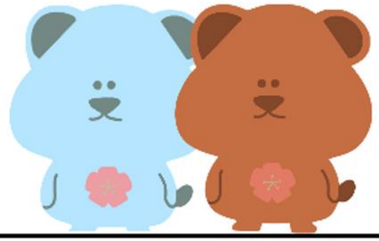
Human Protein Atlas Image Classification

1팀 이지혜 한예송 홍재령

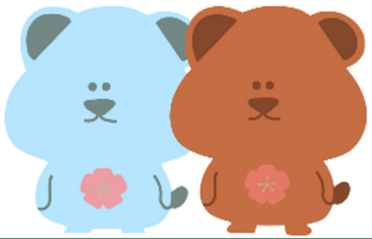
EWHA EURON

목차



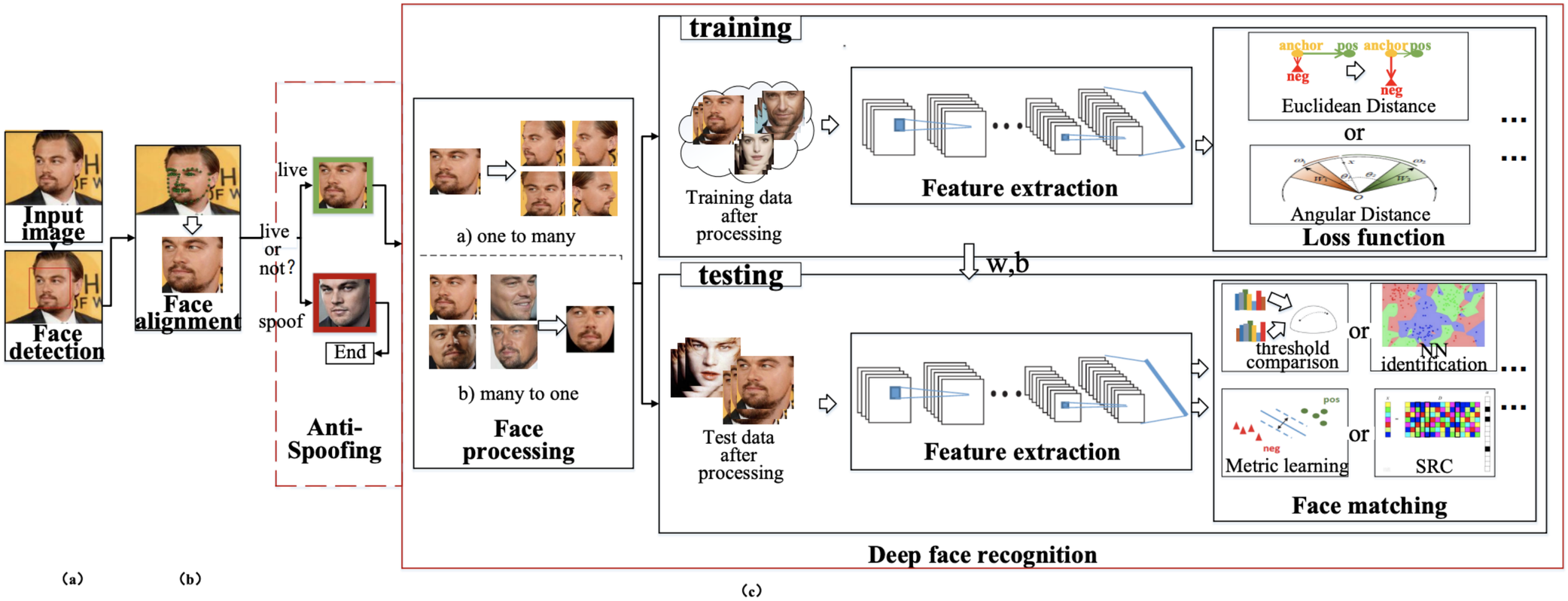
EWHA EURON

Loss function



EWHA EURON

Face Recognition



Loss function – Angular loss

Softmax function

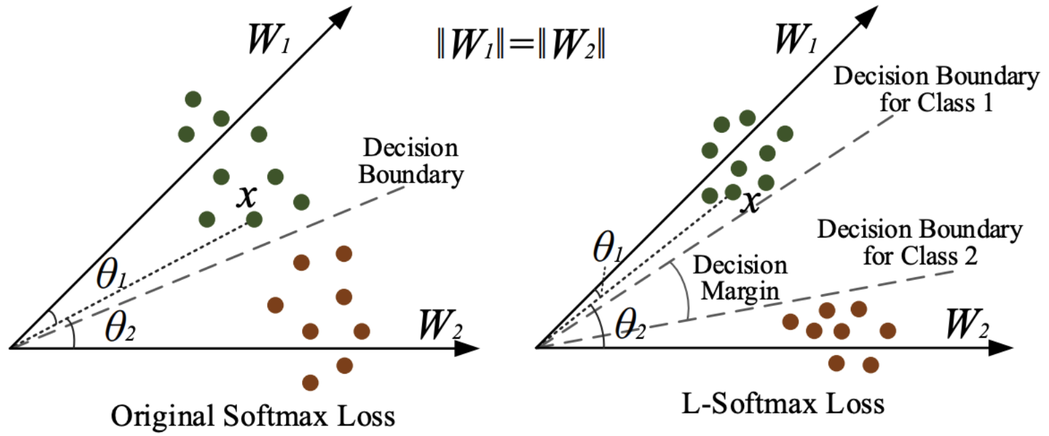
$$p_1 = \frac{\exp(\boldsymbol{W}_1^T \boldsymbol{x} + b_1)}{\exp(\boldsymbol{W}_1^T \boldsymbol{x} + b_1) + \exp(\boldsymbol{W}_2^T \boldsymbol{x} + b_2)} \quad (1)$$

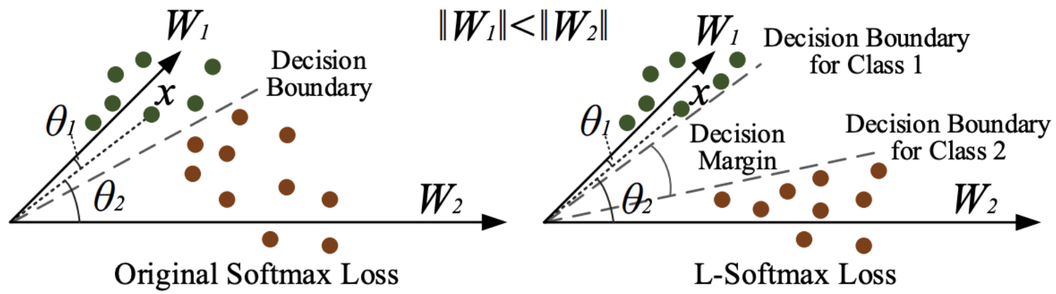
$$p_2 = \frac{\exp(\boldsymbol{W}_2^T \boldsymbol{x} + b_2)}{\exp(\boldsymbol{W}_1^T \boldsymbol{x} + b_1) + \exp(\boldsymbol{W}_2^T \boldsymbol{x} + b_2)} \quad (2)$$

Softmax loss function

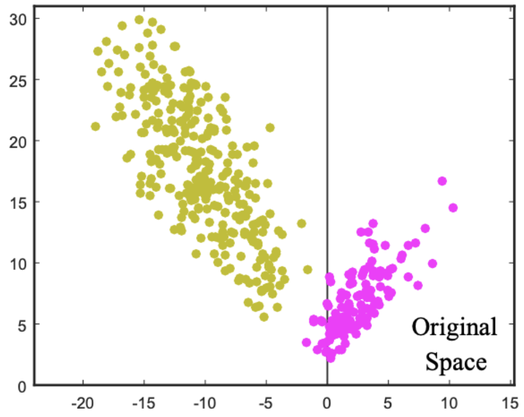
$$L_i = -\log\left(\frac{e^{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_j e^{\boldsymbol{W}_j^T \boldsymbol{x}_i + b_j}}\right)$$

$$= -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i,i}) + b_{y_i}}}{\sum_j e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_{j,i}) + b_j}}\right)$$

