Disentangled Representation with Dual-stage Feature Learning for Face Anti-spoofing

Yu-Chun Wang*, Chien-Yi Wang†, Shang-Hong Lai*†
*National Tsing Hua University, †Microsoft AI R&D Center, Taiwan
yuchun@gapp.nthu.edu.tw chiwa@microsoft.com shlai@microsoft.com

1. Experiments

1.1. Cross-dataset testing

We evaluate the generalization and robustness of our model on cross-dataset testing. We follow the state-of-the-art works that experiment on CASIA-MFSD and Replay-Attack cross-dataset protocols. The performances are measured in HTER. The results are shown in Table 2. Our cross-dataset's results are not remarkable as unknown attack testing results, but we obtain comparable HTER with other state-of-the-art works from Replay-Attack to CASIA-MFSD and CASIA-MFSD to Replay-Attack. Because we design our method for detecting unknown attacks, we rely on the Spoof-encoder and the Live-encoder to disentangle the difference between real and attack data. In our option, as a result of having different illumination, scene, and camera sensors simultaneously, cross datasets' live features may not share. It may inhibit the disentangled feature learning in our framework.

We fine-tune our model with few testing images to further observe the performance of our method. It is obvious from Table 3 that when a small number of testing images is added to the training dataset, the model accuracy improved obviously.

1.2. Visualization and Analysis

As shown in Figure 1, we visualize the feature embeddings F_L and F_S of our Live-encoder E_L and Spoof-encoder E_S simultaneously to show that these features encode different information. We adopt the leave-one-out method on the SiW-M dataset of different unknown attack types. We randomly choose 1000 real data and 1000 unseen spoof-type data from the testing set. We utilize t-SNE to convert the feature embeddings F_L and F_S in one figure. We can observe that live features and spoof features scatter apart. It means that our Spoof-encoder E_S without learning correlated features, which already learned from the Live-encoder E_L.

2. Computing Cost

2.1. Comparison with other methods

We utilize floating-point operations(FLOPs) to measure the model complexity and compare it with other disentanglement representation learning-based methods. In the inference stage, Live-info Framework and Disentanglement Framework are both abandoned. Therefore, the speed of our approach can achieve 8.23±0.1(ms) on NVIDIA GeForce GTX 1080 GPU. The comparison of computing cost of our method with other SOTA disentanglement representation learning methods is summarized in Table 1.

Method	inference FLOPs	# of parameters	inference time (GTX 1080)
DST[5]	17.1G	6M	-
DRL[8]	19.6G	14.4M	12.8 ±0.03ms
Ours	10G	16M	8.23±0.1ms

Table 1: Comparison of computing cost between our method and other SOTA methods

3. Model Architecture

3.1. The first training stage

Our Live-info Framework consists of an encoder E_L and a decoder D_L. The details of their network structures and input sizes are illustrated in Table 4. Each Conv2d layer is followed by a batch normalization layer and a Rectified Linear Unit (ReLU) activation function. We use $*$ as the symbol for the Conv2d layer which normalizes with the instance normalization layer instead of batch normalization layer. We use # as the symbol for the Conv2d layer without adopting Rectified Linear Unit (ReLU) activation function.

3.2. The second training stage

The second stage contains two parts, the Disentanglement Framework and Spoof cue module. Our Disentanglement Framework consists of two encoders, E_L and E_S, a

