on independent fake parts of a image, while a single attention cannot attend to all the fake information precisely. The multiple

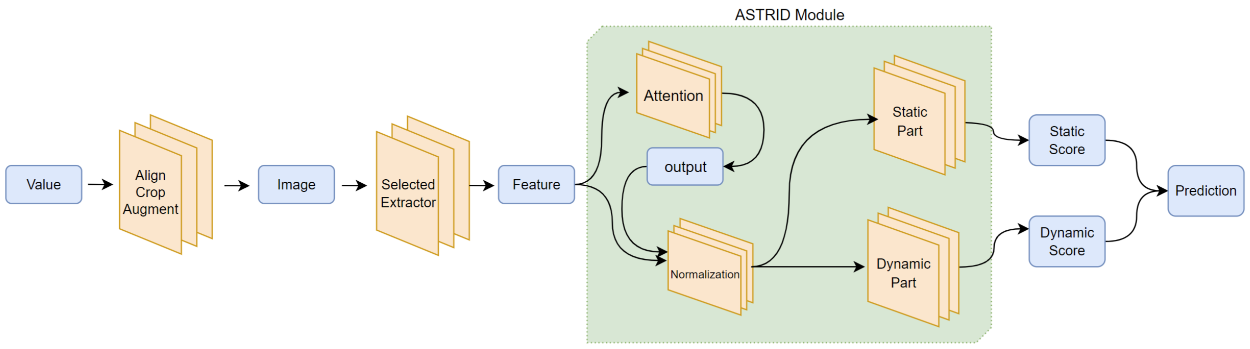

Figure 2. Model Architecture

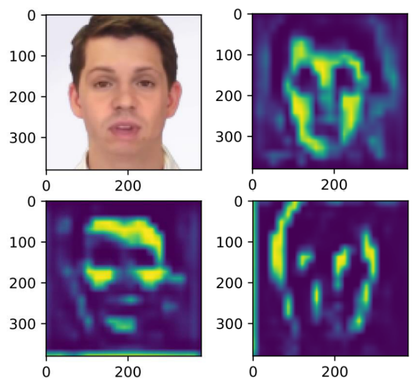

Figure 3. Multiple Attention Regions, retrieved from [13]

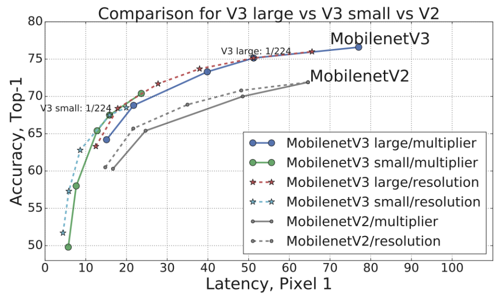

Figure 4. Performance of MobileNetV3 as a function of different multipliers and resolutions [6]

attention regions trained by our model should look like Figure 3

Skip Connection and Mixing Feature This layer concatenates (low-level) features extracted by the pretrained model and the (high-level) attention produced by our model. The mixed feature representation is then passed through a normalization layer. The final output is then fed to the static model and dynamic model. Instead of directly using the attention output, it froms a variant of residual block with a skip connection from the previous feature extraction layer.

Static (Sampling Picture) The input pictures are from frames sampled randomly from the videos in the dataset. This sampling method made the frames spread across the dataset more evenly, covering more diversities of different faces and with fewer frames per face. The evaluation of a small set of testing data is shown in the graph. See Figure 5

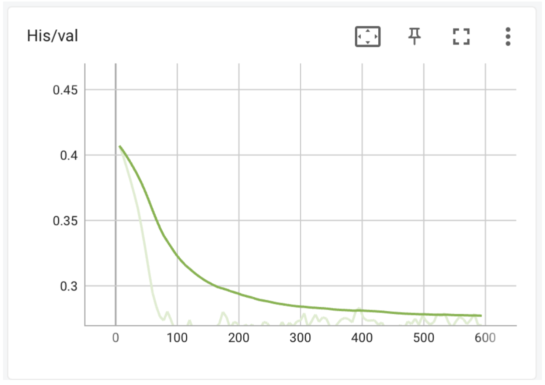

Figure 5. Static(Sampling Picture)

Static (Sampling Video) Another way is to sample by videos, which randomly takes a fixed amount of frames from a fixed amount of videos randomly selected in the dataset, holding the assumption that more clustered data would give a better result. However, sampling from each video cut down the amount of diversity of different faces and put more weight on each sampled video. As shown in the graph, sampling with pictures outperformed sampling with the video approach in loss evaluation. See Figure 6

Our best static module consists of a self-attention layer to focus on important part of the input feature representation, a batchnorm layer to improve the gradient flow, and two fully-connected layers to classify the images.

