1-2. Face Landmark Detection

PFLD
(A Practical Facial Landmark Detector)

PFLD: A Practical Facial Landmark Detector

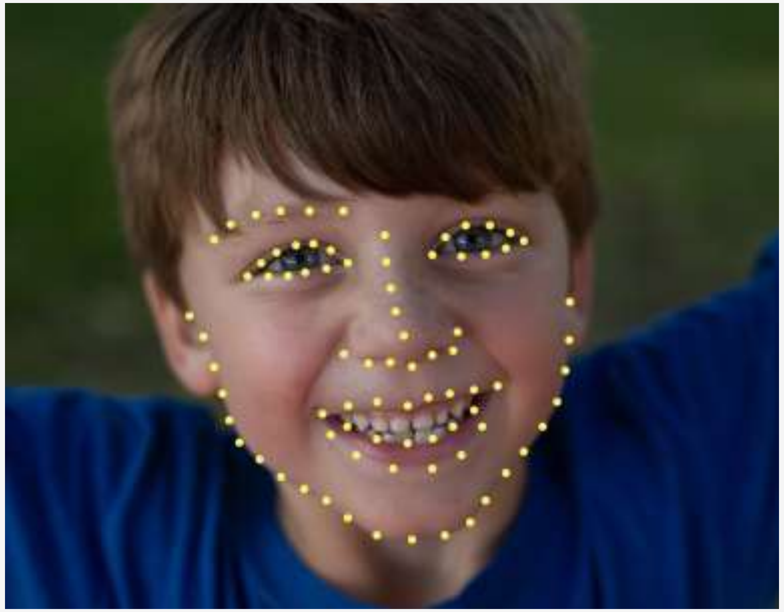
Xiaojie Guo[1], Siyuan Li[1], Jinke Yu[1], Jiawan Zhang[1], Jiayi Ma[2], Lin Ma[3], Wei Liu[3], and Haibin Ling[4]
[1]Tianjin University [2]Wuhan University [3]Tencent AI Lab [4]Temple University

References
https://arxiv.org/pdf/1902.10859.pdf

Face Landmark Detection이란?

Detecting and localizing specific points or landmarks on a face, such as the eyes, nose, mouth, and chin.

사람의 상태를 파악할 수 있다. (표정, 고개의 기울어짐 등)



References
(Left) https://prlabhotelshoe.tistory.com/4
(Middle) https://www.plugger.ai/blog/the-top-7-use-cases-for-facial-landmark-detection
(Right) http://blog.dlib.net/2018/01/correctly-mirroring-datasets.html
https://paperswithcode.com/task/facial-landmark-detection

CONTENT

Introduction

Introduction

Figure 1: Example faces with different poses, expressions, lightings, occlusions, and image qualities. The green markers are detected landmarks via our method. The processing speed achieves over 140 fps on an Android phone with Qualcomm ARM 845 processor.

Introduction

#1 Local Variation

Expression, local extreme lighting *(e.g., highlight and shading), and* ***occlusion*** *... Landmarks of some regions may* ***deviate*** *from their normal positions or even* ***disappear.***

#2 Global Variation

Pose *and* ***imaging quality*** *are two main factors globally affecting the appearance of faces in images*

#3 Data Imbalance

An available dataset exhibits an ***unequal distribution*** *between its classes/attributes.*

#4 Model Efficiency

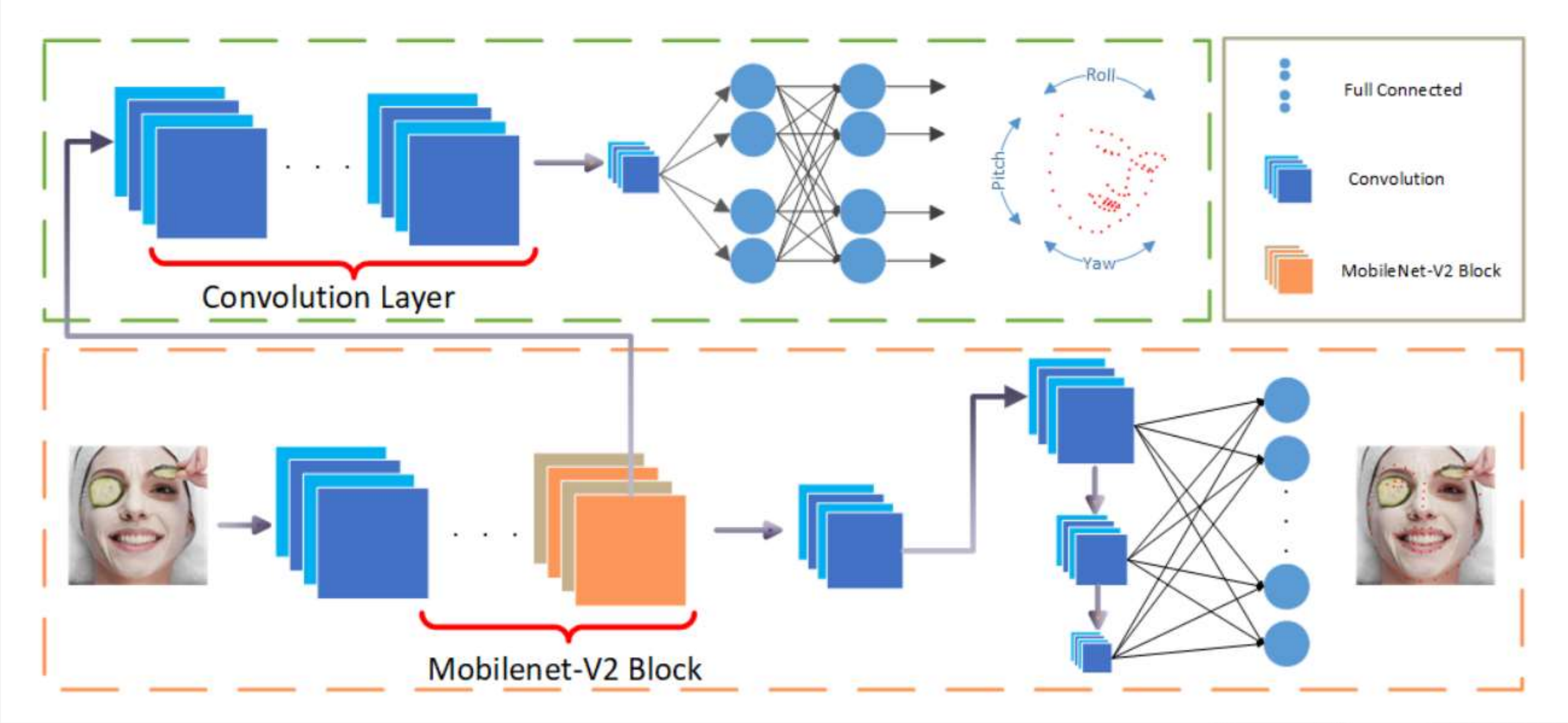
Another two constraints on applicability are ***model size*** *and* ***computing requirement****.*

Introduction

- *This paper investigates a neat model with promising detection accuracy under* ***wild environments*** *(e.g., unconstrained pose, expression, lighting, and occlusion conditions) and* ***super real-time speed*** *on a* ***mobile device.***
- *More concretely, we customize an* ***endto-end single stage network*** *associated with acceleration techniques.*
- *During the* ***training phase****, for each sample,* ***rotation information*** *is estimated for geometrically regularizing landmark localization, which is then NOT involved in the testing phase.*
- *A novel loss is designed to, besides considering the* ***geometrical regularization****, mitigate the issue of* ***data imbalance*** *by adjusting weights of samples to different states, such as large pose, extreme lighting, and occlusion, in the training set.*
- *Extensive experiments are conducted to demonstrate the efficacy of our design and reveal its superior performance over* ***state-ofthe-art*** *alternatives on widely-adopted* ***challenging benchmarks****.*
- *Our model can be merely* ***2.1Mb*** *of size and reach over* ***140 fps*** *per face on a* ***mobile phone*** *(****Qualcomm ARM 845 processor****) with high precision, making it attractive for large-scale or real-time applications.*

Proposed

PFLD Architecture



Loss function

$$\mathcal{L} := \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N}\boxed{\gamma_n\boxed{\|\mathbf{d}_n^m\|}},$$

$$\mathcal{L} := \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N}\boxed{\left(\sum_{c=1}^{C}\omega_n^c\sum_{k=1}^{K}(1-\cos\theta_n^k)\right)}\|\mathbf{d}_n^m\|_2^2.$$

M : Input image
N : face landmark
C : attribute class
K : 3d pose

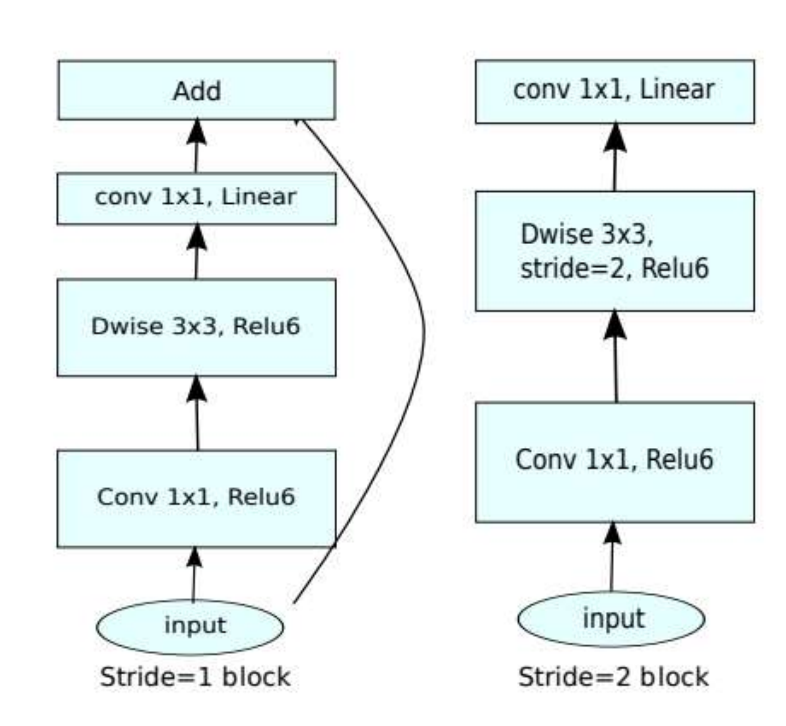
*1) it plays in a coupled way between **3D pose estimation** and **2D distance measurement**, which is much more reasonable than simply adding two concerns*

*2) it is intuitive and **easy** to be computed both **forward** and **backward**, comparing*

*3) it makes the network work in a **single-stage** manner instead of cascaded, which improves the optimality.*