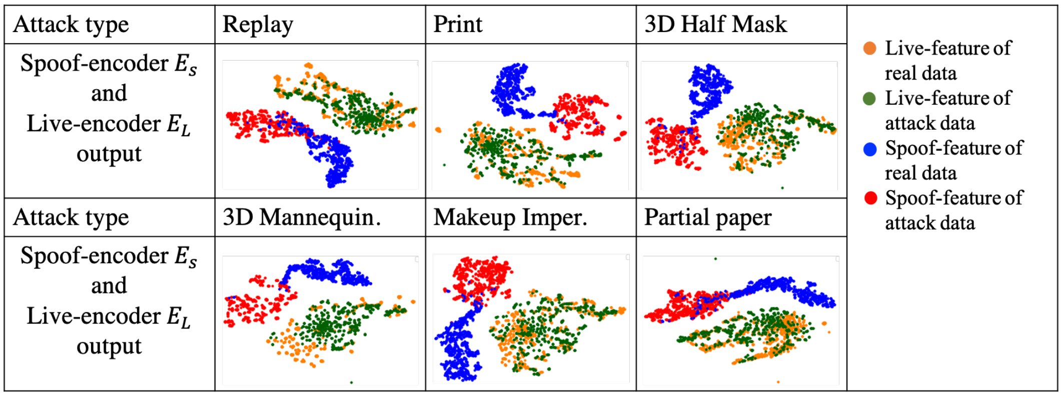

Figure 1: **Visualization of feature distributions of the SiW-M dataset by t-SNE[6].** we visualize the feature embeddings F_L and F_S of our Live-encoder E_L and Spoof-encoder E_S simultaneously.

Method	Train	Test	Train	Test
	CASIA-MFSD	Repaly-Attack	CASIA-MFSD	Replay-Attack
FaceDe-S[2]	28.5%		41.1%	
Auxiliary[4]	27.6%		**28.4**%	
STASN[7]	31.3%		30.9%	
BASN[3]	23.6%		29.9%	
DRL[8]	**22.4**%		30.3%	
Ours	22.6%		32.77%	

Table 2: The cross-dataset testing results on CASIA-MFSD[9] and Replay-Attack[1] datasets.

decoder D_{syn}, and a discriminator D . The Spoof cue module consists of a decoder D_{map} and a classifier C_{aux}. The details of their network structures and input sizes are illustrated in Table 5. Each Conv2d layer is followed by a batch normalization layer and a Rectified Linear Unit (ReLU) activation function. We use $*$ as the symbol for the Conv2d layer without adopting the batch normalization layer. We use $\&$ as the symbol for the Conv2d layer, which employs Leaky ReLU instead of normal ReLU as the activation function. Using $\#$ as the symbol for the Conv2d layer without adopting Rectified Linear Unit (ReLU) activation function.

References

[1] Ivana Chingovska, André Anjos, and Sébastien Marcel. On the effectiveness of local binary patterns in face anti-spoofing. In *2012 BIOSIG-proceedings of the international conference of biometrics special interest group (BIOSIG)*, pages 1–7. IEEE, 2012.

[2] Amin Jourabloo, Yaojie Liu, and Xiaoming Liu. Face de-spoofing: Anti-spoofing via noise modeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 290–306, 2018.

[3] Taewook Kim, YongHyun Kim, Inhan Kim, and Daijin Kim. Basn: Enriching feature representation using bipartite auxiliary supervisions for face anti-spoofing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[4] 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.

[5] Yaojie Liu, Joel Stehouwer, and Xiaoming Liu. On disentangling spoof trace for generic face anti-spoofing. In *European Conference on Computer Vision*, pages 406–422. Springer, 2020.

[6] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[7] Xiao Yang, Wenhan Luo, Linchao Bao, Yuan Gao, Dihong Gong, Shibao Zheng, Zhifeng Li, and Wei Liu. Face anti-spoofing: Model matters, so does data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3507–3516, 2019.

[8] Ke-Yue Zhang, Taiping Yao, Jian Zhang, Ying Tai, Shouhong Ding, Jilin Li, Feiyue Huang, Haichuan Song, and Lizhuang Ma. Face anti-spoofing via disentangled representation learning. In *European Conference on Computer Vision*, pages 641–657. Springer, 2020.

Method	Train	Test	Train	Test
	CASIA-MFSD	Repaly-Attack	CASIA-MFSD	Replay-Attack
w/o fine-tuning	22.6%		32.77%	
w fine-tuning (8)	16%		18.61%	
w fine-tuning (16)	15.5%		12.5%	
w fine-tuning (32)	8.75%		11.94%	
w fine-tuning (64)	**5.74%**		**10.0%**	

Table 3: The cross-dataset testing results with a small amount of data used for model fine-tuning on CASIA-MFSD[9] and Replay-Attack[1] datasets.

