AUTO-FAS: SEARCHING LIGHTWEIGHT NETWORKS FOR FACE ANTI-SPOOFING

Zitong Yu[1], Yunxiao Qin[2*], Xiaqing Xu[3], Chenxu Zhao[3], Zezheng Wang[3], Zhen Lei[4], Guoying Zhao[1]*

[1]CMVS, University of Oulu [2]Northwestern Polytechnical University [3]Aibee
[4]NLPR, Institute of Automation, Chinese Academy of Sciences

ABSTRACT

With the development of mobile devices, it is hopeful and pressing to deploy face recognition and face anti-spoofing (FAS) model on cell phone or portable devices. Most of existing face anti-spoofing methods focus on building computational costly detector for better spoofing face detection performance. However, these detectors are unfriendly to be deployed on the mobile device for real-time FAS applications. In this paper, we propose a neural architecture search (NAS) based method called Auto-FAS, intending to discover well-suitable lightweight networks for mobile-level face anti-spoofing. In Auto-FAS, a special search space is designed to restrict the model's size, and pixel-wise binary supervision is used to improve the model's performance. We demonstrate both the effectiveness and efficiency of the proposed approach on three public benchmark datasets, which shows the potential real-time FAS application for mobile devices.

***Index Terms*—** Face anti-spoofing, mobile, neural architecture search

1. INTRODUCTION

Nowadays, face recognition has been widely used in lots of identity verification and payment applications. Face anti-spoofing (FAS), intending to protect face recognition from presentation attacks, has also developed rapidly. Traditional face anti-spoofing methods [1, 2] usually extract hand-crafted features from the facial images to capture the spoofing patterns, which is efficient but limited performance for such face security applications. Recently, deep learning based face anti-spoofing methods [3, 4, 5, 6, 7] train expert-designed models to detect the presentation attacks. These models usually have huge parameters and are computational costly, so that they are usually deployed in backend servers.

As mobile devices are widely used in daily lives, the demand for deploying face recognition and the corresponding face anti-spoofing on mobile devices arises. Deploying face anti-spoofing on mobile devices has several advantages, *e.g.,* convenient for customer to use, little network data transmission, *etc.*. However, considering the limited computational performance and the storage of mobile devices, the mobile-level model should be lightweight.

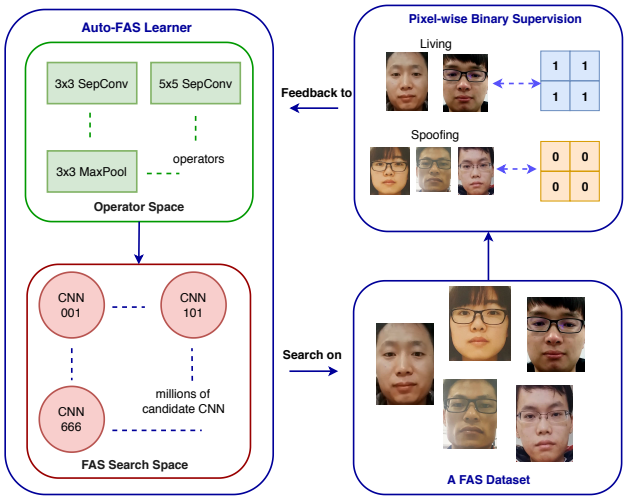


Fig. 1. Our Auto-FAS learns to search for a well-suited architecture on a specific FAS dataset, and it is supervised by the deep pixel-wise binary supervision during searching. Auto-FAS discovers lightweight networks from a FAS search space, which consists of a large number of candidate architectures. These candidates are generated by combining basic operations, such as a 3×3 separate convolutional layer and a 3×3 max pooling operation.

In order to obtain lightweight models, researchers traditionally utilize pruning [8], quantization [9] or distillation [10] methods to compress heavy models. However, these methods more or less damage the model's performance in the compressing procedure. In this paper, we propose a neural architecture search (NAS) based method (Auto-FAS) to discover a well-performing lightweight model for mobile-level face anti-spoofing.

NAS methods can be categorized into three directions: 1) Gradient based [11, 12], 2) Evolution algorithm based [13], and 3) Reinforcement learning based [14, 15]. Compared with reinforcement learning and evolution based network search methods, gradient based methods search the network architecture more efficiently, mainly because they relaxes the discrete operations decision process as weighted summation of different operations. In this paper, we choose the gradient based methods to search the lightweight face anti-spoofing

* denotes equal contribution

model in Auto-FAS for its fast convergence.