MobileNet v2

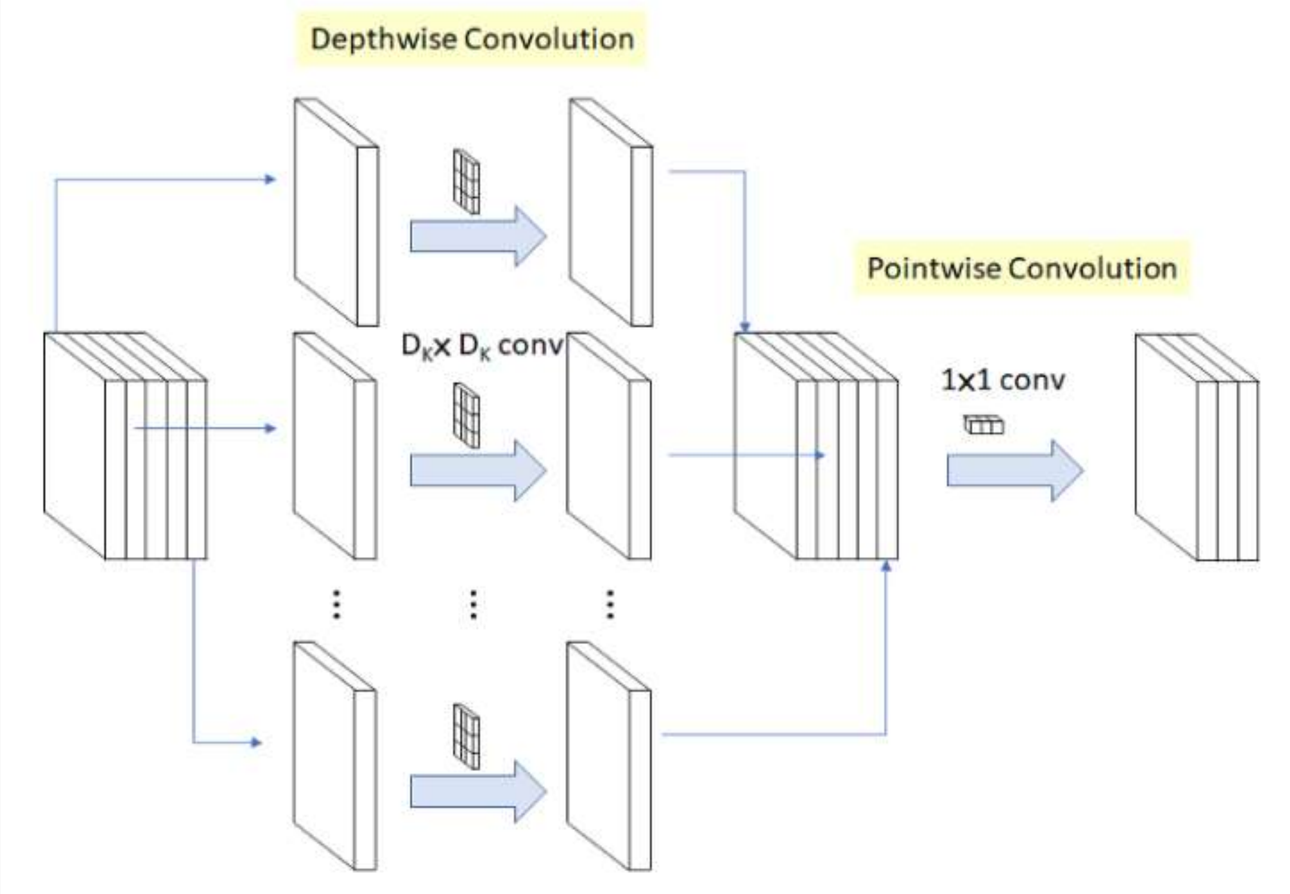
- Depthwise Seperable Convolutions (Depthwise Convolution + Pointwise Convolution)
- Linear Bottlenecks
- Inverted Residuals
- ReLU6



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

References
https://arxiv.org/pdf/1801.04381.pdf

MobileNet v2 - Depthwise Seperable Convolution



- Depthwise Convolution

$$D_k \cdot D_k \cdot D_G \cdot D_G \cdot M$$

- Pointwise Convolution

$$N \cdot D_G \cdot D_G \cdot M$$

- Depthwise Seperable Convolution

$$M \cdot D_G \cdot D_G \cdot (D_k \cdot D_k + N)$$

- Standard Convolution

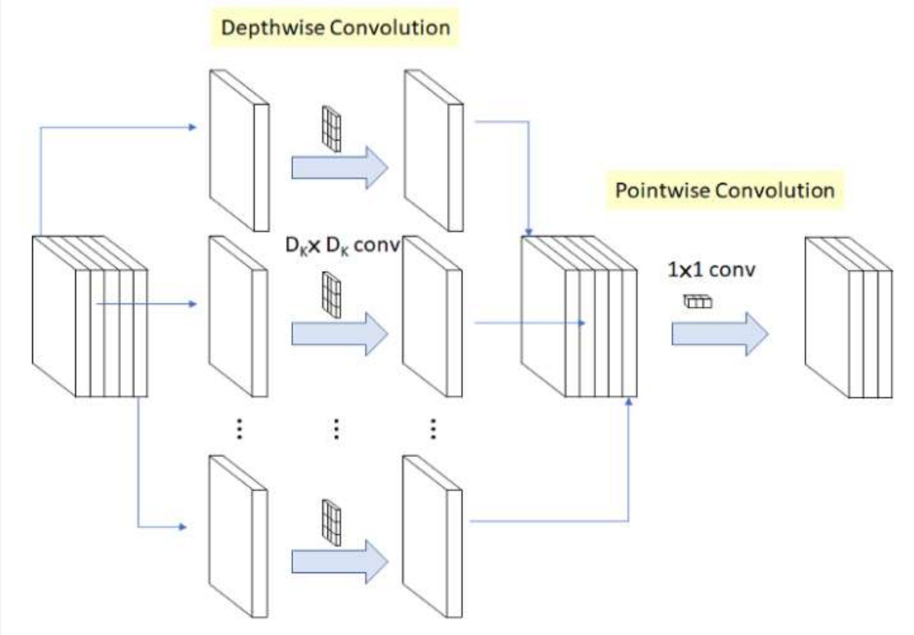
$$D_k \cdot D_k \cdot M \cdot D_G \cdot D_G \cdot N$$

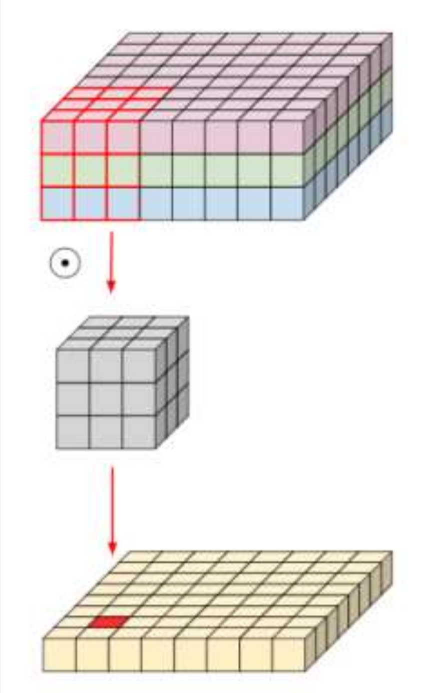
- Depthwise Seperable Coonvolution /Standard Convolution

$$\frac{1}{N} + \frac{1}{D_k^2}$$

References
https://towardsdatascience.com/review-mobilenetv1-depthwise-separable-convolution-light-weight-model-a382df364b69

MobileNet v2 - Depthwise Seperable Convolution





- Depthwise Convolution

$$D_k \cdot D_k \cdot D_G \cdot D_G \cdot M$$

- Pointwise Convolution

$$N \cdot D_G \cdot D_G \cdot M$$

- Depthwise Seperable Convolution

$$M \cdot D_G \cdot D_G \cdot (D_k \cdot D_k + N)$$

- Standard Convolution

$$D_k \cdot D_k \cdot M \cdot D_G \cdot D_G \cdot N$$

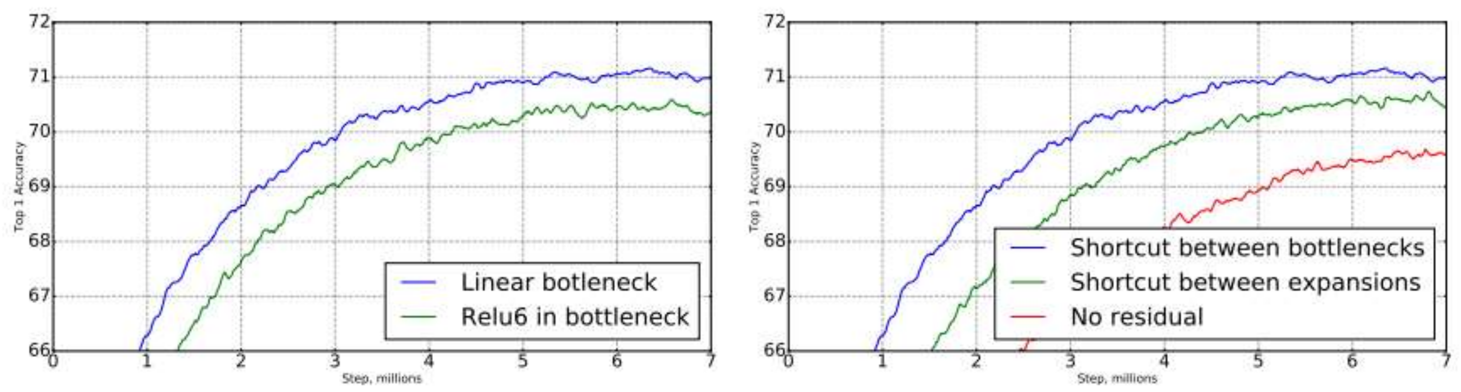
- Depthwise Seperable Coonvolution / Standard Convolution

$$\frac{1}{N} + \frac{1}{D_k^2}$$

References
(Left) https://towardsdatascience.com/review-mobilenetv1-depthwise-separable-convolution-light-weight-model-a382df364b69
(Right) https://kuklife.tistory.com/121

MobileNet v2 - Linear Bottlenecks

- 비선형함수 (ReLU)는 정보 손실을 발생시킴
- 채널 수가 적은 레이어에서는 Linear 함수를 사용

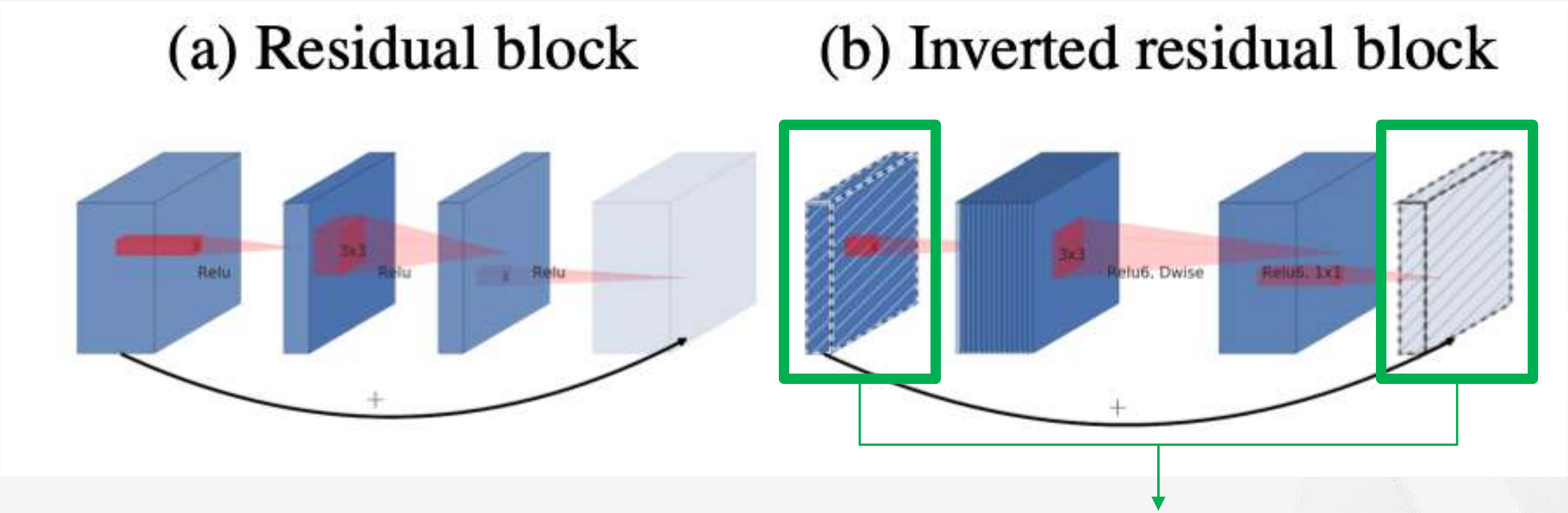


(a) Impact of non-linearity in the bottleneck layer. (b) Impact of variations in residual blocks.

