1-3. Face Recognition

ArcFace

ArcFace: Additive Angular Margin Loss for Deep Face Recognition

Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou

Abstract—Recently, a popular line of research in face recognition is adopting margins in the well-established softmax loss function to maximize class separability. In this paper, we first introduce an Additive Angular Margin Loss (ArcFace), which not only has a clear geometric interpretation but also significantly enhances the discriminative power. Since ArcFace is susceptible to the massive label noise, we further propose sub-center ArcFace, in which each class contains K sub-centers and training samples only need to be close to any of the K positive sub-centers. Sub-center ArcFace encourages one dominant sub-class that contains the majority of clean faces and non-dominant sub-classes that include hard or noisy faces. Based on this self-propelled isolation, we boost the performance through automatically purifying raw web faces under massive real-world noise. Besides discriminative feature embedding, we also explore the inverse problem, mapping feature vectors to face images. Without training any additional generator or discriminator, the pre-trained ArcFace model can generate identity-preserved face images for both subjects inside and outside the training data only by using the network gradient and Batch Normalization (BN) priors. Extensive experiments demonstrate that ArcFace can enhance the discriminative feature embedding as well as strengthen the generative face synthesis.

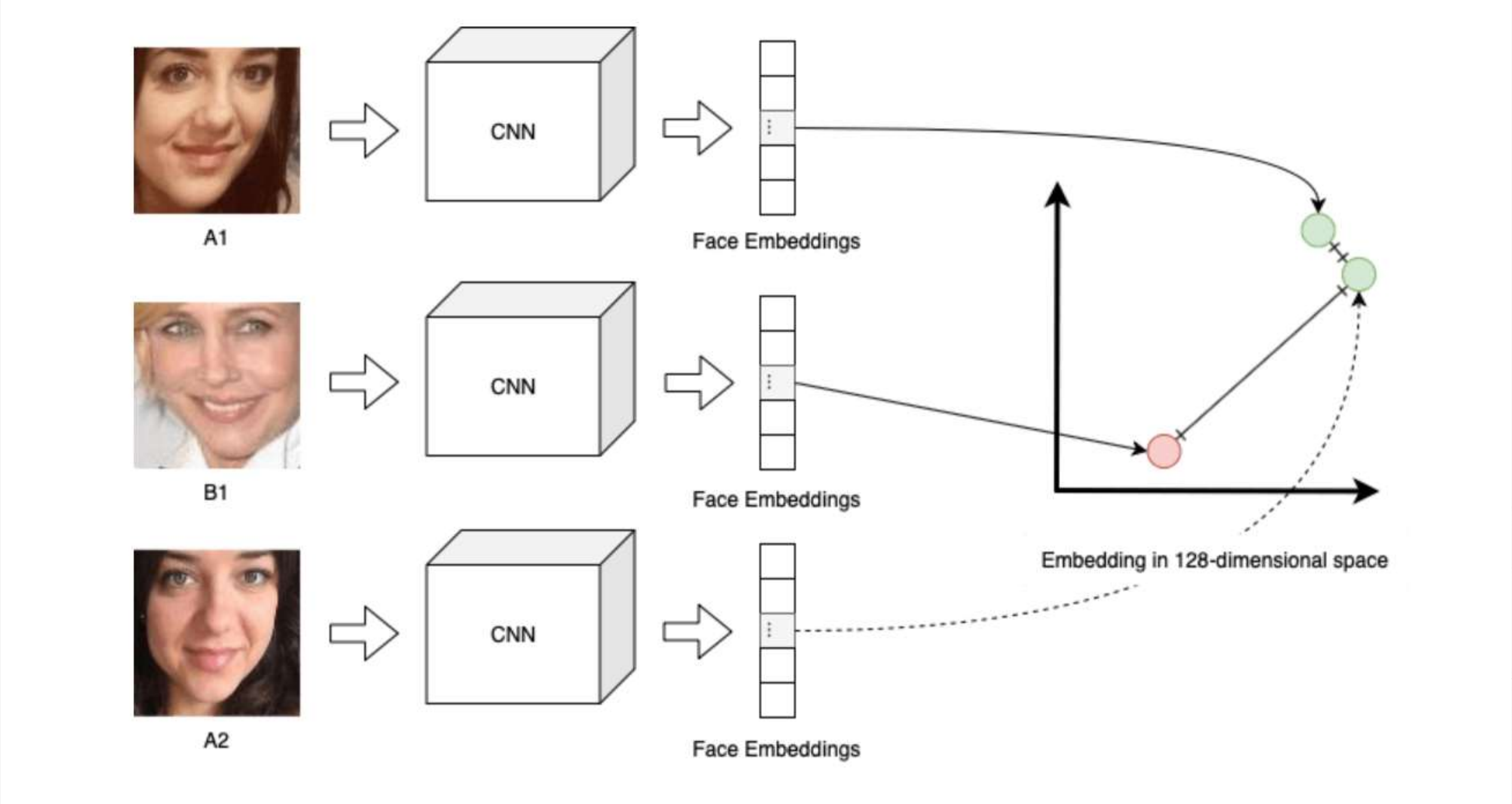
Index Terms—Large-scale Face Recognition, Additive Angular Margin, Noisy Labels, Sub-class, Model Inversion

