2020학년도 2학기 데이터분석 캡스톤디자인 중간보고서

Generate a Personal Emoji and Real-time Emotion Recognition

소프트웨어융합학과

2018102092

권민지

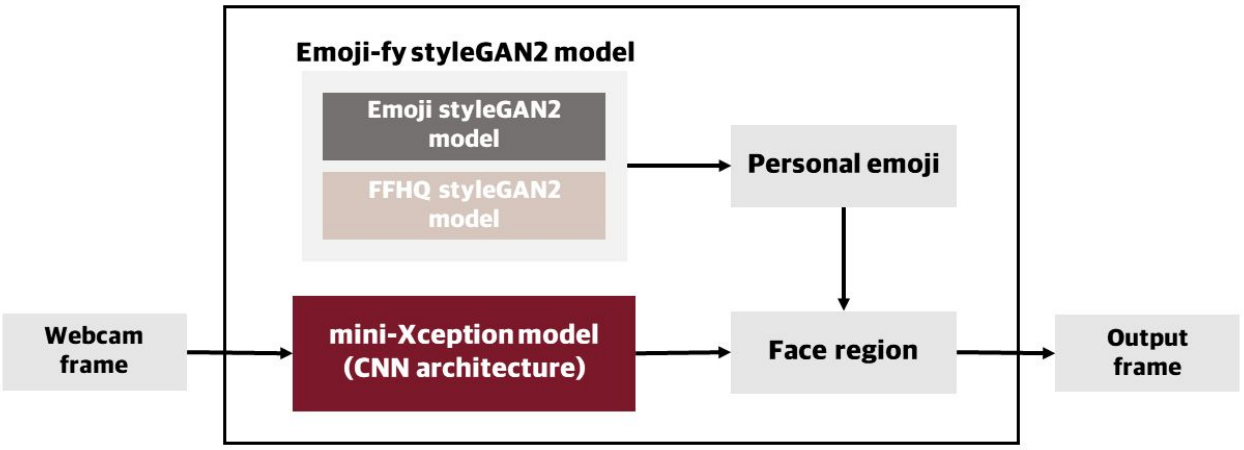
I. 연구 주제

실시간으로 받아온 webcam 영상에서 사용자의 얼굴 영역을 찾아 표정을 구분하고, 각 표정에 맞는 personal emoji를 렌더링한다.

II. 구현 목표

1. 기존의 emotion recognition CNN model(Keras)의 정확도를 66%보다 향상시킨다.
2. Emoji를 학습시킨 styleGAN2 모델과 FFHQ를 학습시킨 styleGAN2 모델의 네트워크를 혼합하여 새로운 모델을 구축한다.
3. 실시간으로 webcam의 영상을 받아와 검출된 얼굴 영역에서 감지한 표정에 따라 styleGAN2 모델을 통해 만든 personal emoji를 렌더링한다.

III. 프로그램 설계



[그림 1] 프로그램 설계도

프로그램의 전체적인 구조는 [그림 1]과 같다. StyleGAN2 모델을 이용하여 personal emoji를 생성하고, 준비해둔 emoji를 real-time으로 검출한 얼굴 영역 위에 표정에 걸맞는 것을 렌더링한다. Emotion recognition에 사용되는 mini-Xception 모델의 경우, 파라미터의 수를 완전히 줄여 실시간으로 얼굴 표정을 검출해도 속도에 큰 문제가 없으나, styleGAN의 경우 GPU 메모리 부족과 시간이 오래 걸리는 점을 감안하여 personal emoji의 경우는 미리 생성하여 emoji list에 넣어둘 계획이다.