References
https://arxiv.org/pdf/1801.04381.pdf

MobileNet v2 - Inverted Residuals

- Residual block과 비슷한 구조
- 일반적인 Wide –narrow – wide 형태, Inverted Residuals는 narrow –wide –narrow 형태
- 메모리 사용량을 줄일 수 있음

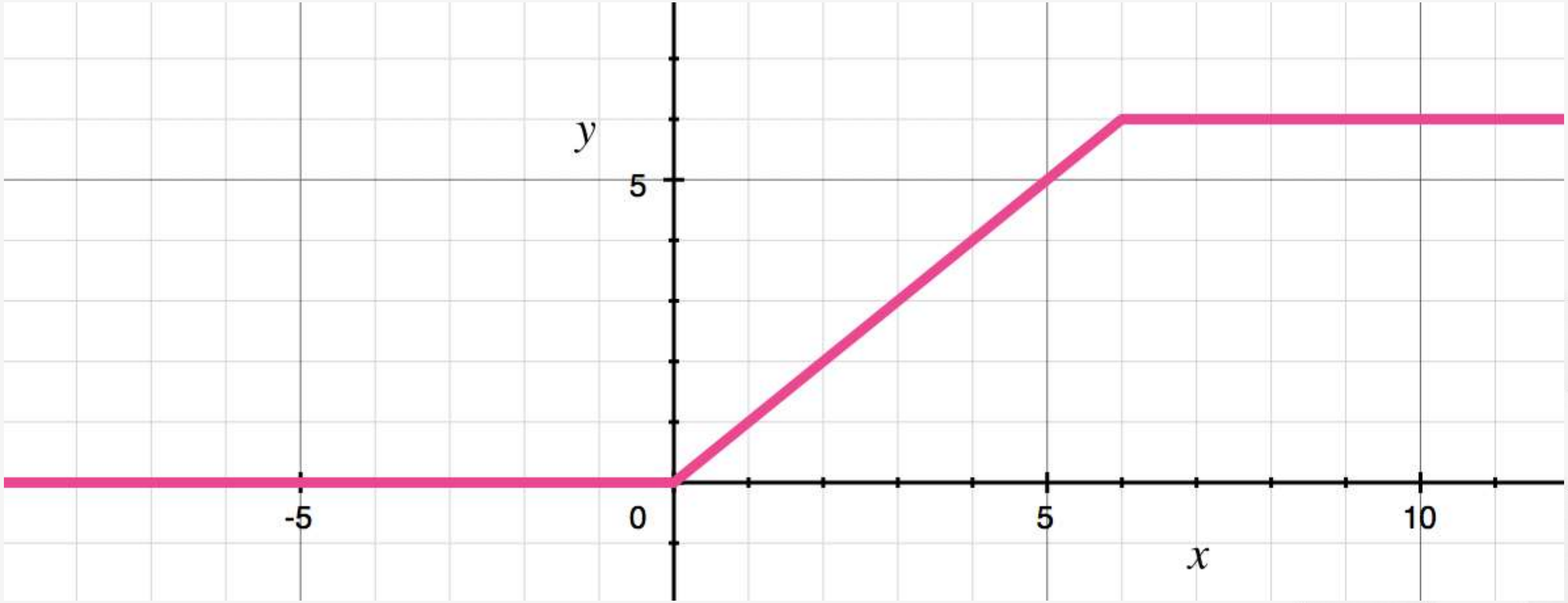


Linear Bottleneck

References
https://arxiv.org/pdf/1801.04381.pdf

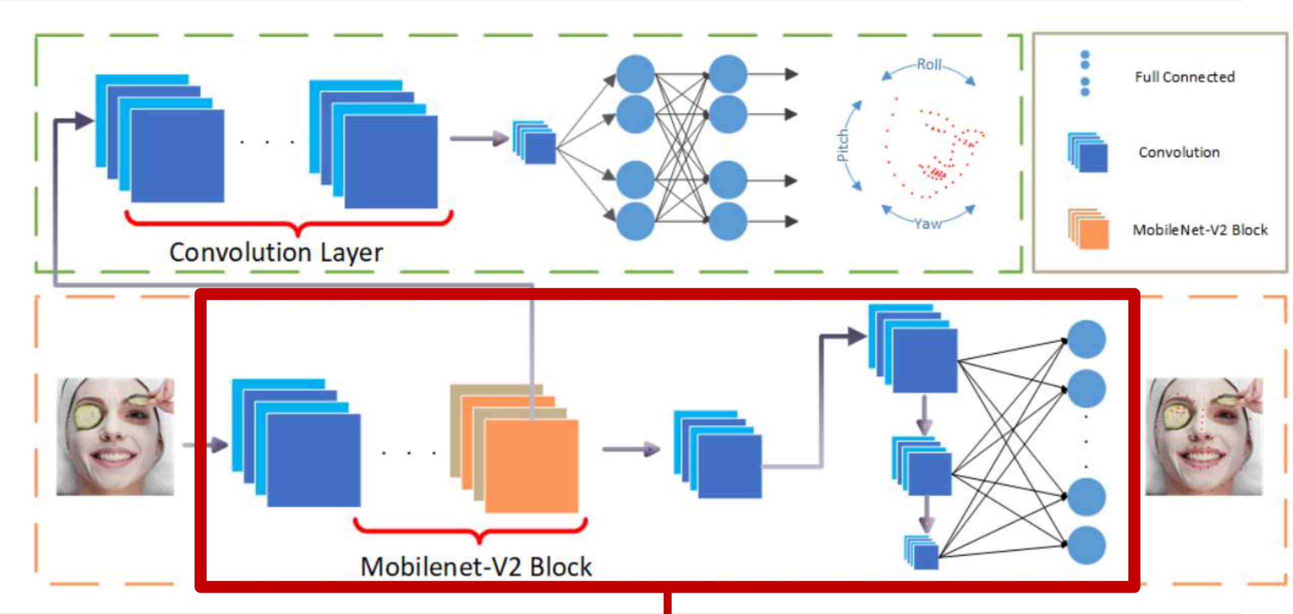
MobileNet v2 – ReLU6

- $\min(\max(0, x), 6)$
- 딥러닝 모델 최적화제 좋은 활성화 함수



References
https://gaussian37.github.io/dl-concept-relu6/

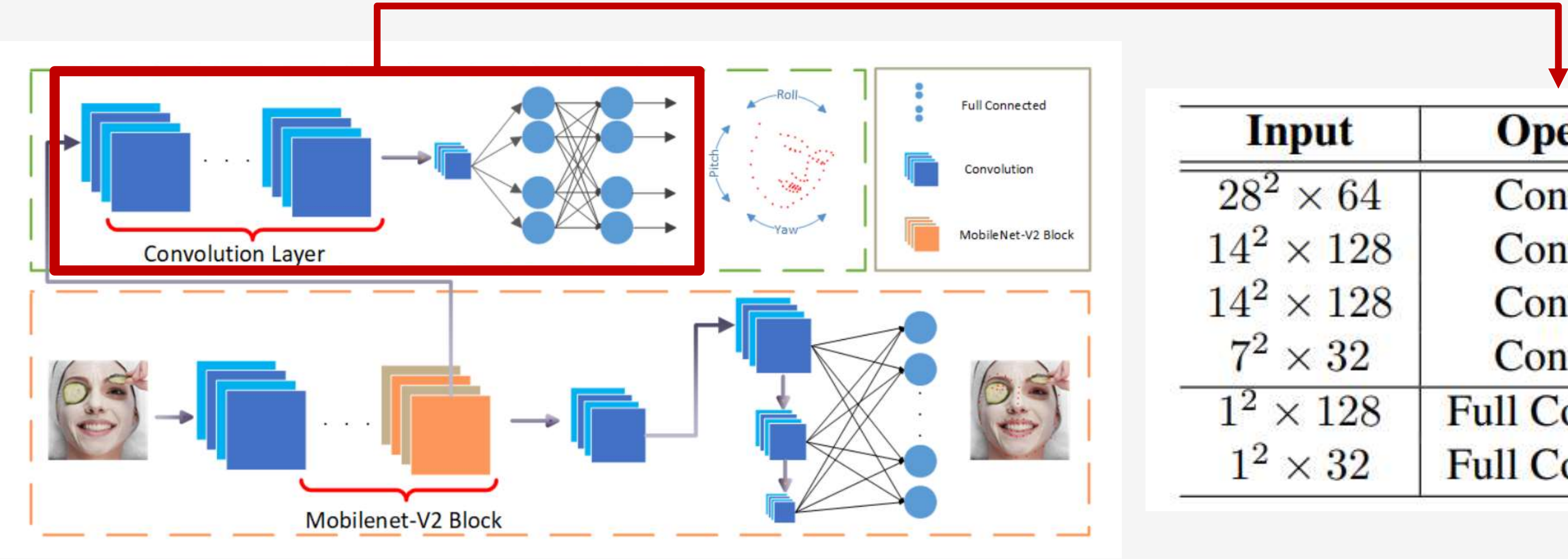
Backbone Network



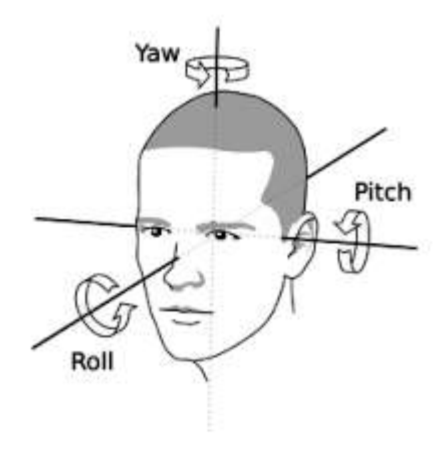
Input	Operator	t	c	n	s
$112^2 \times 3$	Conv3 $\times$ 3	-	64	1	2
$56^2 \times 64$	Depthwise Conv3 $\times$ 3	-	64	1	1
$56^2 \times 64$	Bottleneck	2	64	5	2
$28^2 \times 64$	Bottleneck	2	128	1	2
$14^2 \times 128$	Bottleneck	4	128	6	1
$14^2 \times 128$	Bottleneck	2	16	1	1
(S1) $14^2 \times 16$	Conv3 $\times$ 3	-	32	1	2
(S2) $7^2 \times 32$	Conv7 $\times$ 7	-	128	1	1
(S3) $1^2 \times 128$	-	-	128	1	-
S1, S2, S3	Full Connection	-	136	1	-

- Multi-Scale Fully-Connected (MS-FC) – Global structure on faces -> high accuracy localizing landmarks
- Simple Model, Hight performance
- Mobilenet
- Compressed by adjusting the width parameter

Auxiliary Network



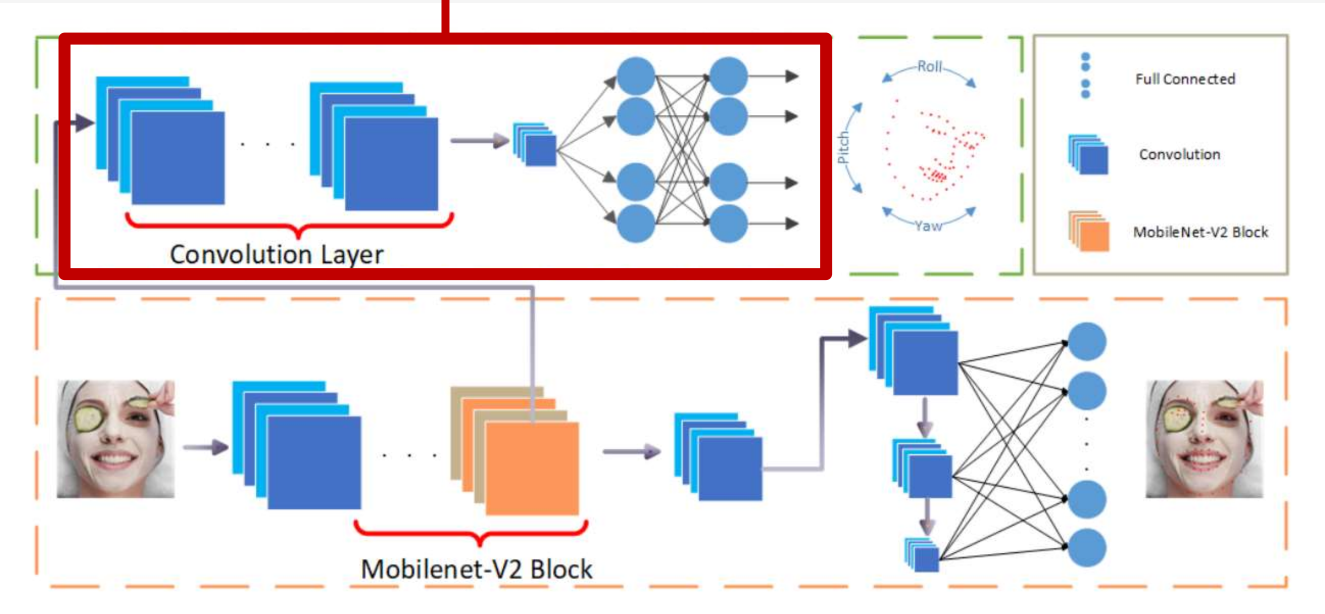
Input	Operator	c	s
$28^2 \times 64$	Conv3 $\times$ 3	128	2
$14^2 \times 128$	Conv3 $\times$ 3	128	1
$14^2 \times 128$	Conv3 $\times$ 3	32	2
$7^2 \times 32$	Conv7 $\times$ 7	128	1
$1^2 \times 128$	Full Connection	32	1
$1^2 \times 32$	Full Connection	3	-



