2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Deep Tree Learning for Zero-shot Face Anti-Spoofing

Yaojie Liu, Joel Stehouwer, Amin Jourabloo, Xiaoming Liu
Department of Computer Science and Engineering
Michigan State University, East Lansing MI 48824
{liuyaoj1, stay.jb, jourablo, liuxm}@msu.edu

Abstract

Face anti-spoofing is designed to prevent face recognition systems from recognizing fake faces as the genuine users. While advanced face anti-spoofing methods are developed, new types of spoof attacks are also being created and becoming a threat to all existing systems. We define the detection of unknown spoof attacks as Zero-Shot Face Anti-spoofing (ZSFA). Previous ZSFA works only study 1*-*2 *types of spoof attacks, such as print/replay, which limits the insight of this problem. In this work, we investigate the ZSFA problem in a wide range of* 13 *types of spoof attacks, including print, replay, 3D mask, and so on. A novel Deep Tree Network (DTN) is proposed to partition the spoof samples into semantic sub-groups in an unsupervised fashion. When a data sample arrives, being know or unknown attacks, DTN routes it to the most similar spoof cluster, and makes the binary decision. In addition, to enable the study of ZSFA, we introduce the first face anti-spoofing database that contains diverse types of spoof attacks. Experiments show that our proposed method achieves the state of the art on multiple testing protocols of ZSFA.*

1. Introduction

Face is one of the most popular biometric modalities due to its convenience of usage, e.g., access control, phone unlock. Despite the high recognition accuracy, face recognition systems are not able to distinguish between real human faces and fake ones, e.g., photograph, screen. Thus, they are vulnerable to face spoof attacks, which deceives the systems to recognize as another person. To safely use face recognition, face anti-spoofing techniques are required to detect spoof attacks before performing recognition.

Attackers can utilize a wide variety of mediums to launch spoof attacks. The most common ones are replaying videos/images on digital screens, i.e., replay attack, and printed photograph, i.e., print attack. Different methods are proposed to handle replay and print attacks, based on either handcrafted features [7, 35, 38] or CNN-based features [4, 18, 20, 32]. Recently, high-quality 3D custom mask is also used for attacking, i.e., 3D mask attack. In [29–31], methods for detecting print/replay attacks are found to be less effective for this new spoof, and hence the authors leverage the remote photoplethysmography (r-PPG) to detect the heart rate pulse as the spoofing cue. Further, facial makeup may also influence the outcome of recognition, i.e., makeup attack [12]. Many works [11–13] study facial makeup, despite not as an anti-spoofing problem.

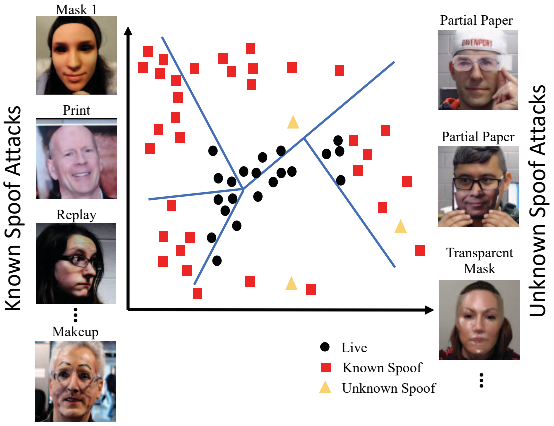


Figure 1: To detect unknown spoof attacks, we propose a Deep Tree Network (DTN) to unsupervisely learn a hierarchic embedding for known spoof attacks. Samples of unknown attacks will be routed through DTN and classified at the destined leaf node.

All aforementioned methods present algorithmic solutions to the *known* spoof attack(s), where models are trained and tested on the *same* type(s) of spoof attacks. However, in real-world applications, attackers can also initiate spoof attacks that we, the algorithm designers, are not aware of, termed *unknown* spoof attacks[1]. Researchers increasingly pay attention to the generalization of anti-spoofing models, i.e., how well they are able to detect spoof attacks that have never been seen during the training? We define the prob-

[1] There is subtle distinction between 1) *unseen attacks*, attack types that are known to algorithm designers so that algorithms could be tailored to them, but their data are unseen during training; 2) *unknown attacks*, attack types that are neither known to designers nor seen during training. We do not differentiate these two cases and term both unknown attacks.

978-1-7281-3293-8/19/$31.00 ©2019 IEEE
DOI 10.1109/CVPR.2019.00481

lem of detecting unknown face spoof attacks as **Zero-Shot Face Anti-spoofing (ZSFA)**. Despite the success of face anti-spoofing on known attacks, ZSFA, on the other hand, is a new and unsolved challenge to the community.

The first attempts on ZSFA are [3, 45]. They address ZSFA between print and replay attacks, and regard it as an outlier detection problem for live faces (a.k.a. real human faces). With handcrafted features, the live faces are modeled via standard generative models, e.g., GMM, auto-encoder. During testing, an unknown attack is detected if it lies outside the estimated live distribution. These ZSFA works have three drawbacks:

Lacking spoof type variety: Prior models are developed w.r.t. print and replay attacks only. The respective feature design may not be applicable to different unknown attacks.

No spoof knowledge: Prior models only use live faces, without leveraging the available known spoof data. While the unknown attacks are different, the known spoof attacks may still provide valuable information to learn the model.

Limitation of feature selection: They use handcrafted features such as LBP to represent live faces, which were shown to be less effective for known spoof detection [27, 32, 37, 48]. Recent deep learning models [20, 32] show the advantage of CNN models for face anti-spoofing.

This work aims to address all three drawbacks. Since one ZSFA model may perform differently when the unknown spoof attack is different, it should be evaluated on a wide range of unknown attacks types. In this work, we substantially expand the study of ZSFA from 2 types of spoof attacks to 13 types. Besides print and replay attacks, we include 5 types of 3D mask attacks, 3 types of makeup attacks, and 3 partial attacks. These attacks cover both impersonation spoofing, i.e., attempt to be authenticated as someone else, and obfuscation spoofing, i.e., attempt to cover attacker's own identity. We collect the first face anti-spoofing database that includes these diverse spoof attacks, termed Spoof in the Wild database with Multiple Attack Types (SiW-M).

To tackle the broader ZSFA, we propose a Deep Tree Network (DTN). Assuming there are both homogeneous features among different spoof types and distinct features within each spoof type, a tree-like model is well-suited to handle this case: learning the homogeneous features in the early tree nodes and distinct features in later tree nodes. Without any auxiliary labels of spoof types, DTN learns to partition data in an unsupervised manner. At each tree node, the partition is performed along the direction of the largest data variation. In the end, it clusters the data into several sub-groups at the leaf level, and learns to detect spoof attacks for each sub-group independently, shown in Fig. 1. During the testing, a data sample is routed to the most similar leaf node to produce a binary decision of live vs. spoof.

In summary, our contributions in this work include :

- Conduct an extensive study of zero-shot face anti-spoofing on 13 different types of spoof attacks;
- Propose a Deep Tree Network (DTN) to learn features hierarchically and detect unknown spoof attacks;
- Collect a new database for ZSFA and achieve the state-of-the-art performance on multiple testing protocols.