[9] Zhiwei Zhang, Junjie Yan, Sifei Liu, Zhen Lei, Dong Yi, and Stan Z Li. A face antispoofing database with diverse attacks. In *2012 5th IAPR international conference on Biometrics (ICB)*, pages 26–31. IEEE, 2012.

Encoder		Decoder	
E_L input : Image (3, 256, 256)		D_{syn} input : Concatenated features (1024,8,8)	
		D_L input : Live feature (512, 8, 8)	
E_S input : Image (3, 256, 256)		D_{map} input : Spoof feature (512, 8, 8)	
Layer	chan./Stri./kernel	Layer	chan./Stri./kernel
Conv2d	64, 2, 7	Conv2d*	256, 1, 3
MaxPool2d	- , 2, 3	Conv2d*#	256, 1, 3
Conv2d	64, 1, 3	Conv2d*#	256, 1, 1
Conv2d #	64, 1, 3	Conv2d	256, 1, 3
Conv2d	64, 1, 3	Conv2d #	256, 1, 3
Conv2d #	64, 1, 3	Conv2d*	128, 1, 3
Conv2d	128, 2, 3	Conv2d*#	128, 1, 3
Conv2d #	128, 1, 3	Conv2d*#	128, 1, 1
Conv2d#	128, 2, 1	Conv2d	128, 1, 3
Conv2d	128, 1, 3	Conv2d #	128, 1, 3
Conv2d #	128, 1, 3	Conv2d*	64, 1, 3
Conv2d	256,2,3	Conv2d*#	64, 1, 3
Conv2d #	256, 1, 3	Conv2d*#	64, 1, 1
Conv2d #	256, 2, 1	Conv2d	64, 1, 3
Conv2d	256,1,3	Conv2d #	64, 1, 3
Conv2d #	256, 1, 3	Conv2d*	64, 1, 3
Conv2d	512, 2, 3	Conv2d*#	64, 1, 3
Conv2d #	512, 1, 3	Conv2d*#	64, 1, 1
Conv2d #	512, 2, 1	Conv2d	64, 1, 3
Conv2d	512, 1, 3	Conv2d #	64, 1, 3
Conv2d #	512, 1, 3	Conv2d*	3, 1, 3
		Conv2d*#	3, 1, 3
		Conv2d*#	3, 1, 1
		Conv2d	3, 1, 3
		Conv2d#	3, 1, 3

Table 4: The details of the encoder and the decoder of our proposed method.

Classifier		Discriminator	
C_{aux} input : Overlapped image (3, 256, 256)		Discriminator input : Image or Reconstruction (3, 256, 256)	
Layer	chan./Stri./kernel	Layer	chan./Stri./kernel
Conv2d	64, 2, 7	Conv2d*&	256, 2, 4
MaxPool2d	- , 2, 3	Conv2d &	512, 2, 4
Conv2d &	64, 1, 3	Conv2d &	1024, 2, 4
Conv2d #	64, 1, 3	Conv2d &	2048, 2, 4
Conv2d	64, 1, 3	Conv2d #*	2048, 1, 4
Conv2d #	64, 1, 3	Linear	1, - , -
Conv2d	128, 2, 3		
Conv2d #	128, 1, 3		
Conv2d #	128, 2, 1		
Conv2d	128, 1, 3		
Conv2d #	128, 1, 3		
Conv2d	256, 2, 3		
Conv2d #	256, 1, 3		
Conv2d #	256, 2, 1		
Conv2d	256, 1, 3		
Conv2d #	256, 1, 3		
Conv2d	512, 2, 3		
Conv2d #	512, 1, 3		
Conv2d #	512, 2, 1		
Conv2d	512, 1, 3		
Conv2d #	512, 1, 3		

Table 5: The details of the classifier and the discriminator of our proposed method.