In previous NAS algorithms [11, 12], cross-entropy loss is utilized as the supervision for both the searching and the retraining stage. However, face anti-spoofing task differs greatly from the generic object classification task. The FAS task prefers to capture more detailed patch-based clues for distinguishing spoofing faces from living faces, while the generic object classification task focus more on semantic features. Compared with cross-entropy, dense pixel-wise binary cross-entropy[16] has been proved to be more effective for FAS task. Thus, in Auto-FAS, we adopt pixel-wise binary supervision in both the searching and the retraining stage.

Our contributions are summarized as the follows:

- We propose a neural architecture search based method (Auto-FAS) to search a lightweight face anti-spoofing model which simultaneously satisfies the requiremet of small parameters, low computational cost and high performance. To the best of our knowledge, it is the first work to introduce deep pixel-wise binary supervision based NAS method to search mobile-level networks specifically for face anti-spoofing task.
- We executed comprehensive experiments on OULU-NPU[17], CASIA-MFSD [18] and Replay-Attack [19] datasets for intra- and cross-testing. All experimental results show that the searched mobile-level model (0.27M parameters) performs comparable with the other state-of-the-art methods.

2. METHODOLOGY

Instead of manually designing a lightweight model for mobile-level face anti-spoofing problem, we search such a model automatically. In this paper, inspired by recent developed gradient based NAS methods, we propose a novel neural architecture search based method (Auto-FAS) to discover lightweight networks for face anti-spoofing task. Here, we will detail our Auto-FAS method, including the search space, the search strategy and the supervision.

2.1. Auto-FAS

In Auto-FAS, the searched network is a cascade of several cells, and each cell is a directed acyclic graph (DAG) containing N nodes. Each node of the graph is formed using a feature $x^{(i)}$. The edge which connects node $x^{(i)}$ and $x^{(j)}$ is denoted as (i, j), and on this edge, $x^{(i)}$ passes forward to node $x^{(j)}$ through operation $f^{(i,j)}$. Node $x^{(j)}$ is a summation of all the forward results of pre-nodes. Therefore, node $x^{(j)}$ can be presented as

$$x_j = \sum_i f^{(i,j)}(x_i), \tag{1}$$

where $0 \leq i < j \leq N-1$. The operation $f^{(i,j)}$ is a composition of several candidates (*e.g.*, convolution, pooling, and

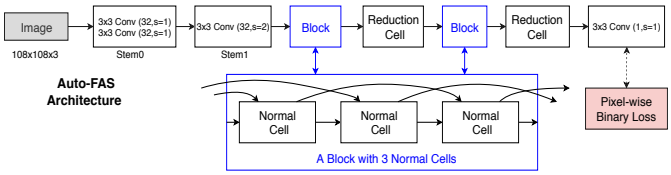


Fig. 2. The strategy to design Auto-FAS architecture based on the discovered cell. Both normal and reduction cells receive the outputs of two previous cells as inputs, as illustrated in the middle.

etc.). So the operation $f^{(i,j)}$ can be represented as

$$\begin{aligned} f^{(i,j)}(x_i) &= \sum_{o \in \mathcal{O}} \beta_o^{(i,j)} \cdot o(x_i), \\ \beta_o^{(i,j)} &= \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} exp(\alpha_{o'}^{(i,j)})}, \end{aligned} \tag{2}$$

where $\mathcal{O}$ is the set of candidate operations, and $o(x_i)$ is the output of operation o with node x_i as the input. $\beta_o^{(i,j)}$ is the weight of the operation o in $f^{(i,j)}(x_i)$, and when $\beta_o^{(i,j)}$ getting larger and larger, $f^{(i,j)}$ is more and more determined by the operation o. $\alpha_o^{(i,j)}$ if a trainable variable that is used to calculate $\beta_o^{(i,j)}$ with the softmax function. As a summary, all the trainable $\alpha_o^{(i,j)}$ determines the networks architecture. So, the task of searching the network architecture turns to optimizing all $\alpha_o^{(i,j)}$ in the network.

2.1.1. Search Space

To ensure both the efficiency and effectiveness of the searched architectures, Auto-FAS holds a search space specially designed for mobile-level face anti-spoofing task, which is illustrated in Fig.2.

