컴퓨터 공학 특강

FaceNet

A Unified Embedding for Face Recognition and Clustering

한양대학교 HANYANG UNIVERSITY

인공지능 연구실
김지성, 조건희

Introduction

FaceNet

Identification

FaceNet

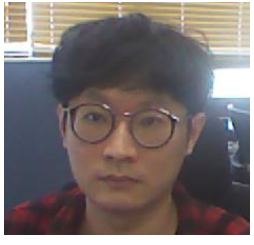
array([-0.07440512, 0.13833548, 0.01550988, -0.04143589, -0.11708137,
0.01741652, -0.09219746, -0.0411015 , 0.12901564, -0.03510666,
0.26017255, -0.01175268, -0.23214489, -0.10993981, -0.06433399,
0.14533579, -0.18374404, -0.0706907 , 0.01960303, 0.03927957,
0.17917837, 0.10840175, 0.06728961, 0.01386871, -0.0921066 ,
-0.32157087, -0.06875613, -0.11010353, 0.02105946, -0.09806988,
-0.09223023, -0.01714757, -0.16107245, -0.04865564, 0.05062422,
0.04144309, -0.03346116, -0.03011878, 0.15263124, 0.01069992,
-0.23673587, 0.05740198, 0.03050802, 0.23846291, 0.20607698,
0.01160336, 0.00998583, -0.15661725, 0.09741502, -0.11773828,
0.08130169, 0.15210505, 0.14222656, 0.02321066, 0.00600557,
-0.07693202, -0.02959017, 0.15295519, -0.13042481, 0.03233573,
0.1088061 , -0.05199346, -0.01501178, -0.08011279, 0.17588341,
0.02202863, -0.12124902, -0.26303217, 0.06608438, -0.123976 ,
-0.14779539, 0.1516934 , -0.16154116, -0.1716671 , -0.25228465,
0.01856503, 0.36660314, 0.04856375, -0.18907131, 0.05604936,
-0.05154709, -0.04398936, 0.08519959, 0.14640823, 0.00993889,
0.02739849, -0.11102622, 0.01848426, 0.23328975, -0.11351117,
-0.04641433, 0.22538319, -0.00492372, 0.10828383, 0.02823116,
0.02502055, -0.02946196, 0.0702677 , -0.09549975, -0.0334047 ,
0.00996118, -0.08729131, -0.04567208, 0.09973253, -0.14260028,
0.10422572, -0.00379006, 0.05201333, 0.02785442, -0.05933321,
-0.09983464, -0.02418252, 0.13822252, -0.24259827, 0.2536535 ,
0.12104575, 0.14091188, 0.07011457, 0.10865125, 0.04752614,
0.02150964, -0.04581762, -0.23496597, -0.01055925, 0.11252803,
-0.05433004, 0.10194337, -0.02596316])

128d 벡터

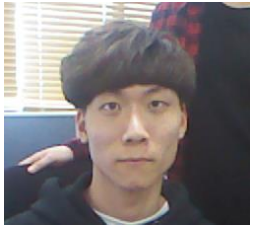
Introduction

목표

A인물의 사진1



A인물의 사진2



B인물의 사진1

FaceNet

FaceNet

FaceNet

128d 벡터

array([-0.07440512, 0.13833548, 0.01550988, -0.04143589, -0.11708137,
0.01741652, -0.09219746, -0.0411015, 0.12901564, -0.03510666,
0.26017255, -0.01175268, -0.23214489, -0.10993981, -0.06433399,
0.14533579, -0.18374404, -0.0706907, 0.01960303, 0.0927957,
0.17917837, 0.10840175, 0.06728961, 0.01386871, -0.0921066,
-0.32157087, -0.06875613, -0.11010353, 0.02105946, -0.09806988,
-0.09223023, -0.01714757, -0.16107245, -0.04865564, 0.05062422,
0.04144309, -0.03346116, -0.03011878, 0.15263124, 0.01069992,
-0.23673587, 0.05740198, 0.03050802, 0.23846291, 0.20607698,
0.01160336, 0.00998583, -0.15661725, 0.09741502, -0.11773828,
0.08130169, 0.15210505, 0.14222656, 0.02321066, 0.00600557,
-0.07693202, -0.02959017, 0.15295519, -0.13042481, 0.03233573,
0.1088061, -0.05199346, -0.01501178, -0.08011279, 0.17588341,
0.02202863, -0.12124902, -0.26303217, 0.06608438, -0.123976 ,
-0.14779539, 0.1516934, -0.16154116, -0.1716671 , -0.25228465,
0.01856503, 0.36660314, 0.04856375, -0.18907131, 0.05604936,
-0.05154709, -0.04398936, 0.08519959, 0.14640823, 0.00993889,
0.02739849, -0.11102622, 0.01848426, 0.23328975, -0.11351117,
-0.04641433, 0.22538319, -0.00492372, 0.10828383, 0.02823116,
0.02502055, -0.02946196, 0.0702677 , -0.09549975, -0.0334047 ,
0.00996118, -0.08729131, -0.04567208, 0.09973253, -0.14260028,
0.10422572, -0.00379006, 0.05201333, 0.02785442, -0.05933321,
-0.09983464, -0.02418252, 0.13822252, -0.24259827, 0.2536535 ,
0.12104575, 0.14091188, 0.07011457, 0.10865125, 0.04752614,
0.02150964, -0.04581762, -0.23496597, -0.01055925, 0.11252803,
-0.05433004, 0.10194337, -0.02596316])

