Searching Central Difference Convolutional Networks for Face Anti-Spoofing (Appendix)

1. Appendix A: Derivation and Code of CDC

Here we show the detailed derivation (Eq.(4) in draft) of CDC in Eq. (1) and Pytorch code of CDC in Fig. 1.

$$
\begin{aligned}
y(p_0) &= \theta \cdot \underbrace{\sum_{p_n \in \mathcal{R}} w(p_n) \cdot (x(p_0 + p_n) - x(p_0))}_{\text{central difference convolution}} \\
&\quad + (1-\theta) \cdot \underbrace{\sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n)}_{\text{vanilla convolution}} \\
&= \theta \cdot \underbrace{\sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n)}_{\text{vanilla convolution}} + \theta \cdot (\underbrace{-\sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0)}_{\text{central difference term}}) \\
&\quad + (1-\theta) \cdot \underbrace{\sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n)}_{\text{vanilla convolution}} \\
&= (\theta + 1 - \theta) \cdot \underbrace{\sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n)}_{\text{vanilla convolution}} + \theta \cdot (\underbrace{-x(p_0) \cdot \sum_{p_n \in \mathcal{R}} w(p_n)}_{\text{central difference term}}) \\
&= \underbrace{\sum_{p_n \in \mathcal{R}} w(p_n) \cdot x(p_0 + p_n)}_{\text{vanilla convolution}} + \theta \cdot (\underbrace{-x(p_0) \cdot \sum_{p_n \in \mathcal{R}} w(p_n)}_{\text{central difference term}}) .
\end{aligned}
\tag{1}
$$

```
import torch.nn as nn
import torch.nn.functional as F
class CDC(nn.Module):
    def __init__(self, IC, OC, K=3, P=1, theta=0.7):
        # IC, OC: in_channels, out_channels
        # K, P: kernel_size, padding
        # theta: hyperparameter in CDC
        super(CDC, self).__init__()
        self.vani = nn.Conv2d(IC, OC, kernel_size=K, padding=P)
        self.theta = theta

    def forward(self, x):
        # x: input features with shape [N,C,H,W]
        out_vanilla = self.vani(x)

        kernel_diff = self.conv.weight.sum(2).sum(2)
        kernel_diff = kernel_diff[:, :, None, None]
        out_CD = F.conv2d(input=x, weight=kernel_diff, padding=0)

        return out_vanilla - self.theta * out_CD
```

Figure 1. Python code of CDC based on Pytorch.

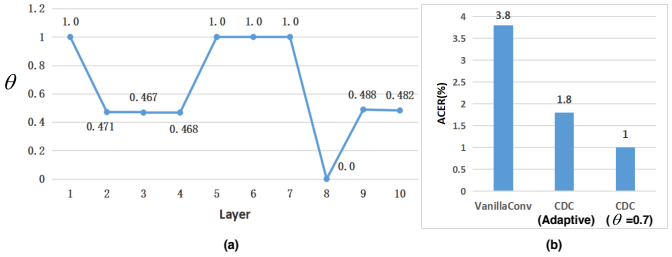

Figure 2. Adaptive CDC with learnable θ for each layer. (a) The learned θ weights for the first ten layers. (b) Performance comparison on Protocol-1 OULU-NPU.

2. Appendix B: Adaptive θ for CDC

Although the best hyperparameter $\theta = 0.7$ can be manually measured for face anti-spoofing task, it is still troublesome to find the best-suited θ when applying Central Difference Convolution (CDC) to other datasets/tasks. Here we treat θ as the data-driven learnable weights for each layer. A simple implementation is to utilize $Sigmoid(\theta)$ to guarantee the output range within $[0, 1]$.

As illustrated in Fig. 2(a), it is interesting to find that the values of learned weights in low (2nd to 4th layer) and high (8th to 10th layer) levels are relatively small while that in mid (5th to 7th layer) level are large. It indicates that the central difference gradient information might be more important for mid level features. In terms of the performance comparison, it can be seen from Fig. 2(b) that adaptive CDC achieves comparable results (1.8% vs. 1.0% ACER) with CDC using constant $\theta = 0.7$.