Network level. Unlike the state-of-the-art method [6] using 256x256x6 facial image (RGB and HSV color space) as input, our Auto-FAS only takes 108x108x3 facial image (RGB) as input and predict the 14x14 binary map, which is more suitable for mobile-level application. We search for two kinds of cells in networks, i.e., a normal cell and a reduction cell. For the normal cell, each operator function has the stride of 1 while the first operator function has the stride of 2 for the reduction cell. The input nodes of each cell are propagated from the output nodes of two previous cells.

Cell level. Each cell contains 7 nodes, including two input nodes, four intermediate nodes and one output node. The edge connections to the intermediate nodes denote summation operation while the output node concatenates all results from intermediate nodes.

Operator level. There are six operator candidates in Auto-FAS, i.e., 'none', 'max_pool_3x3', 'avg_pool_3x3', 'skip_connect', 'sep_conv_3x3' and 'sep_conv_5x5'. The reason for only considering separate convolutions is that such operator is proved to be more efficient in mobile-level networks. As a result, the total searching space is $2 \times 6^{(2+3+4+5)} = 2 \times 6^{14}$.

2.1.2. Search Strategy

Following the similar bi-level optimization strategy [11], we denote $\alpha = \{\alpha_o^{(i,j)}\}$ as the set of all $\alpha_o^{(i,j)}$, and α presents the network architecture. Then the prediction of the network can be formulated as

$$\hat{y} = \mathcal{F}(x; \theta, \alpha), \tag{3}$$

where θ is the network's weight. Weight θ and architecture α are optimized alternatively on the train and validation set. On the train set, the optimizing of θ can be formulated as

$$\theta(\alpha) = \theta - \gamma_1 \cdot \nabla_\theta L_{train}(\theta, \alpha), \tag{4}$$

where $\theta(\alpha)$ is the update result of θ conditioned by current architecture α. γ_1 and L_{train} are the learning rate and loss on the train set, respectively. On the validation set, the optimizing of α can be formulated as

$$\begin{aligned} \alpha =& \alpha - \gamma_2 \cdot \nabla_\alpha L_{val}(\theta(\alpha), \alpha) \\ =& \alpha - \gamma_2 \cdot \nabla_\alpha L_{val}(\theta - \gamma_1 \cdot \nabla_\theta L_{train}(\theta, \alpha), \alpha), \end{aligned} \tag{5}$$

where γ_2 and L_{val} are the learning rate and the loss on the validation set, respectively. By alternatively optimizing θ and α with Eq.(4) and Eq.(5), the searching stage converges gradually. After convergence,the operations with the largest weight $max_{o \in \mathcal{O}, o \neq none}\, \beta_o^{(i,j)}$ and two incoming edges with two largest $max_{o \in \mathcal{O}, o \neq none}\, \beta_o^{(i,j)}$ are adopted to form the final discrete architecture.

2.2. Supervision

Traditionally, simple binary softmax loss function is used to supervise the face anti-spoofing model to learn discrimination between spoofing and living faces. Recently, some work show that pixel-wised facial depth supervision [5] which provides dense supervision for the training of face anti-spoofing model, and leads the model learn better discrimination. However, collecting facial depth label is somewhat costly. In this paper, to take the advantage of pixel-wise supervision [16], and to improve the network search efficiency, we use a pixel-wise binary cross-entropy loss to supervise the search procedure. It can be seen in Fig. 1 that the binary pixel-wise label is easy to generate, which can be treated as expand of the patch-level binary mask of spoofing attacks. The loss function can be formulated as

$$L(x) = \frac{1}{m \cdot n} \sum_{i=1}^{n} \sum_{j=1}^{m} -y_{ij} \cdot log(\hat{y_{ij}}) - (1 - y_{ij}) \cdot log(1 - \hat{y_{ij}}), \tag{6}$$

where y is the binary pixel-wise label, and m and n are the height and width of y, respectively. $\hat{y}$ is the model's prediction of input x with the model's weight θ with respect to architecture parameters α. $y_{i,j}$ is the value at the i-th row and j-th col of y, so is $\hat{y_{ij}}$.

Table 1. Intra testing result on OULU-NPU.