array([-0.08902847, 0.14762324, 0.05718813, -0.05012994, -0.09676401,
0.02028476, -0.08940892, -0.0386399, 0.10385386, -0.031321 ,
0.2729257 , -0.02359607, -0.23816103, -0.09732635, -0.03333118,
0.1333721 , -0.19538365, -0.07668579, 0.00275577, 0.03931012,
0.15097558, 0.08741103, 0.0678171, 0.01446525, -0.09825172,
-0.33090472, -0.06750867, -0.01570621, 0.01662679, -0.11775655,
-0.10916033, -0.04655562, -0.1787585, -0.05166516, 0.04875493,
0.0426253, -0.0591871, -0.04663718, 0.16380328, 0.00791238,
-0.22012679, 0.04296945, 0.04917528, 0.23990902, 0.21356051,
0.00606929, 0.00836444, -0.13134141, 0.09845343, -0.162596 ,
0.11051578, 0.15520622, 0.12857951, 0.05687075, 0.01556353,
-0.09330104, -0.04209765, 0.14811578, -0.10172325, 0.06637652,
0.11120091, -0.05348029, -0.02427355, -0.06907345, 0.17457779,
0.04438576, -0.13614309, -0.27176958, 0.04076207, -0.10192671,
-0.13134097, 0.16820309, -0.17274556, -0.17039801, -0.25199199,
0.0466072 , 0.38125145, 0.03306871, -0.19173753, 0.01305585,
-0.07093169, -0.05639979, 0.07696941, 0.15101001, 0.02203412,
0.0188837 , -0.10816664, 0.04892536, 0.23276457, -0.10933174,
-0.06033619, 0.2151441 , -0.01290991, 0.10794014, 0.02028765,
0.05325544, -0.03414371, 0.055746 , -0.10889978, -0.02816046,
0.01456824, -0.07452301, -0.06755616, 0.11638117, -0.13528842,
0.10086828, 0.00977207, 0.03836292, 0.02850841, -0.05801099,
-0.10640397, -0.03940248, 0.12043379, -0.25706992, 0.24780291,
0.14584672, 0.13685006, 0.05027201, 0.09039105, 0.03249723,
0.0475209, -0.02310574, -0.21513188, -0.00798578, 0.09137717,
-0.01307906, 0.06952387, -0.03653527])

array([-0.07440512, 0.13833548, 0.01550988, -0.04143589, -0.11708137,
0.01741652, -0.09219746, -0.0411015, 0.12901564, -0.03510666,
0.26017255, -0.01175268, -0.23214489, -0.10993981, -0.06433399,
0.14533579, -0.18374404, -0.0706907, 0.01960303, 0.0927957,
0.17917837, 0.10840175, 0.06728961, 0.01386871, -0.0921066 ,
-0.32157087, -0.06875613, -0.11010353, 0.02105946, -0.09806988,
-0.09223023, -0.01714757, -0.16107245, -0.04865564, 0.05062422,
0.04144309, -0.03346116, -0.03011878, 0.15263124, 0.01069992,
-0.23673587, 0.05740198, 0.03050802, 0.23846291, 0.20607698,
0.01160336, 0.00998583, -0.15661725, 0.09741502, -0.11773828,
0.08130169, 0.15210505, 0.14222656, 0.02321066, 0.00600557,
-0.07693202, -0.02959017, 0.15295519, -0.13042481, 0.03233573,
0.1088061 , -0.05199346, -0.01501178, -0.08011279, 0.17588341,
0.02202863, -0.12124902, -0.26303217, 0.06608438, -0.123976 ,
-0.14779539, 0.1516934 , -0.16154116, -0.1716671 , -0.25228465,
0.01856503, 0.36660314, 0.04856375, -0.18907131, 0.05604936,
-0.05154709, -0.04398936, 0.08519959, 0.14640823, 0.00993889,
0.02739849, -0.11102622, 0.01848426, 0.23328975, -0.11351117,
-0.04641433, 0.22538319, -0.00492372, 0.10828383, 0.02823116,
0.02502055, -0.02946196, 0.0702677 , -0.09549975, -0.0334047 ,
0.00996118, -0.08729131, -0.04567208, 0.09973253, -0.14260028,
0.10422572, -0.00379006, 0.05201333, 0.02785442, -0.05933321,
-0.09983464, -0.02418252, 0.13822252, -0.24259827, 0.2536535 ,
0.12104575, 0.14091188, 0.07011457, 0.10865125, 0.04752614,
0.02150964, -0.04581762, -0.23496597, -0.01055925, 0.11252803,
-0.05433004, 0.10194337, -0.02596316])