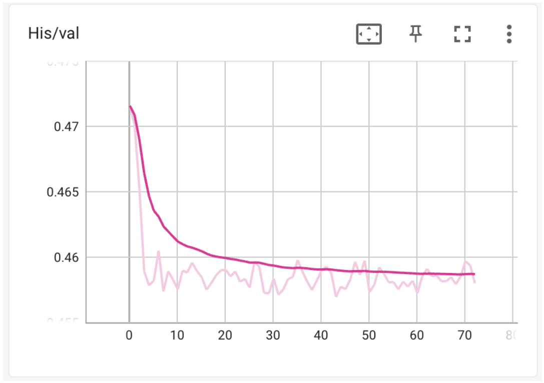

Figure 6. Static(Sampling Video)

Dynamic (Sampling Video) Although previous studies ([10], [5], [9]) indicate that the a complex model for sequential information does not perform well, we still want to dig out some ways to help the static part make predictions. The motivation is that even though the facial features themselves look realistic, the movement of face boundary may not match the movement of facial features in the sense of geometry.

According to the sample videos from the dataset that we accessed ourselves, it is hard to tell whether each frame is fake, but it is rather easy to determine the results from the inconsistent changes in facial expressions across consecutive frames. To take into consideration of consistency measurement, the dynamic approach samples the fixed amount of adjacent frames from randomly selected videos in the dataset and arranges the frames sequentially in a series of data. The dynamic approach is combined with the static approach sampling from the videos, with adjustments to the strength, and performs a respectively better result. See Figure 7

A sequential model is needed for the dynamic part. We choose to use the transformer model, in particular its encoder part. Considering the selected frames of a video as a sequential input, we feed the feature representations of each frame to the encoder, together with a positional encoding to avoid order invariance. After that, we compute the mean value over the output sequence, and use a fully-connected layer to perform classification.

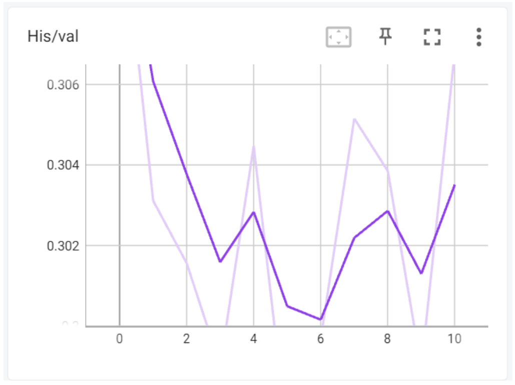

Figure 7. Dynamic(Sampling Video)

4.3. Loss Evaluation

Quantitatatively, we use the loss function introduced by [4]:

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \quad (1)$$

where

- n is the number of videos being predicted
- $\hat{y}_i$ is the predicted probability of the video being FAKE
- y_i is 1 if the video is FAKE, 0 if REAL
- $\log()$ is the natural (base e) logarithm

The usage of logarithm provides strong punishments for being most confident but wrong. A useful comment to understand this metric: if a model always output 0.5, this LogLoss is around 0.69.

5. Experiments

5.1. Ablation Study of Various Components

StaPic The static model that samples pictures instead of videos in each batch.

StaNew & StaNew + Mix The New Static Model replaces the single FC classifier with the self-attention module, applies batch normalization on direct attention output and is connected to 2 fully-connected layers. The StaNew + Mix model is basically the previous one with input replaced by the mixed and normalized feature.

Dyn and Dyn + Mix As stated in the previous sections, we decide to use the transformer model as our dynamic block. Similarly, the idea of skip connection can be applied to the Dyn + Mix variant.

Model	Tr.	Ave. Tr.	Val.	Var
StaPic	⇕	0.578	0.270	Medium
Baseline	⇕	0.385	0.818	Huge
StaNew	0.274	0.389	0.111	Medium
+ Mix	0.298	0.442	0.470	Small
+ Dyn	0.308	0.447	0.305	Medium
+ Mix	0.214	0.390	0.376	Depends

Table 2. Ablation Study

Experiment on small dataset The small dataset consists about 700 videos x 30 frames

See Table 2 for the results. Remark:

- For the StaPic and baseline model, the variation of the last-iteration loss is very large, e.g. from ≈ 0.4 to ≈ 0.9 and from < 0.3 to > 1, respectively.
- The variation of Dyn + Mix depends on what the dynamic strength it takes (see the corresponding section below).

On the right of Figure 8 shows the validation loss of these strategies. From top to down they are baseline, StaNew + Mix, Dyn + Mix, Dyn and Sta.