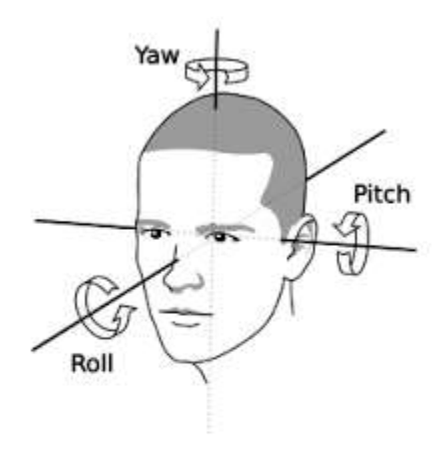
- Human faces that are of strong regularity and structure from the frontal view
- Estimate the 3D rotation information including yaw, pitch, and roll angles
- Landmark prediction may be too inaccurate especially at the beginning.
- Do not have frontal face with respect to each training sample

References
https://www.pinterest.co.kr/pin/127226758208193432/

Auxiliary Network



Input	Operator	c	s
$28^2 \times 64$	Conv3 $\times$ 3	128	2
$14^2 \times 128$	Conv3 $\times$ 3	128	1
$14^2 \times 128$	Conv3 $\times$ 3	32	2
$7^2 \times 32$	Conv7 $\times$ 7	128	1
$1^2 \times 128$	Full Connection	32	1
$1^2 \times 32$	Full Connection	3	-



- NO extra annotation used for computing the Euler angles

1) Predefine ONE standard face (**averaged over a bunch of frontal faces**);

2) Use the corresponding 11 landmarks of each face and the reference ones to estimate the **rotation matrix**

3) Compute the **Euler angles** from the rotation matrix

- Input of the auxiliary net is from the 4-th block of the backbone net

References
https://www.pinterest.co.kr/pin/127226758208193432/

Experimental Results

Implementation details

- Input image size : 112 x 112
- Batch size : 256
- Optimizer : Adam
- Weight decay : 10e-6
- Momentum : 0.9
- Learning rate : 10e-4
- Nvidia GTX 1080Ti

Dataset

300W

- LFPW, AFW, HELEN, XM2VTS, IBUG
- 68개 Face landmarks
- Trainnig : 3148, Test : 689

AFLW

- 21개 Face landmarks
- Trainnig : 20000, Test : 4386

Processing speed

Model	SDM [38]	SAN [9]	LAB [34]	**PFLD 0.25X**	**PFLD 1X**
Size (Mb)	10.1	270.5+528	50.7	**2.1**	**12.5**
Speed	16ms (C)	343ms(G)	2.6s(C)/60ms(G*)	**1.2ms(C)/1.2ms(G)/7ms(A)**	**6.1ms(C)/3.5ms(G)/26.4ms(A)**

Normalized Mean Error

Method	Common	Challenging	Fullset
Inter-pupil Normalization (IPN)			
RCPR [4]	6.18	17.26	8.35
CFAN [42]	5.50	16.78	7.69
ESR [5]	5.28	17.00	7.58
SDM [38]	5.57	15.40	7.50
LBF [24]	4.95	11.98	6.32
CFSS [46]	4.73	9.98	5.76
3DDFA [48]	6.15	10.59	7.01
TCDCN [45]	4.80	8.60	5.54
MDM [29]	4.83	10.14	5.88
SeqMT [12]	4.84	9.93	5.74
RAR [37]	4.12	8.35	4.94
DVLN [35]	3.94	7.62	4.66
CPM [33]	3.39	8.14	4.36
DCFE [30]	3.83	7.54	4.55
TSR [22]	4.36	7.56	4.99
LAB [34]	3.42	6.98	4.12
PFLD 0.25X	**3.38**	**6.83**	**4.02**
PFLD 1X	**3.32**	**6.56**	**3.95**
PFLD 1X+	**3.17**	**6.33**	**3.76**
Inter-ocular Normalization (ION)			
PIFA-CNN [15]	5.43	9.88	6.30
RDR [36]	5.03	8.95	5.80
PCD-CNN [19]	3.67	7.62	4.44
SAN [9]	3.34	6.60	3.98
PFLD 0.25X	**3.03**	**5.15**	**3.45**
PFLD 1X	**3.01**	**5.08**	**3.40**
PFLD 1X+	**2.96**	**4.98**	**3.37**

Inter-purpil Normalizeion

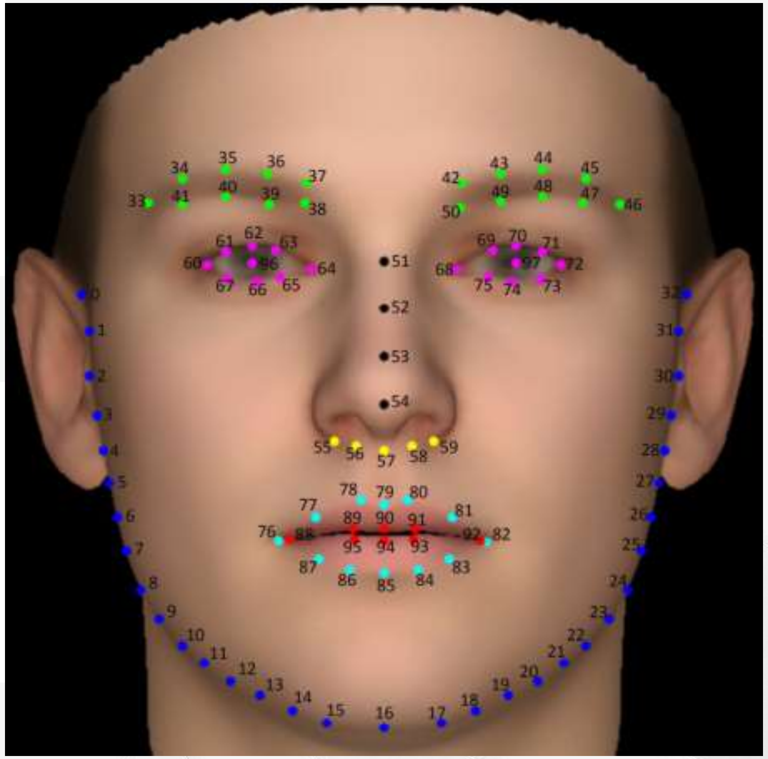
$$L = dist(p_{96}, p_{97})$$

$$NME(S, S^*) = \frac{1}{N} \sum_{i=1}^{N} \frac{\| s_i - s_i^* \|_2^2}{L}$$

Inter-purpil Normalizeion

$$L = dist(p_{60}, p_{72})$$

$$NME(S, S^*) = \frac{1}{N} \sum_{i=1}^{N} \frac{\| s_i - s_i^* \|_2^2}{L}$$



Normalized Mean Error

Method	RCPR [4]	CDM [41]	SDM [38]	ERT [16]	LBF [24]	CFSS [46]	CCL [47]
AFLW	5.43	3.73	4.05	4.35	4.25	3.92	2.72
Method	Binary-CNN [3]	PCD-CNN [19]	TSR [22]	CPM [33]	SAN [9]	**PFLD 0.25X**	**PFLD 1X**
AFLW	2.85	2.40	2.17	2.33	1.91	**2.07**	**1.88**

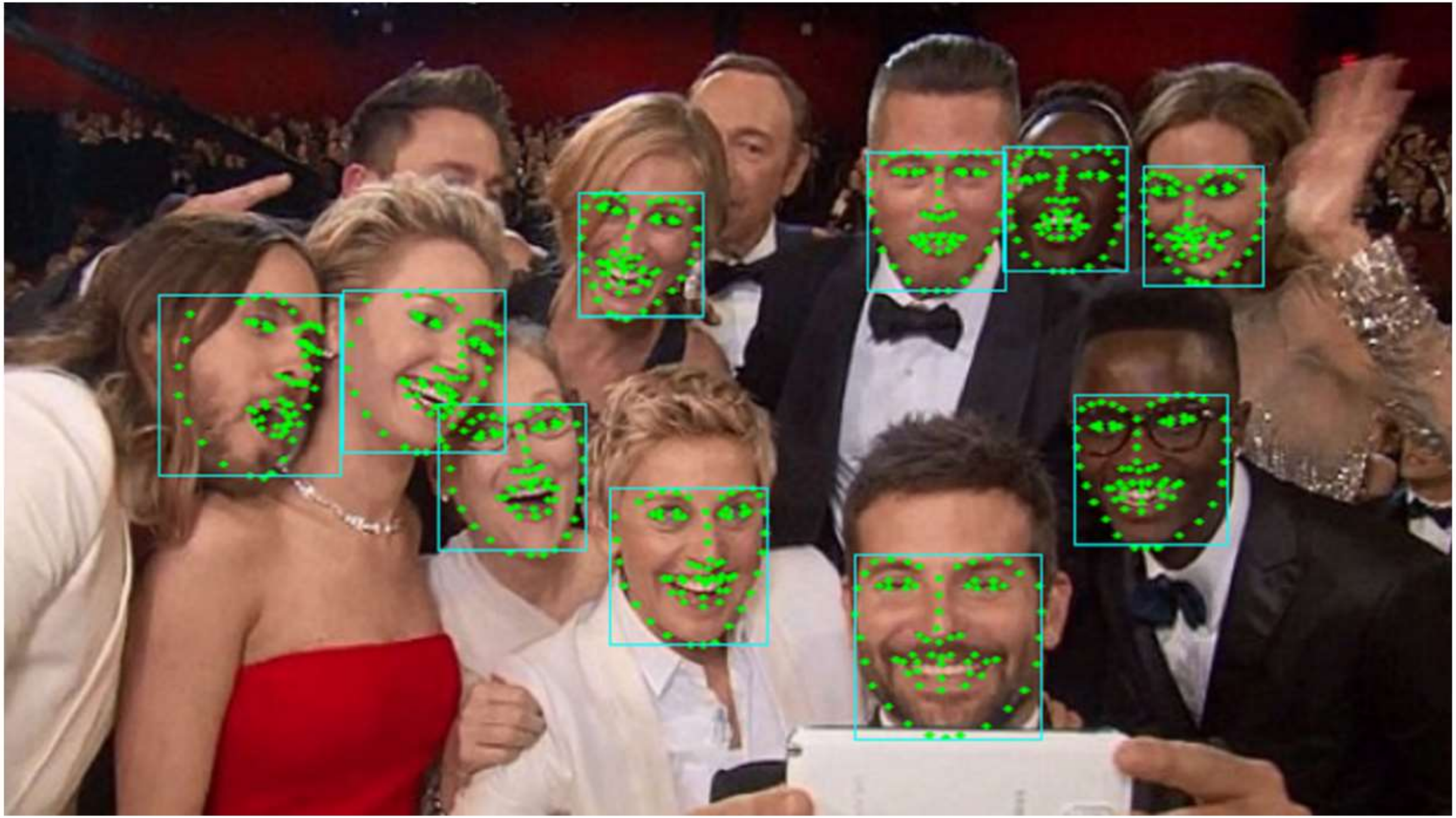
Table 5: Comparison in normalized mean error on the AFLW-full dataset.

$$L = \sqrt{Width * height}$$

$$NME(S,S^*) = \frac{1}{N}\sum_{i=1}^{N}\frac{\| s_i - s_i^* \|_2^2}{L}$$

Qualitative results

Figure 4: Qualitative results on several challenging faces by our PFLD 0.25X. We can observe that even with extreme lighting, expression, occlusion, and blur interferences, PFLD 0.25X can obtain visually pleasant results.