Euclidean distance

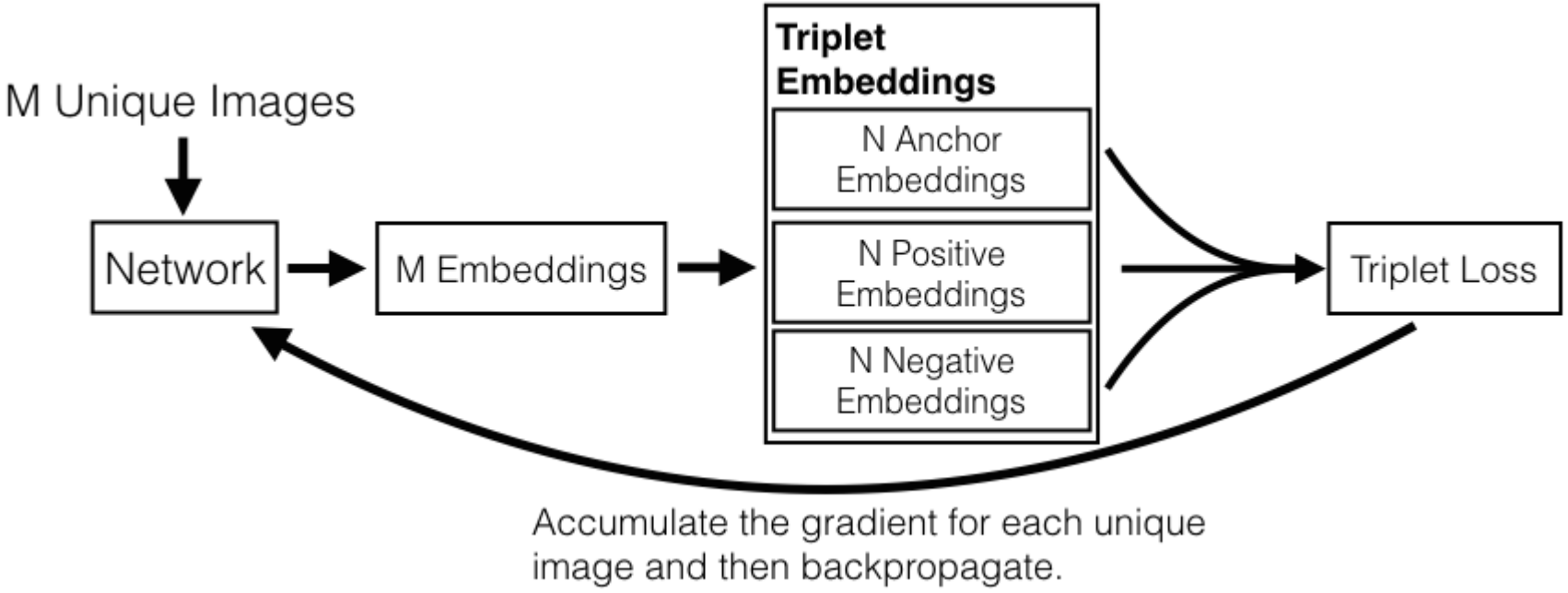
0.78

1.27

1.33

Introduction

구조



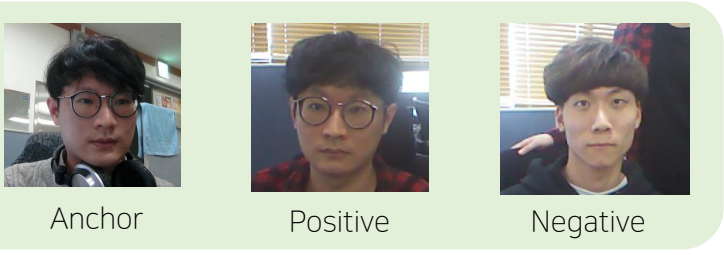
FaceNet Architecture

Method

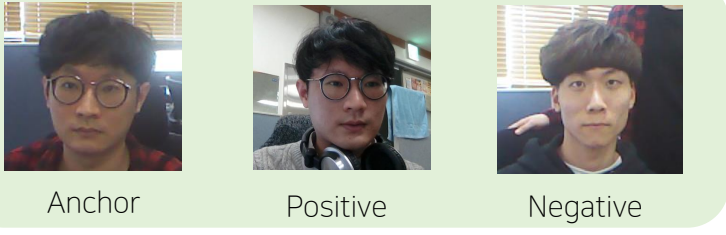
FaceNet - Triplet

Triplet : 세 개의 데이터

- Anchor(x_i^a) : 기준 얼굴의 벡터
- Positive(x_i^p) : 기준과 같은 인물의 얼굴의 벡터
- Negative(x_i^n) : 기준과 다른 인물의 얼굴의 벡터

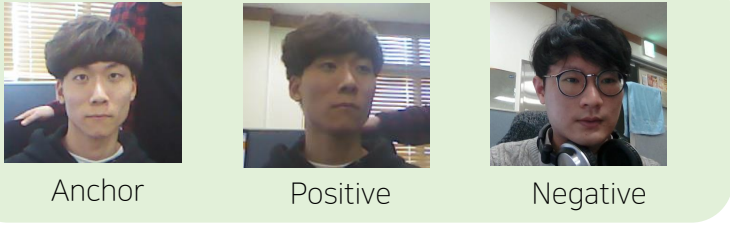


Triplet 1



Triplet 2

...



Triplet N

Method

FaceNet - Triplet

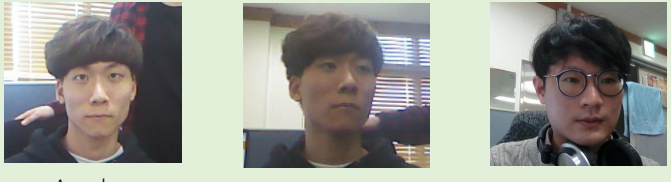
Triplet : 세 개의 데이터

- Anchor(x_i^a) : 기준 얼굴의 벡터
- Positive(x_i^p) : 기준과 같은 인물의 얼굴의 벡터
- Negative(x_i^n) : 기준과 다른 인물의 얼굴의 벡터

Triplet 1

Triplet 2

...