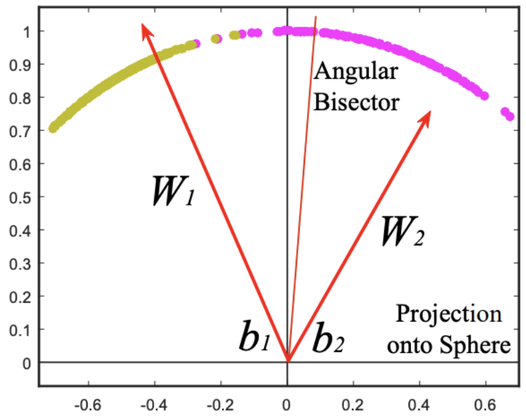




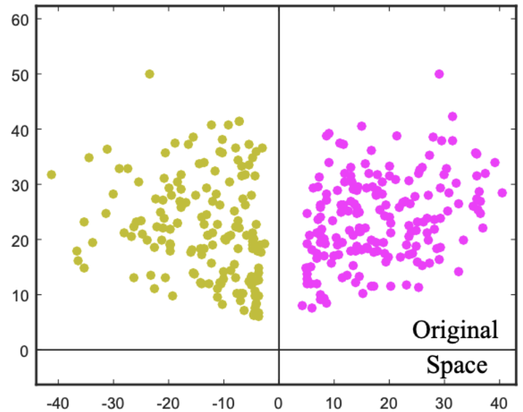
발전된 형태의 Loss function



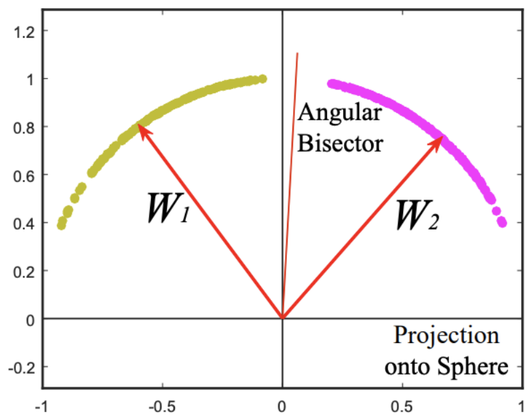
(a) Original Softmax Loss



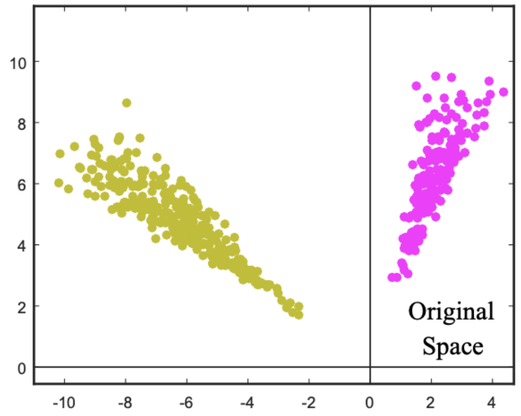
(b) Original Softmax Loss



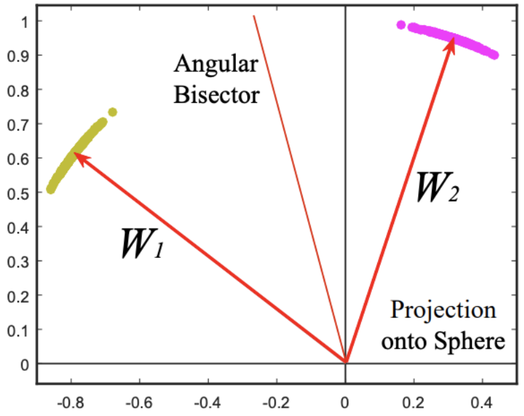
(c) Modified Softmax Loss



(d) Modified Softmax Loss



(e) A-Softmax Loss



(f) A-Softmax Loss

TABLE V

DECISION BOUNDARIES FOR CLASS 1 UNDER BINARY CLASSIFICATION CASE, WHERE $\hat{x}$ IS THE NORMALIZED FEATURE. [106]

Loss Functions	Decision Boundaries
Softmax	$(W_1 - W_2)\,x + b_1 - b_2 = 0$
L-Softmax [104]	$\|x\|\,(\|W_1\|\,cos(m\theta_1) - \|W_2\|\,cos(\theta_2)) > 0$
A-Softmax [84]	$\|x\|\,(cosm\theta_1 - cos\theta_2) = 0$
CosFace [105]	$\hat{x}\,(cos\theta_1 - m - cos\theta_2) = 0$
ArcFace [106]	$\hat{x}\,(cos\,(\theta_1 + m) - cos\theta_2) = 0$

다양한 angular loss function

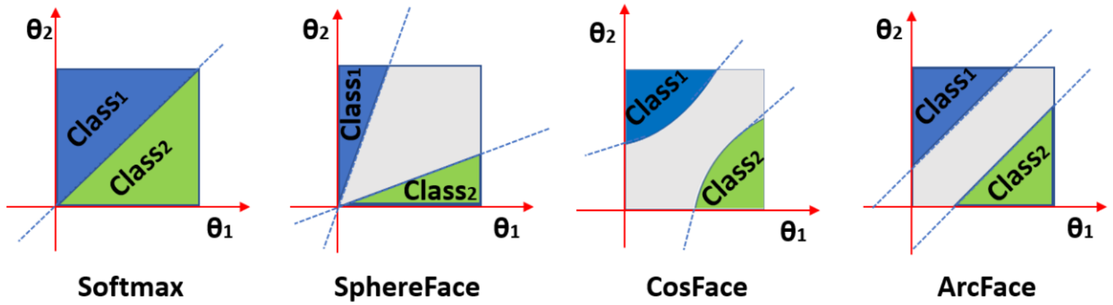
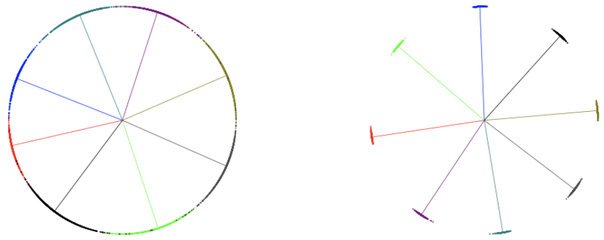


Figure 5. Decision margins of different loss functions under binary classification case. The dashed line represents the decision boundary, and the grey areas are the decision margins.

TABLE V

DECISION BOUNDARIES FOR CLASS 1 UNDER BINARY CLASSIFICATION CASE, WHERE $\hat{x}$ IS THE NORMALIZED FEATURE. [106]

Loss Functions	Decision Boundaries
Softmax	$(W_1 - W_2)\, x + b_1 - b_2 = 0$
L-Softmax [104]	$\|x\| \left(\|W_1\| \, cos(m\theta_1) - \|W_2\| \, cos(\theta_2)\right) > 0$
A-Softmax [84]	$\|x\| \left(cosm\theta_1 - cos\theta_2\right) = 0$
CosFace [105]	$\hat{x} \left(cos\theta_1 - m - cos\theta_2\right) = 0$
ArcFace [106]	$\hat{x} \left(cos\left(\theta_1 + m\right) - cos\theta_2\right) = 0$



(a) Softmax (b) ArcFace

Figure 3. Toy examples under the softmax and ArcFace loss on 8 identities with 2D features. Dots indicate samples and lines refer to the centre direction of each identity. Based on the feature normalisation, all face features are pushed to the arc space with a fixed radius. The geodesic distance gap between closest classes becomes evident as the additive angular margin penalty is incorporated.

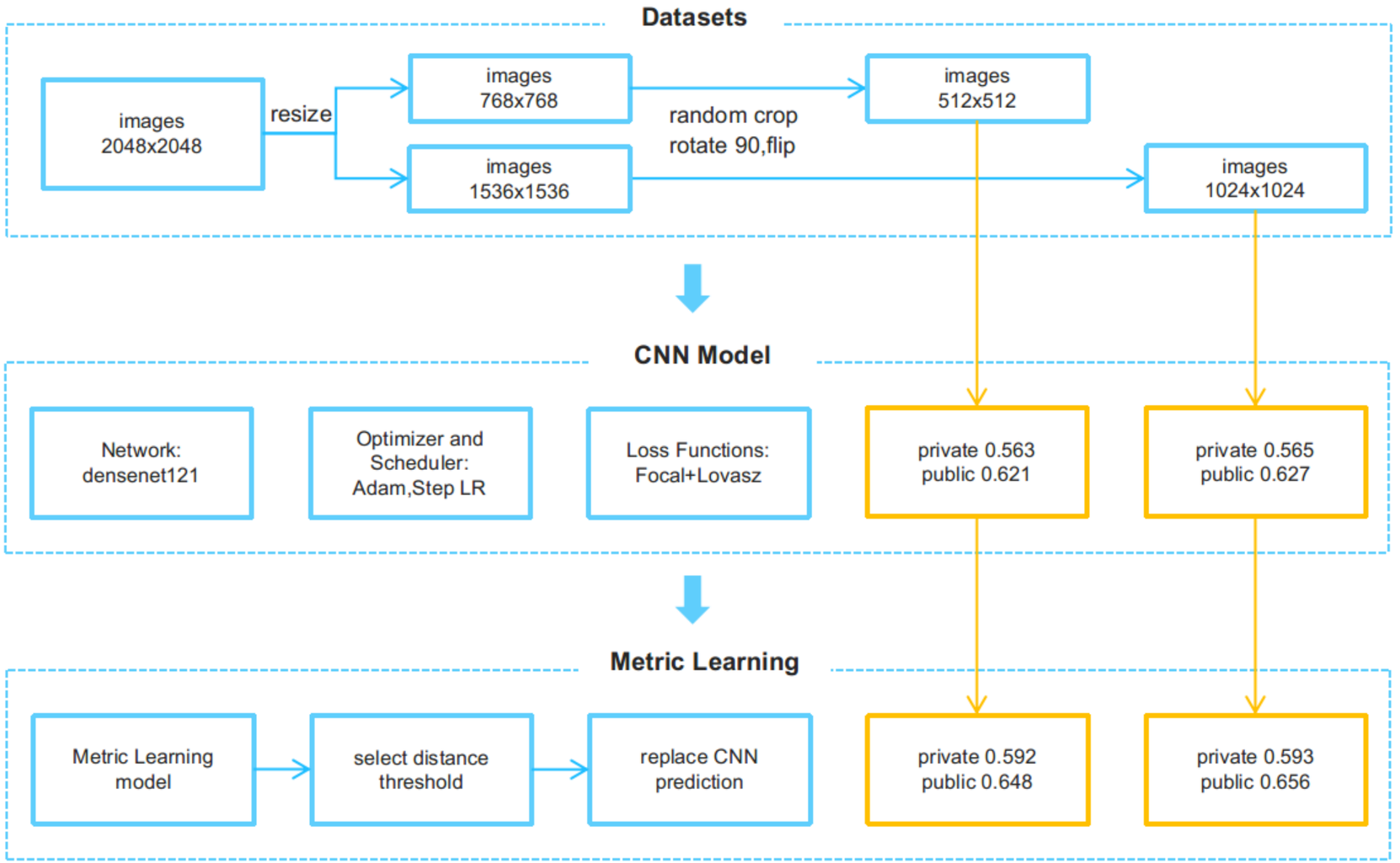
ArcFace의 성능

Loss Functions	LFW	CFP-FP	AgeDB-30
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 [18]	99.42	-	-
SphereFace (1.35)	99.11	94.38	91.70
CosFace [37]	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 (NS)	98.56	89.79	88.72
NS+Intra	98.75	93.81	90.92
NS+Inter	98.68	90.67	89.50
NS+Intra+Inter	98.73	94.00	91.41
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