Conclusion

Conclusion

*Three aspects of facial landmark detectors need to be concerned for being competent on large-scale and/or real-time tasks, which are **accuracy, efficiency,** and **compactness**.*

*This paper proposed a practical facial landmark detector, termed as PFLD, which consists of two subnets, i.e. the **backbone network** and the **auxiliary network**.*

*Considering the **geometric regularization** and **data imbalance** issue, a novel loss was designed.*

*The extensive experimental results demonstrate the superior performance of our design over the **state-of-the-art** methods in terms of accuracy, model size, and processing speed.*

*PFLD only adopts the **rotation information** (yaw, roll and pitch angles) as the geometric constraint. It is expected to employ other geometric/structural information to help further **improve the accuracy**.*

[Practice 2] Golden Ratio Calculator

CONTENT

01
실습 소개

02
데이터셋

03
실습 튜토리얼

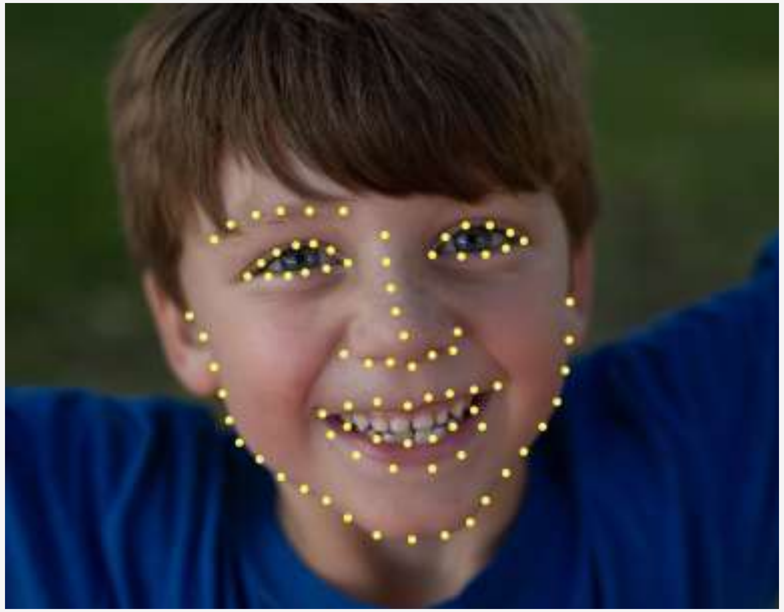
04
실습 결과

실습 소개

Face Landmark Detection이란?

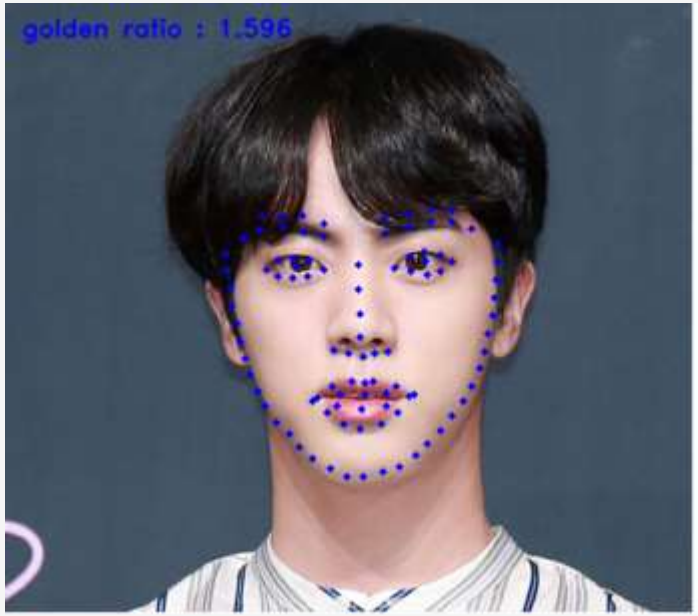
Detecting and localizing specific points or landmarks on a face, such as the eyes, nose, mouth, and chin.

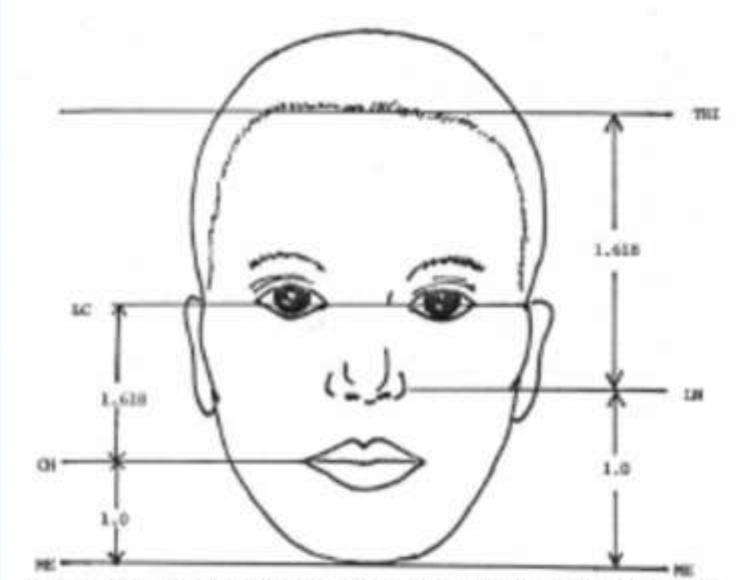
사람의 상태를 파악할 수 있다. (표정, 고개의 기울어짐 등)



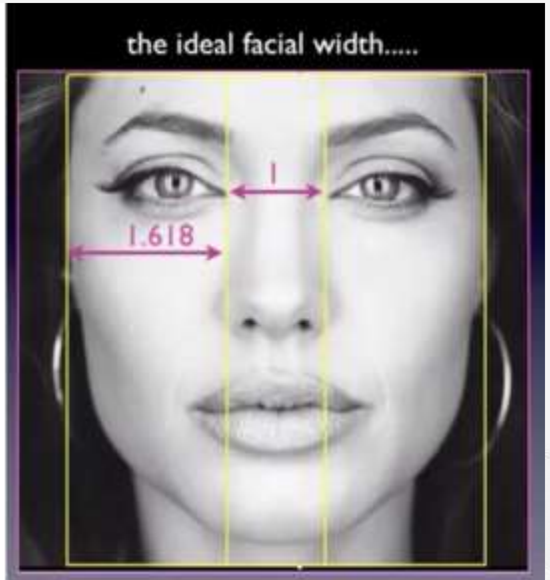
References
(Left) https://prlabhotelshoe.tistory.com/4
(Middle) https://www.plugger.ai/blog/the-top-7-use-cases-for-facial-landmark-detection
(Right) http://blog.dlib.net/2018/01/correctly-mirroring-datasets.html
https://paperswithcode.com/task/facial-landmark-detection

[실습2] 얼굴의 황금비 계산기





출처 : https://www.researchgate.net/figure/The-divine-proportions-of-the-face-and-the-golden-ratio_fig2_221472863

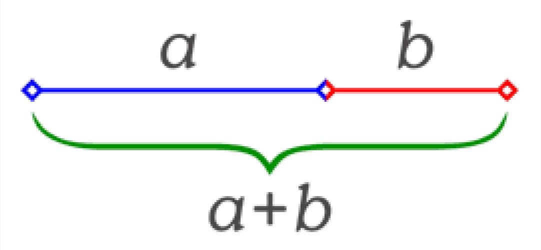


출처 : https://www.quora.com/Why-are-certain-facial-features-considered-attractive

황금비란?