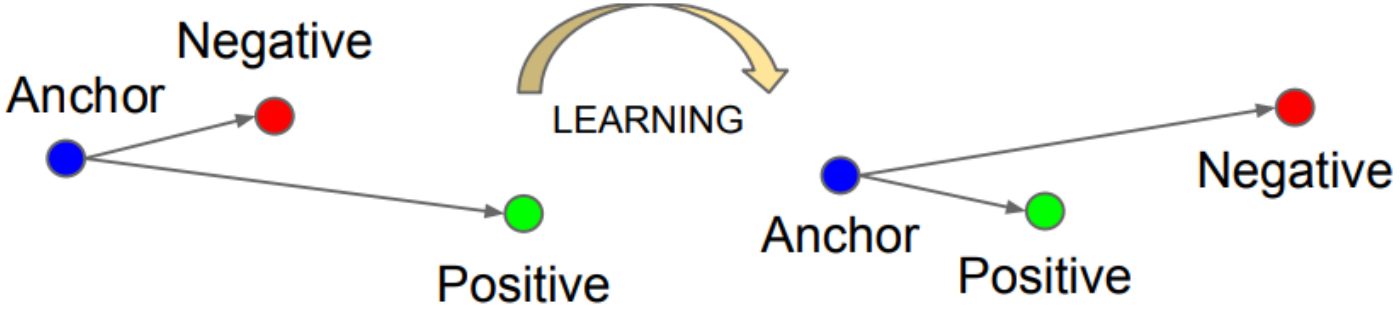
Triplet N

Method

FaceNet - Triplet

Triplet : 세 개의 데이터

- Anchor(x_i^a) : 기준 얼굴의 벡터
- Positive(x_i^p) : 기준과 같은 인물의 얼굴의 벡터
- Negative(x_i^n) : 기준과 다른 인물의 얼굴의 벡터



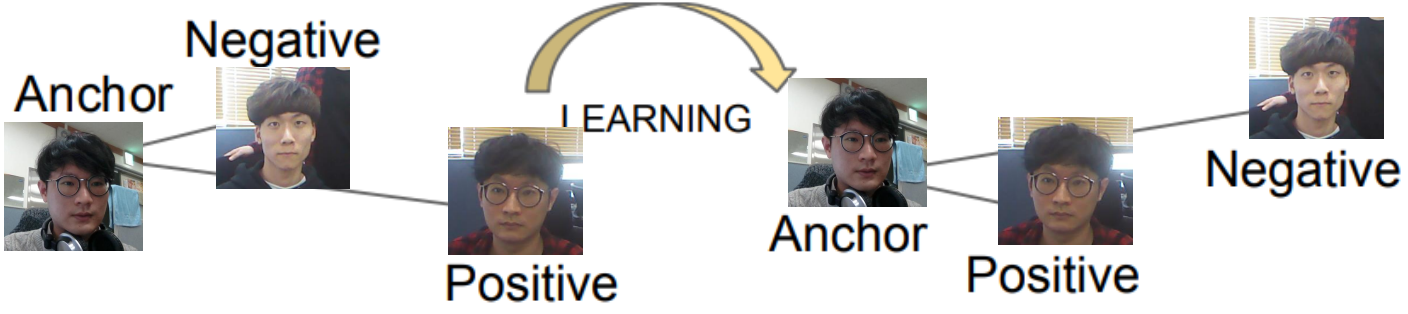
기준 얼굴과 같은 얼굴은 가깝도록, 기준 얼굴과 다른 얼굴은 멀도록

Method

FaceNet - Triplet

Triplet : 세 개의 데이터

- Anchor(x_i^a) : 기준 얼굴의 벡터
- Positive(x_i^p) : 기준과 같은 인물의 얼굴의 벡터
- Negative(x_i^n) : 기준과 다른 인물의 얼굴의 벡터



기준 얼굴과 같은 얼굴은 가깝도록, 기준 얼굴과 다른 얼굴은 멀도록

Method

Triplet - Loss

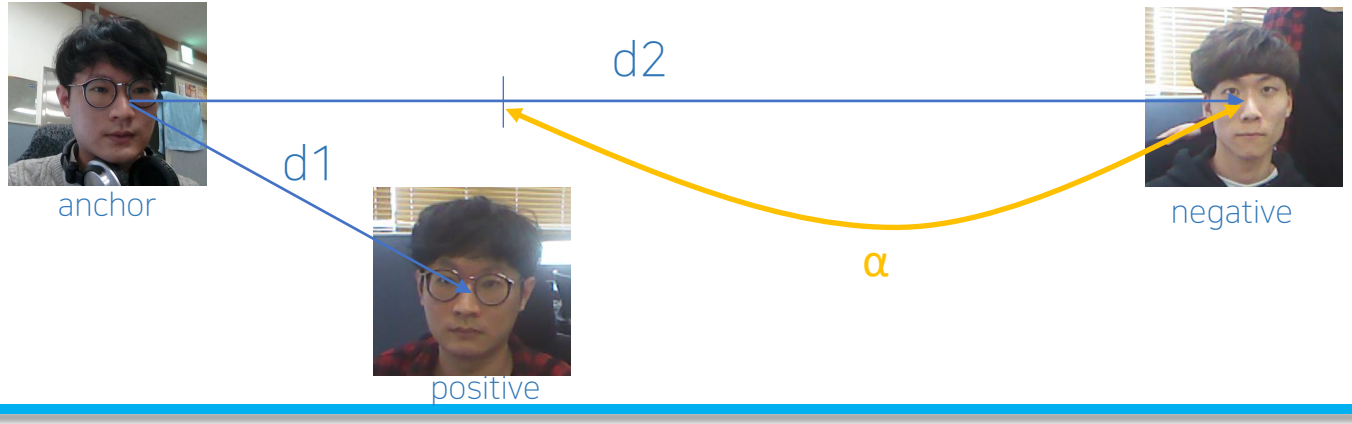
d1 d2

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \,, \tag{1}$$

- x : 이미지
- $f(x)$: 임베딩 함수
- α : 마진

- x_i^a : 기준 얼굴(anchor) 이미지
- x_i^p : 기준과 같은 인물(positive)의 얼굴 이미지
- x_i^n : 기준과 다른 인물(negative)의 얼굴 이미지

Anchor와 Negative의 제곱거리가 Anchor와 Positive의 제곱거리보다 α 만큼 떨어져 있고 싶다!