References
https://arxiv.org/pdf/1801.07698.pdf

CONTENT

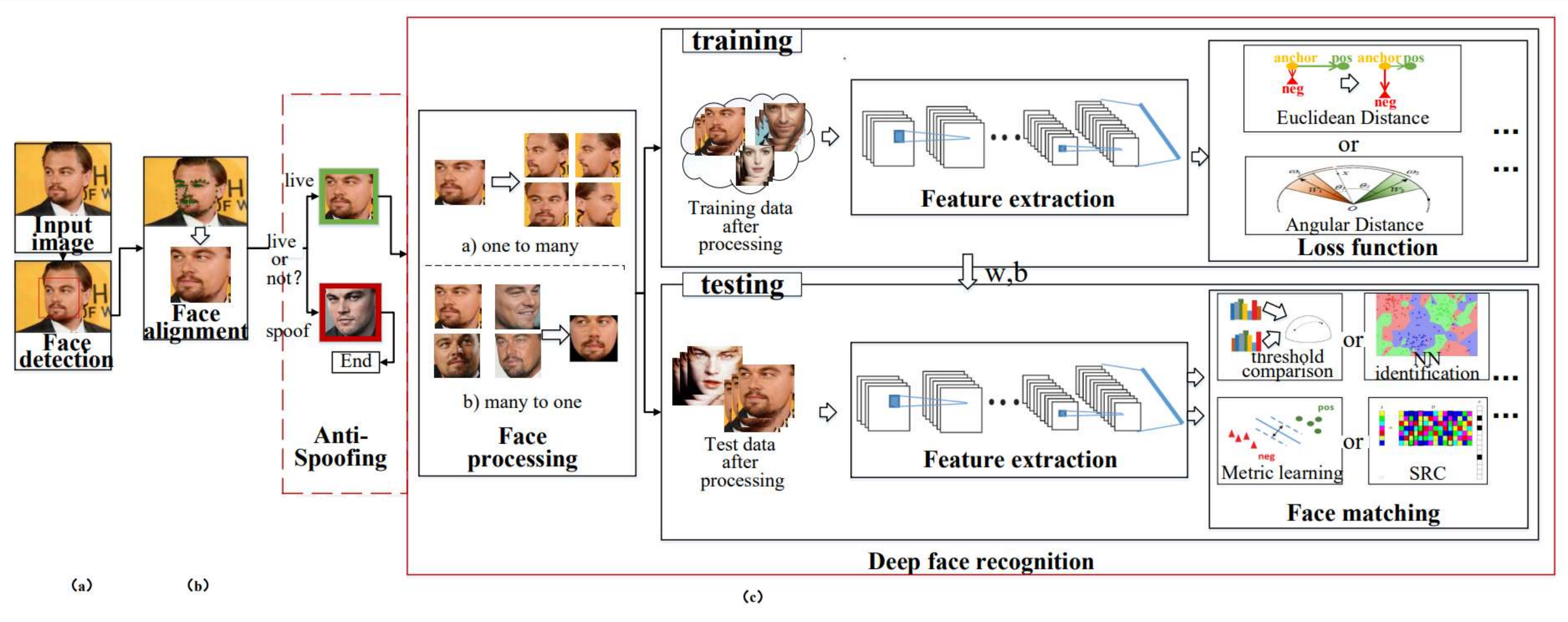
Introduction

Face Recognition



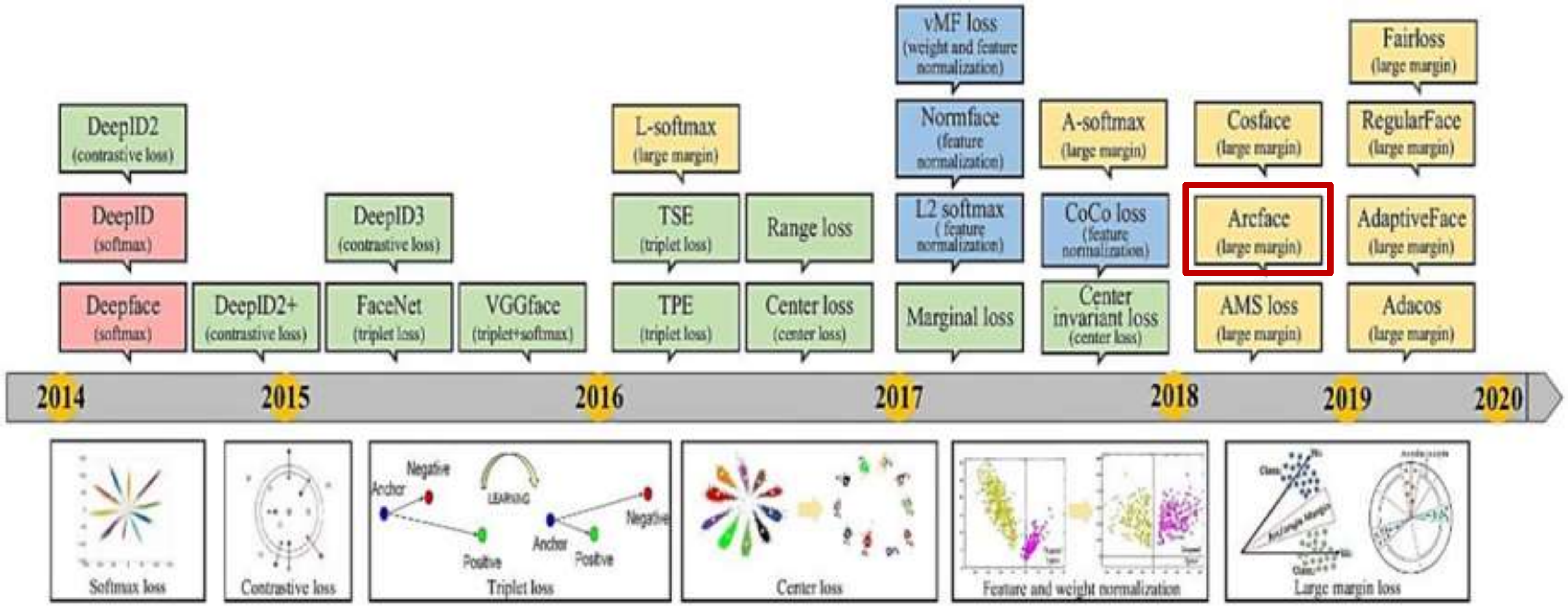
References
https://www.analyticsvidhya.com/blog/2022/04/face-recognition-system-using-python/#h-understand-the-working-of-face-recognition

Face Recognition



References
https://www.sciencedirect.com/science/article/abs/pii/S0925231220316945

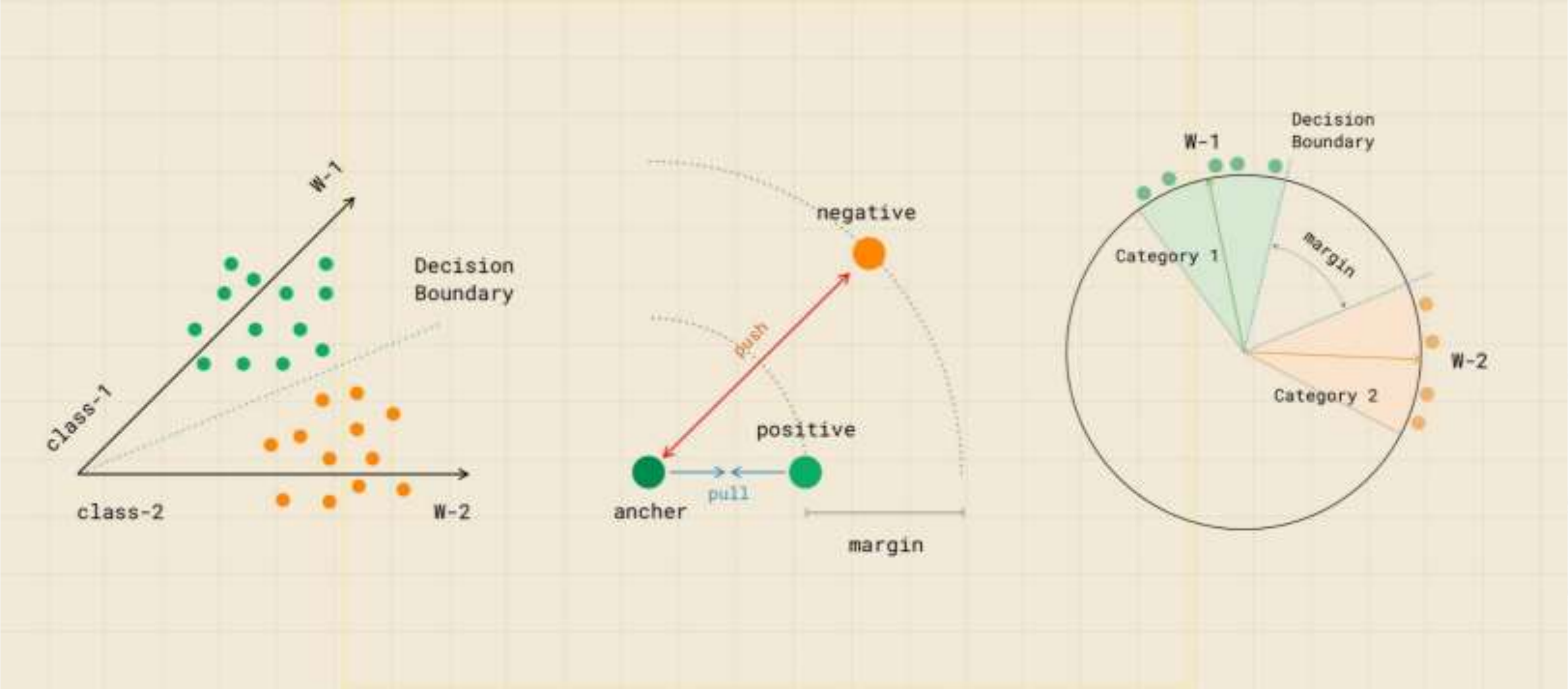
Loss Functions



References
https://www.researchgate.net/figure/History-of-loss-function-development-26_fig1_367762234

Loss Functions

- Softmax
- 거리기반
- Angular margin 기반



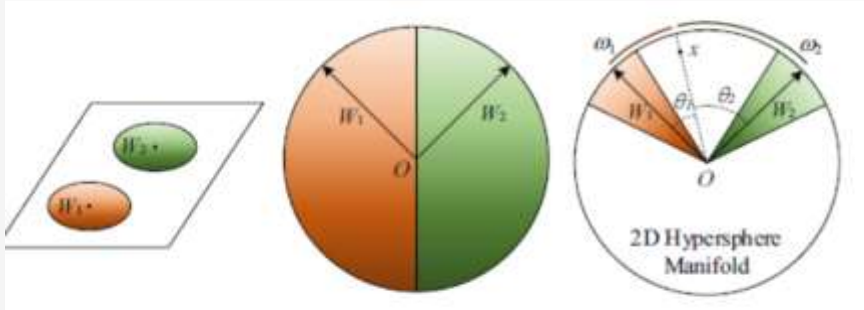
References
https://kakaoenterprise.github.io/deepdive/200723

Loss functions – Angular margin

SphereFace

- W 정규화 및 angular 공간을 이용해 feature를 구분
- Angular margin을 통해 intra-class variance를 최소화
  inter-class variance를 최대화

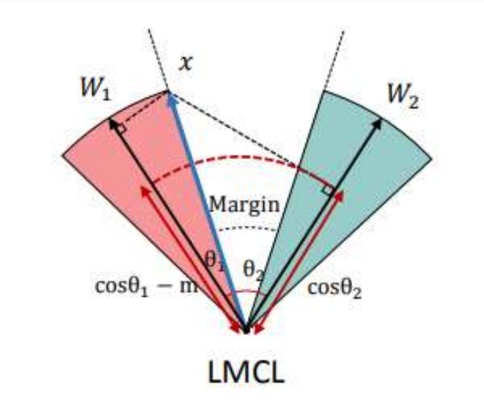
$$L_{\text{ang}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})}}{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})}+\sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right)$$