2. Prior Work

Face Anti-spoofing Image-based face anti-spoofing refers to face anti-spoofing techniques that only take RGB images as input without extra information such as depth or heat. In early years, researchers utilize liveness cues, such as eye blinking and head motion, to detect print attacks [24, 36, 37, 39]. However, when encountering unknown attacks, such as photograh with eye portion cut, and video replay, those methods suffer from a total failure. Later, research move to a more general texture analysis and address print and replay attacks. Researchers mainly utilize handcrafted features, e.g., LBP [7, 16, 17, 35], HoG [25, 47], SIFT [38] and SURF [8], with traditional classifiers, e.g., SVM and LDA, to make a binary decision. Those methods perform well on the testing data from the same database. However, while changing the testing conditions such as lighting and background, they often have a large performance drop, which can be viewed as an overfitting issue. Moreover, they also show limitations in handling 3D mask attacks, mentioned in [30].

To overcome the overfitting issue, researchers make various attempts. Boulkenafet et al. extract the spoofing features in HSV+YCbCR space [7]. Works in [2, 5, 6, 18, 46] consider features in the temporal domain. Recent works [2, 4] augment the data by using image patches, and fuse the scores from patches to a single decision. For 3D mask attacks, the heart pulse rate is estimated to differentiate 3D mask from real faces [28, 30]. In the deep learning era, researchers propose several CNN works [4, 18, 20, 27, 32, 37, 48] that outperform the traditional methods.

Zero-shot learning and unknown spoof attacks Zero-shot object recognition, or more generally, zero-shot learning, aims to recognize objects from unknown classes [40], i.e., object classes unseen in training. The overall idea is to associate the known and unknown classes via a semantic embedding, whose embedding spaces can be attributes [26], word vector [19], text description [49] and human gaze [22].

Zero-shot learning for unknown spoof attack, i.e., ZSFA, is a relatively new topic with unique properties. Firstly, unlike zero-shot object recognition, ZSFA emphasizes the detection of spoof attacks, instead of recognizing specific spoof types. Secondly, unlike generic objects with rich semantic embedding, there is no explicit well-defined semantic embedding for spoof patterns [20]. As elaborated in Sec. 1, prior ZSFA works [3, 45] only model the live data via handcrafted features and standard generative models, with

Table 1: Comparing our SiW-M with existing face anti-spoofing datasets.

Dataset	Year	Num. of subj./vid.	Face variations			Spoof attack types					Total num. of spoof types
			pose	expression	lighting	replay	print	3D mask	makeup	partial	
CASIA-FASD [50]	2012	50/600	Frontal	No	No	1	2	0	0	0	3
Replay-Attack [15]	2012	$50/1,200$	Frontal	No	Yes	1	1	0	0	0	2
HKBU-MARs [30]	2016	$35/1,008$	Frontal	No	Yes	0	0	2	0	0	2
Oulu-NPU [9]	2017	$55/5,940$	Frontal	No	No	1	1	0	0	0	2
SiW [32]	2018	$165/4,620$	$[-90°, 90°]$	Yes	Yes	1	1	0	0	0	2
SiW-M	2019	$493/1,630$	$[-90°, 90°]$	Yes	Yes	1	1	5	3	3	13

several drawbacks. In this work, we propose a deep tree network to unsupervisely learn the semantic embedding for known spoof attacks. The partition of the data naturally associates certain semantic attributes with the sub-groups. During the testing, the unknown attacks are projected to the embedding to find the closest attributes for spoof detection.

Deep tree networks Tree structure is often found helpful in tackling language-related tasks such as parsing and translation [14], due to the intrinsic relation of words and sentences. E.g., tree models are applied to joint vision and language problems such as visual question reasoning [10]. Tree structure also has the property for learning features hierarchically. Face alignment works [23, 41] utilize the regression trees to estimate facial landmarks from coarse to fine. Xiong et al. propose a tree CNN to handle the large-pose face recognition [44]. In [21], Kaneko et al. propose a GAN with decision trees to learn hierarchically interpretable representations. In our work, we utilize tree networks to learn the latent semantic embedding for ZSFA.

Face anti-spoofing databases Given the significance of a good-quality database, researchers have released several face anti-spoofing databases, such as CASIA-FASD [50], Replay-Attack [15], OULU-NPU [9], and SiW [32] for print/replay attacks, and HKBU-MARs [30] for 3D mask attacks. Early databases such as CASIA-FASD and Replay-Attack [50] have limited subject variety, pose/expression/lighting variations, and video resolutions. Recent databases [9, 30, 32] improve those aspects, and also set up diverse evaluation protocols. However, up to now, all databases focus on either print/replay attacks, or 3D mask attacks. To provide a comprehensive study of face anti-spoofing, especially the challenging ZSFA, we for the first time collect the database with diverse types of spoof attacks, as in Tab. 1. The details of our database are in Sec. 4.

3. Deep Tree Network for ZSFA

The main purposes of DTN are twofold: 1) discover the semantic sub-groups for known spoofs; 2) learn the features in a hierarchical way. The architecture of DTN is shown in Fig. 2. Each tree node consists of a Convolutional Residual Unit (CRU) and a Tree Routing Unit (TRU), while the leaf node consists of a CRU and a Supervised Feature Learning (SFL) module. CRU is a block with convolutional layers and the short-cut connection. TRU defines a node routing function to route a data sample to one of the child nodes. The routing function partitions all visiting data along the direction with the largest data variation. SFL module concatenates the classification supervision and the pixel-wise supervision to learn the spoofing features.

3.1. Unsupervised Tree Learning

3.1.1 Node Routing Function

For a TRU node, let's assume the input $\boldsymbol{x} = f(\mathbf{I}\,|\,\theta) \in \mathbb{R}^m$ is the vectorized feature response, $\mathbf{I}$ is data input, θ is the parameters of the previous CRUs, and $\mathcal{S}$ is the set of data samples $\mathbf{I}_k, k = 1, 2, ..., K$ that visit this TRU node. In [44], Xiong et al. define a routing function as:

$$\varphi(\boldsymbol{x}) = \boldsymbol{x}^T \cdot \boldsymbol{v} + \tau, \tag{1}$$

where $\boldsymbol{v}$ denotes the projection vector and τ is the bias. Data $\mathcal{S}$ can then be split into $\mathcal{S}_{left} : \{\mathbf{I}_k | \varphi(\boldsymbol{x}_k) < 0, \mathbf{I}_k \in \mathcal{S}\}$ and $\mathcal{S}_{right} : \{\mathbf{I}_k | \varphi(\boldsymbol{x}_k) \geq 0, \mathbf{I}_k \in \mathcal{S}\}$, and directed to the left and right child node, respectively. To learn this function, they propose to maximize the distance between the mean of $\mathcal{S}_{left}$ and $\mathcal{S}_{right}$, while keeping the mean of $\mathcal{S}$ centered at 0. This unsupervised loss is formulated as:

$$\mathcal{L} = \frac{(\frac{1}{N} \sum\limits_{I_k \in \mathcal{S}} \varphi(\boldsymbol{x}_k))^2}{(\frac{1}{N_l} \sum\limits_{I_k \in \mathcal{S}_{left}} \varphi(\boldsymbol{x}_k) - \frac{1}{N_r} \sum\limits_{I_k \in \mathcal{S}_{right}} \varphi(\boldsymbol{x}_k))^2}, \tag{2}$$

where N, N_l, N_r denote the number of samples in each set.