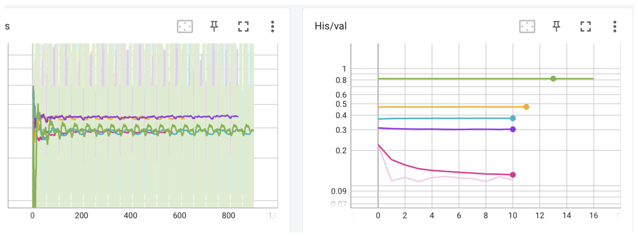

Figure 8. Strategies

Analysis:

- Baseline vs StaNew (see Figure 9): The StaNew model significantly overperform the baseline mode. It has higher loss in training, but much lower in validation. This indicates that the StaNew model has stronger generalization ability.

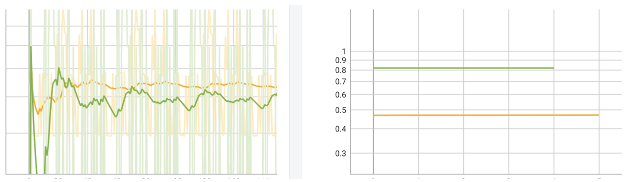

Figure 9. Baseline vs StaNew

- Direct connect vs + Mix (see Figure 10): The mixture of feature output (original feature) and attention output (deeper feature) which serves as a deep residual connect, has smaller variation (grey part) than the ones without this connect (purple part). However, this mixing has worse performance on small data. However, we expect the mixing will overperform the baseline on the entire data.

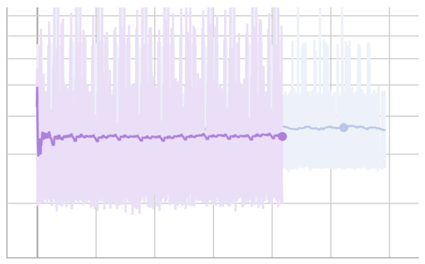

Figure 10. Direct connect vs Mixing

- Dyn vs Sta (see Figure 11): Similarly outperforms in training, but Dyn is worse in Val. A possible reason is that the dynamic model contains a large transformer block, which is harder to train on a small dataset.

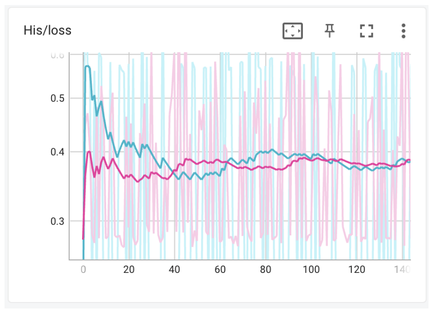

Figure 11. Dynamic vs Static

Meanwhile, the dynamic model consumes a larger amount of GPU memory, and possibly only suits to smaller batch size in the same hardware setting.

5.2. Hyperparameters

Learning Rate Scheduler We mainly use Cosine Annealing Warm Restarts [8], ReduceLR On Plateau(reduce learning rate when a metric has stopped improving), and OneCycleLR [11]. They are robust and efficient to find suitable learning rate with less effort on fine-tuning. To be specific, Cosine scheduler will spend more and more time on large learning rate but still keep touching to small learning rate in each period; ReduceLR On Plateau will adjust based on performance; OneCycleLR will start with a fairly small learning rate and then increase to the desired maximum learning rate, and then eventually decrease to almost 0 to further fine-tune.

Data Size The full DFDC dataset is abundant but requires great computational power and large storage. Our data preprocessing produces about 40 GB of cropped images for only 4 chunks of data among 50 in the full dataset.

Figure 12 shows the difference between training and validation on large and small data. The small one is in green and the large one is in blue.

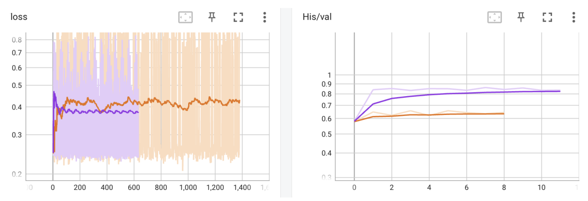

Figure 12. Large and small data

As we can see, the model on smaller data actually performs worse in training set.However, it displays a obvious trend of convergence in validation loss, while the model reduces the validation loss extremely slow on the larger data.

Batch Size The purple curve represents batchsize 6, while the orange one represents batchsize 2.