CosFace

- Feature x 정규화
- Angular margin을 cosine 값 자체에 더함

$$L_{lmc} = \frac{1}{N}\sum_i -\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)}+\sum_{j\neq y_i} e^{s\cos(\theta_{j,i})}},$$



References
https://arxiv.org/pdf/1801.09414.pdf

Proposed

Architecture

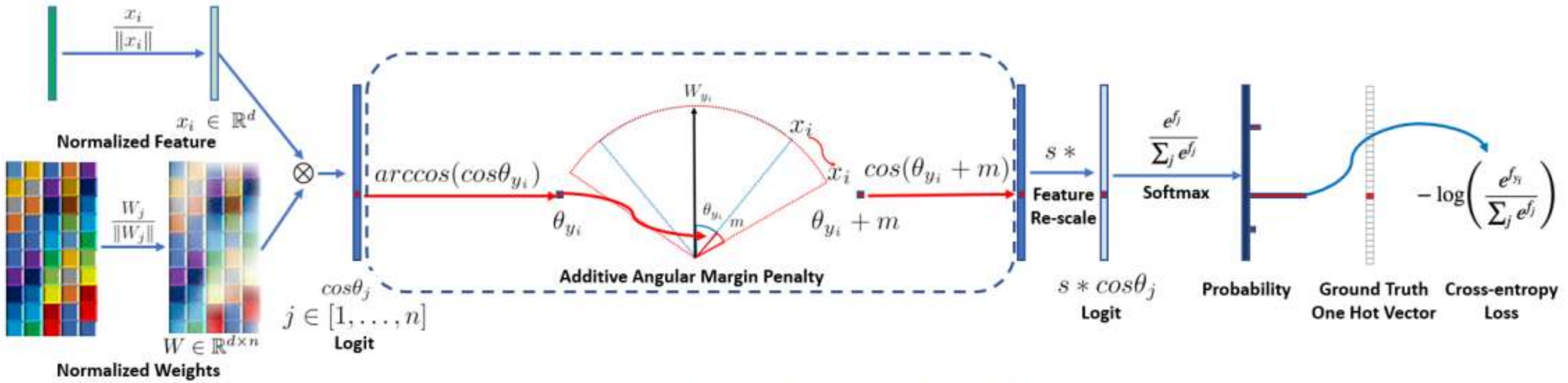


Figure 2. Training a DCNN for face recognition supervised by the ArcFace loss. Based on the feature x_i and weight W normalisation, we get the $\cos\theta_j$ (logit) for each class as $W_j^T x_i$. We calculate the $arccos\theta_{y_i}$ and get the angle between the feature x_i and the ground truth weight W_{y_i}. In fact, W_j provides a kind of centre for each class. Then, we add an angular margin penalty m on the target (ground truth) angle θ_{y_i}. After that, we calculate $\cos(\theta_{y_i} + m)$ and multiply all logits by the feature scale s. The logits then go through the softmax function and contribute to the cross entropy loss.

ArcFace

Softmax Loss

$$L_1 = -\log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{N} e^{W_j^T x_i + b_j}}$$

$b_j = 0$

$W_j^T x_i = \|W_j\| \|x_i\| cos\theta_j$

$\|W_j\| = 1$

$\|x_i\| = s$

Hypersphere Loss

$$L_2 = -\log \frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos\theta_j}}$$

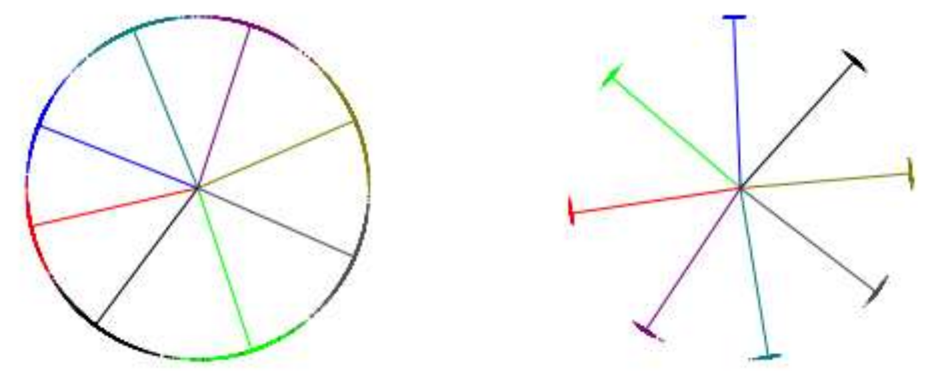
Add margin

ArcFace Loss

$$\boxed{L_3 = -\log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos\theta_j}}}$$

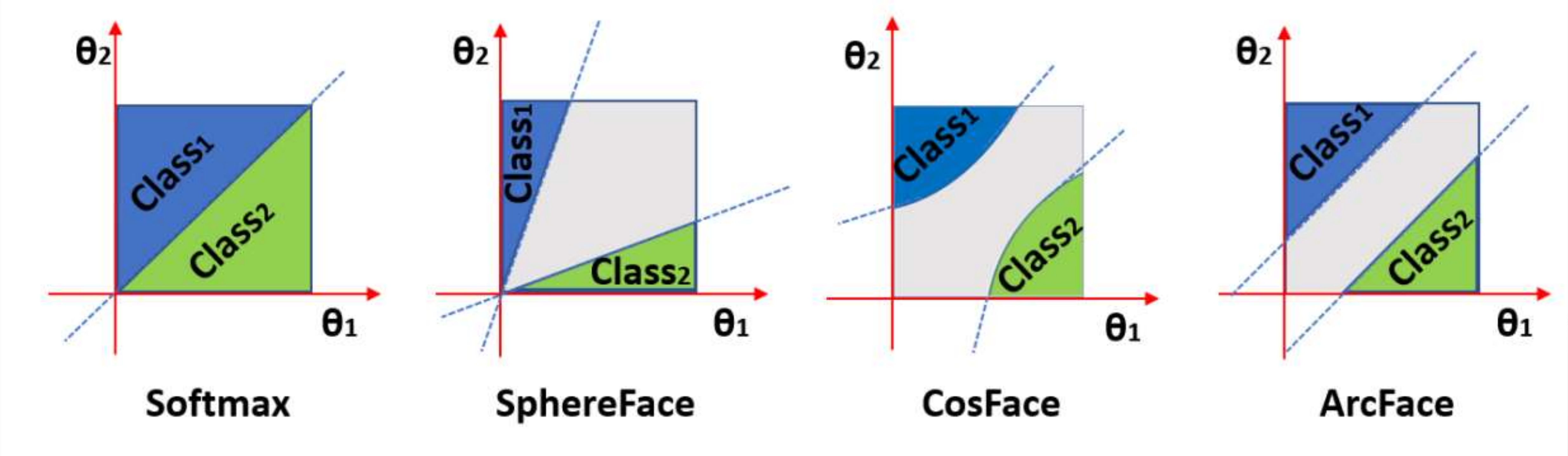
Softmax vs ArcFace

Toy Example – 1500 images, 8 classes



(a) Norm-Softmax (b) ArcFace

Decision Margins of Loss Functions



Experimental Results

Implementation Details

- Image size : 112 x 112
- Embedding channels : 512
- S (rescale-factor): 64
- m (margin) : 0.5

- Batch size : 512
- Learning rate : 0.1
- Momentum : 0.9
- Weight decay :5e-4

TABLE 1
Face datasets for training and testing. "(D)" refers to the distractors. IBUG-500K is the training data automatically refined by the proposed sub-center ArcFace. LFR2019-Image and LFR2019-Video are the proposed large-scale image and video test sets.

Datasets	#Identity	#Image/Video
CASIA [56]	10K	0.5M
VGG2 [9]	9.1K	3.3M
MS1MV0 [37]	100K	10M
MS1MV3 [88]	93K	5.1M
Celeb500K [38]	500K	50M
IBUG-500K	493K	11.96M
LFW [89]	5,749	13,233
YTF [90]	1,595	3,425
CFP-FP [74]	500	7,000
CPLFW [75]	5,749	11,652
AgeDB [76]	568	16,488
CALFW [77]	5,749	12,174
MegaFace [78]	530	1M (D)
IJB-B [79]	1,845	76.8K
IJB-C [80]	3,531	148.8K
LFR2019-Image [88]	5.7K	1.58M(D)
LFR2019-Video [88]	10K	200K

Verification Results (%)

TABLE 2
Verification results (%) of different loss functions ([CASIA, ResNet50, Loss*]).