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

EWHA EURON

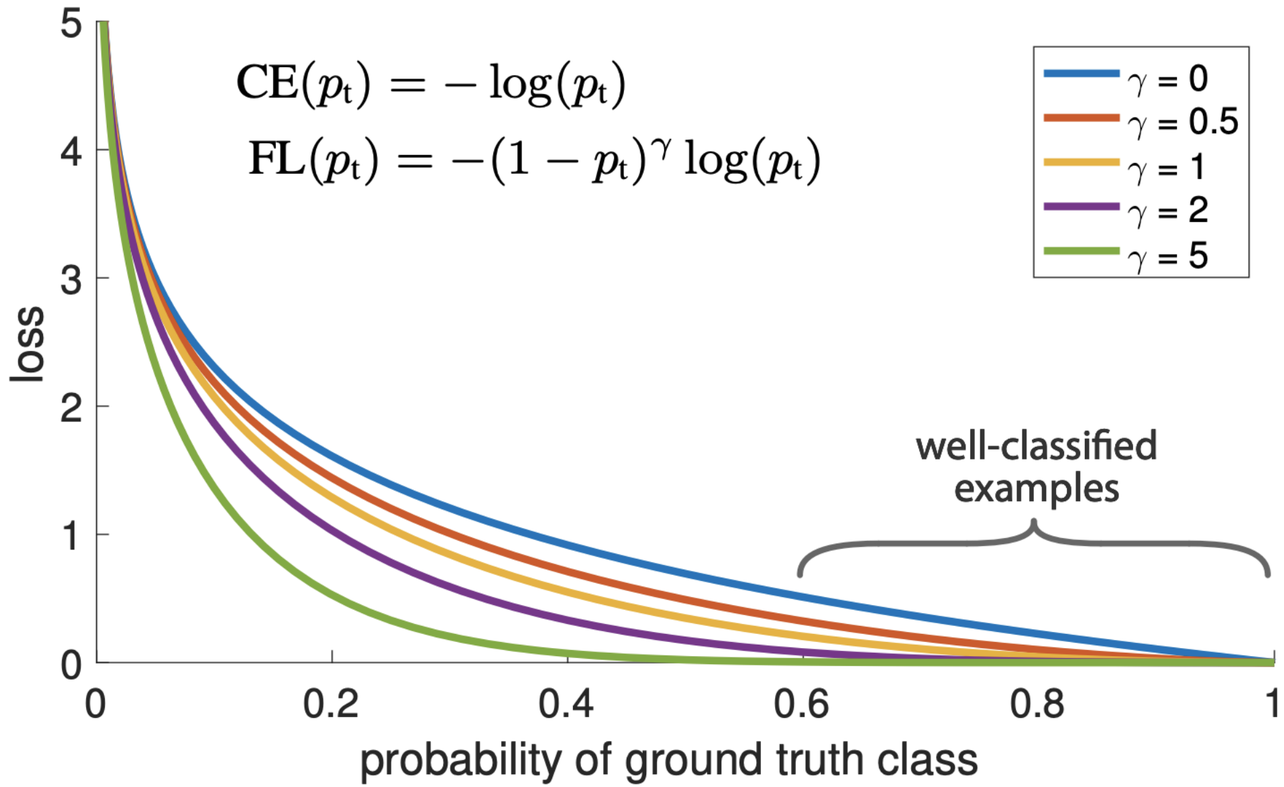
pipeline


Datasets
images
2048x2048
resize
images
768x768
images
1536x1536
random crop
rotate 90,flip
images
512x512
images
1024x1024
CNN Model
Network:
densenet121
Optimizer and
Scheduler:
Adam,Step LR
Loss Functions:
Focal+Lovasz
private 0.563
public 0.621
private 0.565
public 0.627
Metric Learning
Metric Learning
model
select distance
threshold
replace CNN
prediction
private 0.592
public 0.648
private 0.593
public 0.656

Loss: focal loss

CE: 기존의 cross entropy criterion
FL: focal loss

γ: focusing parameter
예측이 쉬운 example에 대한 loss의 비중을 낮추는 역할



분류가 잘 된 example에 대한 loss ⬇
분류가 잘 안된(혹은 어려운) example에 대한 loss ⬆
분류가 잘 안된 example에 초점을 두어 학습하게끔 해준다.

Loss: Lovasz loss

Jaccard loss: 두 집합 사이의 유사도를 측정하는 방법을 loss function으로 이용

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Lovasz loss

For a segmentation output $\tilde{\boldsymbol{y}}$ and ground truth $\boldsymbol{y}^*$, we define the set of mispredicted pixels for class c as

$$\mathbf{M}_c(\boldsymbol{y}^*, \tilde{\boldsymbol{y}}) = \{\boldsymbol{y}^* = c, \tilde{\boldsymbol{y}} \neq c\} \cup \{\boldsymbol{y}^* \neq c, \tilde{\boldsymbol{y}} = c\}. \quad (5)$$

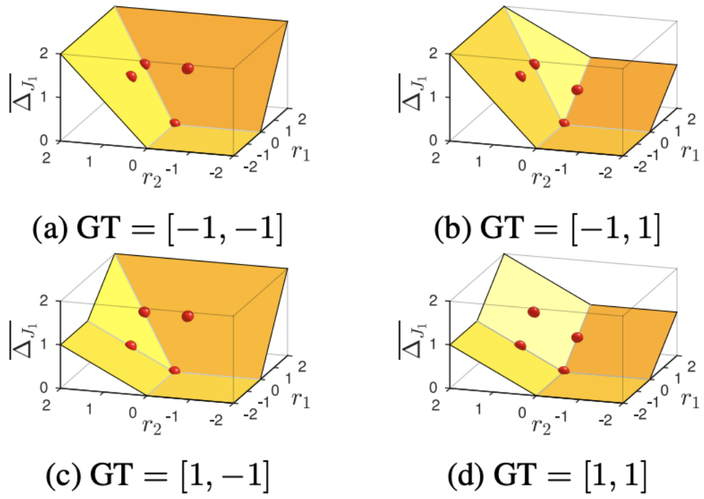
For a fixed ground truth $\boldsymbol{y}^*$, the Jaccard loss in Eq. (4) can be rewritten as a function of the set of mispredictions

$$\Delta_{J_c} : \mathbf{M}_c \in \{0,1\}^p \mapsto \frac{|\mathbf{M}_c|}{|\{\boldsymbol{y}^* = c\} \cup \mathbf{M}_c|}. \quad (6)$$

$$\text{loss}(\boldsymbol{F}) = \overline{\Delta_{J_1}}(\boldsymbol{m}(\boldsymbol{F})) \quad (10)$$

Semantic segmentation에 적합

2.1. Foreground-background segmentation



(a) GT $= [-1,-1]$ (b) GT $= [-1,1]$

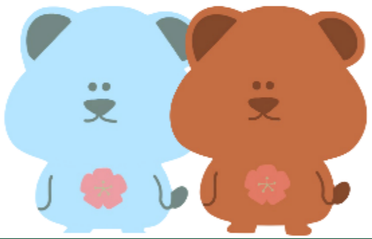
(c) GT $= [1,-1]$ (d) GT $= [1,1]$

Figure 1: Lovász hinge in the case of two pixel predictions for the four possible ground truths GT, as a function of the relative margins $r_i = 1 - F_i(\boldsymbol{x})\, y_i^*$ for $i = 1, 2$. The red dots indicate the values of the discrete Jaccard index.