Prot.	Method	APCER(%)	BPCER(%)	ACER(%)
1	DeepPixBiS [16]	0.83	0.0	0.42
	HKBU [20]	9.6	18.3	14.0
	NWPU [20]	8.8	21.7	15.2
	Auxiliary [6]	1.6	1.6	**1.6**
	Auto-FAS (Ours)	12.1	5.0	8.5
2	DeepPixBiS [16]	11.39	0.56	5.97
	HKBU [20]	13.9	5.6	9.7
	NWPU [20]	12.5	26.7	19.6
	Auxiliary [6]	2.7	2.7	**2.7**
	Auto-FAS (Ours)	19.4	4.8	12.1
3	DeepPixBiS [16]	11.7±19.6	10.6±14.1	11.1±9.4
	HKBU [20]	12.8±11.0	11.4±9.0	12.1±6.5
	NWPU [20]	3.2±2.6	33.9±10.3	18.5±4.4
	Auxiliary [6]	2.7±1.3	3.1±1.7	**2.9±1.5**
	Auto-FAS (Ours)	11.3±7.3	9.2±5.2	10.2±3.3
4	DeepPixBiS [16]	36.7±29.7	13.3±14.1	25.0±12.7
	HKBU [20]	33.3±37.9	27.5±20.4	30.4±20.8
	NWPU [20]	30.8±7.4	44.2±23.3	37.5±9.4
	Auxiliary [6]	9.3±5.6	10.4±6.0	**9.5±6.0**
	Auto-FAS (Ours)	17.9±9.8	9.2±9.7	13.6±7.5

3. EXPERIMENT

In this section, we will show detail of our experiment, including the experiment setting, results and analysis.

3.1. Database

In this paper, we evaluate the proposed Auto-FAS on three popular face anti-spoofing databases. They are OULU-NPU[17], CASIA-MFSD [18] and Replay-Attack [19]. OULU-NPU contains 4950 high-resolution real and spoofing videos, and four protocols to validate the generalization (e.g., unseen illumination and attack medium) of models. We strictly obey these protocols in our experiment. CASIA-MFSD and Replay-Attack contain lots of low-resolution real and spoofing videos, and they are used for cross testing.

3.2. Metrics

Three metrics are used in the experiment on OULU-NPU. They are: 1) Attack Presentation Classification Error Rate (APCER); 2) Bona Fide Presentation Classification Error Rate (BPCER); 3) ACER which calculates the mean of APCER and BPCER as shown in Eq.(7). Metric Half Total Error Rate (HTER) is used in the cross testing experiments.

$$ACER = (APCER + BPCER)/2. \tag{7}$$

3.3. Implementation Details

The proposed Auto-FAS is implemented with Pytorch library. 8 NVIDIA GTX 1080Ti GPUs are used for searching, and the total batch size is 256. Partial channel connections [12] are adopted for accelerating search process. For optimizing network weight θ, we use the momentum SGD optimizer with initial learning rate of 0.05 following cosine schedule, and the

Table 2. Cross-dataset testing result on CASIA-MFSD and Replay-Attack. HTER (%) is the metric here.

Method	Train	Test	Train	Test
	CASIA-MFSD	Repaly-Attack	Repaly-Attack	CASIA-MFSD
Motion-Mag [21]	50.1		47.0	
Spectral cubes [22]	34.4		50.0	
FaceDs [23]	28.5		41.1	
Auxiliary [6]	27.6		**28.4**	
Auto-FAS (Ours)	**15.6**		40.7	

Table 3. Performance of searched networks with different supervision on OULU-NPU Protocol-1.

Supervision	APCER(%)	BPCER(%)	ACER(%)
Binary cross-entropy	12.5	16.0	14.2
Pixel-wise binary cross-entropy	**12.1**	**5.0**	**8.5**

momentum is set to 0.9, and weight decay to 1e-4. For architecture parameters α, we use the Adam optimizer with fixed learning rate of 6e-3, and set the momentum to $\{0.5, 0.999\}$ and weight decay to 1e-3. We search the network for 40 epochs on the Protocol-1 of OULU-NPU. Fig. 3 visualizes the searched network architecture.