However, in practice, minizing Equ. 2 might not lead to a satisfactory solution. Firstly, the loss can be minimized by increasing the norm of either $\boldsymbol{v}$ or $\boldsymbol{x}$, which is a trivial solution. Secondly, even when the norms of $\boldsymbol{v}, \boldsymbol{x}$ are constrained, Equ. 2 is affected by the density of data $\mathcal{S}$ and can be sensitive to the outliers. In other words, the zero expectation of $\varphi(\boldsymbol{x})$ does not necessarily result in a balanced partition of data $\mathcal{S}$. Local minima could be achieved when all data are split to one side. In some cases, the tree may suffer from collapsing to a few (even one) leaf nodes.

To better partition the data, we propose a novel routing function and an unsupervised loss. Regardless of τ, the dot product between $\boldsymbol{x}^T$ and $\boldsymbol{v}$ can be regarded as projecting $\boldsymbol{x}$ to the direction of $\boldsymbol{v}$. We design $\boldsymbol{v}$ such that we can observe the largest variation after projection. Inspired by the concept of PCA, the optimal solution naturally becomes the largest PCA basis of data $\mathcal{S}$. To achieve this, we first constrain $\boldsymbol{v}$ to

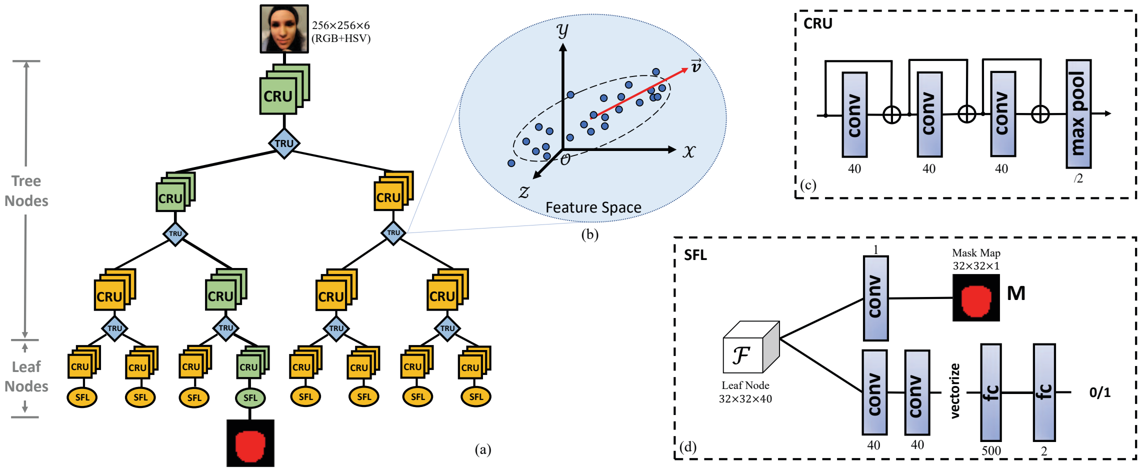


Figure 2: The proposed Deep Tree Network (DTN) architecture. (a) the overall structure of DTN. A tree node consists of a Convolutional Residual Unit (CRU) and a Tree Routing Unit (TRU), and a leaf node consists of a CRU and a Supervised Feature Learning (SFL) module. (b) the concept of Tree Routing Unit (TRU): finding the base with largest variations; (c) the structure of each Convolutional Residual Unit (CRU); (d) the structure of the Supervised Feature Learning (SFL) in the leaf nodes.

be norm 1 and reformulate Equ. 1 as:

$$\varphi(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \cdot \boldsymbol{v}, \quad \|\boldsymbol{v}\| = 1, \tag{3}$$

where $\boldsymbol{\mu}$ is the mean of data $\mathcal{S}$. Then, finding $\boldsymbol{v}$ is identical to finding the largest eigenvector of the covariance matrix $\bar{\boldsymbol{X}}_{\mathcal{S}}^T\bar{\boldsymbol{X}}_{\mathcal{S}}$, where $\bar{\boldsymbol{X}}_{\mathcal{S}} = \boldsymbol{X}_{\mathcal{S}} - \boldsymbol{\mu}$, and $\boldsymbol{X}_{\mathcal{S}} \in \mathbb{R}^{N\times K}$ is the data matrix. Based on the definition of eigen-analysis $\bar{\boldsymbol{X}}_{\mathcal{S}}^T\bar{\boldsymbol{X}}_{\mathcal{S}}\boldsymbol{v} = \lambda\boldsymbol{v}$, our optimization aims to maximize:

$$\underset{\boldsymbol{v},\theta}{\arg\max}\ \lambda = \underset{\boldsymbol{v},\theta}{\arg\max}\ \boldsymbol{v}^T\bar{\boldsymbol{X}}_{\mathcal{S}}^T\bar{\boldsymbol{X}}_{\mathcal{S}}\boldsymbol{v}. \tag{4}$$

The loss for learning the routing function is formulated as:

$$\mathcal{L}_{route} = \exp(-\alpha\boldsymbol{v}^T\bar{\boldsymbol{X}}_{\mathcal{S}}^T\bar{\boldsymbol{X}}_{\mathcal{S}}\boldsymbol{v}) + \beta\mathrm{Tr}(\bar{\boldsymbol{X}}_{\mathcal{S}}^T\bar{\boldsymbol{X}}_{\mathcal{S}}), \tag{5}$$

where α, β are scalars, and set as 1e-3, 1e-2 in our experiments. We apply the exponential function on the first term to make the maximization problem bounded. The second term is introduced as a regularizer to prevent trivial solutions by constraining the trace of covariance matrix of $\bar{\boldsymbol{X}}_{\mathcal{S}}$.

3.1.2 Tree of Known Spoofs

With the routing function, we can build the entire binary tree. Fig. 2 shows a binary tree of depth of 4, with 8 leaf nodes. As mentioned early in Sec. 3, the tree is designed to find the semantic sub-groups from all known spoofs, and is termed as spoof tree. Similarly, we may also train live tree with live faces only, as well as general data tree with both live and spoof data. Compared to spoof tree, live and general data tree have some drawbacks. Live tree does not convey semantic meaning for the spoof, and the attributes learned at each node cannot help to route and better detect spoof; General data tree may result in imbalanced sub-groups, where samples of one class outnumber another. Such imbalance would cause bias for supervised learning in the next stage.

Hence, when we compute Equ. 5 to learn the routing functions, we only consider the spoof samples to construct $\boldsymbol{X}_{\mathcal{S}}$. To have a balanced sub-group for each leaf, we suppress the responses of live data to zero, so that all live data can be evenly partitioned to the child nodes. Meanwhile, we also suppress the responses of the spoof data that do not visit this node, so that every node models the distribution of a unique spoof subset.