image ground truth cross-entropy hinge loss Lovász hinge

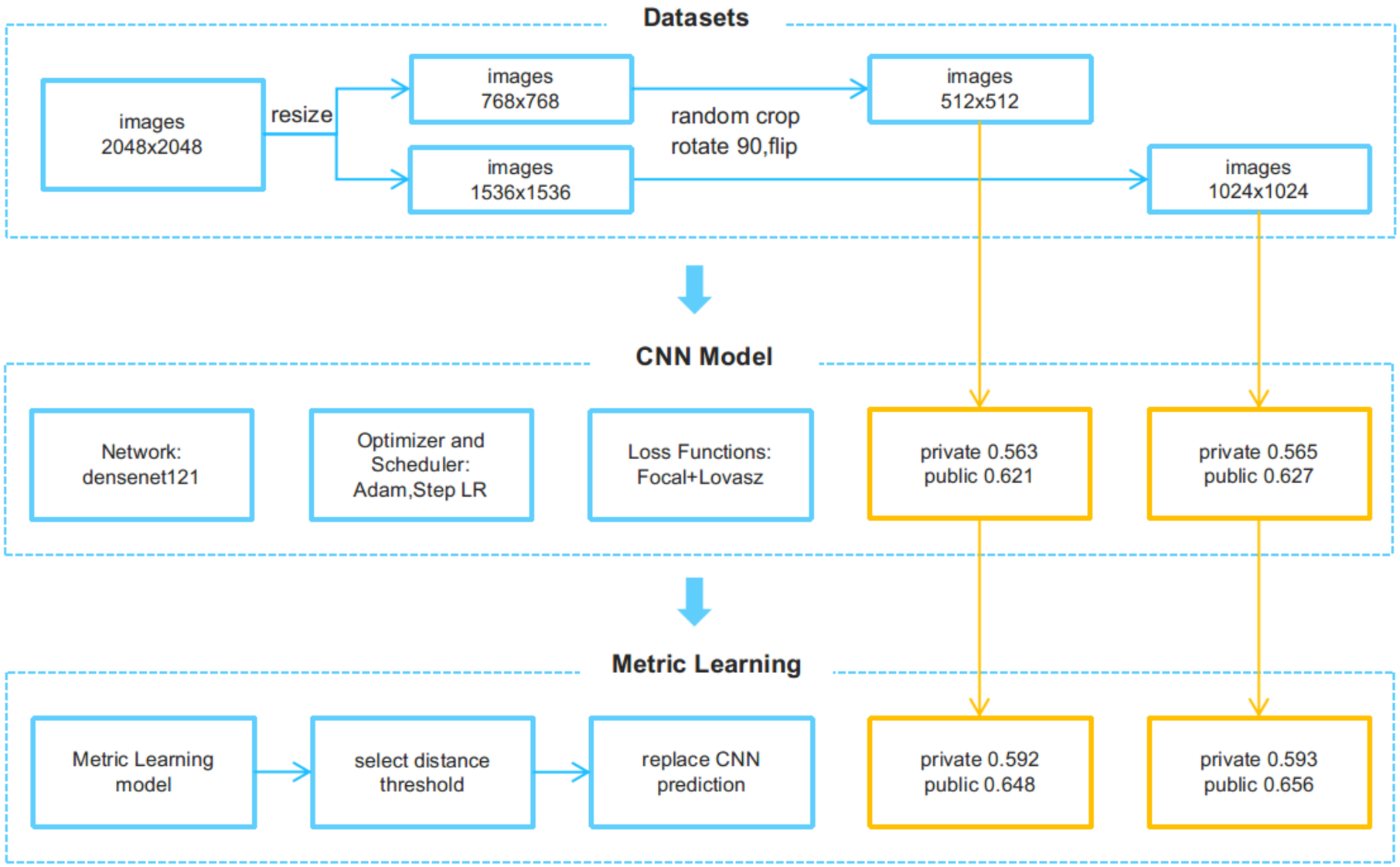
1st Place Solution

(1) 전체적인 Pipeline
(2) desnet121
(3) Metric learning



EWHA EURON

#2-1 전체적인 Pipeline



#2-2 Model Structure - densenet121

densenet

Densely Connected Convolutional Networks

architecture that

- 딥러닝 네트워크가 더욱 깊고 효율적으로 훈련되도록 하는 데에 초점을 맞춤
- 계층 간에 더 짧은 connections 사용

- 각 layer가 더 깊이 있는 layer들과 서로 연결되어있는 CNN → layer 간 maximum information flow 가짐
- l번째 layer는 i개의 input을 가지고, 이전 convolutional block들의 feature map으로 이루어짐
- 총 l개의 layer가 있을 때 connection이 l개가 아닌 (l(l+1))/2개가 있음

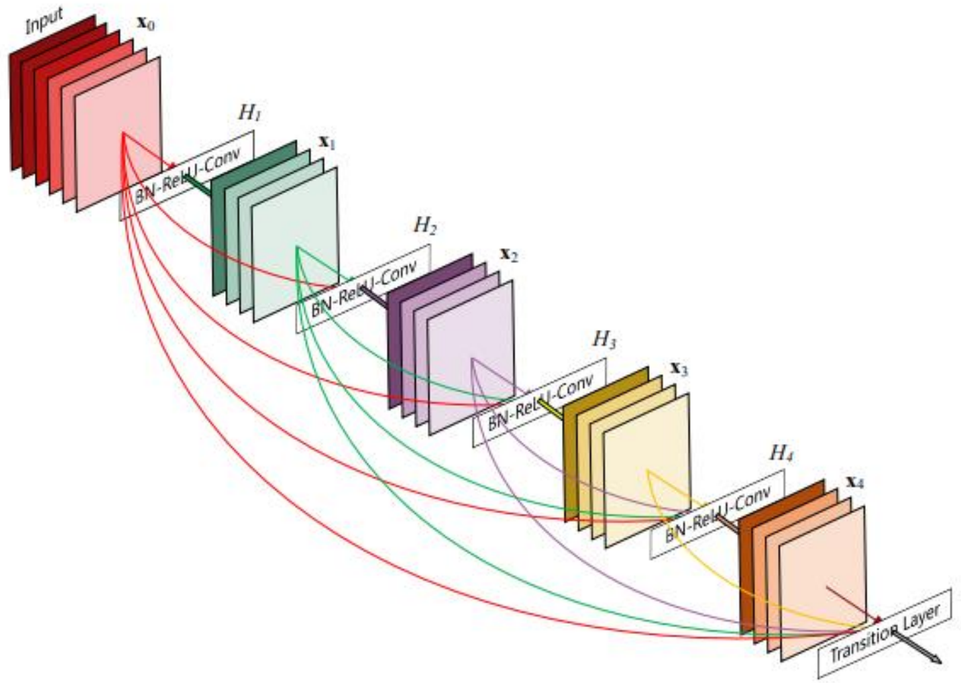


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

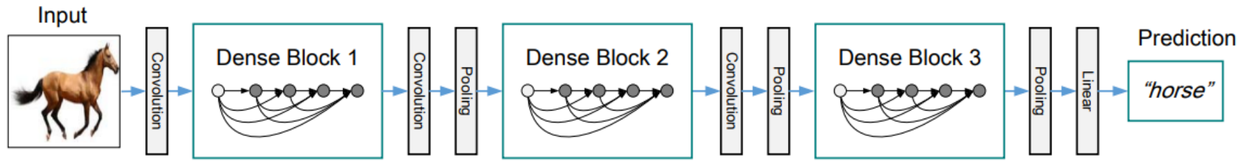


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

#2-2 Model Structure - densenet121

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56				
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 3		
Transition Layer (1)	56 × 56				
	28 × 28				
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 3		
Transition Layer (2)	28 × 28				
	14 × 14				
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 3		
Transition Layer (3)	14 × 14				
	7 × 7				
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 3 × 3 conv]	[3 × 3 conv]	[3 × 3 conv]
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

1 7x7 Convolution
58 3x3 Convolution
61 1x1 Convolution
4 AvgPool
1 Fully Connected Layer