3.4. Experiment on OULU-NPU

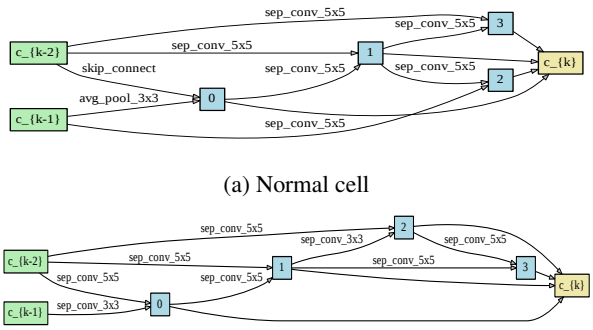
The experimental result of Auto-FAS on OULU-NPU dataset is shown in Tab.1. We can see that on all protocols of OULU-NPU, Auto-FAS outperforms HKBU [20] and NWPU [20] with clear margin, which indicates that the searched model is robust to all protocols. Another phenomenon is that the more challenging the protocol is, the smaller the margin between Auto-FAS and Auxiliary becomes (protocol 3 and 4 are more challenging than protocol 1 and 2). Specifically, compared with DeepPixBiS [16] which is supervised by pixel-wise binary loss, Auto-FAS decreases the ACER by 8.1% and 45.6% on protocol 3 and 4, respectively. This experimental result reveals that the proposed Auto-FAS performs satisfactorily.

3.5. Cross Testing Experiment

The cross testing experiment is conducted on the dataset CASIA-MFSD and Replay-Attack. There are two protocols. One is training the model on CASIA-MFSD and then testing it on Replay-Attack. Another is training on Replay-Attack and testing on CASIA-MFSD. The experiment result is shown in Tab.2. We can see that our Auto-FAS achieves the state-of-the-art performance on the first protocol. Compared with the Auxiliary model, it decreases the ACER by 43.5%. And on the second protocol, Auto-FAS also achieves comparable performance. The cross testing result indicates that Auto-FAS generalizes well across different databases.

3.6. Impacts of Different Supervision

In this experiment, we search two networks on OULU-NPU Protocol-1 with two different supervisions, respectively. The two supervisions are traditional binary cross-entropy and deep pixel-wise binary cross-entropy. It can be seen from



(a) Normal cell

(b) Reduction cell

Fig. 3. Neural architecture searched by Auto-FAS.

Table 4. The efficiency and performance comparison on OULU-NPU Protocol-1.

Method	Params (Mb)	GFLOPS	ACER (%)
ResNet-18	11.69	**0.48**	11.91
Auxiliary (Depth) [6]	2.20	8.50	**7.90**
Auto-FAS (Ours)	**0.27**	0.53	8.50

Tab.3 that the searched network supervised by pixel-wise binary loss achieves 8.5% ACER, outperforming the network searched with binary loss with a large margin.

3.7. Efficiency Analysis

A model deployed in mobile devices should has small size and low FLOPS to save storage and computational cost, respectively. We analyse the model parameters, and FLOPS of Auto-FAS and other popular models in face anti-spoofing task, and show the results in Tab.4. The Auxiliary (Depth) model is the Auxiliary model trained only with facial depth label. For fair comparison, the Auxiliary (Depth) model is also trained on 108x108x3 RGB facial image, so is the ResNet-18 model. It can be seen in Tab.4 that the Auto-FAS is the most efficient model as it has only 0.27Mb parameters and 0.53 GFLOPS. In contrast, ResNet-18 has much more parameters and performs worse than Auto-FAS. Auxiliary (Depth) outperforms Auto-FAS slightly, however it needs more parameters and computational cost. All the results show that Auto-FAS is promising and reliable to be applied to practical mobile-level face anti-spoofing.

4. CONCLUSION

In this paper, we search a lightweight face anti-spoofing model (Auto-FAS) for mobile-level applications. Auto-FAS takes advantage of pixel-wised binary supervision to search well-performing tiny model for face anti-spoofing. The experimental results show that Auto-FAS makes a good trade-off between efficiency and accuracy, indicating the searched lightweight model is suitable to be applied on mobile devices.

Acknowledgement This work was supported by the Academy of Finland for project MiGA (Grant 316765), ICT 2023 project (Grant 328115), Infotech Oulu and the Chinese National Natural Science Foundation Projects (Grant No. 61876178). As well, the authors acknowledge CSC–IT Center for Science, Finland, for computational resources.