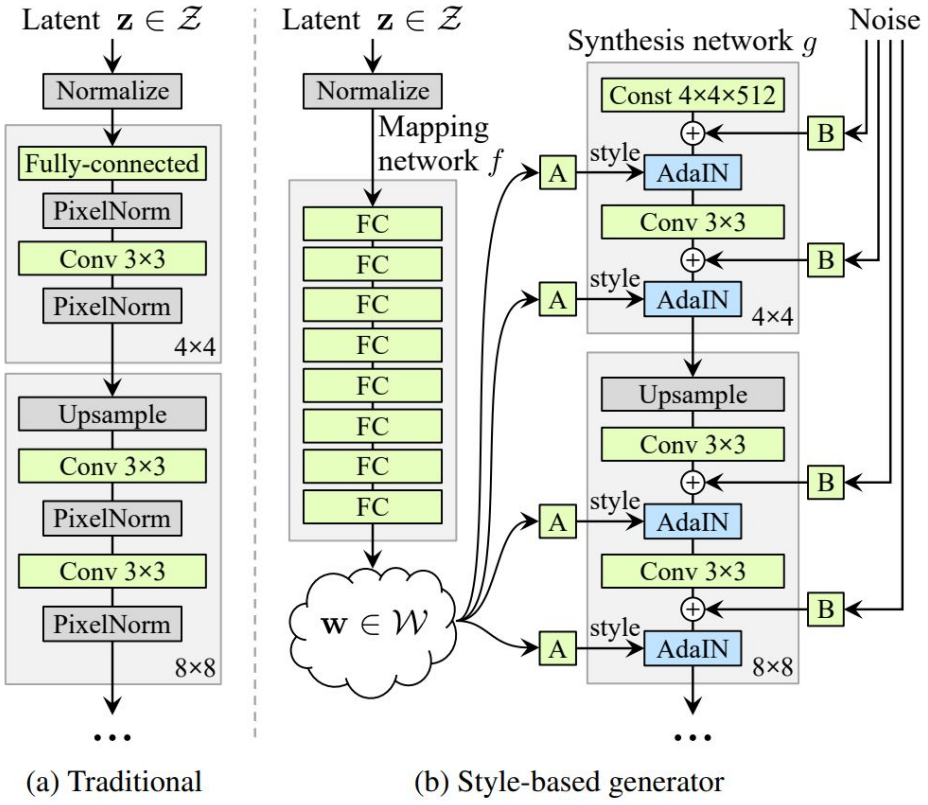
IV. 이론

1. StyleGAN(A Style-Based Generator Architecture for Generative Adversarial Networks) [1-2]

StyleGAN은 PGGAN(Progressive Growing of GANs) 구조에서 generator architexture를 재구성했다. 이로 인해 기존 PGGAN에서 불가능했던 Image synthesis process(이미지 합성 프로세스)를 조정하는 것이 가능하게 되었다. StyleGAN은 학습된 constant input, 즉 mapping network로부터 학습되어진 입력 값으로부터 이미지를 생성한다. Intermediate latent vector w 로부터 얻은 각각의 style들을 conv layer를 거치는 과정마다 적용하는 것이 가능하고, 이로 인해 이미지의 strength를 조절하는 것이 가능하다.

1.1 PGGAN의 generator architecture

PGGAN은 styleGAN의 base model이 되는 모델로, latent vector Z 가 normalization을 거쳐 모델에 바로 input value로 들어간다[그림 2 (a)]. 이런 식으로 latent vector가 바로 input으로 들어가게 되면, latent space는 반드시 training data의 probability density를 따라야 한다. Training data의 분포에 완전하게 관여하게 되기 때문에, 생성되는 이미지에서 머리카락 길이, 시선과 같은 한 가지 특성에만 관여하는 것이 불가능하다. 이는 generator model이 학습을 하는 동안 latent space가 entanglement하게 만들어졌음을 의미한다. 다시 말해, latent vector Z 의 specific value를 변경해도 생성되는 이미지의 하나의 특성(머리카락 길이, 성별 등)에만 영향을 주는 것이 어렵다. 사람의 얼굴 데이터 셋이 동양인 20%, 서양인 60%, 아프리카인 20%의 데이터 분포를 가지고 있다 가정하자. 이 데이터 셋을 이용해 PGGAN 구조의 model을 학습시키면, latent space는 데이터 비율이 많은 서양인을 계속해서 생성해내려고 할 것이다. 그만큼 데이터 분포가 서양인에 편향되게 되며, 이를 latent space가 entanglement하다, 학습에 사용하는 data의 probability density를 따른다고 한다.



[그림 2]PGGAN과 StyleGAN2의 generator architecture[1].