6, 12, 24, 16은 각 블록마다의 layer의 수
→ 총 121개의 layer를 가짐

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

EWHA EURON

#2-3 Metric Learning – 일반 Classification과 비교

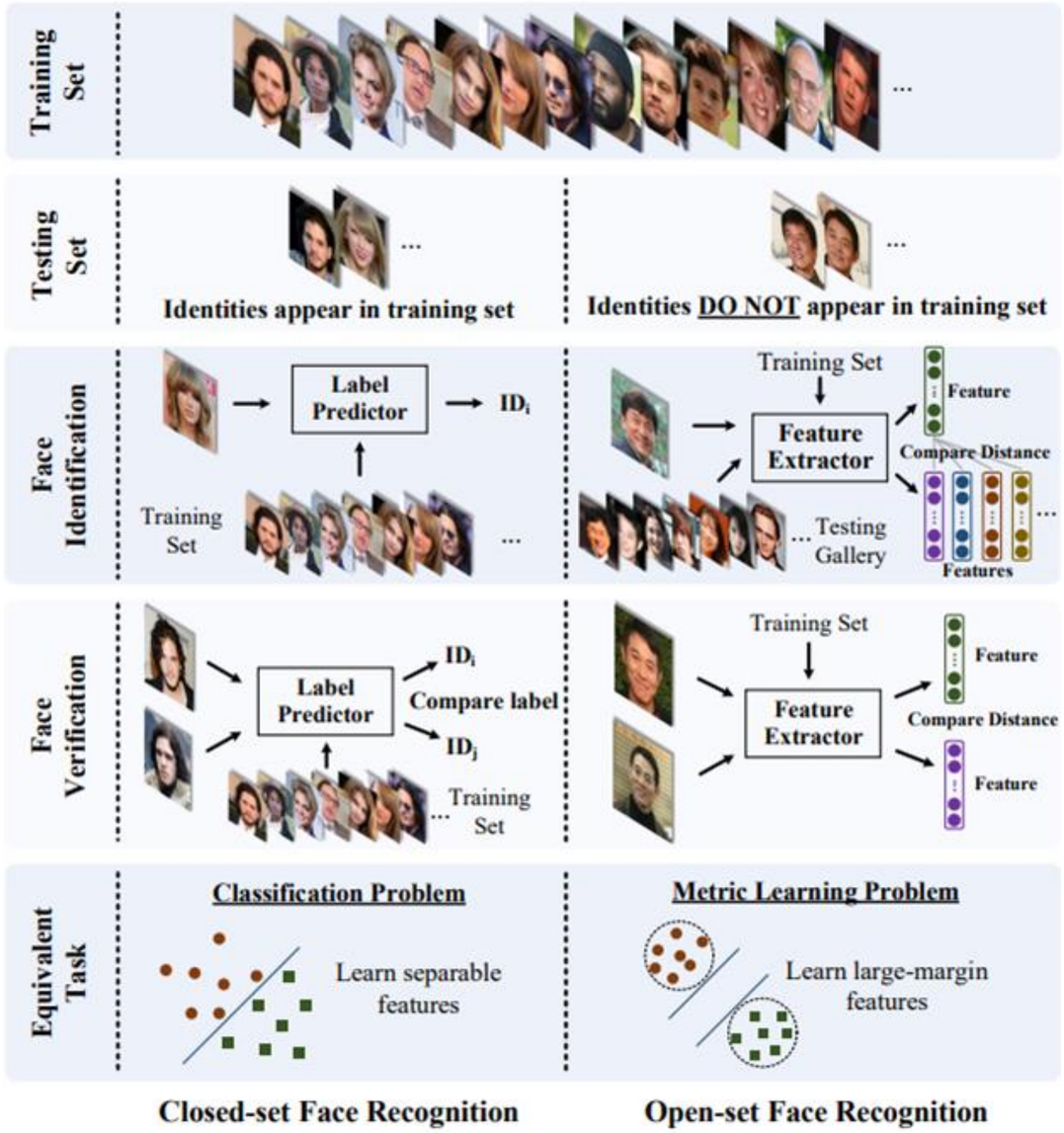


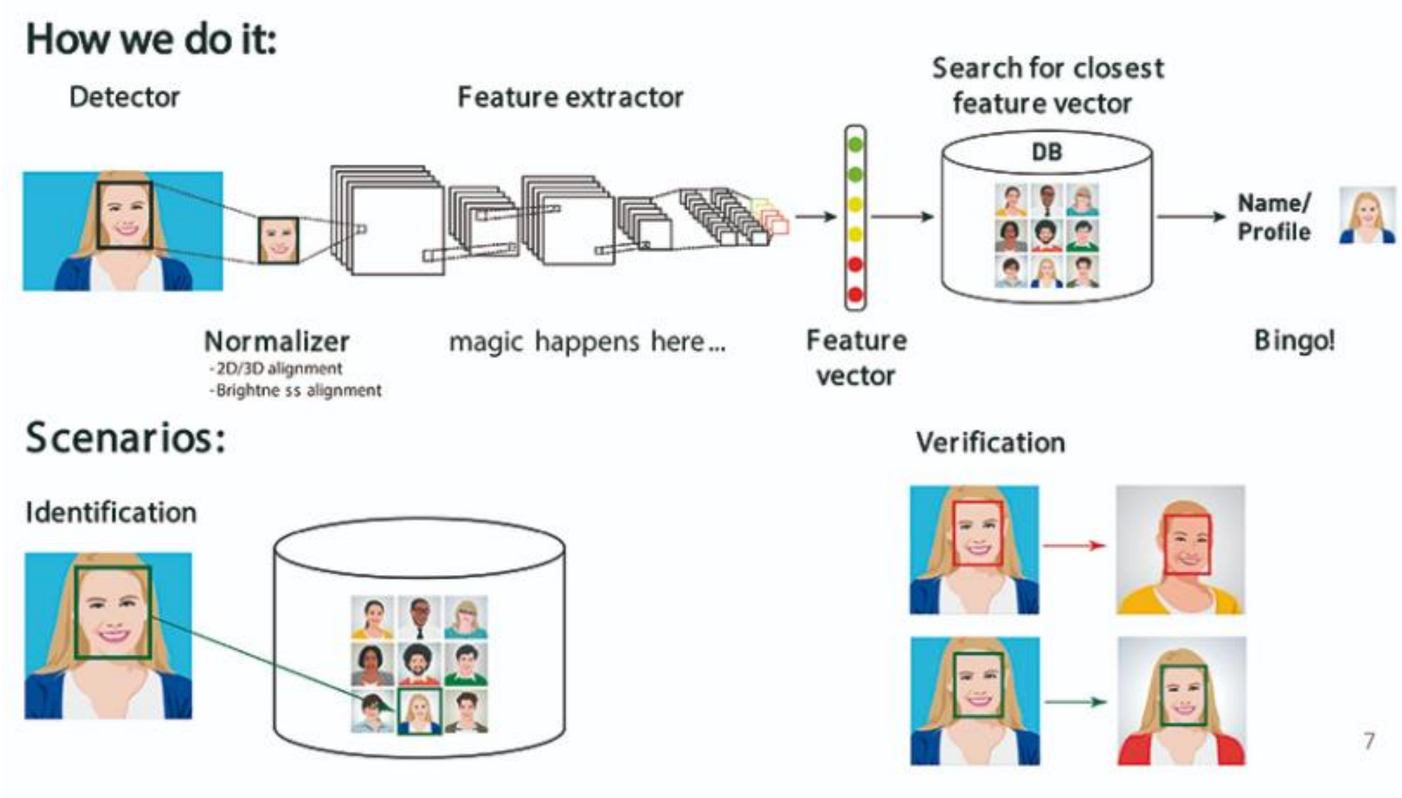
Figure 1: Comparison of open-set and closed-set face recognition.

1. 일반적인 Classification은 **학습한 이미지에 대해서만 인식이 가능**
   → Metric Learning은 **학습하지 않은 이미지도 DB로 구축**만 해둔다면 인식 가능

2. 일반적인 Classification은 **Feature 간의 Decision Boundary**를 찾도록 학습 (Learn separable features)
   → Metric Learning은 **비유사한 Feature들을 멀리 떨어지도록 학습** (Learn Large-Margin features)

EWHA EURON

#2-3 Metric Learning - 원리



거리 공간(metric space)은
두 개체 사이의 거리가 정의된 공간
= 유사한 개체는 가까이, 유사하지 않은 개체는 멀리 위치한 공간

Deep metric learning ?

= Deep neural network

= Distance

딥러닝 모델로 거리 공간을 학습한다

- 데이터 간의 유사도를 잘 수치화하는 거리 함수 (metric function)를 학습
- **얼굴인식(Face Recognition)**이나 **상품 검색(Image Retrieval)** 등에 많이 쓰임

Goal : **find a new metric to make two classes more separable**

#2-3 Metric Learning

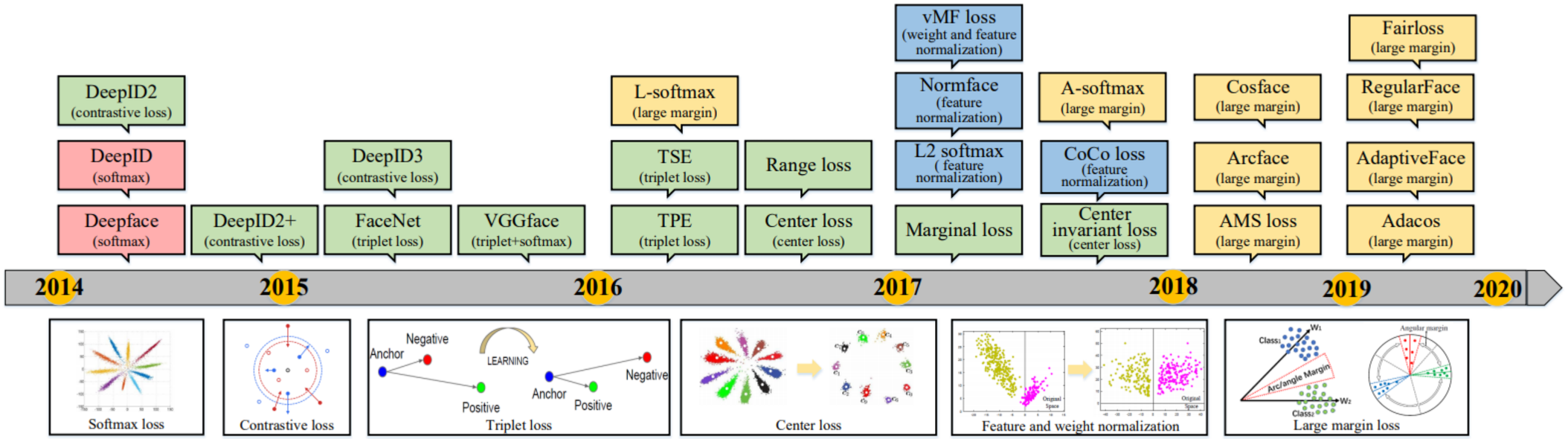
핵심은 크게 두 가지:

1. 어떤 **거리(유사도)**를 사용할 것인가?
   - Euclidean, Cosine Similarity 등

2. 유사도 학습을 위해서, 어떤 **Loss로 학습**할 것인가?
   - Contrastive Loss, Triplet Loss, Margin Loss 등

Metric Learning Model:

- Network : resnet50
- Augmentations : Rotate 90,flip
- Loss Functions : **ArcFaceLoss**
- Optimizer : Adam
- Scheduler : lr = 10e-5, 50 epochs