Loss Functions	LFW	CFP-FP	AgeDB
ArcFace (0.4)	99.53	95.41	94.98
ArcFace (0.45)	99.46	95.47	94.93
ArcFace (0.5)	**99.53**	**95.56**	**95.15**
ArcFace (0.55)	99.41	95.32	95.05
SphereFace [13]	99.42	-	-
SphereFace (1.35)	99.11	94.38	91.70
CosFace [14]	99.33	-	-
CosFace (0.35)	99.51	95.44	94.56
CM1 (1, 0.3, 0.2)	99.48	95.12	94.38
CM2 (0.9, 0.4, 0.15)	99.50	95.24	94.86
Softmax	99.08	94.39	92.33
Norm-Softmax ($s = 64$)	98.56	89.79	88.72
Norm-Softmax ($s = 20$)	99.20	94.61	92.65
Norm-Softmax+Intra	99.30	94.85	93.58
Norm-Softmax+Inter	99.22	94.73	92.94
Norm-Softmax+Intra+Inter	99.31	94.88	93.76
Triplet (0.35)	98.98	91.90	89.98
ArcFace+Intra	99.45	95.37	94.73
ArcFace+Inter	99.43	95.25	94.55
ArcFace+Intra+Inter	99.43	95.42	95.10
ArcFace+Triplet	99.50	95.51	94.40

The CASIA-WebFace dataset is used for face verification and face identification tasks.
The dataset contains 494,414 face images of 10,575 real identities collected from the web.

Verification Performance (%)

TABLE 5
Verification performance (%) of different methods on CFP-FP, CPLFW, AgeDB and CALFW. ([Dataset*, ResNet100, ArcFace])

Method	CFP-FP	CPLFW	AgeDB	CALFW
Center Loss [72]	-	77.48	-	85.48
SphereFace [13]	-	81.40	-	90.30
VGGFace2 [9]	-	84.00	-	90.57
MV-Softmax [53]	98.28	92.83	97.95	**96.10**
Search-Softmax [108]	95.64	89.50	97.75	95.40
FaceGraph [109]	96.90	92.27	97.92	95.67
CurricularFace [54]	98.36	93.13	98.37	96.05
MS1MV3, R100, ArcFace	98.79	93.21	98.23	96.02
IBUG500K, R100, ArcFace	**98.87**	**93.43**	**98.38**	**96.10**

TABLE 4
Verification performance (%) of different methods on LFW and YTF. ([Dataset*, ResNet100, ArcFace])

Method	#Image	LFW	YTF
DeepID [1]	0.2M	99.47	93.20
Deep Face [2]	4.4M	97.35	91.4
VGG Face [4]	2.6M	98.95	97.30
FaceNet [3]	200M	99.63	95.10
Baidu [95]	1.3M	99.13	-
Center Loss [72]	0.7M	99.28	94.9
Range Loss [73]	5M	99.52	93.70
Marginal Loss [17]	3.8M	99.48	95.98
SphereFace [13]	0.5M	99.42	95.0
SphereFace+ [84]	0.5M	99.47	-
CosFace [14]	5M	99.73	97.6
RegularFace [51]	3.1M	99.61	96.7
UniformFace [52]	6.1M	99.8	97.7
DAL [96]	0.5M	99.47	-
FTL [97]	5M	99.55	-
Fair Loss [98]	0.5M	99.57	96.2
Unequal-training [20]	0.55M	99.53	96.04
Noise-Tolerant [19]	1M noisy	99.72	97.36
AdaptiveFace [50]	5M	99.62	-
AFRN [99]	3.1M	**99.85**	97.1
PFE [100]	4.4M	99.82	97.36
DUL [101]	3.6M	99.78	96.78
RDCFace [102]	1.7M	99.80	97.10
HPDA [103]	5M	99.80	-
URFace [104]	5M	99.78	-
CircleLoss [105]	3.6M	99.73	96.38
GroupFace [55]	5.8M	**99.85**	97.8
BioMetricNet [106]	3.8M	99.80	**98.06**
BroadFace [107]	5.8M	**99.85**	98.0
IBUG500K,R100,BroadFace	11.96M	99.83	98.03
MS1MV3, R100, ArcFace	5.1M	99.83	98.02
IBUG500K, R100, ArcFace	11.96M	99.83	98.01

Conclusion

Advantages of ArcFace

4E

Engaging

*ArcFace can not only **enhance the discriminative power** but also strengthen the generative power.*

Effective

*Using IBUG-500K as the training data, ArcFace, achieves **state-of-the-art performance** on ten face recognition benchmarks. including large-scale image and video datasets collected by us*

Easy

*ArcFace only needs several lines of code and is **extremely easy to implement** in the computational-graph-based deep learning. frameworks*

Efficient

*ArcFace only adds **negligible computational complexity** during training.*

Advantages of ArcFace

Easy

*ArcFace only needs several lines of code and is **extremely easy to implement** in the computational-graph-based deep learning frameworks.*

```python
class ArcFace(torch.nn.Module):
    """ ArcFace (https://arxiv.org/pdf/1801.07698v1.pdf):
    """
    def __init__(self, s=64.0, margin=0.5):
        super(ArcFace, self).__init__()
        self.s = s
        self.margin = margin
        self.cos_m = math.cos(margin)
        self.sin_m = math.sin(margin)
        self.theta = math.cos(math.pi - margin)
        self.sinmm = math.sin(math.pi - margin) * margin
        self.easy_margin = False


    def forward(self, logits: torch.Tensor, labels: torch.Tensor):
        index = torch.where(labels != -1)[0]
        target_logit = logits[index, labels[index].view(-1)]

        with torch.no_grad():
            target_logit.arccos_()
            logits.arccos_()
            final_target_logit = target_logit + self.margin
            logits[index, labels[index].view(-1)] = final_target_logit
            logits.cos_()
        logits = logits * self.s
        return logits
```