Formally, for each node, we maximize the routing function responses of spoof data that visit this node (denoted as $\mathcal{S}$), while minimizing the responses of other data (denoted as $\mathcal{S}^-$), including all live data and spoof data that don't visit this node, i.e., that visit neighboring nodes. To achieve this objective, we define the following loss:

$$\mathcal{L}_{uniq} = -\frac{1}{N}\sum_{\mathbf{I}_k\in\mathcal{S}}\left\|\bar{\boldsymbol{x}}_k^T\boldsymbol{v}\right\|^2 + \frac{1}{N^-}\sum_{\mathbf{I}_k\in\mathcal{S}^-}\left\|\bar{\boldsymbol{x}}_k^T\boldsymbol{v}\right\|^2. \tag{6}$$

3.2. Supervised Feature Learning

Given the routing functions, a data sample $\mathbf{I}_k$ will be assigned to one of the leaf nodes. Let's first define the feature output of leaf node as $\mathcal{F}(\mathbf{I}_k \mid \theta)$, shortened as $\mathcal{F}_k$ for simplicity. At each leaf node, we define two node-wise supervised tasks to learn discriminative features: 1) binary classification drives the learning of a high-level understanding of live vs. spoof faces, 2) pixel-wise mask regression draws CNN's attention to low-level local feature learning.

Classification supervision To learn a binary classifier, as shown in Fig. 2(d), we apply two additional convolution layers and two fully connected layers on $\mathcal{F}_k$ to generate a feature vector $\mathbf{c}_k \in \mathbb{R}^{500}$. We supervise the learning via the

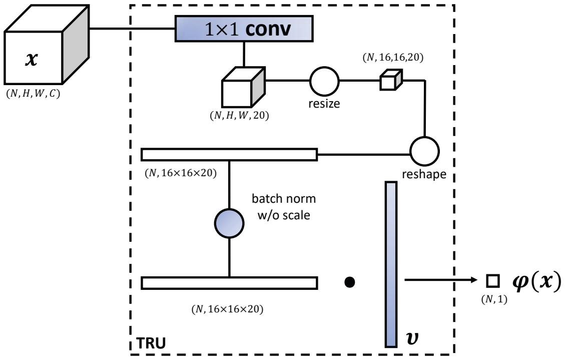


Figure 3: The structure of the Tree Routing Unit (TRU).

softmax cross entropy loss:

$$\mathcal{L}_{class} = \frac{1}{N}\sum_{I_k\in\mathcal{S}}\left\{(1-y_k)\text{log}(1-p_k) - y_k\text{log}p_k\right\} \quad (7)$$

$$p_k = \frac{\exp(\mathbf{w}_1{}^T\mathbf{c}_k)}{\exp(\mathbf{w}_0{}^T\mathbf{c}_k)+\exp(\mathbf{w}_1{}^T\mathbf{c}_k)}, \quad (8)$$

where $\mathcal{S}$ represents all the data samples that arrive this leaf node, N denotes the number of samples in $\mathcal{S}$, $\{\mathbf{w}_0, \mathbf{w}_1\}$ are the parameters in the last fully connected layer, and y_k is the label of data sample k (1 denotes spoof, and 0 live).

Pixel-wise supervision We also concatenate another convolution layer to $\mathcal{F}_k$ to generate a map response $\mathbf{M}_k \in \mathbb{R}^{32\times32}$. Inspired by the prior work [32], we leverage the semantic prior knowledge of face shapes and spoof attack position to provide a pixel-wise supervision. Using the dense face alignment model [33], we provide a binary mask $\mathbf{D}_k \in \mathbb{R}^{32\times32}$, shown in Fig. 4, to indicate the pixels of spoof mediums. Thus, for a leaf node, the loss function for the pixel-wise supervision is:

$$\mathcal{L}_{mask} = \frac{1}{N}\sum_{I_k\in\mathcal{S}} \left\|\mathbf{M}_k - \mathbf{D}_k\right\|_1. \quad (9)$$

Overall loss Finally, we apply the supervised losses on p leaf nodes, the unsupervised losses on q TRU nodes, and formulate our training loss as:

$$\mathcal{L} = \sum_{i=1}^{p}(\alpha_1\mathcal{L}^i_{class}+\alpha_2\mathcal{L}^i_{mask})+\sum_{j=1}^{q}(\alpha_3\mathcal{L}^j_{route}+\alpha_4\mathcal{L}^j_{uniq}), \quad (10)$$

where α_1,α_2,α_3,α_4 are the regularization coefficients for each term, and are set as 0.001, 1.0, 2.0, 0.001 respectively. For a 4-layer DTN, $p = 8$ and $q = 7$.

3.3. Network Architecture

Deep Tree Network (DTN) DTN is the main framework of the proposed model. It takes $\mathbf{I} \in \mathbb{R}^{256\times256\times6}$ as input, where the 6 channels are RGB+HSV color spaces. We concatenate three 3×3 convolution layers with 40 channels and 1 max-pooling layer, and group them as one Convolutional Residual Unit (CRU). Each convolution layer is equipped with ReLU and group normalization layer [43], due to the dynamic batch size in the network. We also apply a shortcut connection for each convolution layer. For each tree node, we deploy one CRU before the TRU. At the leaf node, DTN produces the feature representation of input $\mathbf{I}$ as $\mathcal{F}(\mathbf{I}\,|\,\theta) \in \mathbb{R}^{32\times32\times40}$, then uses one 1×1 convolution layer to generate the binary mask map $\mathbf{M}$.

Tree Routing Unit (TRU) TRU is the module routing the data sample to one of the child CRUs. As shown in Fig. 3, it first compresses the feature by using an 1×1 convolution layer, and resizing the response spatially. For the root node, we compress the CRU feature to $\mathbf{x} \in \mathbb{R}^{32\times32\times10}$, and for later tree node, we compress the CRU feature to $\mathbf{x} \in \mathbb{R}^{16\times16\times20}$. Compressing the input feature to a smaller size helps to reduce the burden of computating and saving the covariance matrix in Equ. 5. E.g., the vectorized feature for the first CRU is $\boldsymbol{x} \in \mathbb{R}^{655,360}$, and the covariance matrix of $\boldsymbol{x}$ can take $\sim$ 400GB in memory. However, after compression the vectorized feature is $\boldsymbol{x} \in \mathbb{R}^{10,240}$, and the covariance matrix of $\boldsymbol{x}$ only needs $\sim$ 0.1GB of memory.

After that, we vectorize the output and apply the routing function $\varphi(\boldsymbol{x})$. To compute $\boldsymbol{\mu}$ in Equ. 3, instead of optimizing it as a variable of the network, we simply apply a batch normalization layer without scaling to save the moving average of each mini-batch. In the end, we project the compressed CRU response to the largest basis $\boldsymbol{v}$ and obtain the projection coefficient. Then we assign the samples with negative coefficient to the left child CRU and the samples with positive coefficient to the right child CRU.

Implementation details With the overall loss in Equ. 10, our proposed network is trained in an end-to-end fashion. All losses are computed based on each mini-batch. DTN modules and TRU modules are optimized alternately. While optimizing DTN, we keep the parameters of TRUs fixed and vice versa.