Method

Triplet - Loss

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \,, \tag{1}$$

- x : 이미지
- $f(x)$: 임베딩 함수
- α : 마진

- x_i^a : 기준 얼굴(anchor) 이미지
- x_i^p : 기준과 같은 인물(positive)의 얼굴 이미지
- x_i^n : 기준과 다른 인물(negative)의 얼굴 이미지

Anchor와 Negative의 제곱거리가 Anchor와 Positive의 제곱거리보다 α 만큼 떨어져 있고 싶다!

$$\text{LOSS} = \sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]_+ . \tag{2}$$

$$= \sum_i^N \text{Max}(\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha, 0)$$

Method

Triplet – Triplet Selection

각 Triplet이 학습하는데 미치는 영향은 전부 다르다.
학습에 미치는 영향이 큰 Triplet들을 학습하는 것이 수렴속도를 높인다.
따라서 어려운 문제를 골라서 해결하기 위해 Hard Negative, Hard Positive가 있는 Triplet 선택해야 한다.

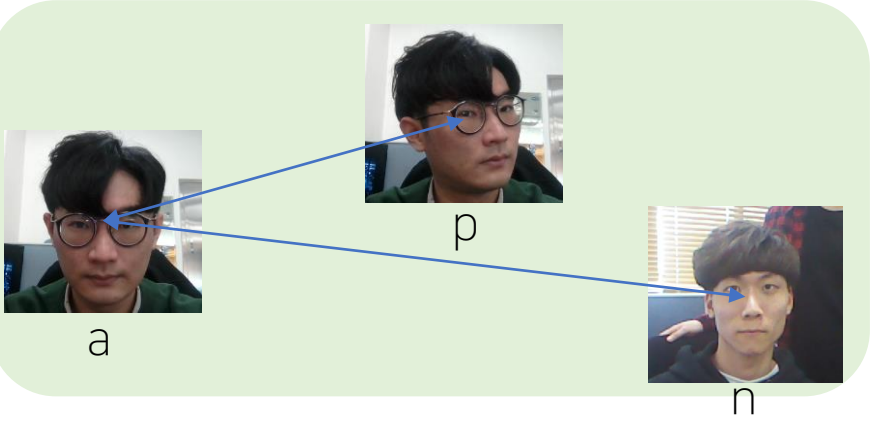
x_i^a Anchor가 주어졌을 때

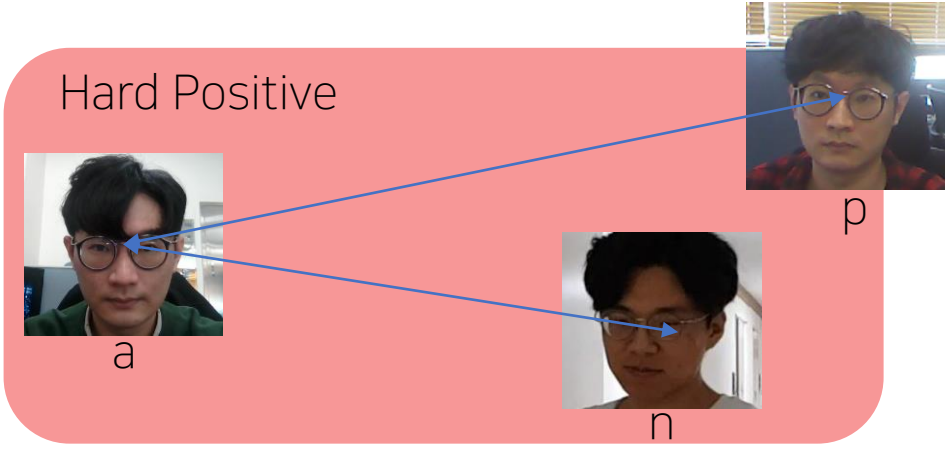
$\text{argmin}_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2$ Hard Negative : 가장 가까이 있는 Negative

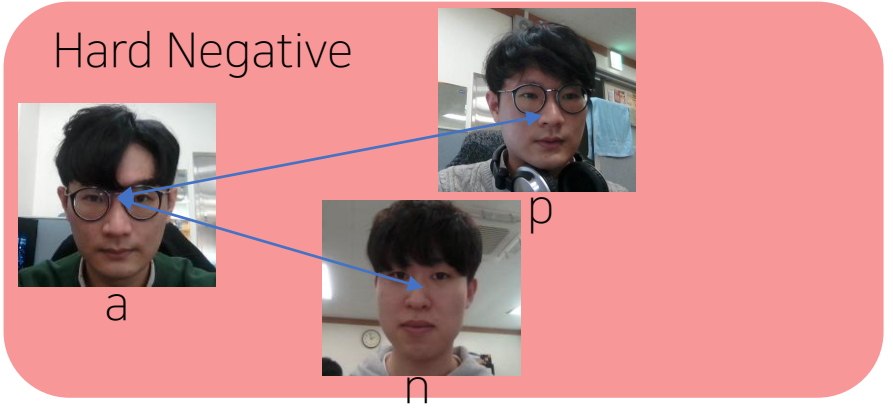
$\text{argmax}_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$ Hard Positive : 가장 멀리 있는 Positive