[Practice 3] Face Recognition

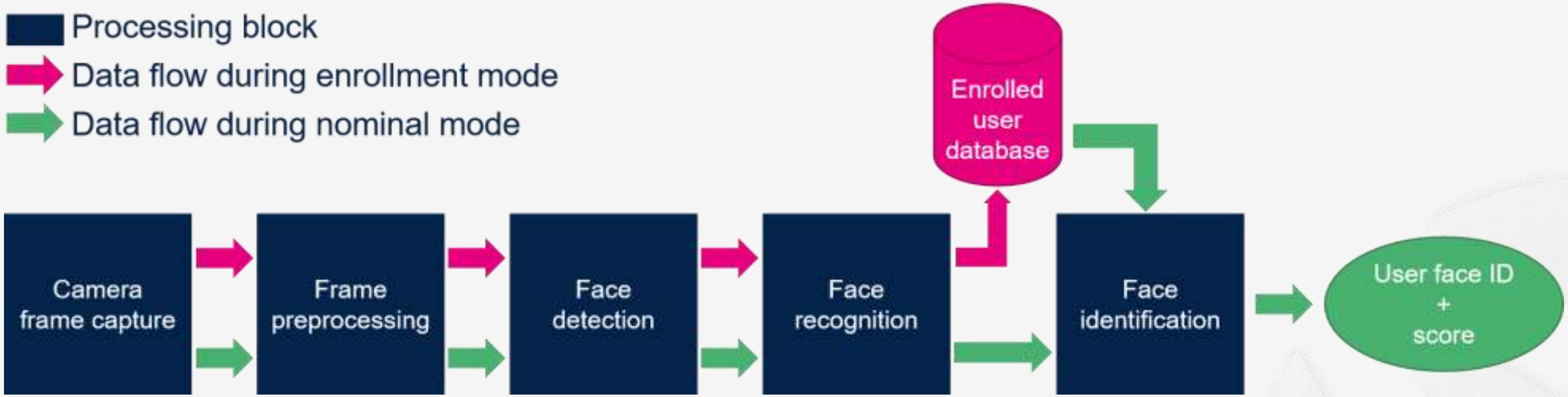
CONTENT

실습 소개

Face Recognition

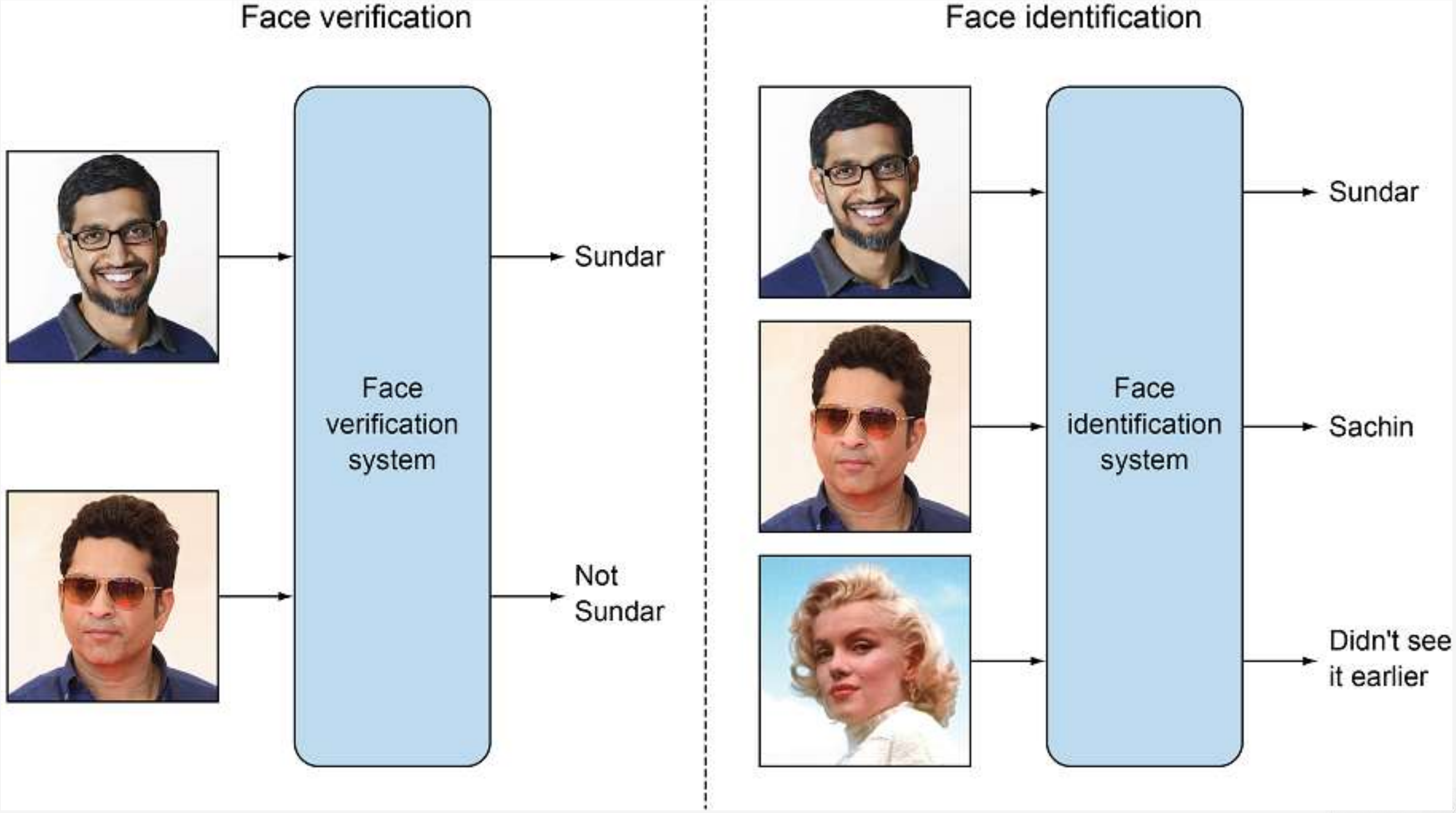
얼굴을 포함하는 입력 정지 영상 또는 비디오에 대해 얼굴 영역의 자동적인 검출 및 분석을 통해 해당 얼굴이 어떤 인물인지 판별해 내는 기술

- 얼굴 검증 (Face Verification)
- **얼굴 식별 (Face Identification)**



References
https://wikidocs.net/151311
https://wiki.st.com/stm32mpu/wiki/TFLite_Cpp_face_recognition

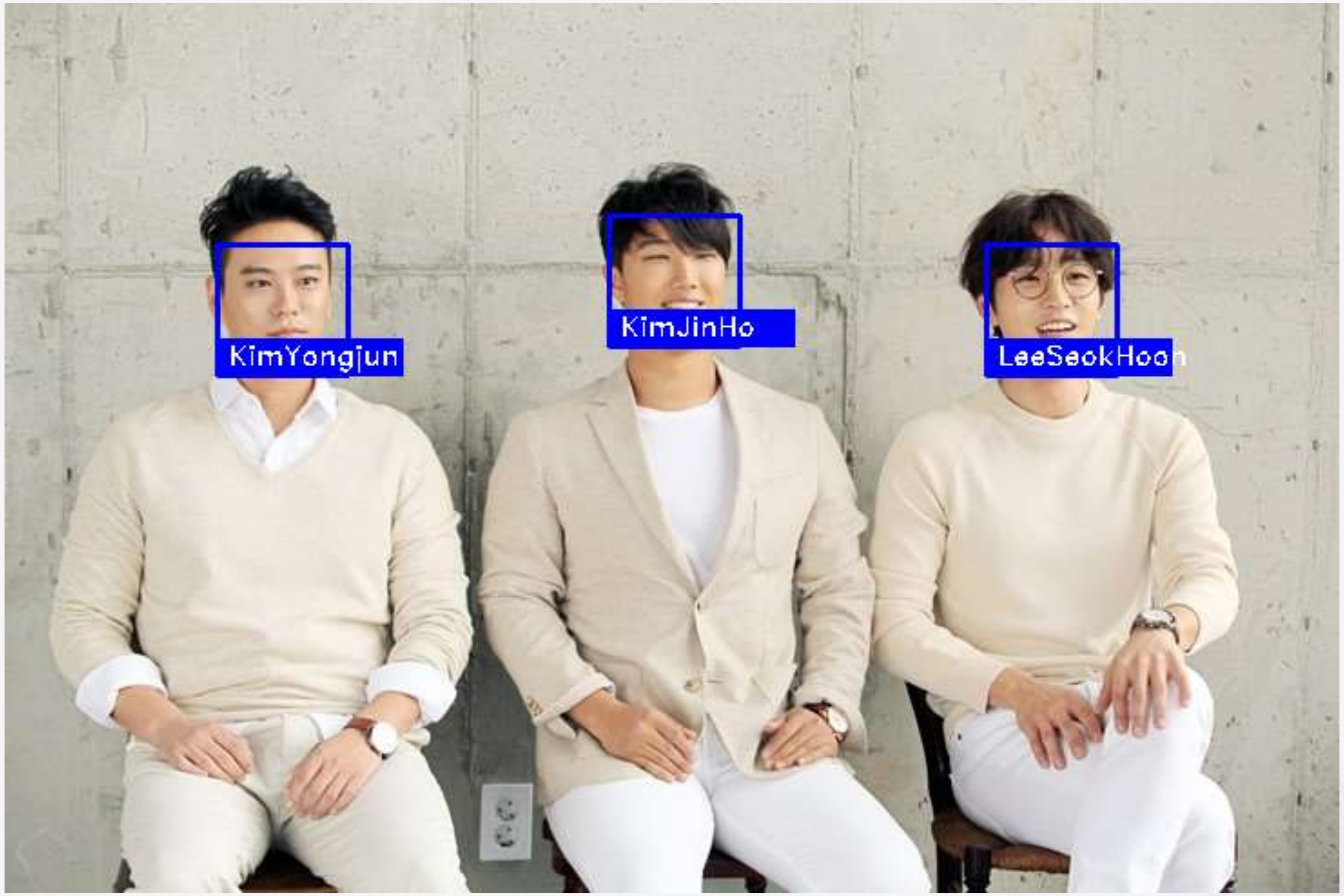
얼굴 검증 (Face Verification) vs 얼굴 식별 (Face Identification)



References
http://lacienciadelcafe.com.ar/kids-jbl-headphones/parka-arm%C3%A9e-de-l//iproov-on-twitter-what-s-the-difference-between-face-pp-24027720