4. Spoof in the Wild Database with Multiple Attack Types

To benchmark face anti-spoofing methods specifically for unknown attacks, we collect the Spoof in the Wild database with Multiple Attack Types (SiW-M). Compared with the previous databases in Tab. 1, SiW-M shows a great diversity in spoof attacks, subject identities, environments and other factors.

For spoof data collection, we consider two spoofing scenarios: *impersonation*, which entails the use of spoof to be recognized as someone else, and *obfuscation*, which entails the use to remove the attacker's own identity. In total, we collect 968 videos of 13 types of spoof attacks listed hieratically in Fig 4. For all 5 mask attacks, 3 partial attacks, ob-

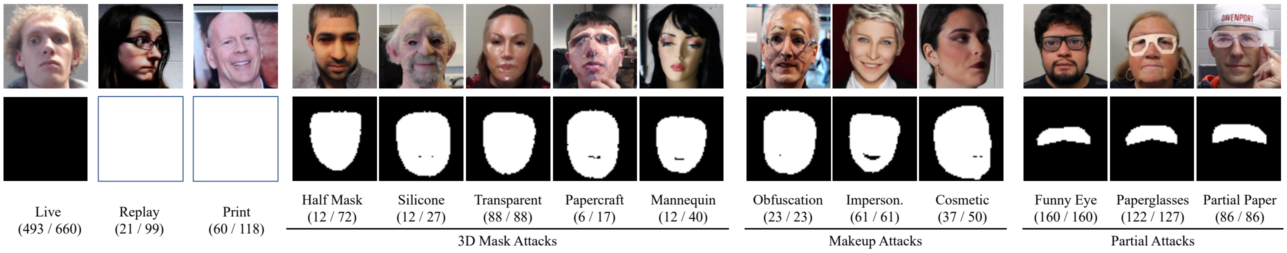


Figure 4: The examples of the live faces and 13 types of spoof attacks. The second row shows the ground truth masks for the pixel-wise supervision $\mathbf{D}_k$. For (m, n) in the third row, m/n denotes the number of subjects/videos for each type of data.

fuscation makeup and cosmetic makeup, we record 1080P HD videos. For impersonation makeup, we collect 720P videos from Youtube due to the lack of special makeup artists. For print and replay attacks, we intend to collect videos from harder cases where the existing system fails. Hence, we deploy an off-the-shelf face anti-spoofing algorithm [32] and record spoof videos when the algorithm predicts live.

For live data, we include 660 videos from 493 subjects. In comparison, the number of subjects in SiW-M is 9 times larger than Oulu-NPU [9] and CASIA-FASD [50], and 3 times larger than SiW [32]. In addition, subjects are diverse in ethnicity and age. The live videos are collected in 3 sessions: 1) a room environment where the subjects are recorded with few variations such as pose, lighting and expression (PIE). 2) a different and much larger room where the subjects are also recorded with PIE variations. 3) a mobile phone mode, where the subjects are moving while the phone camera is recording. Extreme pose angles and lighting conditions are introduced. Similar to print and replay videos, we deploy the face anti-spoofing algorithm [32] to find out the videos where the algorithm predicts spoof. Hence, this third session is a harder scenario.

In total, we collect $1,630$ videos and each lasts 5-7 seconds. The 1080P videos are recorded by Logitech C920 webcam and Canon EOS T6. To use SiW-M for the study of ZSFA, we define the leave-one-out testing protocols. Each time we train a model with 12 types of spoof attacks plus the 80% of the live videos, and test on the left 1 attack type plus the 20% of live videos. There is no overlapping subjects between the training and testing sets of live videos.

5. Experimental Results

5.1. Experimental Setup

Databases We evaluate our proposed method on multiple databases. We deploy the leave-one-out testing protocols on SiW-M and report the results of 13 experiments. Also, we test on previous face anti-spoofing databases, including CASIA [50], Replay-Attack [15], and MSU-MFSD [42]), compare with the state of the art.

Evaluation metrics We evaluate with the following metrics: Attack Presentation Classification Error Rate (APCER) [1], Bona Fide Presentation Classification Error Rate (BPCER) [1], the average of APCER and BPCER, Average Classification Error Rate (ACER) [1], Equal Error Rate (EER), and Area Under Curve (AUC). Note that, in the evaluation of unknown attacks, we assume there is no validation set to tune the model and thresholds while calculating the metrics. Hence, we determine the threshold based on the training set and fix it for all testing protocols. A single test sample is one video frame, instead of one video.

Parameter setting The proposed method is implemented in Tensorflow, and trained with a constant learning rate of 0.001 with a batch size of 32. It takes 15 epochs to converge. We randomly initialize all the weights using a normal distribution of 0 mean and 0.02 standard deviation.

5.2. Experimental Comparison

5.2.1 Ablation Study

All ablation studies use the Funny Eye protocol.

Different fusion methods In the proposed model, both the norm of the mask maps and binary spoof scores could be utilized for the final classification. To find the best fusion method, we compute ACER from using map norm, softmax score, the maximum of map norm and softmax score, and the average of two values, and obtain 31.7%, 20.5%, 21.0%, and 19.3% respectively. Since the average score of the mask norm and binary spoof score performs the best, we use it for the remaining experiments. Moreover, we set 0.2 as the final threshold to compute APCER, BPCER and ACER for all the experiments.

Different routing methods Routing is a crucial step to find the best subgroup to detect spoofness of a testing sample. To show the effect of proper routing, we evaluate 2 alternative routing strategies: random routing and pick-one-leaf. Random routing denotes randomly selecting one leaf node for a testing sample to produce prediction; Pick-one-leaf denotes constantly selecting one particular leaf node to produce results, for which we report the mean score and standard deviation of 8 selections. Shown in Tab. 3, both strategies perform worse than the proposed routing function. In addition, the large standard deviation of pick-one-leaf strategy shows the *large* performance difference of 8 subgroups on the *same type* of unknown attacks, and demonstrates the necessity of a proper routing.

Table 2: AUC (%) of the model testing on CASIA, Replay, and MSU-MFSD.

Methods	CASIA [50]			Replay-Attack [15]			MSU [42]			Overall
	Video	Cut Photo	Warped Photo	Video	Digital Photo	Printed Photo	Printed Photo	HR Video	Mobile Video	
OC-SVM$_{RBF}$+BSIF [3]	70.7	60.7	95.9	84.3	88.1	73.7	64.8	87.4	74.7	78.7 ± 11.7
SVM$_{RBF}$+LBP [9]	91.5	91.7	84.5	99.1	98.2	87.3	47.7	99.5	**97.6**	88.6 ± 16.3
NN+LBP [45]	**94.2**	88.4	79.9	99.8	95.2	78.9	50.6	99.9	93.5	86.7 ± 15.6
Ours	90.0	**97.3**	**97.5**	**99.9**	**99.9**	**99.6**	**81.6**	**99.9**	97.5	$\mathbf{95.9 \pm 6.2}$

Table 3: Compare models with different routing strategies.

Strategies	APCER	BPCER	ACER	EER
Random routing	37.1	**16.1**	26.6	24.7
Pick-one-leaf	51.2 ± 20.0	18.1 ± 4.9	34.7 ± 8.8	24.1 ± 3.1
Proposed routing function	**17.0**	21.5	**19.3**	**19.8**

Table 4: Compare models with different tree losses and strategies. The first two terms of row 2-5 refer to using live or spoof data in tree learning. The last row is our method.