3. Appendix C: Cross-type Testing on SiW-M

Following the same cross-type testing protocol (13 attacks leave-one-out) on SiW-M dataset [3], we compare our proposed methods with three recent face anti-spoofing methods [1, 2, 3] to valid the generalization capacity of unseen attacks. As shown in Table 1, our CDCN++ achieves an overall better ACER and EER, with the improvement of previous state-of-the-art [3] by 24% and 26% respec-

Table 1. The evaluation and comparison of the cross-type testing on SiW-M [3].

Method	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 [1]	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 [2]	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	8.9± 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
DTN [3]	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	17.1±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	16.8 ±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	16.1± 12.2
CDCN (Ours)	APCER	8.2	6.9	8.3	7.4	20.5	5.9	5.0	43.5	1.6	14.0	24.5	18.3	1.2	12.7±11.7
	BPCER	9.3	8.5	13.9	10.9	21.0	3.1	7.0	45.0	2.3	16.2	26.4	20.9	5.4	14.6 ±11.7
	ACER	**8.7**	7.7	11.1	**9.1**	20.7	4.5	5.9	44.2	2.0	15.1	25.4	19.6	3.3	13.6 ±11.7
	EER	**8.2**	7.8	8.3	**7.4**	20.5	5.9	**5.0**	47.8	1.6	14.0	24.5	18.3	1.1	13.1± 12.6
CDCN++ (Ours)	APCER	9.2	6.0	4.2	7.4	18.2	0.0	5.0	39.1	0.0	14.0	23.3	14.3	0.0	10.8±11.2
	BPCER	12.4	8.5	14.0	13.2	19.4	7.0	6.2	45.0	1.6	14.0	24.8	20.9	3.9	14.6±11.4
	ACER	10.8	7.3	**9.1**	10.3	**18.8**	**3.5**	**5.6**	**42.1**	**0.8**	14.0	24.0	**17.6**	**1.9**	**12.7±11.2**
	EER	9.2	5.6	**4.2**	11.1	**19.3**	5.9	**5.0**	**43.5**	**0.0**	14.0	23.3	**14.3**	**0.0**	**11.9±11.8**