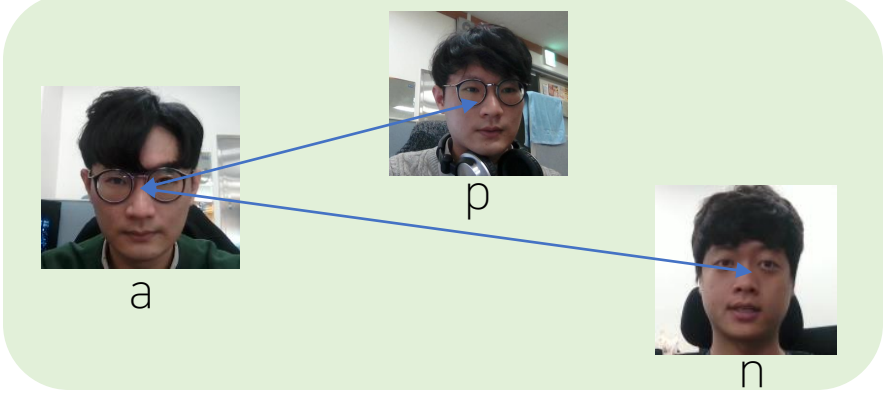
Method

Triplet – Triplet Selection







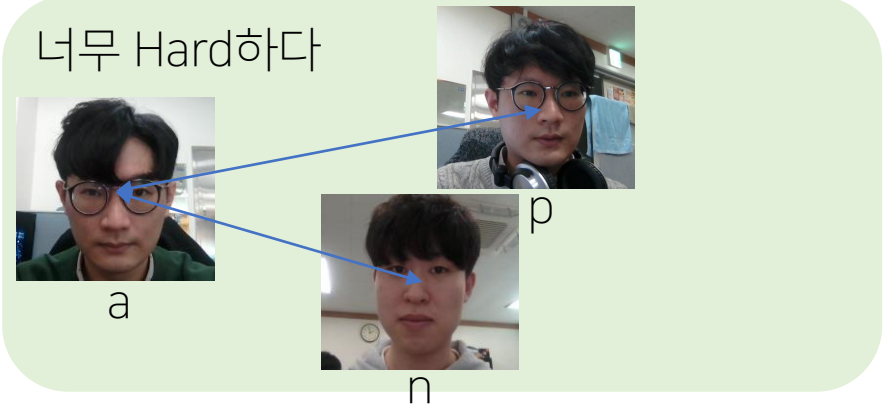


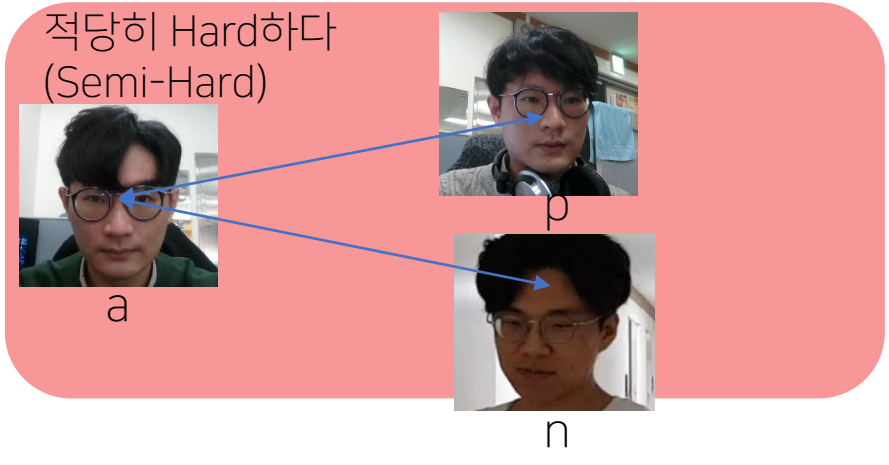
Method

Triplet – Triplet Selection(Detail)

- 가장 Hard한 negative를 선택하면 local-minima에 빠지기 쉽다
  - 학습 초반에 너무 어려운 문제를 해결하기 힘들다
  - Semi-hard negative를 선택해야함
    - Positive보다는 먼 negative 가 있는 Triplet을 선택
    - Positive와 거리가 비슷해서 Hard라고 말할 수 있다

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2 \ . \qquad (3)$$





Method

Triplet – Triplet Selection

두 명의 사람이 각 10장의 사진을 가질 때 가능한 Triplet의 수 :
Class x Anchor x Positive x Negative = 2 x 10 x 9 x 10 = 1800

매번 모든 triplet을 계산해서 Semi-Hard 한 문제를 찾는 것은 너무 비 효율적이다.

Mini Batch가 필요하다!

Method

Triplet – mini batch

이미지셋

랜덤으로 한 ID에서 40개

40개

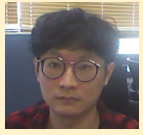
Negative faces 중 랜덤으로 1960개

1960개

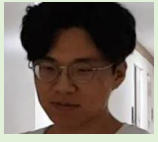
크기 2000의 미니배치

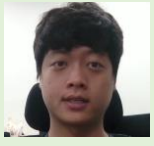
Method

Triplet – mini batch


40개

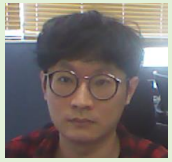





1960개

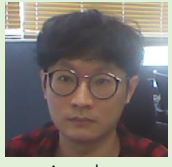
크기 2000의 미니배치

FaceNet


Anchor Positive Negative

Triplet 1

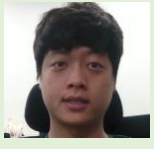


Anchor Positive Negative

Triplet 2

...



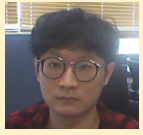


Anchor Positive Negative

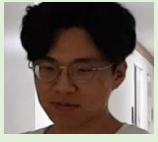
Triplet N

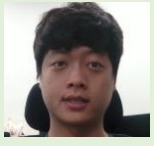
Method

Triplet – mini batch


40개

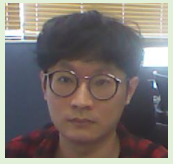





1960개

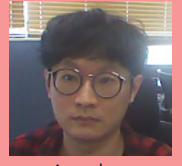
크기 2000의 미니배치

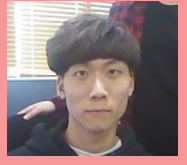
FaceNet


Anchor Positive Negative

Triplet 1



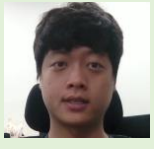


Anchor Positive Negative

Triplet 2

...