1.2 StyleGAN의 generator architecture

StyleGAN은 PGGAN과 같이 latent space에서 나온 latent vector Z를 model에 input으로 바로 넣지 않고, mapping network f를 거쳐서 나온 w를 뽑아내게 된다[그림 2 (b)]. 다시 말해, 확률적으로 생성한 latent vector로부터 이미지를 생성하지 않고, 고정된 값을 이용하여 이미지를 생성하는 것이다. PGGAN의 경우, latent vector가 바로 model에 들어가게 되면 training data의 probability density를 따라야 하기 때문에 머리카락 길이, 사람의 시선과 같은 stochastic variation을 변경하는 데 있어 한계가 있다. 그러나 StyleGAN은 training data의 probability density를 따르지 않도록 Non-linear function인 mapping network를 사용하였고, 이로 인해 latent vector 간의 상관관계를 줄이게 되어 latent space가 좀 더 disentanglement하게 만들 수 있다.

[그림 2 (b)]에서 사용되는 latent space, fully-connected layer, w의 차원은 모두 512이다. Mapping network f를 이용해 z 값을 w 값으로 만들어주고, 이것을 synthesis network g에서 style을 입혀주는 데에 사용한다. 이때, PGGAN과 같이 model에 직접적으로 넣어주는 것이 아닌, 모든 디코더의 layer마다 style을 입혀주며 학습을 시킨다는 중요한 특징이 있다.

PGGAN과 styleGAN 모두 단계적으로 해상도를 상승시키는 Progressive Growing 방식을 사용하지만, PGGAN과 달리 styleGAN은 고정된 learned constant tensor(4x4x512)를 이용해 이미지를 생성한다. PGGAN의 경우 학습 초기에 random input을 사용하여 generator의 초기 이미지를 생성한 반면, styleGAN은 image features가 w와 AdaIN으로 control이 되므로 PGGAN과 같이 초기 입력을 랜덤하게 넣어주지 않고, 이미 학습되어진 tensor를 사용하게 된다. 이런 식으로 feature entanglement를 줄일 수 있으며 network가 input vector에 의존하지 않고 mapping network를 거쳐 나온 w로만 사용하는 것이 network 학습에 더 쉽다.

1.3 스타일 변환을 가능하게 하는 AdaIN

AdaIN(Adaptive Instance Normalization)은 Xun Huang에 의해 2017년 제안된 기존 이미지에 스타일을 적용하여 원하는 스타일의 이미지로 생성하는 것을 돕는 정규화 방법으로, 콘텐츠 입력 x와 스타일 입력 y를 이용해 정규화한다[수식 1].

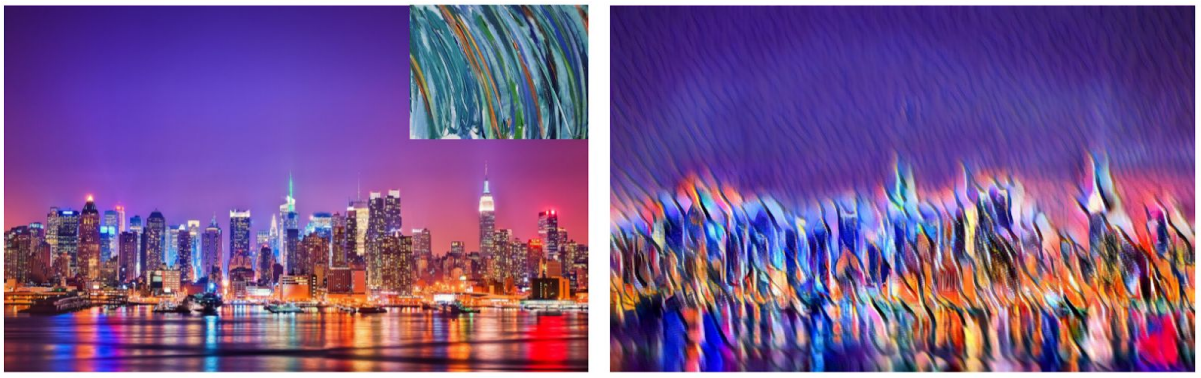
$$AdaIN(x,\ y)\ =\ \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right)\ +\mu(y)\ \cdots\ (1)$$

AdaIN은 [그림 3, 4]와 같이 만약 우리가 적용하고자 하는 스타일의 이미지의 느낌을 살려서 원본 이미지를 재가공하는 것을 가능하게 해준다. AdaIN은 스타일과 콘텐츠 이미지의 총합량만을 정규화하고, 학습 파라미터를 사용하지 않는다. 이에 따라 훈련 데이터에서 본 적 없던 스타일을 적용하는 스타일 변환이 가능하다.