[실습3] Face Recognition

ArcFace Loss로 학습한 모델로 그룹 가수 멤버 인식하기



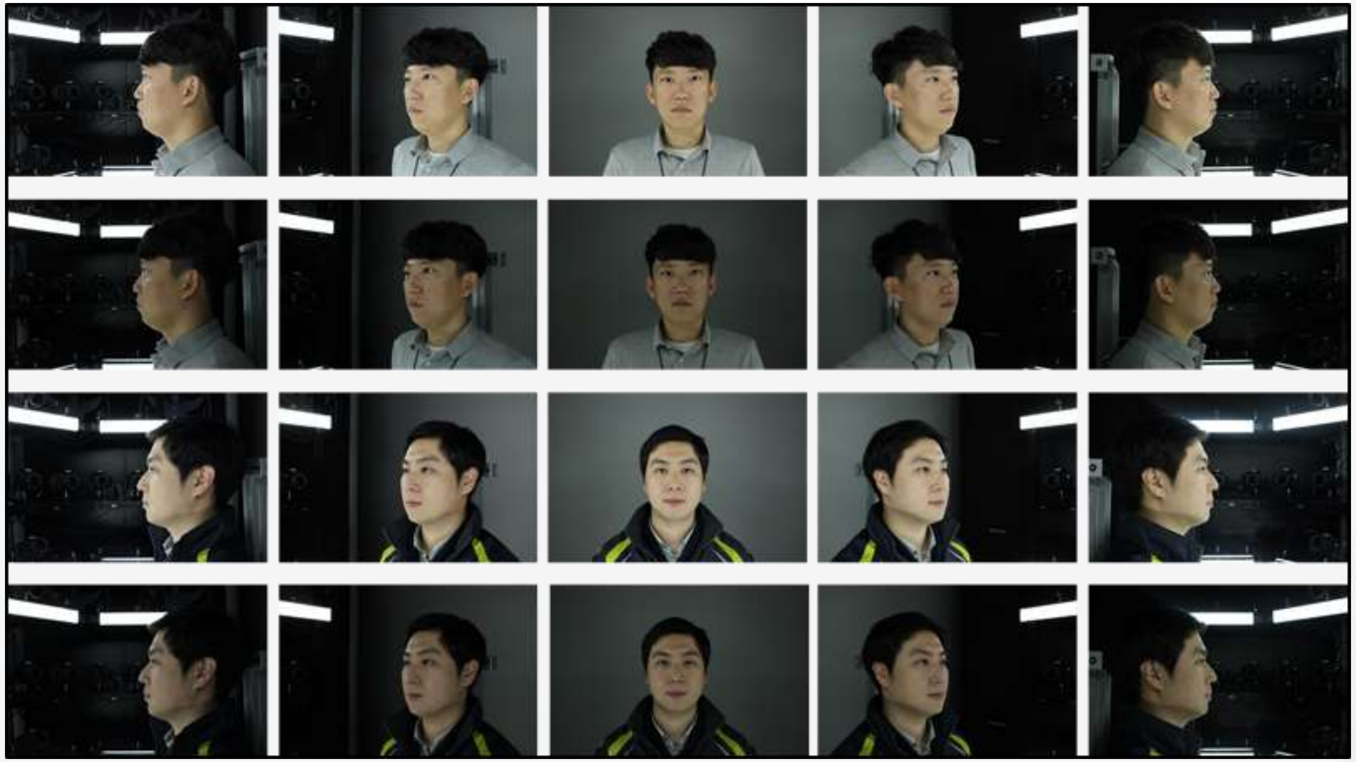
데이터셋

데이터셋 소개 – KFace

K-FACE: A Large-Scale KIST Face Database in Consideration with Unconstrained Environments

Yeji Choi*[1,2], Hyunjung Park *[1,3], Gi Pyo Nam[1], Haksub Kim[1], Heeseung Choi[1], Junghyun Cho[1] and Ig-Jae Kim†[1,3]

[1]Korea Institute of Science and Technology (KIST)
[2]Yonsei University
[3]KIST-School, University of Science and Technology



References
https://arxiv.org/pdf/2103.02211.pdf

데이터셋 소개

Paper : https://arxiv.org/pdf/2103.02211.pdf

Github : https://github.com/k-face/k-face_2019

공식사이트 : https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=&topMenu=&aihubDataSe=realm&dataSetSn=83

한국인 안면 이미지 데이터셋

1000명의 얼굴, 32,400,000장

얼굴별 데이터 종류

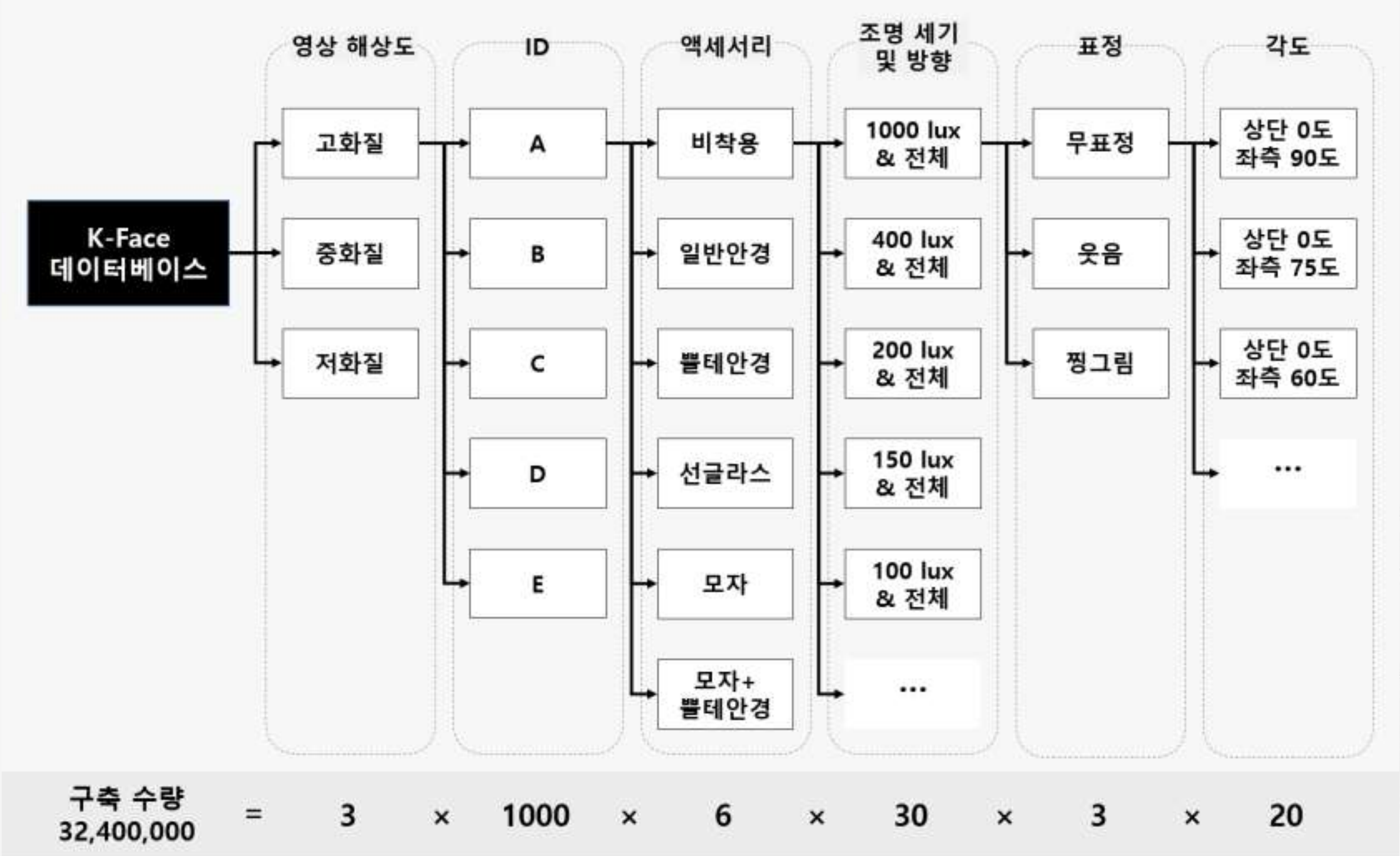
- 각도 (20 views)
- 조도 (30 lightings)
- 가림 (안경 등 6종)
- 표정 (3개 표정)
- 해상도 (3개 해상도)

데이터셋 다운로드

신청사이트 : https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=&topMenu=&aihubDataSe=realm&dataSetSn=83

데이터베이스 구조

8자리 파일명 : ID_액세서리속성_조명속성_표정속성_포즈속성



References
https://github.com/k-face/k-face_2019

데이터베이스 세부 사항

방향		
	+	-
수직	상단	하단
수평	우측	좌측

Accessory	
S001	보통
S002	일반 안경
S003	뿔테 안경
S004	선글라스
S005	모자
S006	모자+뿔테 안경

Light			
	Lux	방향	
		수직	수평
L1	1000	전체	
L2	400	전체	
L3	200	전체	
L4	150	전체	
L5	100	전체	
L6	40	전체	
L7	0	전체	
L8	400	+30°	전체
L9	200	+30°	전체
L10	100	+30°	전체
L11	40	+30°	전체
L12	400	-15°	전체
L13	200	-15°	전체
L14	100	-15°	전체
L15	40	-15°	전체
L16	400	전체	+90°
L17	200	전체	+90°
L18	100	전체	+90°
L19	400	전체	+45°
L20	200	전체	+45°
L21	100	전체	+45°
L22	400	전체	+0°
L23	200	전체	+0°
L24	100	전체	+0°
L25	400	전체	-45°
L26	200	전체	-45°
L27	100	전체	-45°
L28	400	전체	-90°
L29	200	전체	-90°
L30	100	전체	-90°