Methods	APCER	BPCER	ACER	EER
MPT [44]	31.4	24.2	27.8	27.3
Live data $\surd$, Spoof data $\surd$, Unique Loss $\times$	**1.4**	73.3	37.3	31.2
Live data $\times$, Spoof data $\surd$, Unique Loss $\times$	70.0	12.7	41.3	44.8
Live data $\surd$, Spoof data $\surd$, Unique Loss $\surd$	54.2	**12.5**	33.4	36.2
Live data $\times$, Spoof data $\surd$, Unique Loss $\surd$	17.0	21.5	**19.3**	**19.8**

Advantage of each loss function We have three important designs in our unsupervised tree learning: route loss $\mathcal{L}_{route}$, data used to compute the route loss, and the unique loss $\mathcal{L}_{uniq}$. To show the effect of each loss and the training strategy, we train and compare networks with each loss excluded and alternative strategies. First, we train a network with the routing function proposed in [44], and then 4 models with different modules on and off, shown in Tab. 4. The model with MPT [44] routes data only to 2 leaf nodes out of 8 (i.e. tree collapse issue), which limits the performance. Models without the unique loss exhibit the imbalance routing issue where sub-groups cannot be trained properly . Models using all data to learn the tree show worse performances than using spoof data only. Finally, the proposed method performs the best among all options.

5.2.2 Testing on existing databases

Following the protocol proposed in [3], we use CASIA [50], Replay-Attack [15] and MSU-MFSD [42] to perform ZSFA testing between replay and print attacks. Tab. 2 compares the proposed method with top three methods selected from over 20 methods in [3, 9, 45]. Our proposed method outperforms the prior state of the art by a convincing margin of 7.3%, and our smaller standard deviation further indicates a consistently good performance among unknown attacks.

5.2.3 Testing on SiW-M

We execute 13 leave-one-out testing protocols on SiW-M. We compare with two of the most recent face anti-spoofing methods [9,32], and set [32] as the baseline, which has demonstrated its SOTA performance on various benchmarks. For a fair comparison with the baseline, we provide the same pixel-wise labeling (as in Fig. 4), and set the same threshold of 0.2 to compute APCER, BPCER, and ACER.

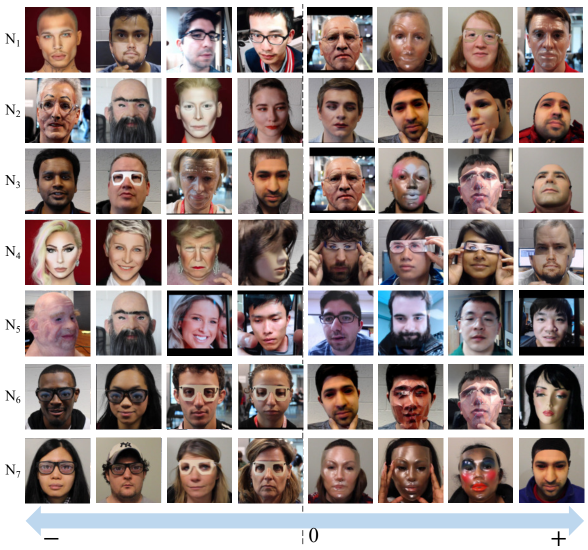


Figure 5: Visulization of the Tree Routing.

As shown in Tab. 5, our method achieves an overall better APCER, ACER and EER, with the improvement of baseline by 55%, 29%, and 5%. Specifically, we reduce the ACERs of transparent mask, funny eye, and paper glasses by 31%, 61%, and 51%, where the baseline models can be considered as total failures since they recognize most of the attacks as live. Note that, ACER is more valuable in the context of ZSFA: no evaluation data for setting threshold and considerably varied thresholds for obtaining the EER performance. For instance, EERs of paper glasses model are similar between the baseline and our method, but with a preset threshold, our method offers a much better ACER.

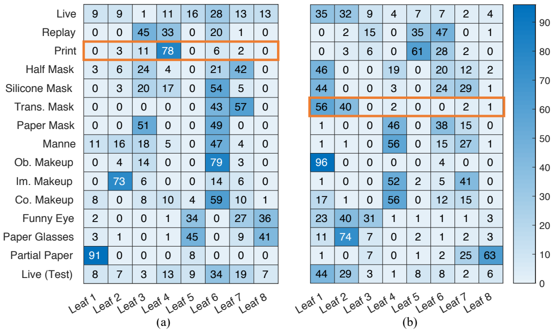
Moreover, the proposed method is a more compact model than [32]. Given the input size of $256 \times 256 \times 6$, the baseline requires 87 GFlops to compute the result while our method only needs 6 GFlops ($\times 15$ smaller). More analysis are shown with visualization in Sec. 5.2.4.

5.2.4 Visualization and Analysis

To provide a better understanding of the tree learning and ZSFA, we visualize the results in several ways. First, we illustrate the tree routing results. In Fig. 5, we rank the spoof data based on the routing function values $\varphi(\boldsymbol{x})$, and provide 8 examples with responses from the smallest to the largest. This offers us an intuitive understanding of what are learned at each tree node. We observe an obvious spoof style transfer: for the first two-layer nodes N_1, N_2 and N_3,

Table 5: The evaluation and comparison of the testing on SiW-M.

Methods	Metrics (%)	Replay	Print	Mask Attacks					Makeup Attacks			Partial Attacks			Average
				Half	Silicone	Trans.	Paper	Manne.	Obfusc.	Imperson.	Cosmetic	Funny Eye	Paper Glasses	Partial Paper	
SVM$_{RBF}$+LBP [9]	APCER	19.1	15.4	40.8	20.3	70.3	**0.0**	4.6	96.9	35.3	**11.3**	53.3	58.5	0.6	32.8 ± 29.8
	BPCER	22.1	21.5	21.9	21.4	20.7	23.1	22.9	21.7	12.5	22.2	18.4	20.0	22.9	21.0 ± 2.9
	ACER	20.6	18.4	31.3	21.4	45.5	11.6	13.8	59.3	23.9	16.7	35.9	39.2	11.7	26.9 ± 14.5
	EER	20.8	18.6	36.3	21.4	37.2	7.5	14.1	51.2	19.8	16.1	34.4	33.0	7.9	24.5 ± 12.9
Auxiliary [32]	APCER	23.7	7.3	27.7	**18.2**	97.8	8.3	16.2	100.0	18.0	16.3	91.8	72.2	0.4	38.3 ± 37.4
	BPCER	**10.1**	**6.5**	**10.9**	**11.6**	**6.2**	**7.8**	**9.3**	11.6	**9.3**	**7.1**	**6.2**	**8.8**	**10.3**	$\mathbf{8.9 \pm 2.0}$
	ACER	16.8	6.9	19.3	**14.9**	52.1	8.0	12.8	55.8	13.7	**11.7**	49.0	40.5	**5.3**	23.6 ± 18.5
	EER	14.0	4.3	**11.6**	**12.4**	**24.6**	7.8	10.0	72.3	10.1	**9.4**	21.4	**18.6**	**4.0**	17.0 ± 17.7
Ours	APCER	**1.0**	**0.0**	**0.7**	24.5	**58.6**	0.5	**3.8**	**73.2**	**13.2**	12.4	**17.0**	**17.0**	**0.2**	$\mathbf{17.1 \pm 23.3}$
	BPCER	18.6	11.9	29.3	12.8	13.4	8.5	23.0	**11.5**	9.6	16.0	21.5	22.6	16.8	16.6 ± 6.2
	ACER	**9.8**	**6.0**	**15.0**	18.7	**36.0**	**4.5**	**7.7**	**48.1**	**11.4**	14.2	**19.3**	**19.8**	8.5	$\mathbf{16.8 \pm 11.1}$
	EER	**10.0**	**2.1**	14.4	18.6	26.5	**5.7**	**9.6**	**50.2**	**10.1**	13.2	**19.8**	20.5	8.8	$\mathbf{16.1 \pm 12.2}$