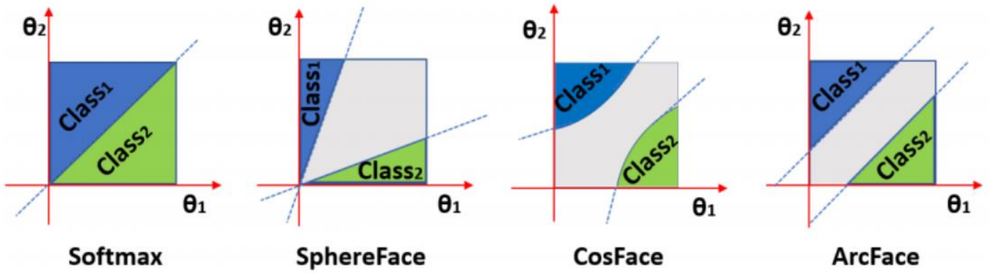


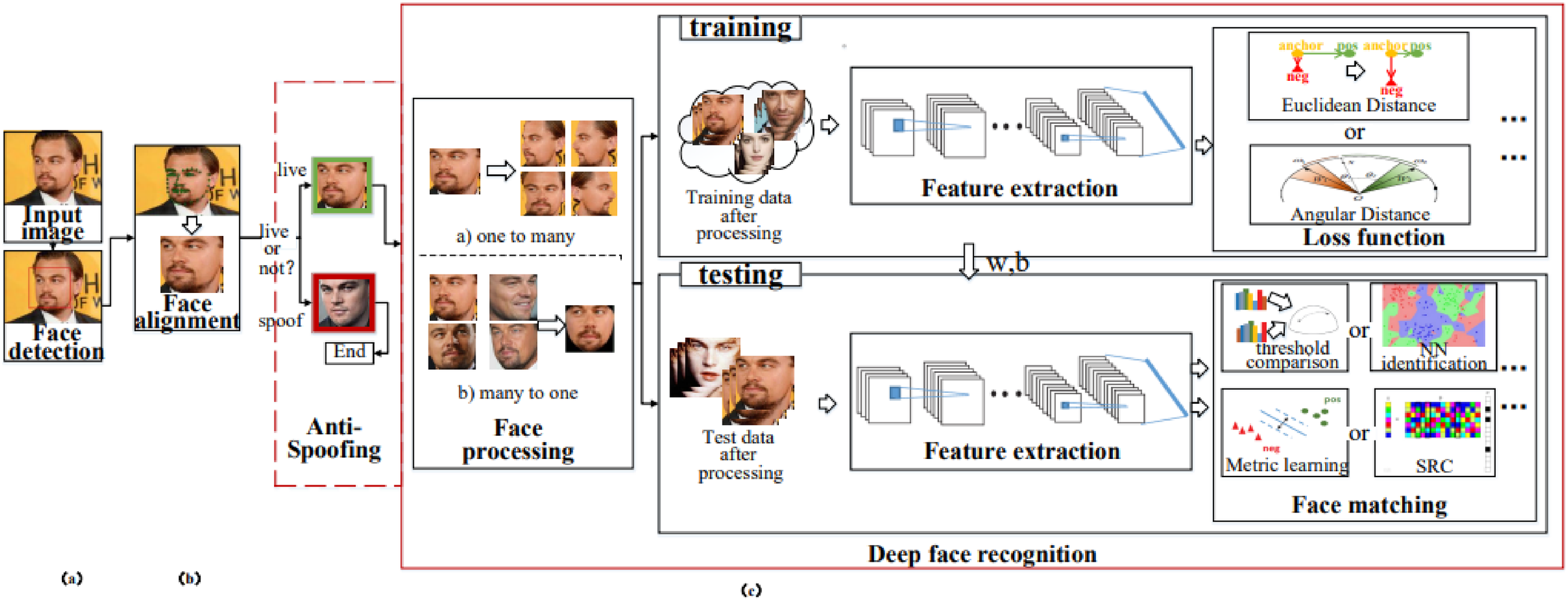
#2-3 Metric Learning – ArcFaceLoss

ArcFaceLoss

Angular/cosine-margin-based Loss

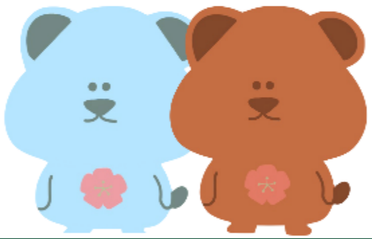
- Decision Boundary를 도입함





4th Place Solution

(1) 문제 재해석 및 사전 지식
(2) GAPNet
(3) DualLoss
(4) Tricks Used



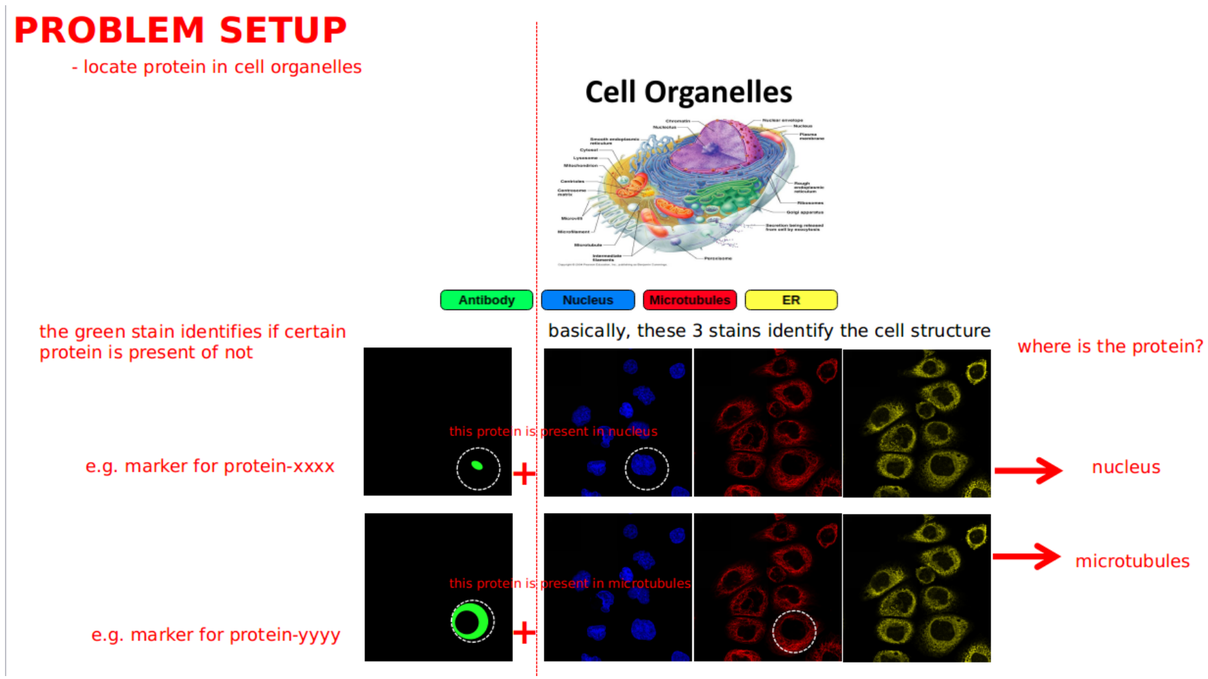
EWHA EURON

#1-1. 문제 재정의

✔1. Cell image에서 protein multi-label classification (총 28개의 class)

✔2. Localization + Classification으로 재정의
-> Attention + GAP (global average pooling)

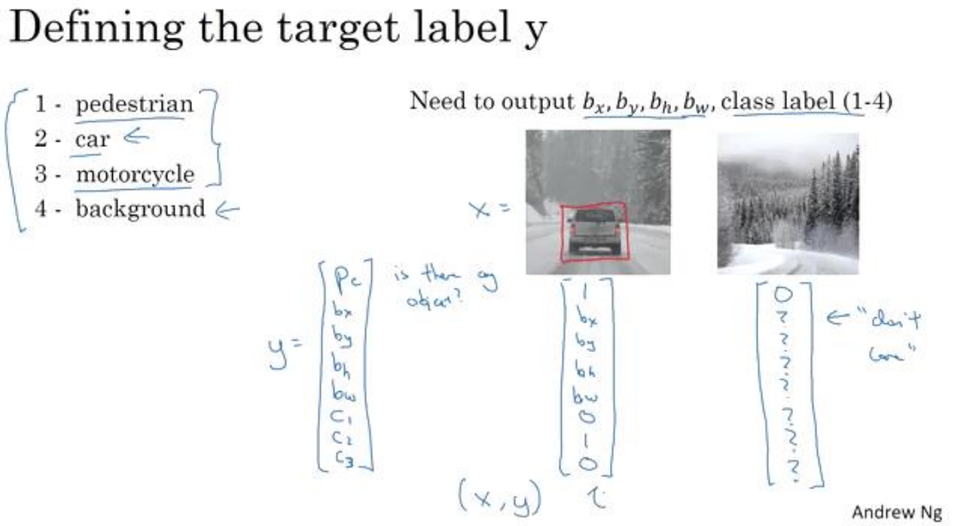
✔3. 여러 개의 protein객체 중 무엇이 어떤 종류인지의 정보 미제공 -> Weak Supervision



#1-2. Classification + Localization

✔1. 이미지의 객체를 분류하는 것 뿐만 아니라 위치까지 감지 (단, 하나의 객체만)

✔2. Output은 Bounding Box의 좌표인 [P_c, bx, by, bh, bw, C1, C2,.. Cn]이다. (bx, by는 중심의 x, y좌표)



❓ Multi-Label인데 왜 Detection을 안썼을까?