Camera (각도)		
	방향	
	수직	수평
C1	0°	+90°
C2	0°	+75°
C3	0°	+60°
C4	0°	+45°
C5	0°	+30°
C6	0°	+15°
C7	0°	+0°
C8	0°	-15°
C9	0°	-30°
C10	0°	-45°
C11	0°	-60°
C12	0°	-75°
C13	0°	-90°
C14	+30°	+45°
C15	+30°	+15°
C16	+30°	0°
C17	+30°	-15°
C18	+30°	-45°
C19	-15°	+30°
C20	-15°	-30°

Expression	
E01	무표정
E02	활짝 웃음
E03	찡그림

References
https://github.com/k-face/k-face_2019

데이터셋 구조

- face_recognition
  - High_Resolution
  - Low_Resolution
  - MIddle_Resolution

19062531.zip
19062542.zip
19062621.zip
19062622.zip
19062641.zip
19062722.zip
19062731.zip
19062732.zip

ID\액세서리속성\조명속성\표정속성\포즈속성.*

19062531 > S001 > L1 > E01

C1.jpg
C1.txt
C2.jpg
C2.txt
C3.jpg
C3.txt
C4.jpg
C4.txt
C5.jpg
C5.txt
C6.jpg
C6.txt
C7.jpg
C7.txt
C8.jpg
C8.txt
C9.jpg
C9.txt
C10.jpg
C10.txt

데이터셋 구조

그림7 | 한국인 안면 이미지 특징점 예시

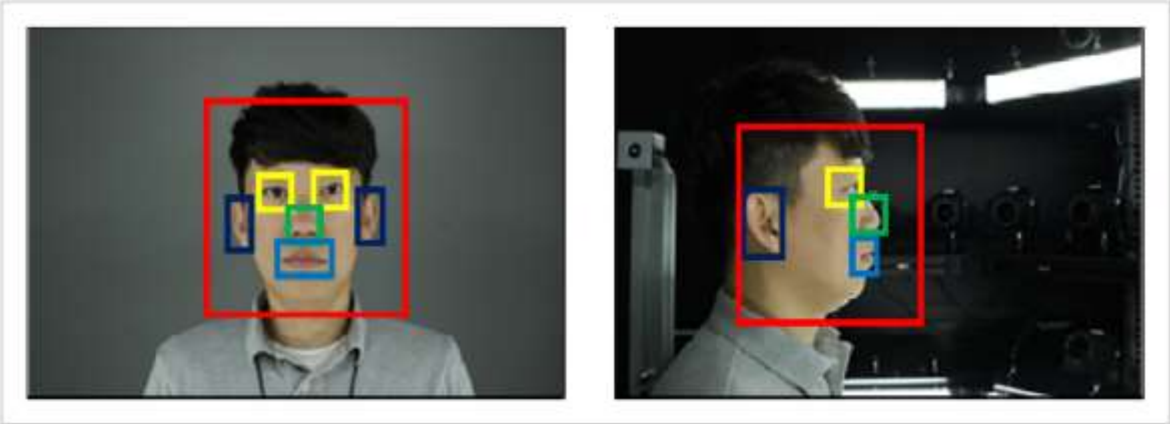


그림8 | 한국인 안면 이미지 바운딩 박스 예시

C1.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

7개의 특징점 좌표

176	116
186	97
190	96
189	137
191	139
246	110
250	110

156	51	111	111
183	88	18	13
172	97	18	29
179	130	16	16
233	88	35	45

바운딩박스의 코너좌표 및 너비, 높이

References
https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=&topMenu=&aihubDataSe=realm&dataSetSn=83

실습 튜토리얼

ArcFace

논문 : https://arxiv.org/pdf/1801.07698.pdf

공식 Github : https://github.com/deepinsight/insightface

Pytorch Github : https://github.com/deepinsight/insightface/tree/master/recognition/arcface_torch

ArcFace: Additive Angular Margin Loss for Deep Face Recognition

Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou

Abstract—Recently, a popular line of research in face recognition is adopting margins in the well-established softmax loss function to maximize class separability. In this paper, we first introduce an Additive Angular Margin Loss (ArcFace), which not only has a clear geometric interpretation but also significantly enhances the discriminative power. Since ArcFace is susceptible to the massive label noise, we further propose sub-center ArcFace, in which each class contains K sub-centers and training samples only need to be close to any of the K positive sub-centers. Sub-center ArcFace encourages one dominant sub-class that contains the majority of clean faces and non-dominant sub-classes that include hard or noisy faces. Based on this self-propelled isolation, we boost the performance through automatically purifying raw web faces under massive real-world noise. Besides discriminative feature embedding, we also explore the inverse problem, mapping feature vectors to face images. Without training any additional generator or discriminator, the pre-trained ArcFace model can generate identity-preserved face images for both subjects inside and outside the training data only by using the network gradient and Batch Normalization (BN) priors. Extensive experiments demonstrate that ArcFace can enhance the discriminative feature embedding as well as strengthen the generative face synthesis.

Index Terms—Large-scale Face Recognition, Additive Angular Margin, Noisy Labels, Sub-class, Model Inversion

실습 환경 구축

Git clone https://github.com/deepinsight/insightface

Pytorch 설치 (torch>=1.12.0)

pip install -r requirement.txt

References

데이터셋 전처리

1) 압축풀기

```
if not os.path.exists("Middle_Resolution"): os.makedirs("Middle_Resolution")
zipfile.ZipFile("Middle_Resolution.zip").extractall("Middle_Resolution")
```

데이터셋 전처리

2) 데이터 선택

```
lux = ["L1", "L3"]
emotion = ["E01", "E02", "E03"]
angle = ["C6", "C7", "C8", "C9"]
accs = ['S001']
img_names = []
txt_names = []
for acc in accs:
    for l in lux:
        for e in emotion:
            for c in angle:
                img_names.append(acc + '/' + l + '/' + e + '/' + c + '.jpg')
                txt_names.append(acc + '/' + l + '/' + e + '/' + c + '.txt')

for file, cls_ in tqdm(zip(file_list, class_names)):
    if not os.path.exists("kface_data/" + cls_):
        os.makedirs("kface_data/" + cls_)

    for img, txt in zip(img_names, txt_names):
        zipfile.ZipFile(os.path.join(root_folder, file)).extract(img)
        zipfile.ZipFile(os.path.join(root_folder, file)).extract(txt)
    shutil.move("S001", "kface_data/" + cls_)
```

데이터셋 전처리

3) 얼굴영역 Crop

```
with open(txt, 'r') as f:
     bbox = f.read().split('\n')[7].split()
     bbox = list(map(int, bbox))
     (x, y, w, h) = bbox

     img = cv2.imread(img)
     img = img[y: y + h, x: x + w]
     img = cv2.resize(img, (112,112))
     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

데이터셋 전처리

4) Train / test분리

총 class 400, 390:10 분리

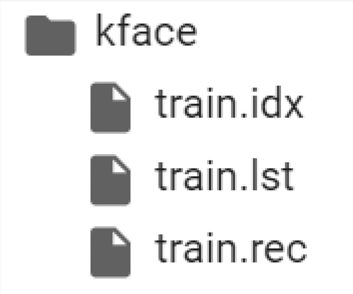
```
if j >= 390:
    base_val = "fr_kface/val/" + str(j - 390)
    if not os.path.exists(base_val):
        os.makedirs(base_val)
    paaa = os.path.join(base_val, str(j-390) + '_' + name)+ '.jpg'
    Image.fromarray(img).save(os.path.join(base_val, str(j-390) + '_' + name) + '.jpg')
else:
    base = "fr_kface/train/" + str(j)
    if not os.path.exists(base):
        os.makedirs(base)
    paaa = os.path.join(base, str(j) + '_' + name)+ '.jpg'
    Image.fromarray(img).save(os.path.join(base, str(j) + '_' + name) + '.jpg')
```

데이터셋 전처리

5) 학습 데이터셋 변환

- train.lst, train.idx, train.rec 생성
- val.lst, val.idx, val.rec 생성

```
!python -m mxnet.tools.im2rec ./train fr_kface/train --list --recursive
!python -m mxnet.tools.im2rec ./  fr_kface/train --recursive --pass-through --pack-label --num-thread 8
!python -m mxnet.tools.im2rec ./val fr_kface/val --list --recursive
!python -m mxnet.tools.im2rec ./  fr_kface/val --recursive --pass-through --pack-label --num-thread 8
```



Training

python train_v2.py configs/kface_r50_onegpu

Fine tuning – msceleb1m

```
weight_path = 'model_data/backbone.pth'
backbone.module.load_state_dict(torch.load(weight_path, map_location = 'cuda'))
```

Training

Config file - kface_r50_onegpu