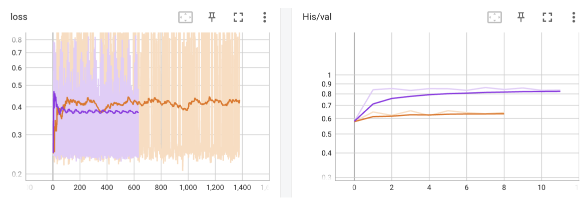

Figure 13. Batchsize

As shown in Figure 13, smaller batchsize performs worse than the larger one. Since we determined to sample videoes instead of pictures in each batch, smaller batchsize results in huge loss in the diversity of data per batch, and hence leads to slower learning speed of the model and larger variation.

Dynamic Strength We compose the static loss and dynamic loss by a weight $w \in (0, 1)$,

$$\mathcal{L}_{\text{total}} = w \cdot \mathcal{L}_{\text{static}} + (1 - w) \cdot \mathcal{L}_{\text{dynamic}}$$

And the results of $w = 0.8$ (0.2 strength for Dyn) and $w = 0.95$ (0.05) are as follows:

Model	Tr.	Ave. Tr.	Val.	Var
Dyn (0.2)	0.308	0.447	0.305	Medium
(0.2) + Mix	0.214	0.390	0.376	Medium
(0.05) + Mix	0.374	0.464	0.538	Very small

StaNew	loss	accuracy	loss	accuracy
train	0.447	0.878	0.469	0.875
val	0.280	0.900	0.519	0.849
test	0.541	0.800	0.518	0.851

Table 3. Prediction Results. The two columns on the left are for small data, the two on the right are for large data

StaNew+Mix	loss	accuracy	loss	accuracy
train	0.412	0.888	0.425	0.875
val	0.471	0.833	0.446	0.857
test	0.288	1.000	0.449	0.854

Table 4. Prediction Results. Small data

We observe that larger strength for dynamic loss leads to lower average loss, but large variation. A possible reason is that the transformer in the dynamic block provides more stength of normalization.

5.3. Hyperparameter Tuning

The batch size of 6 is mainly due to a consideration of memory. When running on HKUST UG Cluster which provides 4 GPUs, we are training on a 24 batch size scale before each parameter updating, which should be good enough and no way to tune. With the advanced learning rate scheduler, the effort on learning rate tuning is also little. Weight decaying is tuned based on the level of overfitting, and we mainly use 0.01.

5.4. Prediction Results

See Table 3, 4 for the prediction results of different variants.

In short, the +Mix model shows better regularization ability.

5.5. Reproduce Ability

To ensure the reproduce ability, we set random seed for numpy and torch in every runs. At the same time, to avoid overfitting on random seed, we hash the training configuration file content to produce a random seed, such that as long as using the same core configuration(learning rate, model archtecture, et. al), the same results can be reproduced.

To further help with training, we also provide checkpoint saving and restore module. Status of model parameters, optimizer, scheduler and the training progress can be restored easily.

6. Conclusion and Future Work

We could try to design a more fine-grained training process by adapting different focuses during different periods. The transfer learning of pre-trained model should go first.

At the same time, we could try fine-tuning the learning rate for each parameter group in different modules at different periods of training. For example, the pre-trained model should use a large learning rate when trying to adapt to the new problem context. After that, it should adopt a lower learning rate to produce a relatively stable input for later dynamic attention part, which is far more complex and hard to train. These can be done by adapting multiple optimizers focusing on different peroids of training. Our modular design of training process will make this improvement easy to implement.

References

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. 2

[2] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, and Cristian Canton Ferrer. The deepfake detection challenge (dfdc) dataset, 2020. 2

[3] Tim Esler. Facenet-pytorch: Face recognition using pytorch, 2021. https://github.com/timesler/facenet-pytorch. 2

[4] Facebook. Deepfake detection challenge, 2019. https://www.kaggle.com/c/deepfake-detection-challenge. 1, 4

[5] Cui Hao. kaggle-dfdc, 2019. https://github.com/cuihaoleo/kaggle-dfdc. 4

[6] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019. 2, 3

[7] Tao Hu, Honggang Qi, Qingming Huang, and Yan Lu. See better before looking closer: Weakly supervised data augmentation network for fine-grained visual classification, 2019. 1

[8] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2016. 5

[9] NTech-lab. Deepfake detection challenge, 2019. https://github.com/NTech-Lab/deepfake-detection-challengeh. 1, 4

[10] Selim Seferbekov. Deepfake detection (dfdc) solution by @selimsef, 2019. https://github.com/selimsef/dfdc_deepfake_challenge. 1, 4

[11] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2017. 5

[12] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection. 2020. 1

[13] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. Multi-attentional deepfake detection, 2021. 1, 2, 3