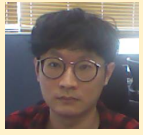


Anchor Positive Negative

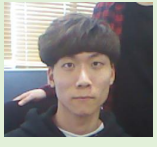
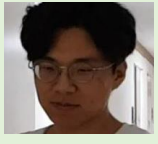
Triplet N

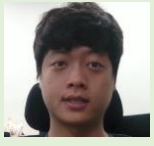
Method

Triplet – mini batch


40개

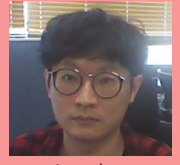



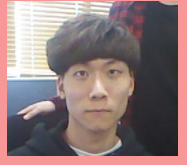


1960개

크기 2000의 미니배치

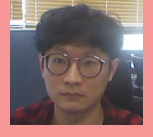
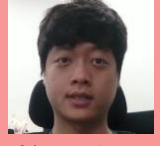
FaceNet





Anchor Positive Negative

Triplet 2

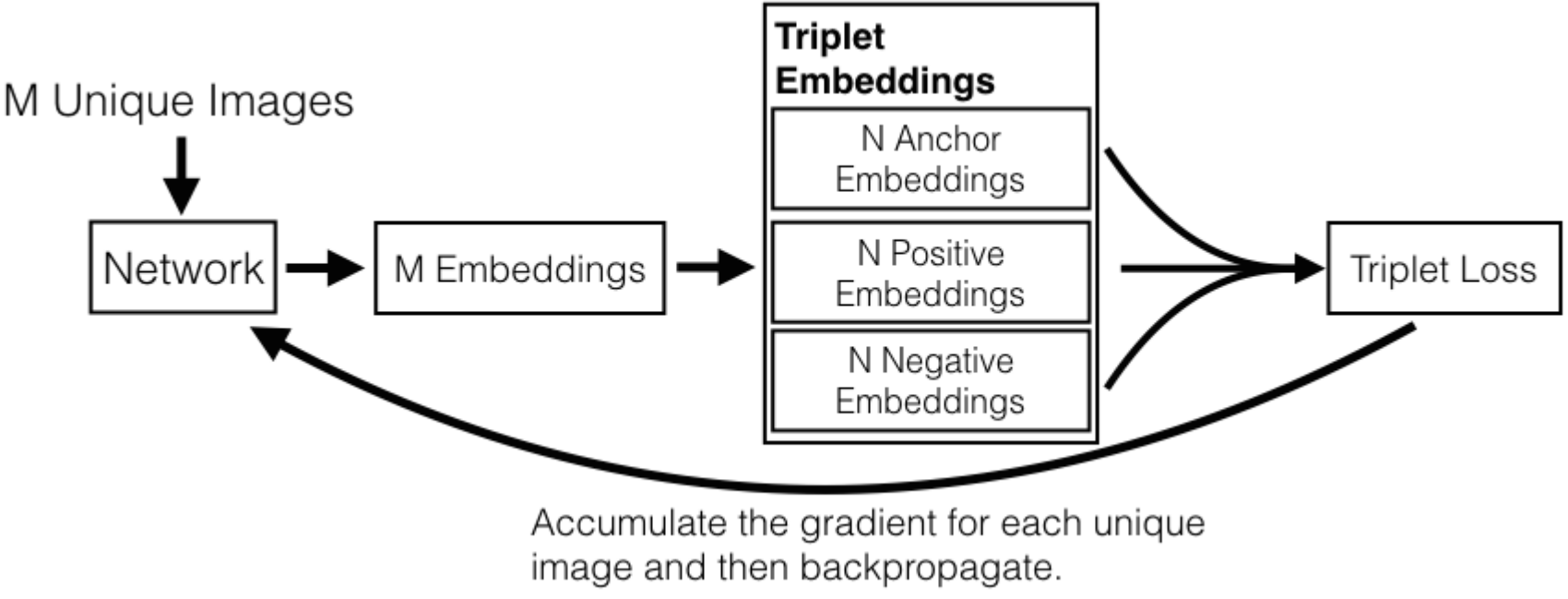




Anchor Positive Negative

Triplet N

Method

Triplet – Triplet Selection(Detail)



FaceNet Architecture

Experiments

실험에 사용된 CNN 모델

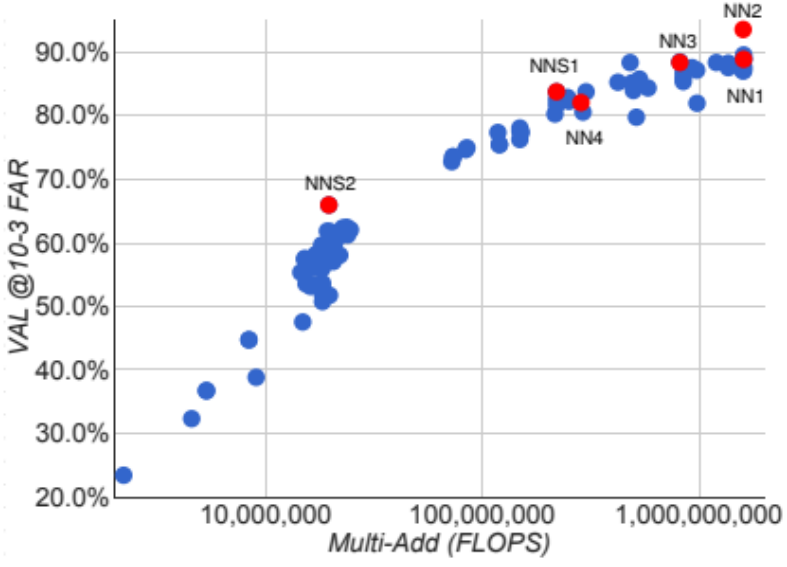
Comparison with the State of the Art (LFW Unrestricted Protocol)