5. REFERENCES

[1] Tiago de Freitas Pereira, André Anjos, José Mario De Martino, and Sébastien Marcel, "Lbp- top based countermeasure against face spoofing attacks," in *ACCV*, 2012, pp. 121–132. 1

[2] Jukka Määttä, Abdenour Hadid, and Matti Pietikäinen, "Face spoofing detection from single images using micro-texture analysis," in *IJCB*, 2011, pp. 1–7. 1

[3] Yunxiao Qin, Chenxu Zhao, Xiangyu Zhu, Zezheng Wang, Zitong Yu, Tianyu Fu, Feng Zhou, Jingping Shi, and Zhen Lei, "Learning meta model for zero- and few-shot face anti-spoofing," *arXiv preprint arXiv:1904.12490*, 2019. 1

[4] Bofan Lin, Xiaobai Li, Zitong Yu, and Guoying Zhao, "Face liveness detection by rppg features and contextual patch-based cnn," in *ICBEA*, 2019, pp. 61–68. 1

[5] Yousef Atoum, Yaojie Liu, Amin Jourabloo, and Xiaoming Liu, "Face anti-spoofing using patch and depth-based cnns," in *IJCB*, 2017, pp. 319–328. 1, 3

[6] Yaojie Liu, Amin Jourabloo, and Xiaoming Liu, "Learning deep models for face anti-spoofing: Binary or auxiliary supervision," in *CVPR*, 2018, pp. 389–398. 1, 2, 3, 4

[7] Zezheng Wang, Chenxu Zhao, Yunxiao Qin, Qiusheng Zhou, and Zhen Lei, "Exploiting temporal and depth information for multi-frame face anti-spoofing," *arXiv preprint arXiv:1811.05118*, 2018. 1

[8] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann, "Learning to prune filters in convolutional neural networks," in *WACV*. IEEE, 2018, pp. 709–718. 1

[9] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017. 1

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015. 1

[11] Hanxiao Liu, Karen Simonyan, and Yiming Yang, "Darts: Differentiable architecture search," *ICLR*, 2019. 1, 2, 3

[12] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong, "Pc-darts: Partial channel connections for memory-efficient differentiable architecture search," *arXiv preprint arXiv:1907.05737*, 2019. 1, 2, 3

[13] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le, "Regularized evolution for image classifier architecture search," in *AAAI*, 2019, vol. 33, pp. 4780–4789. 1

[14] Barret Zoph and Quoc V Le, "Neural architecture search with reinforcement learning," *ICLR*, 2017. 1

[15] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018, pp. 8697–8710. 1

[16] Anjith George and Sébastien Marcel, "Deep pixel-wise binary supervision for face presentation attack detection," in *IJCB*, 2019, number CONF. 2, 3, 4

[17] Zinelabinde Boulkenafet, Jukka Komulainen, Lei Li, Xiaoyi Feng, and Abdenour Hadid, "Oulu-npu: A mobile face presentation attack database with real-world variations," in *FGR*, 2017, pp. 612–618. 2, 3

[18] Zhiwei Zhang, Junjie Yan, Sifei Liu, Zhen Lei, Dong Yi, and Stan Z Li, "A face antispoofing database with diverse attacks," in *IJCB*, 2012, pp. 26–31. 2, 3

[19] Ivana Chingovska, André Anjos, and Sébastien Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," in *Biometrics Special Interest Group*, 2012, pp. 1–7. 2, 3

[20] Zinelabdine Boulkenafet, Jukka Komulainen, Zahid Akhtar, Azeddine Benlamoudi, Djamel Samai, Salah Eddine Bekhouche, Abdelkrim Ouafi, Fadi Dornaika, Abdelmalik Taleb-Ahmed, Le Qin, et al., "A competition on generalized software-based face presentation attack detection in mobile scenarios," in *IJCB*. IEEE, 2017, pp. 688–696. 3, 4

[21] Samarth Bharadwaj, Tejas I Dhamecha, Mayank Vatsa, and Richa Singh, "Computationally efficient face spoofing detection with motion magnification," in *CVPRW*, 2013, pp. 105–110. 4

[22] Allan Pinto, Helio Pedrini, William Robson Schwartz, and Anderson Rocha, "Face spoofing detection through visual codebooks of spectral temporal cubes," *IEEE T-IP*, vol. 24, no. 12, pp. 4726–4740, 2015. 4

[23] Amin Jourabloo, Yaojie Liu, and Xiaoming Liu, "Face de-spoofing: Anti-spoofing via noise modeling," in *ECCV*, 2018, pp. 290–306. 4