(a)

	Leaf 1	Leaf 2	Leaf 3	Leaf 4	Leaf 5	Leaf 6	Leaf 7	Leaf 8
Live	9	9	1	11	16	28	13	13
Replay	0	0	45	33	0	20	1	0
Print	0	3	11	78	0	6	2	0
Half Mask	3	6	24	4	0	21	42	0
Silicone Mask	0	3	20	17	0	54	5	0
Trans. Mask	0	0	0	0	0	43	57	0
Paper Mask	0	0	51	0	0	49	0	0
Manne	11	16	18	5	0	47	4	0
Ob. Makeup	0	4	14	0	0	79	3	0
Im. Makeup	0	73	6	0	0	14	6	0
Co. Makeup	8	0	8	10	4	59	10	1
Funny Eye	2	0	0	1	34	0	27	36
Paper Glasses	3	1	0	1	45	0	9	41
Partial Paper	91	0	0	0	8	0	0	0
Live (Test)	8	7	3	13	9	34	19	7

(b)

	Leaf 1	Leaf 2	Leaf 3	Leaf 4	Leaf 5	Leaf 6	Leaf 7	Leaf 8
Live	35	32	9	4	7	7	2	4
Replay	0	2	15	0	35	47	0	1
Print	0	3	6	0	61	28	2	0
Half Mask	46	0	0	19	0	20	12	2
Silicone Mask	44	0	0	3	0	24	29	1
Trans. Mask	56	40	0	2	0	0	2	1
Paper Mask	1	0	0	46	0	38	15	0
Manne	1	1	0	56	0	15	27	1
Ob. Makeup	96	0	0	0	0	4	0	0
Im. Makeup	1	0	0	52	2	5	41	0
Co. Makeup	17	1	0	56	0	12	15	0
Funny Eye	23	40	31	1	1	1	1	3
Paper Glasses	11	74	7	0	2	1	2	3
Partial Paper	1	0	7	0	1	2	25	63
Live (Test)	44	29	3	1	8	8	2	6

Figure 6: Tree routing distribution of live/spoof data. X-axis denotes 8 leaf nodes, and y-axis denotes 15 types of data. The number in each cell represents the percentage (%) of data that fall in that leaf node. Each row is sum to 1. (a) Print Protocol. (b) Transparent Mask Protocol. Yellow box denotes the unknown attacks.

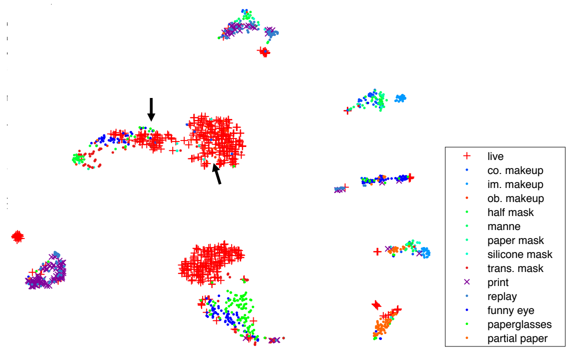


Figure 7: t-SNE Visualization of the DTN leaf features.

the transfer captures the change of general spoof attributes such as image quality and color temperature; for the third-layer tree nodes N_4, N_5, N_6, and N_7, the transfer involves more spoof type specific changes. E.g., N_7 transfers from eye portion spoofs to full face 3D mask spoofs.

Further, Fig. 6 quantitatively analyzes the tree routing distributions of all types of data. We utilize two models, Print and Trans. Mask, to generate the distributions. It can be observed that live samples are relatively more spread out to 8 leaf nodes while the spoof attacks are routed to fewer specific leaf nodes. Two distributions in Fig. 6 (a)&(b) share similar semantic sub-groups, which demonstrates the success of the proposed method on learning a tree. E.g., in both models, about half of trans. mask samples share the same leaf node as ob. makeup. By comparing two distributions, most testing unknown spoofs in both models are successfully routed to the most similar sub-groups.

In addition, we use t-SNE [34] to visualize the feature space of Print model. The t-SNE is able to project the output of the leaf node $\mathcal{F}(\mathbf{I} \mid \theta) \in \mathbb{R}^{32 \times 32 \times 40}$ to 2D by preserving the KL divergence distance. Fig. 7 shows the features of different types of spoof attacks are well-clustered into 8 semantic sub-groups even though we don't provide any auxiliary labels. Based on these sub-groups, the features of unknown print attacks are well lied in the sub-group of replay and silicone mask, and thus are recognized as spoof. Moreover, with the visualization, we can explain the performance variation among different spoof attacks, shown in Tab. 5. Among all, the performance of trans. mask, funny eye, paper glasses and ob. makeup are worse than other protocols. The feature space shows that the live samples lies much closer to those attacks than others ("→" places), and hence it's harder to distinguish them with the live samples. This demonstrates the diverse property of different unknown attacks and the necessity of such a wide range evaluation.

6. Conclusions

This paper tackles the zero-shot face antispoofing problem among 13 types of spoof attacks. The proposed method leverages a deep tree network to route the unknown attacks to the most proper leaf node for spoof detection. The tree is trained in an unsupervised fashion to find the feature base with the largest variation to split the spoof data. We collect SiW-M that contains more subjects and spoof types than any previous databases. Finally, we experimentally show superior performance of the proposed method.