두 수의 비율이 그 합과 두 수중 큰 수의 비율과 같도록 하는 비율로, 근사값이 약 1.618인 무리수

$$\frac{a+b}{a} = \frac{a}{b} = \varphi \qquad \frac{\sqrt{5}+1}{2}$$

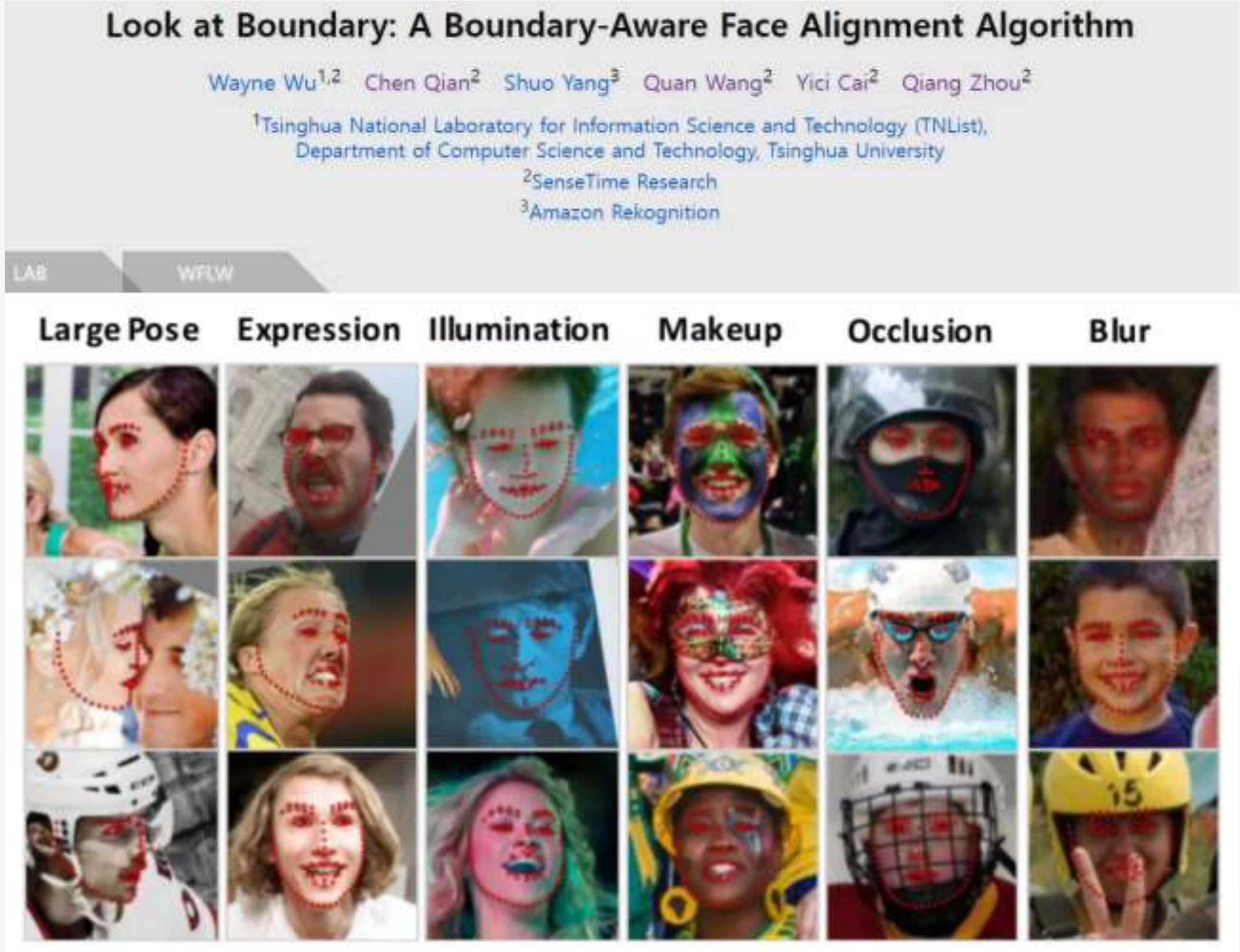


References
https://ko.wikipedia.org/wiki/%ED%99%A9%EA%B8%88%EB%B9%84

데이터셋

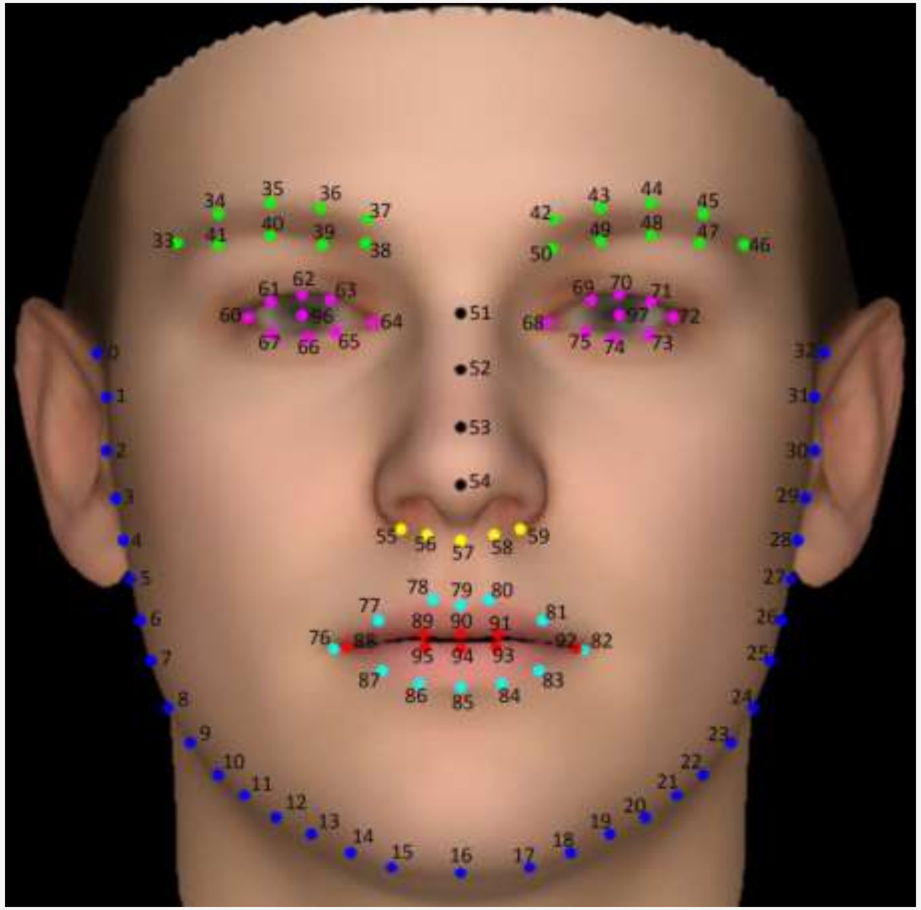
데이터셋 소개

WFLW – Wider Facial Landmarks in-the-wild



데이터셋 소개

WFLW – Wider Facial Landmarks in-the-wild



References
https://wywu.github.io/projects/LAB/WFLW.html

데이터셋 소개

- 데이터셋 소개 페이지 : https://wywu.github.io/projects/LAB/WFLW.html
- 10,000개 얼굴, 98개 annotated landmarks

분류	얼굴 수
Train	7500
test	2500

데이터셋 구조

- WFLW
  - WFLW_annotations
    - list_98pt_rect_attr_train_test
      - README
      - list_98pt_rect_attr_test.txt
      - list_98pt_rect_attr_train.txt
    - list_98pt_test
    - Mirror98.txt
  - WFLW_images

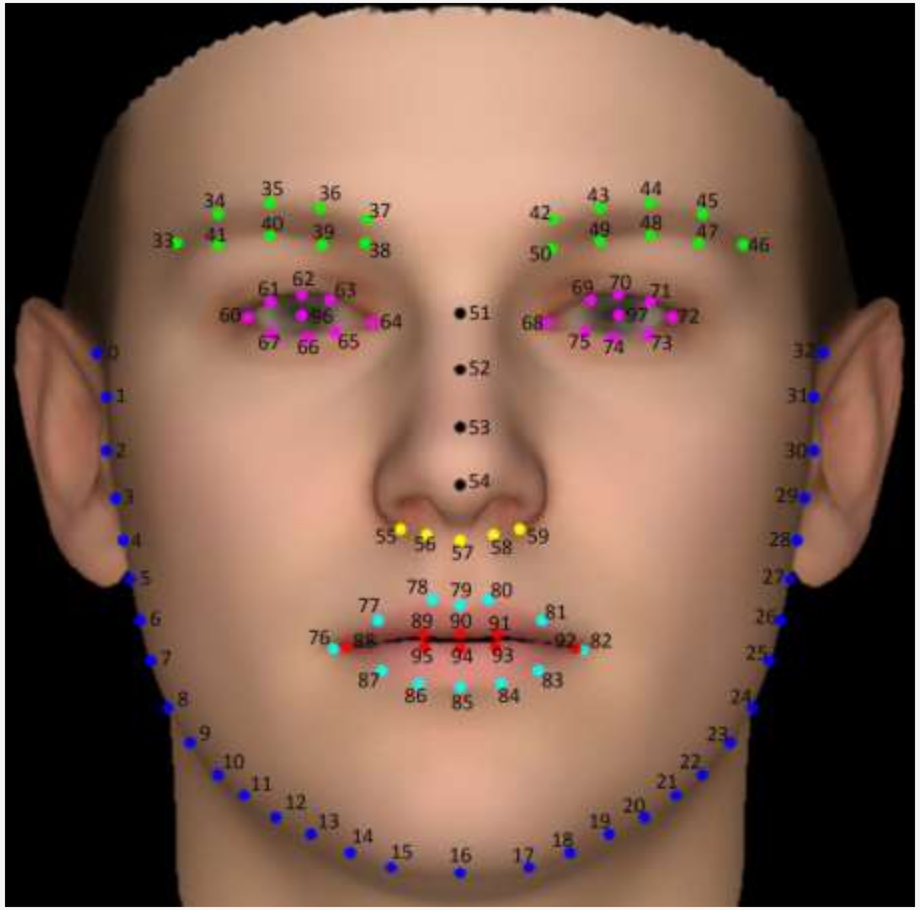
- WFLW_images
  - 0--Parade
  - 1--Handshaking
  - 10--People_Marching
  - 11--Meeting
  - 12--Group

coordinates of 98 landmarks (196) + image name (1)
x0 y0 ... x97 y97 image_name

182.212006 268.895996 184.231026 278.555935 186.344894 288.195481 188.648177 297.791365 191.233207 307.314980 194.196268 316.727393 197.675303 325.960683 201.824785 334.911467 206.775736 343.443634 212.673605 351.346740 219.837267 358.105834 228.194810 363.337202 236.455534 368.705138 243.563716 375.542218 250.925174 382.088737 259.777004 386.346261 269.549779 387.049878 278.298328 384.082885 285.392740 378.152167 291.378812 371.043838 297.780969 364.309057 304.312919 357.698462 310.447236 350.719653 315.930044 343.220593 320.511995 335.141825 324.051533 326.554428 326.568105 317.613186 328.390492 308.500061 330.612962 299.479998 333.487774 290.642392 335.713065 281.622785 337.163003 272.444467 338.227692 263.212006 210.138992 300.615997 222.644974 297.890961 235.363983 298.151978 246.942978 299.391968 257.441010 300.830994 257.611969 304.799957 246.900986 303.914001 235.179001 302.840973 222.606995 301.989990 285.806976 299.045990 295.512970 296.641968 304.544983 294.989014 314.121979 293.123962 323.378967 294.755005 314.194977 296.925964 304.505981 299.115967 295.472961 300.877960 285.771973 302.760956 269.851013 311.436005 269.770491 323.194979 269.382674 334.941630 270.491228 346.539181 253.042007 348.842987 261.490088 352.218771 270.379219 353.896457 277.935988 351.521132 284.542816 347.006134 226.742996 309.906006 232.584430 309.309438 238.440116 308.886997 245.457800 309.204711 252.335205 310.658722 245.617587 313.049272 238.509980 313.282636 232.535767 311.923110 284.890991 308.625000 291.685625 306.388967 298.740838 305.278426 304.445725 305.487436 310.095642 306.337067 304.749064 308.523762 299.148337 309.907617 291.962007 309.990723 246.033005 359.019501 255.168831 361.639041 264.404187 363.931053 269.706619 363.652676 274.989326 361.242112 280.950560 358.649372 286.599365 355.433472 283.359979 363.478759 277.741016 370.000529 269.582112 372.667721 260.176963 371.396279 252.116800 366.373531 247.076996 359.700012 258.005929 364.209840 269.620318 365.757483 278.627904 362.436113 286.014526 356.210266 279.413487 364.753998 269.432083 368.638642 257.285470 366.731658 238.453801 311.748538 299.024561 308.745029 37--Soccer/37_Soccer_soccer_ball_37_45.jpg

32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,46,45,44,43,42,50,49,48,47,37,36,35,34,33,41,40,39,38,51,52,53,54,59,58,57,56,55,72,71,70,69,68,75,74,73,64,63,62,61,60,67,66,65,82,81,80,79,78,77,76,87,86,85,84,83,92,91,90,89,88,95,94,93,97,96

Mirror98.txt



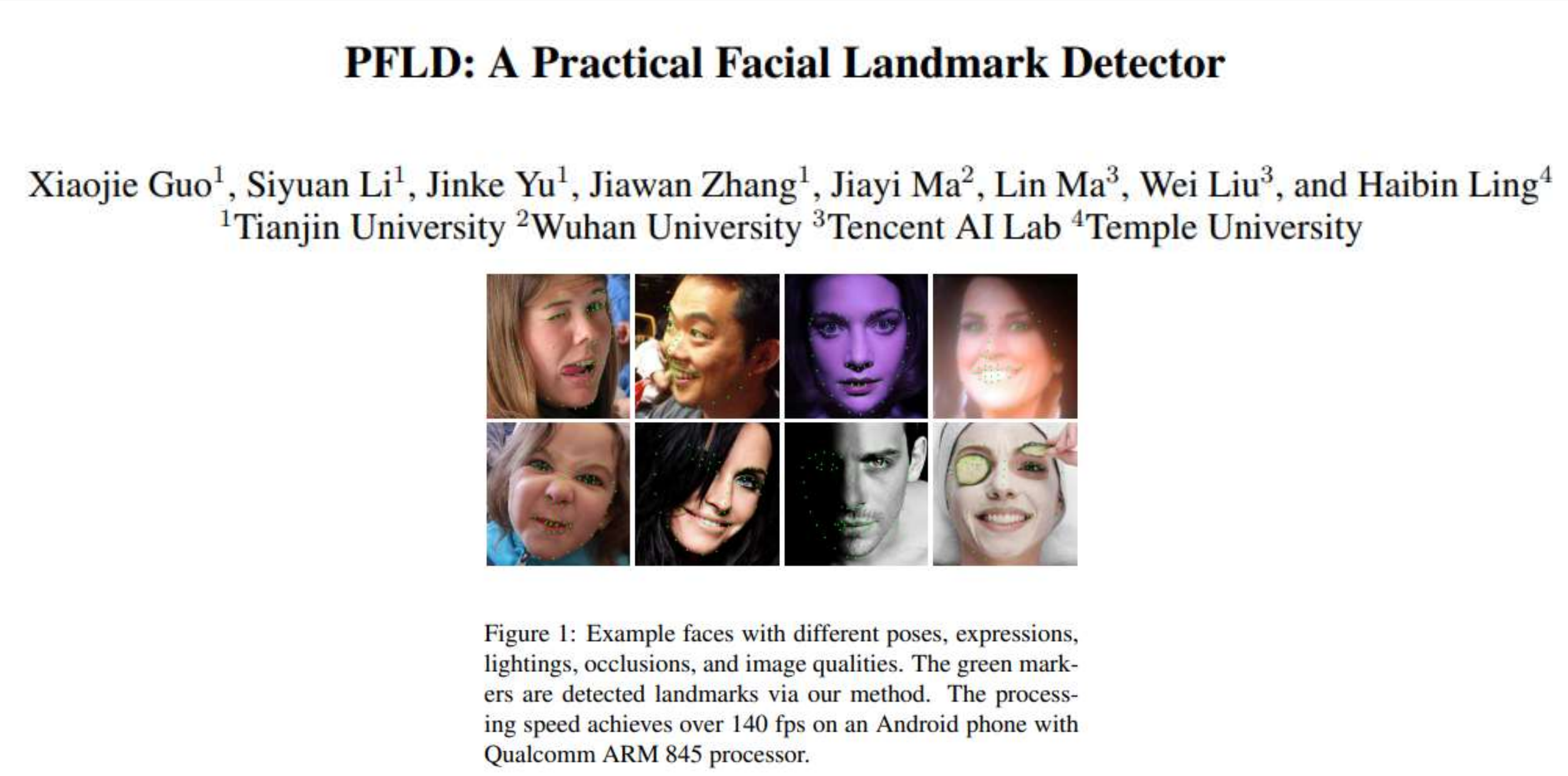
32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0,46,45,44,43,42,50,49,48,47,37,36,35,34,33,41,40,39,38,51,52,53,54,59,58,57,56,55,72,71,70,69,68,75,74,73,64,63,62,61,60,67,66,65,82,81,80,79,78,77,76,87,86,85,84,83,92,91,90,89,88,95,94,93,97,96