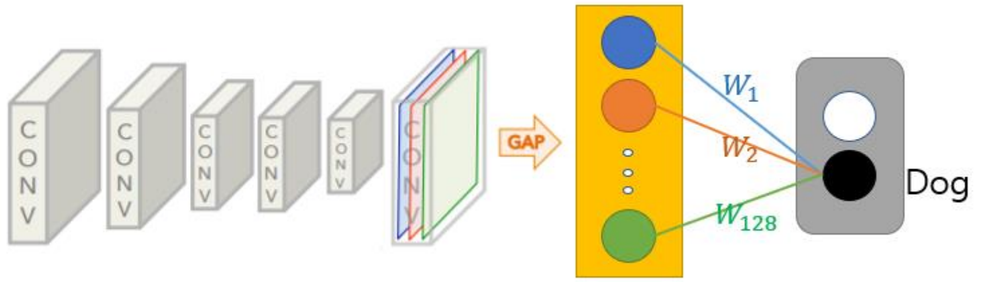
❗ training코드를 찾아보니 threshold를 구하기 위해서 validation과정에서 threshold를 0-1사이의 10000개의 실수로 나누어서 제일 f1 score이 높은 threshold로 선택

#1-3-1. Learning Deep Features For Discriminative Localization

⚠Bounding Box정보가 주어지지 않고 label만 주어질 수 있다.

☑ Class Activation Map을 사용하자

✔CAM (Class Activation Map)
: Convolution + GAP + Softmax



✔각 class별로 같은 이미지를 인식할 때에 다른 곳에 집중하는 결과를 보임
-> 이는 사람이 어떤 물체가 무엇인지 판단할 때 각 물체별 특성을 보고 판단하는 것과 같은 맥락

✔Classification으로 학습 시킨 모델 + threshold를 사용해 bbox검출의 성능이 좋음

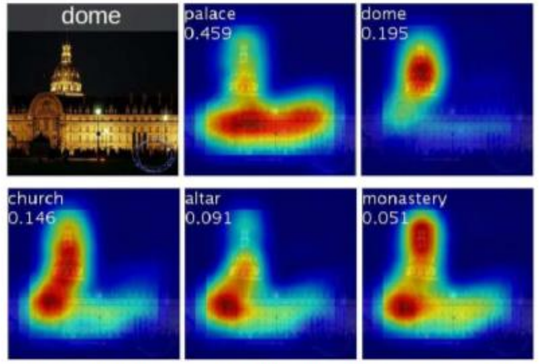


Figure 4. Examples of the CAMs generated from the top 5 predicted categories for the given image with ground-truth as dome. The predicted class and its score are shown above each class activation map. We observe that the highlighted regions vary across predicted classes e.g., *dome* activates the upper round part while *palace* activates the lower flat part of the compound.

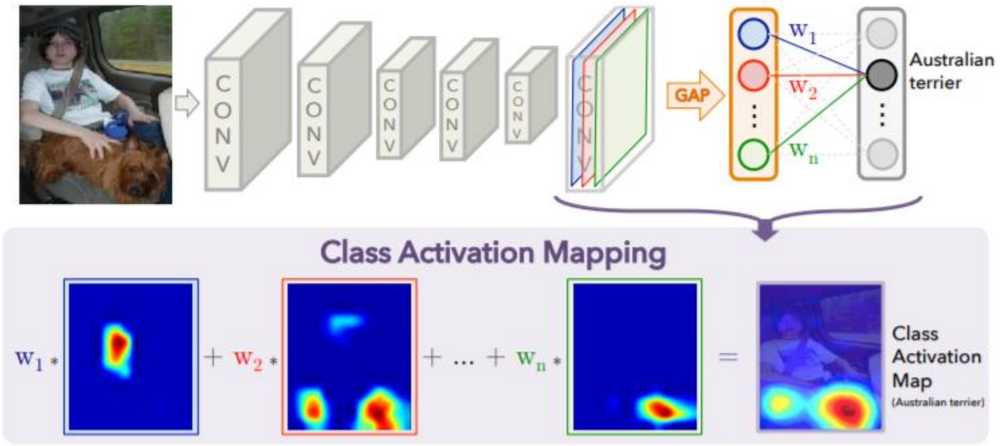
EWHA EURON

#1-3-2. GAP vs GMP

☑ GAP하나만을 추가했는데 localization, 즉 이미지의 위치까지 검출이 가능

✔보통 classification에서 convolution층의 마지막 출력이 (width, height, channel)이고 channel은 예측해야 하는 라벨의 개수

✔Pooling의 목적은 너무 많은 parameter로 인한 overfitting방지를 위한 차원 축소



✔기존의 fully connected layer과 달리 공간적 정보를 잃지 않음

✔GAP 연산 이후의 back propagation에서 이미지의 특징을 더 잘 잡게 함 + Global Context

✔GMP는 가장 두드러지는 영역을 제외한 모든 이미지 영역에서 점수가 낮으면 고려를 전혀 안 하는 문제

EWHA EURON

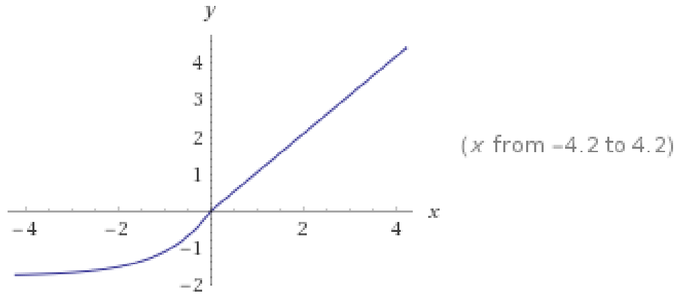
#2-1. GAPNet-PL 소개

👍 주어진 task에 한해서는 좋은 성과를 보임

✔ Low-level convolution 단계에서 이미지 feature인식을 위한 정보를 얻음

✔ ReLU + BN 대신에 SeLU 사용

1. 죽지 않는 gradient
2. Batch Norm과 함께 사용 되어도 빨리 학습



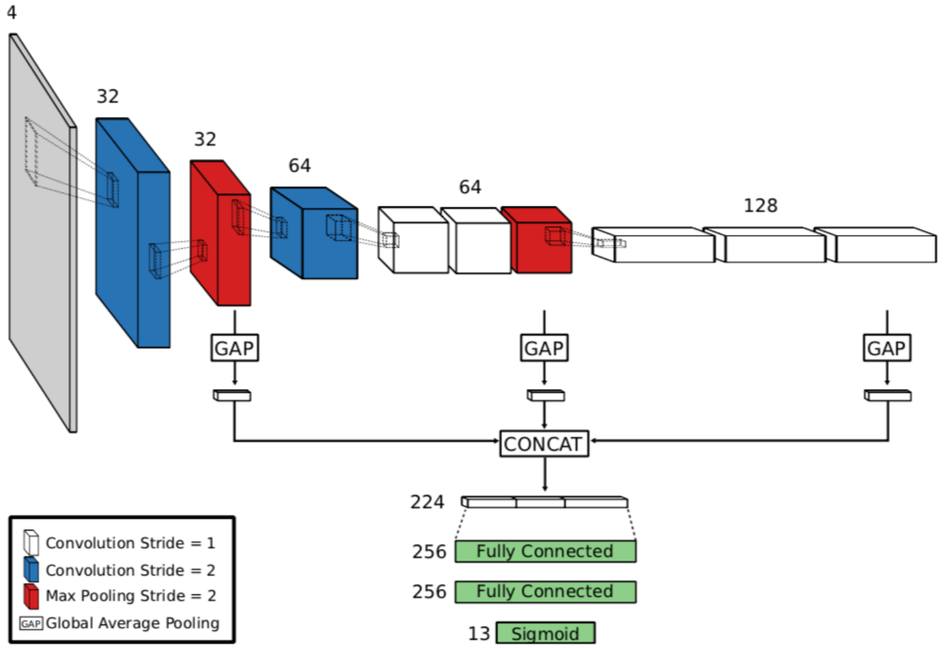


Figure 4: GapNet-PL architecture

$$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

#2-2. GAPNet-PL 적용

👍 Ability For **Multiscale**

- ✔ 5-fold GAP-Net Ensemble
- ✔ GAP layer이전에 SE layer을 추가해 성능 향상
- ✔ **Weighted BCE (pre-training)**
- ✔ Cosine Annealing LR Scheduler
- ✔ Adam Optimizer
- ✔ **F1 loss (fine-tuning)**

❓ F1을 loss function으로?

❗ 일반적으로는 Gradient Update를 하는 것이 불가능한 것으로 알려져 있다. 그래서 modified macro soft f1-score을 사용해서 likelihood value를 continuous sum으로 연산해서 사용

```python
def f1_loss(y_true, y_pred):
    eps = 1e-10 ## zero-division을 방지하기 위한 값
    tp = torch.sum(y_true * y_pred, dim = 0, keepdim = False, dtype = float) ## true-positive
    tn = torch.sum((1-y_true)*(1-y_pred), dim = 0, keepdim = False, dtype = float) ## true-negative
    fp = torch.sum((1-y_true) * y_pred, dim = 0, keepdim = False, dtype = float)) ## false-positive
    fn = torch.sum(y_true * (1-y_pred), dim = 0, keepdim = False, dtype = float) ## false-negative

    p = tp / (tp + fp + eps); r = tp/(tp+fn+eps);
    f1 = 2*p*r / (p+r+eps)
    f1 = torch.where(torch.isnan(f1), torch.zeros_like(f1), f1) ## 조건, 만족 할때 아닐 경우
    return 1-torch.sum(f1)
```

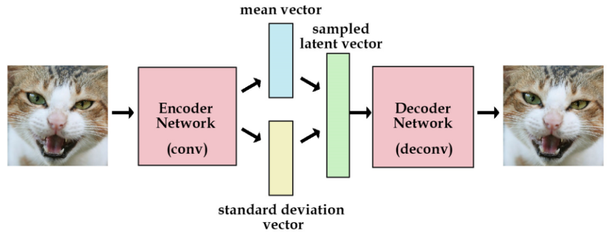
$$F_1 = \left(\frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} \right) = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

EWHA EURON

#3-1. AutoEncoder

✔ Reconstruction Loss + KL Divergence Regularizer

✔ 사실 MLP와 동일하지만 입력 channel# = 출력 channel #이라는 차이가 존재



$$L_i(\phi, \theta, x_i) = \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p_\theta(x_i|z)\right)\right]}_{\textbf{Reconstruction Error}} + \underbrace{KL\left(q_\phi(z|x_i)||p(z)\right)}_{\textbf{Regularization}}$$

? 언제 AutoEncoder을 사용?