Acknowledgment This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. 2017-17020200004. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

[1] ISO/IEC JTC 1/SC 37 Biometrics. information technology biometric presentation attack detection part 1: Framework. international organization for standardization, 2016. https://www.iso.org/obp/ui/iso. 6

[2] A. Agarwal, R. Singh, and M. Vatsa. Face anti-spoofing using Haralick features. In *BTAS*, 2016. 2

[3] S. R. Arashloo, J. Kittler, and W. Christmas. An anomaly detection approach to face spoofing detection: a new formulation and evaluation protocol. *IEEE Access*, 5:13868–13882, 2017. 2, 7

[4] Y. Atoum, Y. Liu, A. Jourabloo, and X. Liu. Face anti-spoofing using patch and depth-based CNNs. In *IJCB*, 2017. 1, 2

[5] W. Bao, H. Li, N. Li, and W. Jiang. A liveness detection method for face recognition based on optical flow field. In *IEEE International Conference on Image Analysis and Signal Processing (IASP)*, 2009. 2

[6] S. Bharadwaj, T. I Dhamecha, M. Vatsa, and R. Singh. Face anti-spoofing via motion magnification and multifeature videolet aggregation. Technical report, 2014. 2

[7] Z. Boulkenafet, J. Komulainen, and A. Hadid. Face anti-spoofing based on color texture analysis. In *ICIP*, 2015. 1, 2

[8] Z. Boulkenafet, J. Komulainen, and A. Hadid. Face anti-spoofing using speeded-up robust features and fisher vector encoding. *IEEE Signal Processing Letters*, 2017. 2

[9] Z. Boulkenafet, J. Komulainen, L. Li, X. Feng, and A. Hadid. OULU-NPU: A mobile face presentation attack database with real-world variations. In *FG*, 2017. 3, 6, 7, 8

[10] Q. Cao, X. Liang, B. Li, G. Li, and L. Lin. Visual question reasoning on general dependency tree. In *CVPR*, 2018. 3

[11] H. Chang, J. Lu, F. Yu, and A. Finkelstein. PairedCycleGAN: Asymmetric style transfer for applying and removing makeup. In *CVPR*, 2018. 1

[12] C. Chen, A. Dantcheva, and A. Ross. Automatic facial makeup detection with application in face recognition. In *ICB*, 2013. 1

[13] C. Chen, A. Dantcheva, and A. Ross. Impact of facial cosmetics on automatic gender and age estimation algorithms. In *IEEE International Conference on Computer Vision Theory and Applications (VISAPP)*, 2014. 1

[14] X. Chen, C. Liu, and D. Song. Tree-to-tree neural networks for program translation. *arXiv preprint arXiv:1802.03691*, 2018. 3

[15] I. Chingovska, A. Anjos, and S. Marcel. On the effectiveness of local binary patterns in face anti-spoofing. In *BIOSIG*, 2012. 3, 6, 7

[16] T. de Freitas Pereira, A. Anjos, J. M. De Martino, and S. Marcel. LBP-TOP based countermeasure against face spoofing attacks. In *ACCV*, 2012. 2

[17] T. de Freitas Pereira, A. Anjos, J. M. De Martino, and S. Marcel. Can face anti-spoofing countermeasures work in a real world scenario? In *ICB*, 2013. 2

[18] L. Feng, L. Po, Y. Li, X. Xu, F. Yuan, T. C. Cheung, and K. Cheung. Integration of image quality and motion cues for face anti-spoofing: A neural network approach. *Journal of Visual Communication and Image Representation*, 2016. 1, 2

[19] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 2

[20] A. Jourabloo, Y. Liu, and X. Liu. Face de-spoofing: Anti-spoofing via noise modeling. In *ECCV*, 2018. 1, 2

[21] T. Kaneko, K. Hiramatsu, and K. Kashino. Generative adversarial image synthesis with decision tree latent controller. In *CVPR*, 2018. 3

[22] N. Karessli, Z. Akata, B. Schiele, A. Bulling, et al. Gaze embeddings for zero-shot image classification. In *CVPR*, 2017. 2

[23] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *CVPR*, 2014. 3

[24] K. Kollreider, H. Fronthaler, M. I. Faraj, and J. Bigun. Real-time face detection and motion analysis with application in liveness assessment. In *TIFS*, 2007. 2

[25] J. Komulainen, A. Hadid, and M. Pietikainen. Context based face anti-spoofing. In *BTAS*, 2013. 2

[26] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 2

[27] L. Li, X. Feng, Z. Boulkenafet, Z. Xia, M. Li, and A. Hadid. An original face anti-spoofing approach using partial convolutional neural network. In *IEEE International Conference on Image Processing Theory Tools and Applications (IPTA)*, 2016. 2

[28] X. Li, J. Komulainen, G. Zhao, P. C. Yuen, and M. Pietikäinen. Generalized face anti-spoofing by detecting pulse from face videos. In *ICPR*, 2016. 2

[29] S. Liu, X. Lan, and P. C. Yuen. Remote photoplethysmography correspondence feature for 3D mask face presentation attack detection. In *ECCV*, 2018. 1

[30] S. Liu, B. Yang, P. C. Yuen, and Guoying Zhao. A 3D mask face anti-spoofing database with real world variations. In *CVPRW*, 2016. 1, 2, 3

[31] S. Liu, P. C. Yuen, S. Zhang, and G. Zhao. 3D mask face anti-spoofing with remote photoplethysmography. In *ECCV*, 2016. 1

[32] Y. Liu, A. Jourabloo, and X. Liu. Learning deep models for face anti-spoofing: Binary or auxiliary supervision. In *CVPR*, 2018. 1, 2, 3, 5, 6, 7, 8

[33] Y. Liu, A. Jourabloo, W. Ren, and X. Liu. Dense face alignment. In *ICCVW*, 2017. 5

[34] L. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 8

[35] J. Määttä, A. Hadid, and M. Pietikäinen. Face spoofing detection from single images using micro-texture analysis. In *IJCB*, 2011. 1, 2

[36] G. Pan, L. Sun, Z. Wu, and S. Lao. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. In *ICCV*, 2007. 2

[37] K. Patel, H. Han, and A. K. Jain. Cross-database face anti-spoofing with robust feature representation. In *CCBR*, 2016. 2

[38] K. Patel, H. Han, and A. K. Jain. Secure face unlock: Spoof detection on smartphones. In *TIFS*, 2016. 1, 2

[39] R. Shao, X. Lan, and P. C. Yuen. Deep convolutional dynamic texture learning with adaptive channel-discriminability for 3D mask face anti-spoofing. In *IJCB*, 2017. 2

[40] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013. 2

[41] R. Valle and M. José. A deeply-initialized coarse-to-fine ensemble of regression trees for face alignment. In *ECCV*, 2018. 3

[42] D. Wen, H. Han, and A. K. Jain. Face spoof detection with image distortion analysis. In *TIFS*, 2015. 6, 7

[43] Y. Wu and K. He. Group normalization. In *ECCV*, 2018. 5

[44] C. Xiong, X. Zhao, D. Tang, K. Jayashree, S. Yan, and T. Kim. Conditional convolutional neural network for modality-aware face recognition. In *ICCV*, 2015. 3, 7

[45] F. Xiong and W. Abdalmageed. Unknown presentation attack detection with face RGB images. In *BTAS*, 2018. 2, 7

[46] Z. Xu, S. Li, and W. Deng. Learning temporal features using LSTM-CNN architecture for face anti-spoofing. In *ACPR*, 2015. 2

[47] J. Yang, Z. Lei, S. Liao, and S. Z. Li. Face liveness detection with component dependent descriptor. In *ICB*, 2013. 2

[48] Z. Yang, J.and Lei and S. Z. Li. Learn convolutional neural network for face anti-spoofing. *arXiv preprint arXiv:1408.5601*, 2014. 2

[49] L. Zhang, T. Xiang, and S. Gong. Learning a deep embedding model for zero-shot learning. In *CVPR*, 2017. 2

[50] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li. A face antispoofing database with diverse attacks. In *ICB*, 2012. 3, 6, 7