References
https://wywu.github.io/projects/LAB/WFLW.html

실습 튜토리얼

PFLD

- 논문 : https://arxiv.org/pdf/1902.10859.pdf
- Tensorlow Github : https://github.com/guoqiangqi/PFLD
- Pytorch Github : https://github.com/polarisZhao/PFLD-pytorch



PFLD: A Practical Facial Landmark Detector

Xiaojie Guo[1], Siyuan Li[1], Jinke Yu[1], Jiawan Zhang[1], Jiayi Ma[2], Lin Ma[3], Wei Liu[3], and Haibin Ling[4]
[1]Tianjin University [2]Wuhan University [3]Tencent AI Lab [4]Temple University

Figure 1: Example faces with different poses, expressions, lightings, occlusions, and image qualities. The green markers are detected landmarks via our method. The processing speed achieves over 140 fps on an Android phone with Qualcomm ARM 845 processor.

실습 환경 구축 / 데이터셋 다운로드

- 실습환경 구축

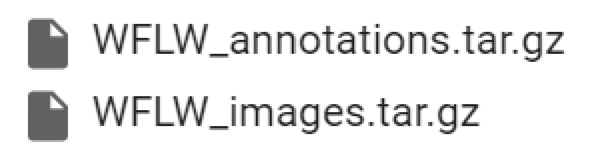
Git clone https://github.com/polarisZhao/PFLD-pytorch.git

pip3 install -r requirements.txt

- 데이터셋 다운로드

WFLW Training and Testing Images [Google Drive] [Baidu Drive]

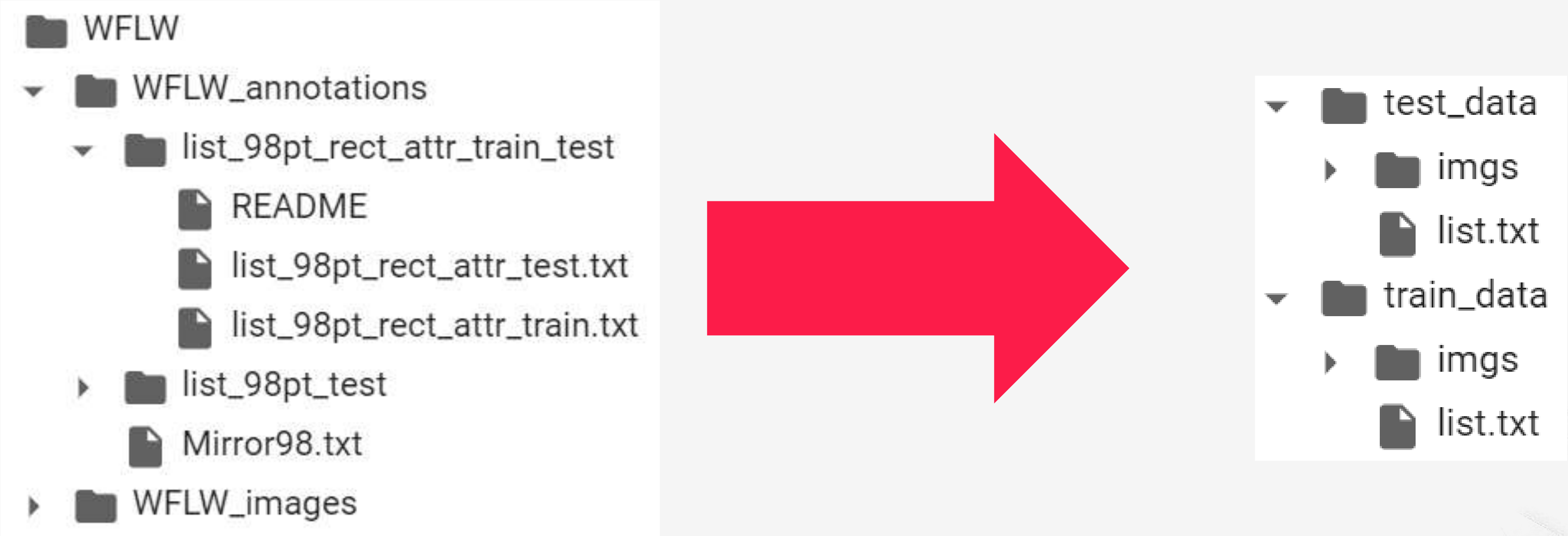
WFLW Face Annotations



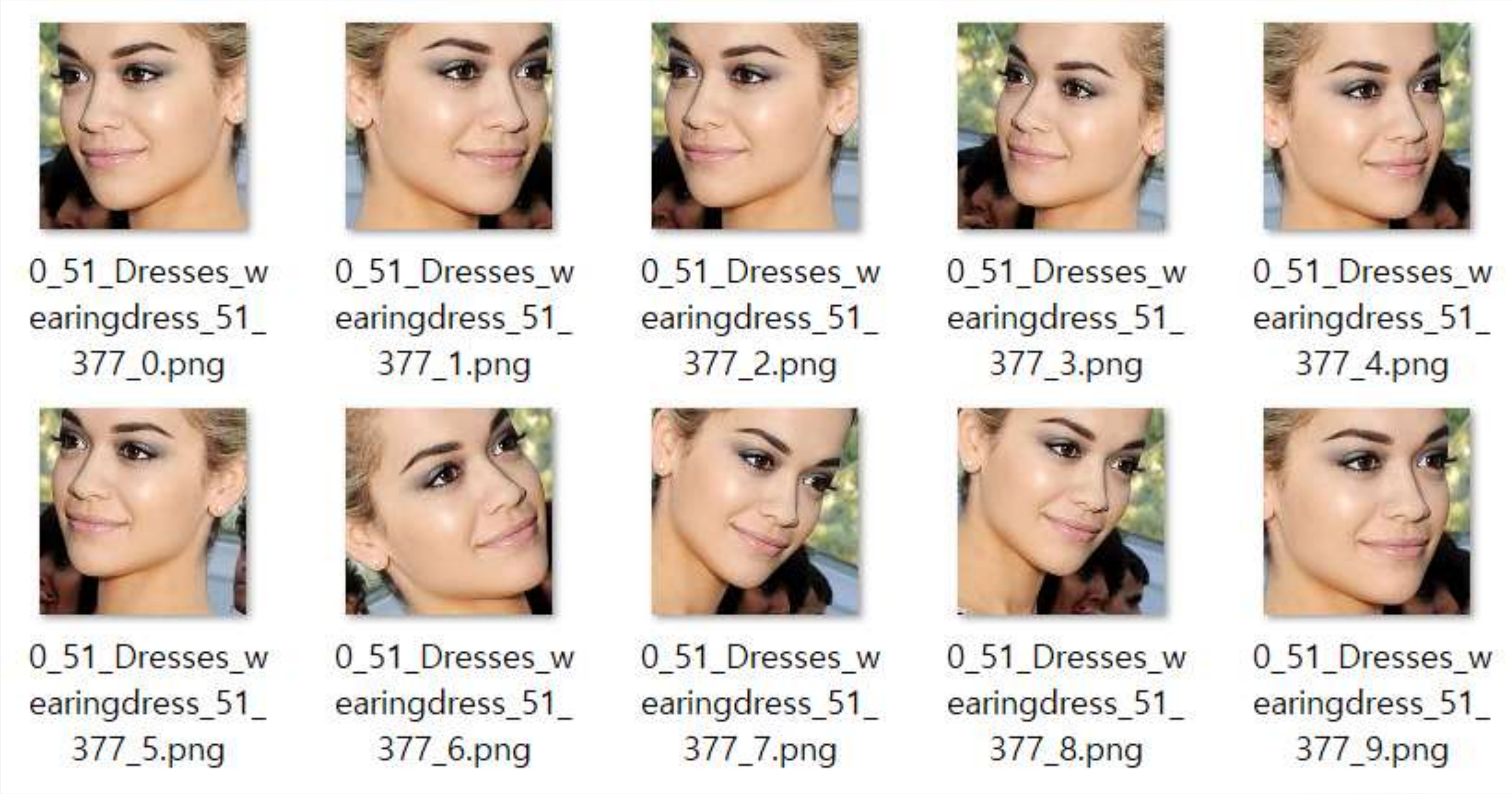
Unzip *.tar.gz

Move Mirror98.txt to WFLW_annotations

데이터셋 전처리


WFLW
WFLW_annotations
list_98pt_rect_attr_train_test
README
list_98pt_rect_attr_test.txt
list_98pt_rect_attr_train.txt
list_98pt_test
Mirror98.txt
WFLW_images
test_data
imgs
list.txt
train_data
imgs
list.txt

데이터셋 전처리


0_51_Dresses_wearingdress_51_377_0.png
0_51_Dresses_wearingdress_51_377_1.png
0_51_Dresses_wearingdress_51_377_2.png
0_51_Dresses_wearingdress_51_377_3.png
0_51_Dresses_wearingdress_51_377_4.png
0_51_Dresses_wearingdress_51_377_5.png
0_51_Dresses_wearingdress_51_377_6.png
0_51_Dresses_wearingdress_51_377_7.png
0_51_Dresses_wearingdress_51_377_8.png
0_51_Dresses_wearingdress_51_377_9.png