! Input Data의 Feature을 추출할 때 자주 사용
Pre-Training을 할 때 많이 사용 (가중치 초기화에 용이)
Denoising, 혹은 Anomaly Detection에 사용
GAN보다 목적하고자 하는 바가 분명해서 학습이 잘된다고 함

EWHA EURON

#3-2-1. AutoEncoder 응용

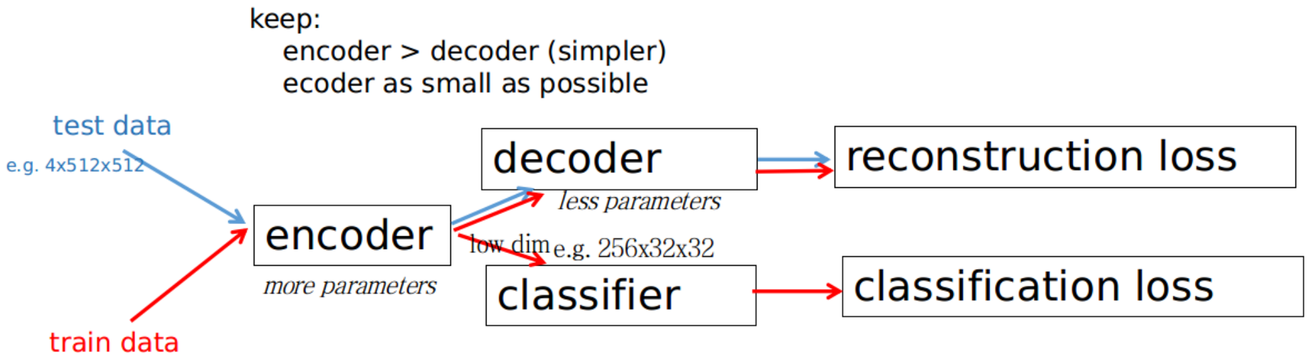
✔Encoder : ResNet-34
✔Decoder : 6개의 block

```
class Decoder(nn.Module):
    def __init__(self, blocks = 6, start_filters = 512):
        super().__init__()
        self.blocks = nn.ModuleList([
            nn.Conv2D(start_filters // (2**i), 3, 3),
            nn.ReLU(inplace = False),
            nn.Upsample(2)
        ] for i in range(blocks))
    def forward(self, x):
        for i, l in enumerate(self.blocks):
            x = self.blocks[l](x)
        return x
```

1.인코더로 먼저 모델이 시작을 하고,
AutoEncoder처럼 인코더는 디코더와 연결이 되어 학습

2.encoder에 classification head가 연결이 되어 Binary Cross Entropy + F1으로 훈련

✔ 이렇게 decoder을 따로 학습 시킴으로서 encoder이 이미지를 reconstruct할 수 있을 만큼 미세한 feature을 잘 잡을 수 있게 됨



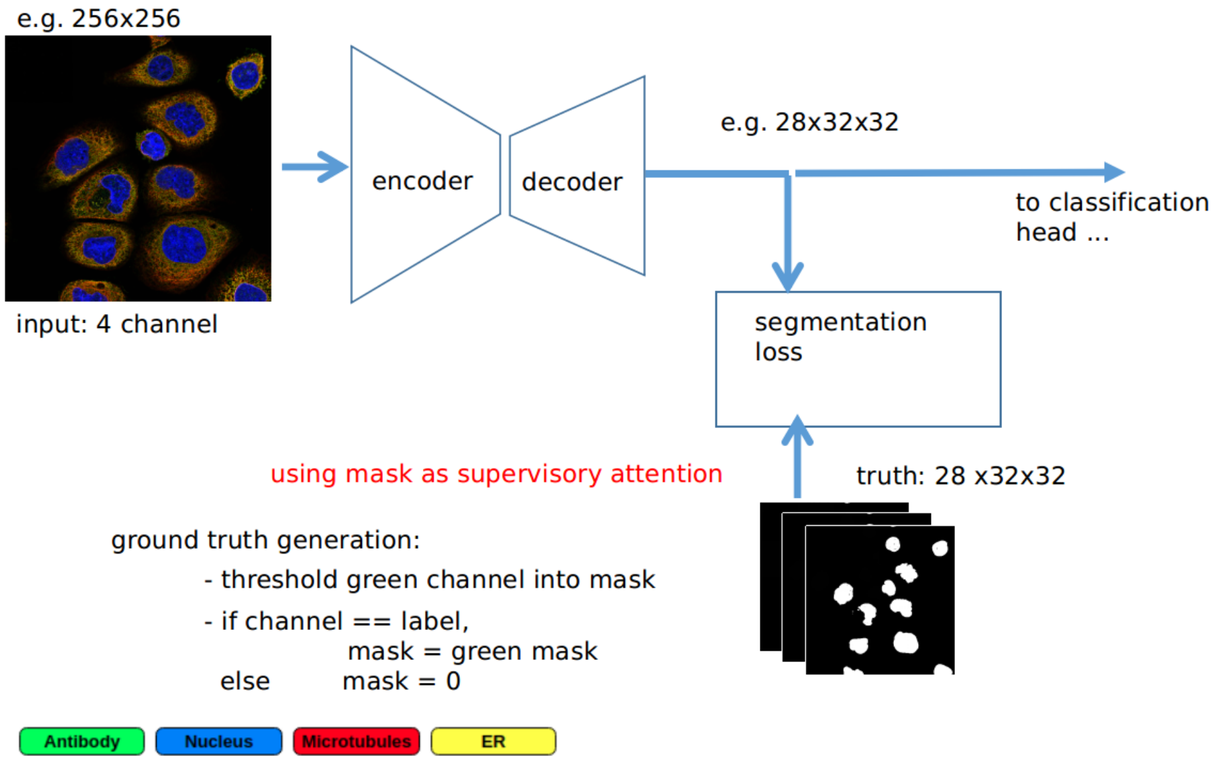
#3-2-2. AutoEncoder 응용

✔그냥 classification loss만 사용했을 때 Network output을 시각화 한 결과 f1 score이 높아도 경계선이 분명하지 않음

✔ 총 28개의 라벨로 예측 -> segmentation output에서 정답 라벨인 것의 채널에서 threshold이상의 score을 해당 픽셀로 갖는다면 그 부분은 mask로 예측

✔앞선 GAPNet과 ensemble진행

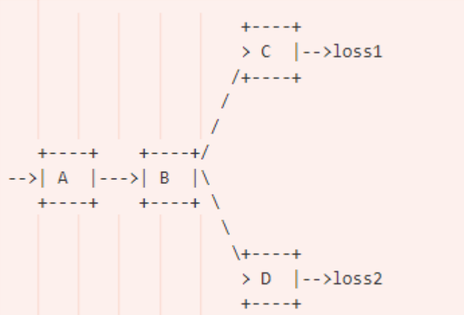
✔연산량이 많아서 최대한 decoder은 적은 파라미터 수로 구현



"segmentation loss works like a regulariser that ensures that that activations of the 32x32x128 layer are 'nice'."

#3-3. Dual Loss 적용

✔ Reconstruction Loss : 가중치 10.0 (MSE Loss사용)
✔ Classification Loss : 가중치 1.0 (F1 Loss + BCE Loss 사용)




```
x = Dense(28, name='b17_d3')(x)
mpx = MaxPooling2D((2, 2), strides=(2, 2), name='b17_b6_o')(x) ## Encoder Output

img_out = decoder_block(mpx,7)
img_out = Dense(3,activation='relu', name='img_out')(img_out) ## Decoder Output -> MSE Loss

x = Activation('sigmoid', name='predictions')(x) ## Classification -> Weighted Cross Entropy Loss
```

✔ Tf로 구현해서 multi output classification을 사용함
-> 위의 loss의 역전파는 A, B, C에 아래는 A, B, D의 parameter update를 함

❓ 따로 손실을 계산하면 pytorch의 경우 retain graph 에러가 계속 발생
-> keras에서는 최종 loss는 loss weight를 반영한 손실함수의 합으로 계산 (tensorflow fit_generator)

EWHA EURON

#4-1. Tricks Used

✔ 희귀한 클래스의 경우에는 이미지 데이터도 적음 -> 도메인 지식을 활용하거나 수동으로 segmentation처리 등을 해주면 학습에 도움

✔ 다양한 이미지 크기 -> train set에서 (1024, 1024)가 이미지 라벨이 B라면 test set에서 동일한 이미지의 크기를 갖는 경우에 라벨도 동일할 확률이 큼

✔ Balanced Sampling

✔ Multi-label 이미지에 대해서 예측의 정확도가 낮을 확률이 높은데, 이에 대비하기 위해서는 역시나 도메인 지식을 활용해서 label의 co-occurence를 도입하거나 rnn, lstm, 혹은 network stacking method를 사용

✔ 앙상블을 했을 때의 threshold를 학습

EWHA EURON