```
config.network = "r50"
config.resume = False
config.output = None
config.embedding_size = 512
config.sample_rate = 1.0
config.fp16 = False
config.momentum = 0.9
config.weight_decay = 5e-4
config.batch_size = 16
config.lr = 0.02
config.verbose = 2000
config.dali = False

config.rec = "/content/drive/MyDrive/dataset/face_recognition/kface"
config.num_classes = 390
config.num_image = 9360
config.num_epoch = 20
```

Inference

```
weight_path = './model_data/model.pt'
device = torch.device("cuda" if
torch.cuda.is_available() else "cpu")
model = iresnet50().to(device)
model.load_state_dict(torch.load(weight_path,
map_location = device))
model.eval()
```

```
def face_embedding(model, img, dsize=112,
device='cuda'):
    img = cv2.resize(img, (dsize,dsize))
    img = np.transpose(img, (2, 0, 1))
    img = torch.from_numpy(img).unsqueeze(0).float()
    img.div_(255).sub_(0.5).div_(0.5)
    img = img.to(device)
    embed = model(img).detach().cpu().numpy()
    return l2_norm(embed)
```

[Practice 3] Face Recognition

1) 얼굴 등록

```
known_face_embeddings = []
known_face_names = []

for p in person_list:
    name, _ = os.path.splitext(p)
    known_face_names.append(name)
    person_img = cv2.imread(os.path.join(folder_path, p))
    person_img_gray = cv2.cvtColor(person_img, cv2.COLOR_BGR2GRAY)
    face_detection = face_detector(person_img, 2)
    f = face_detection[0]
    faceAligned = fa.align(person_img, person_img_gray, f)
    known_embed = face_embedding(model, faceAligned, 112)
    known_face_embeddings.append(known_embed)
```

Face Recognition

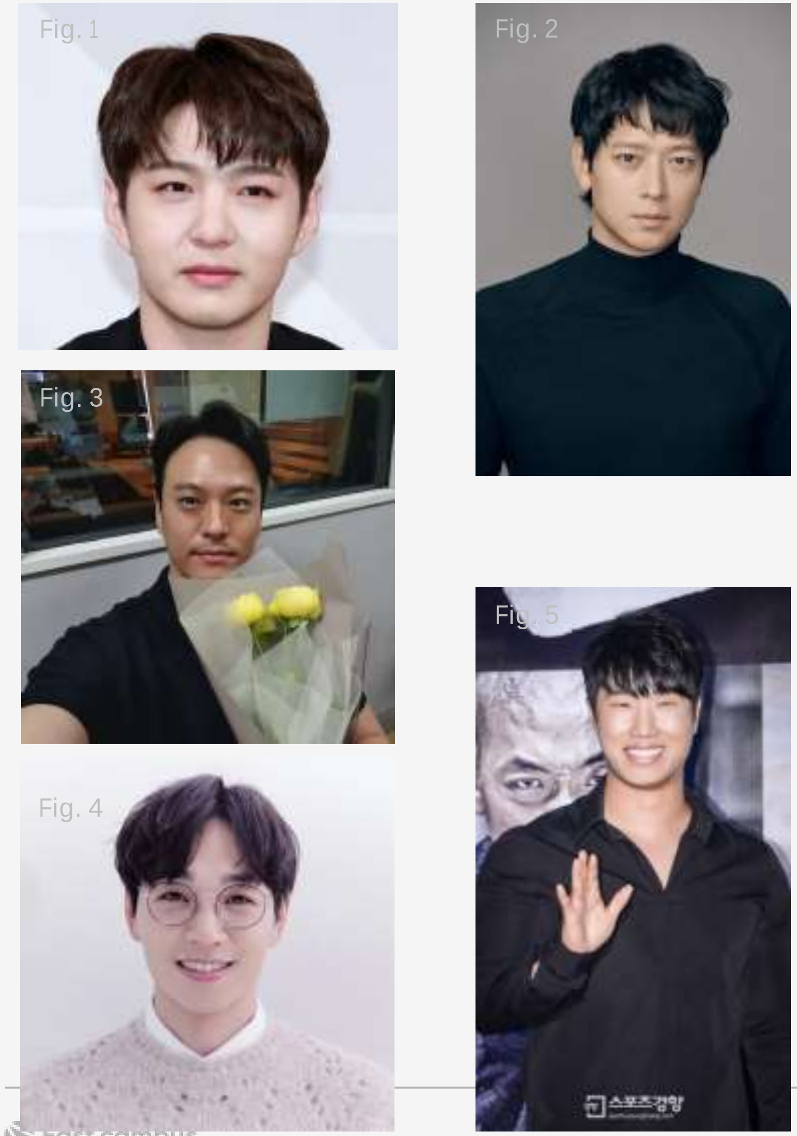
2) Face Recognition

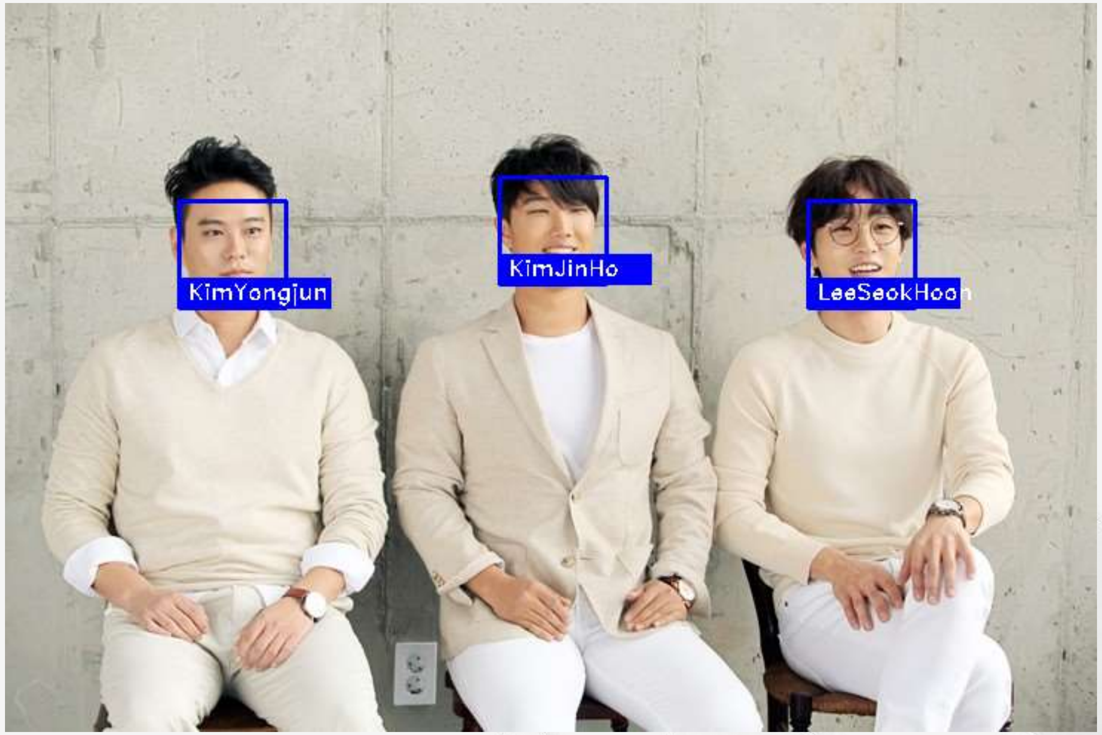
```
face_detection = face_detector(test_img)
for f in face_detection:
    faceAligned = fa.align(test_img, person_img_gray, f)
    camera_embed = face_embedding(model, faceAligned, 112)

    distances = []
    for idx, embd in enumerate(known_face_embeddings):
        embeds_distance = np.subtract(camera_embed, embd)
        embeds_distance = np.sum(np.square(embeds_distance), axis=1)
        distances.append(embeds_distance)
    name = 'unknown'
    if min(distances) < 1.5:
        idx = np.argmin(distances)
        name = known_face_names[idx]
        print(name, min(distances))
    cv2.rectangle(test_img, (f.left(), f.top()), (f.right(), f.bottom()), (255,0,0), 2)
    cv2.rectangle(test_img, (f.left(), f.bottom()-20), (f.right()+30, f.bottom()), (255,0,0), cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(test_img, name, (f.left()+6,f.bottom()-6), font, 0.5, (255,255,255), 1)
```

실습 결과

실행 결과

인물 Database





References
(Fig. 1) 서경스타DB
(Fig. 2) YG엔터테인먼트
(Fig. 3) 김용준인스타그램캡처
(Fig. 4) 멜론프로필
(Fig. 5) https://m.sports.khan.co.kr/view.html?art_id=202207291714003&sec_id=540101

Thank You.