데이터셋 전처리

coordinates of 98 landmarks (196) + image name (1)
x0 y0 ... x97 y97 image_name

182.212006 268.895996 184.231026 278.555935 186.344894 288.195481 188.648177 297.791365 191.233207 307.314980 194.196268 316.727393 197.675303 325.960683 201.824785 334.911467 206.775736 343.443634 212.673605 351.346740 219.837267 358.105834 228.194810 363.337202 236.455534 368.705138 243.563716 375.542218 250.925174 382.088737 259.777004 386.346261 269.549779 387.049878 278.298328 384.082885 285.392740 378.152167 291.378812 371.043838 297.780969 364.309057 304.312919 357.698462 310.447236 350.719653 315.930044 343.220593 320.511995 335.141825 324.051533 326.554428 326.568105 317.613186 328.390492 308.500061 330.612962 299.479998 333.487774 290.642392 335.713065 281.622785 337.163003 272.444467 338.227692 263.212006 210.138992 300.615997 222.644974 297.890961 235.363983 298.151978 246.942978 299.391968 257.441010 300.830994 257.611969 304.799957 246.900986 303.914001 235.179001 302.840973 222.606995 301.989990 285.806976 299.045990 295.512970 296.641968 304.544983 294.989014 314.121979 293.123962 323.378967 294.755005 314.194977 296.925964 304.505981 299.115967 295.472961 300.877960 285.771973 302.760956 269.851013 311.436005 269.770491 323.194979 269.382674 334.941630 270.491228 346.539181 253.042007 348.842987 261.490088 352.218771 270.379219 353.896457 277.935988 351.521132 284.542816 347.006134 226.742996 309.906006 232.584430 309.309438 238.440116 308.886997 245.457800 309.204711 252.335205 310.658722 245.617587 313.049272 238.509980 313.282636 232.535767 311.923110 284.890991 308.625000 291.685625 306.388967 298.740838 305.278426 304.445725 305.487436 310.095642 306.337067 304.749064 308.523762 299.148337 309.907617 291.962007 309.990723 246.033005 359.019501 255.168831 361.639041 264.404187 363.931053 269.706619 363.652676 274.989326 361.242112 280.950560 358.649372 286.599365 355.433472 283.359979 363.478759 277.741016 370.000529 269.582112 372.667721 260.176963 371.396279 252.116800 366.373531 247.076996 359.700012 258.005929 364.209840 269.620318 365.757483 278.627904 362.436113 286.014526 356.210266 279.413487 364.753998 269.432083 368.638642 257.285470 366.731658 238.453801 311.748538 299.024561 308.745029 37--Soccer/37_Soccer_soccer_ball_37_45.jpg

image location(1) + coordinates of 98 landmarks (196) + attributes (6) + euler_angles_landmark (3)
image location, x0 y0 ... x97 y97, pose, expression, illumination, make_up, occlusion, blur, pitch, yaw, roll

/content/drive/MyDrive/PFLD-pytorch/data/test_data/imgs/0_37_Soccer_soccer_ball_37_45_0.png 0.08623407242145945 0.20157444730718085 0.09697358151699634 0.2529571208548039 0.10821752345308344 0.3042313596035572 0.1204690324499252 0.3552731453104222 0.13421914932575632 0.40593070172249 0.14998017980697306 0.4559967365670711 0.16848568206137798 0.5051100710605053

0.5990845700527759 0.6991283335584275 0.6383751402509973 0.6660120537940492 0.6032632056702959 0.7114574351209275 0.5501705737824135 0.732120432752244 0.48556096503075136 0.7219769092316323 0.38539253397190826 0.4295134848736702 0.7075774821829288 0.4135373703976895 0 0 0 0 0 75.08026 -36.433212 -22.542728

데이터셋 전처리

cd root/data

$ python3 SetPreparation.py

데이터 위치가 다르다면 수정 필요

```
if __name__ == '__main__':
    root_dir = os.path.dirname(os.path.realpath(__file__))
    imageDirs = './data/WFLW_images'
    Mirror_file = './data/Mirror98.txt'

    landmarkDirs = ['./data/WFLW_annotations/list_98pt_rect_attr_train_test/list_98pt_rect_attr_test.txt',
                    './data/WFLW_annotations/list_98pt_rect_attr_train_test/list_98pt_rect_attr_train.txt']

    outDirs = ['test_data', 'train_data']
    for landmarkDir, outDir in zip(landmarkDirs, outDirs):
        outDir = os.path.join(root_dir, outDir)
        print(outDir)
        if os.path.exists(outDir):
            shutil.rmtree(outDir)
        os.mkdir(outDir)
        if 'list_98pt_rect_attr_test.txt' in landmarkDir:
            is_train = False
        else:
            is_train = True
        imgs = get_dataset_list(imageDirs, outDir, landmarkDir, is_train)
    print('end')
```

Training / Testing

Training

python train.py --train_batchsize 320 --val_batchsize 320

Testing

python test.py --model_path

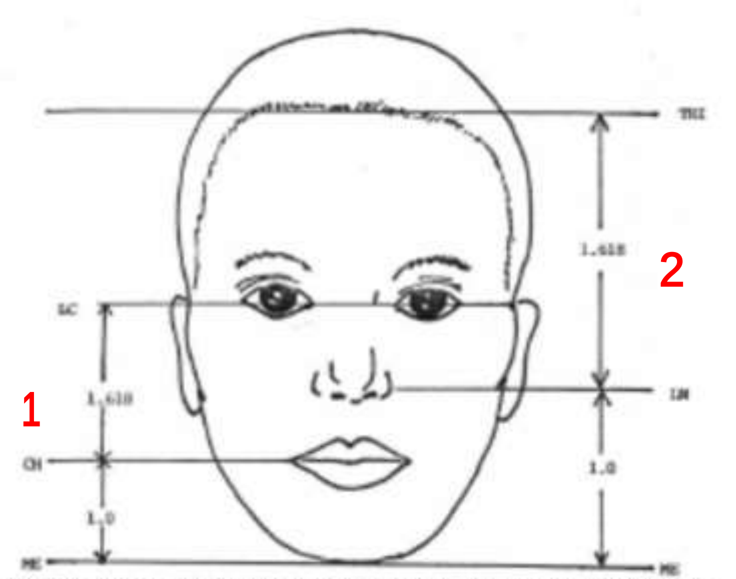
Inference

```python
def landmark_detection(img, det, model_path):
    checkpoint = torch.load(model_path, map_location=device)
    pfld_backbone = PFLDInference().to(device)
    pfld_backbone.load_state_dict(checkpoint['pfld_backbone'])
    pfld_backbone.eval()
    pfld_backbone = pfld_backbone.to(device)
    transform = torchvision.transforms.Compose(
        [torchvision.transforms.ToTensor()])

    height, width = img.shape[:2]
    x1, y1, x2, y2 = (det[:4] + 0.5).astype(np.int32)

    w = x2 - x1 + 1
    h = y2 - y1 + 1
    cx = x1 + w // 2
    cy = y1 + h // 2

    size = int(max([w, h]) * 1.1)
    x1 = cx - size // 2
    x2 = x1 + size
    y1 = cy - size // 2
    y2 = y1 + size

    x1 = max(0, x1)
    y1 = max(0, y1)
    x2 = min(width, x2)
    y2 = min(height, y2)
```

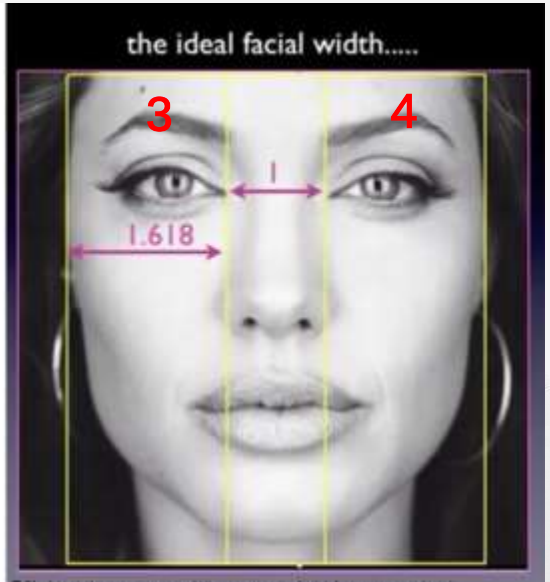
```python
    edx1 = max(0, -x1)
    edy1 = max(0, -y1)
    edx2 = max(0, x2 - width)
    edy2 = max(0, y2 - height)

    cropped = img[y1:y2, x1:x2]
    if (edx1 > 0 or edy1 > 0 or edx2 > 0 or edy2 > 0):
        cropped = cv2.copyMakeBorder(cropped, edy1, edy2, edx1, edx2,
                                     cv2.BORDER_CONSTANT, 0)

    input = cv2.resize(cropped, (112, 112))
    input = transform(input).unsqueeze(0).to(device)
    _, landmarks = pfld_backbone(input)
    pre_landmark = landmarks[0]
    pre_landmark = pre_landmark.cpu().detach().numpy().reshape(
        -1, 2) * [size, size] - [edx1, edy1]

    result = []
    for p in pre_landmark:
        x = p[0] + x1
        y = p[1] + y1
        result.append([int(x), int(y)])

    return result
```

Golden Ratio Calculator 함수

```python
def calc_gr_vertical(pts, bounding_boxes, n =4):
    result = []
    A = pts[64][1] - pts[76][1]
    B = pts[76][1] - pts[16][1]

    result.append(round(A/B,n))

    A = bounding_boxes[1] - pts[59][1]
    B = pts[59][1] - pts[16][1]

    result.append(round(A/B,n))

    A = pts[64][0] - pts[0][0]
    B = pts[68][0] - pts[64][0]

    result.append(round(A/B,n))
    A = pts[32][0] - pts[68][0]
    B = pts[68][0] - pts[64][0]

    result.append(round(A/B,n))
    return result, sum(result)/len(result)
```



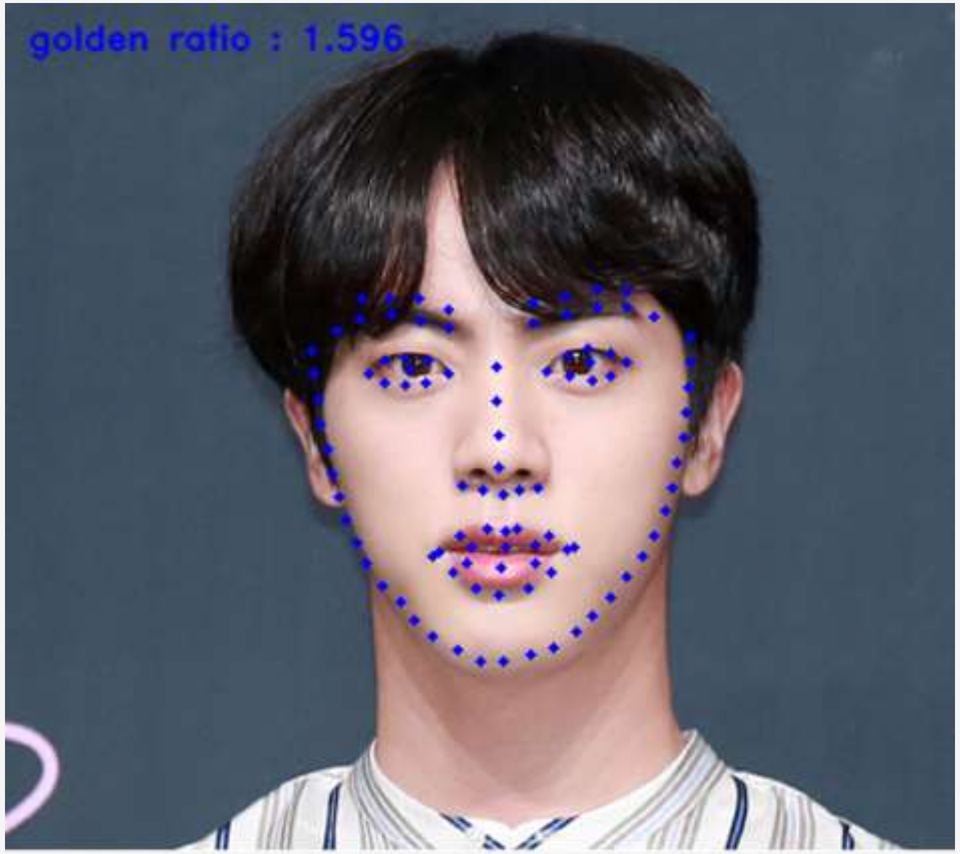
출처 : https://www.researchgate.net/figure/The-divine-proportions-of-the-face-and-the-golden-ratio_fig2_221472863



출처 : https://www.quora.com/Why-are-certain-facial-features-considered-attractive

실습 결과

실행 결과



황금비 결과 리스트 : [1.7391, 1.7844, 1.3953, 1.4651],
황금비 결과 평균값 : 1.596

Thank You.