No.	Method	# Training Images	# Networks	Accuracy
1	Fisher Vector Faces	-	-	93.10
2	DeepFace	4 M	3	97.35
3	DeepFace Fusion	500 M	5	98.37
4	DeepID-2,3	Full	200	99.47
5	FaceNet	200 M	1	98.87
6	FaceNet+ Alignment	200 M	1	99.63
7	VGG Face	2.6 M	1	98.95

Experiments

실험에 사용된 CNN 모델

architecture	VAL
NN1 (Zeiler&Fergus 220×220)	87.9% ± 1.9
NN2 (Inception 224×224)	89.4% ± 1.6
NN3 (Inception 160×160)	88.3% ± 1.7
NN4 (Inception 96×96)	82.0% ± 2.3
NNS1 (mini Inception 165×165)	82.4% ± 2.4
NNS2 (tiny Inception 140×116)	51.9% ± 2.9



FLOPS vs. Accuracy

Testset : Personal Photos

Evaluation

- VAL(d) : Validation Rate
  같은 사람을 같다고 예측한 비율
- FAR(d) : False Accept Rate
  다른 사람을 같다고 예측한 비율

Category1 : Zeiler & Fergus : ZF Net 기반 모델

- NN1
  - 1x1 convolution

Category 2 : GoogLeNet 기반 모델(Inception v1)

- NN2
  - NN1과 비교
  - 약 20배 적은 파라미터
  - 최대 5배 까지 적은 FLOPS
- NN2~4
  - 입력 크기 220x220, 160x160, 96x96
- NNS1~4
  - 모바일을 위한 경량 모델

Experiments

FLOPS vs. Accuracy

#pixels	val-rate
1,600	37.8%
6,400	79.5%
14,400	84.5%
25,600	85.7%
65,536	86.4%

이미지 크기에 따른 정확도

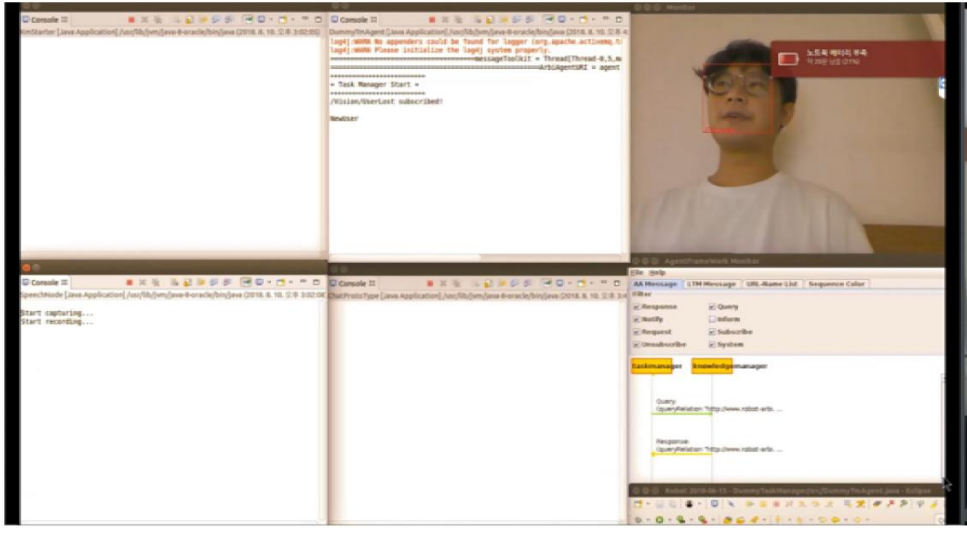
#dims	VAL
64	86.8% ± 1.7
128	87.9% ± 1.9
256	87.7% ± 1.9
512	85.6% ± 2.0

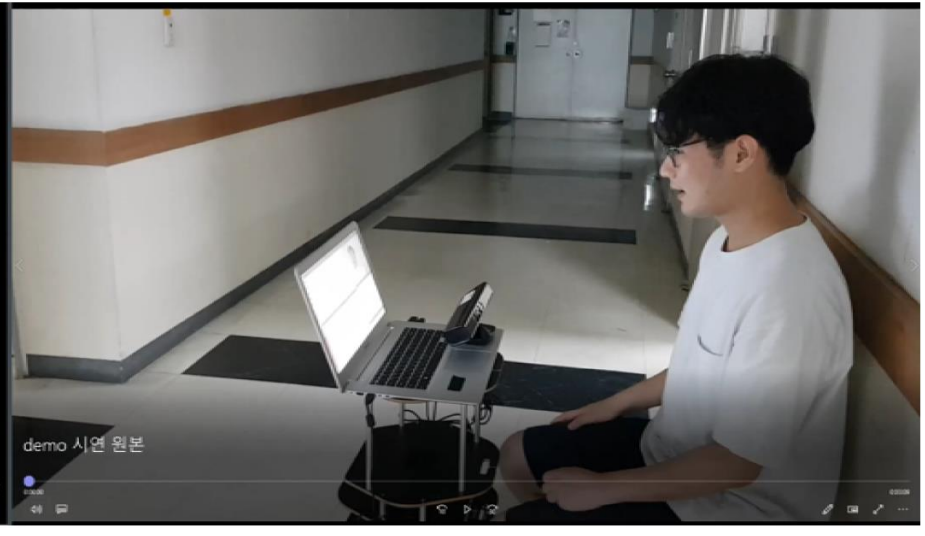
임베딩 벡터 차원에 따른 정확도

#training images	VAL
2,600,000	76.3%
26,000,000	85.1%
52,000,000	85.1%
260,000,000	86.2%

데이터셋 양에 따른 정확도

Experiments






감사합니다.