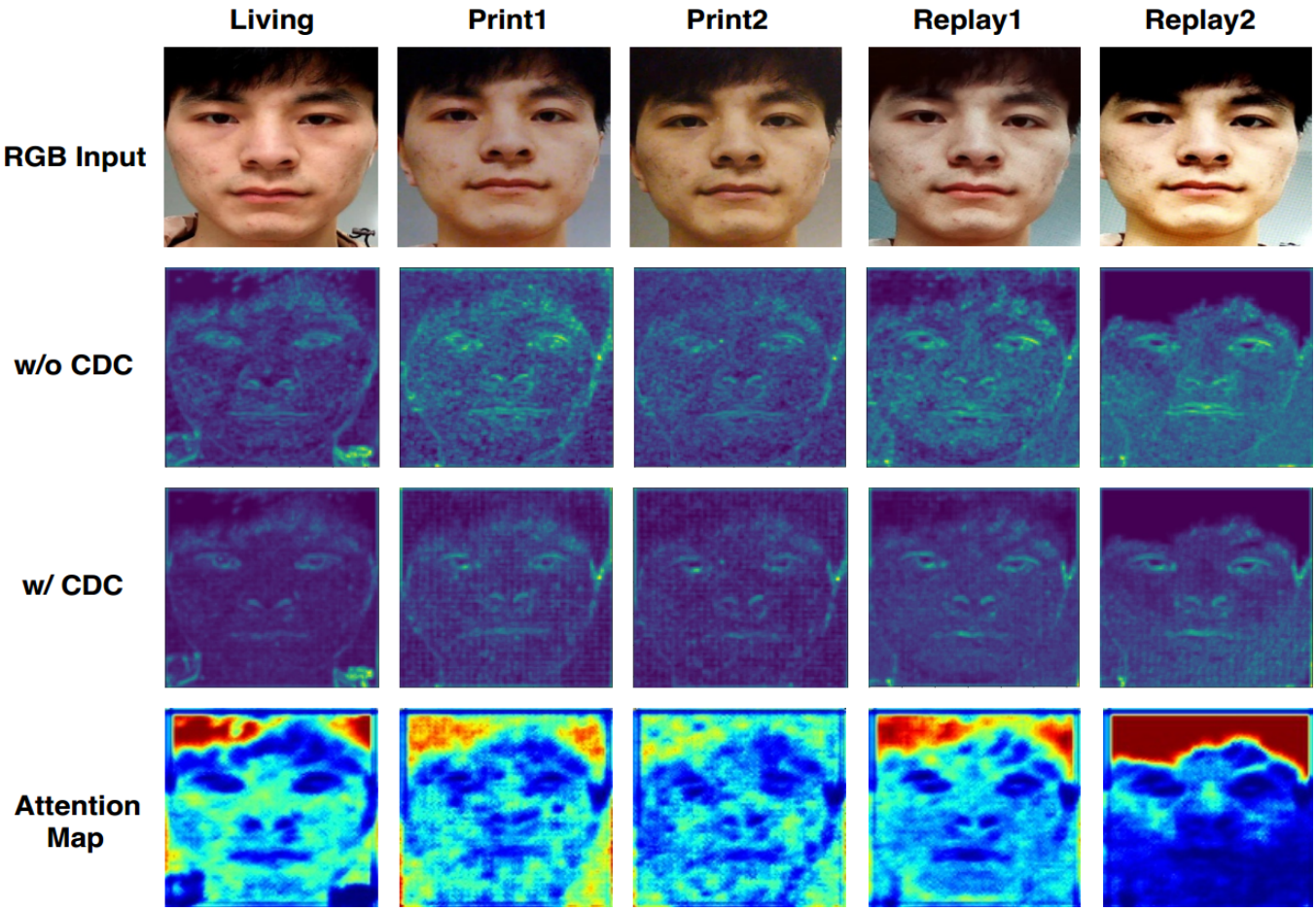

Figure 3. Features visualization on living face (the first column) and spoofing faces (four columns to the right). The four rows represent the RGB images, low-level features w/o CDC, w/ CDC and low-level spatial attention maps respectively. Best view when zoom in.

tively. Specifically, we detect almost all "Impersonation" and "Partial Paper" attacks (EER=0%) while the previous methods perform poorly on "Impersonation" attack. It is obvious that we reduce the both the EER and ACER of Mask attacks ("HalfMask", "SiliconeMask", "Transparent-Mask" and "MannequinHead") sharply, which shows our CDC based methods generalize well on 3D nonplanar attacks.

4. Appendix D: Feature Visualization

The low-level features and corresponding spatial attention maps of MAFM are visualized in Fig. 3. It is clear that both the features and attention maps between living and spoofing faces are quite different. 1) For the low-level features (see 2nd and 3rd row in Fig. 3), neural activation from the spoofing faces seems to be more homogeneous between the facial and background regions than that from living faces. It's worth noting that features with CDC are more likely to capture the detailed spoofing patterns (e.g., **lattice artifacts** in "Print1" and **reflection artifacts** in "Replay2"). 2) For the spatial attention maps of MAFM (see 4th row in Fig. 3), all the regions of hair, face and background have the relatively strong activation for the living faces while the facial regions contribute weakly for the spoofing faces.

References

[1] 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*, pages 612–618, 2017. 1, 2

[2] Yaojie Liu, Amin Jourabloo, and Xiaoming Liu. Learning deep models for face anti-spoofing: Binary or auxiliary supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 389–398, 2018. 1, 2

[3] Yaojie Liu, Joel Stehouwer, Amin Jourabloo, and Xiaoming Liu. Deep tree learning for zero-shot face anti-spoofing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4680–4689, 2019. 1, 2