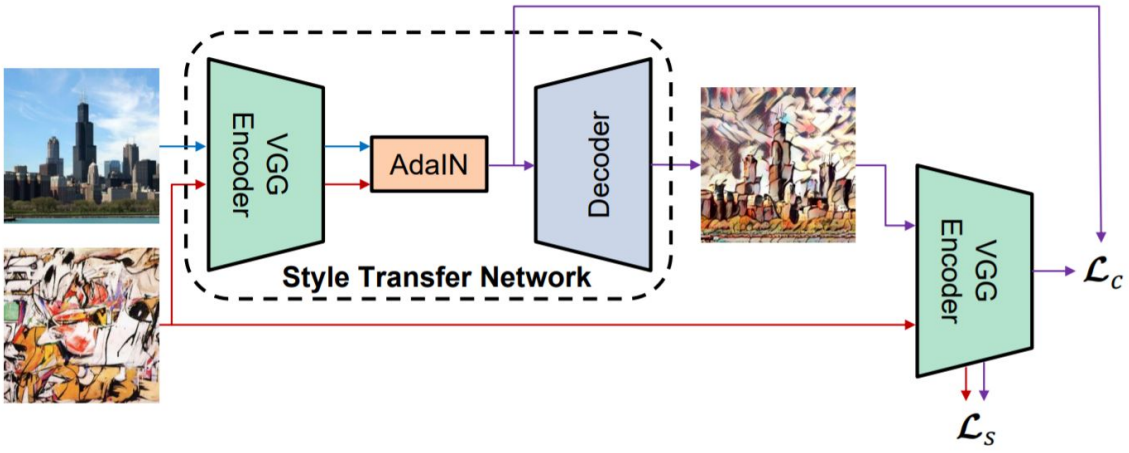
StyleGAN에서의 AdaIN은 [수식 2]의 수식을 사용한다. 정규화된 콘텐츠 정보에 스타일을 사용한 선형 변환을 적용한다는 개념에는 변화가 없지만, 스타일의 표준편차와 평균치 대신 스타일 벡터 w에 선형 변환을 더한 y_s와 y_b 2개의 값을 사용한다. StyleGAN은 non-linear function인 Mapping Network f에서 입력되는 z를 intermediate space W에 매핑하여 w를 만든다. 이 w를 통해 y라는 스타일을 얻고, AdaIN이라는 operations을 통해 각각 Upsample, Conv layer 이후에 적용시켜 style을 입혀 학습시키게 된다[그림 2 (b)]. 이때 w의 shape은 1x512이므로 AdaIN을 이용하여 style을 입힐 때, 즉 style y로 변환할 때, Affine Transformation을 거쳐 shape을 맞춰주게 된다. Style을 입힌다는 것의 의미는 y_s (scale)를

입히고, y_b (bias)를 더하는 과정을 의미한다[수식 2]. 이 style y 는 각 convolution layer를 거치며 AdaIN의 처리에 사용되고, generator를 control한다.

$$AdaIN(x_i, y) = y_{s,i}\frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i} \quad \cdots (2)$$



[그림 3] 왼쪽은 원본 이미지와 적용을 원하는 스타일의 이미지이고, 오른쪽은 왼쪽 이미지 상단의 스타일을 원본 이미지에 적용한 결과물이다[2].



[그림 4] 왼쪽 상단의 이미지에 하단의 이미지의 스타일을 적용하여 스타일 변환을 하면 오른쪽의 이미지가 생성된다[2].

1.4 Style mixing

Mapping network f 로부터 만들어진 intermediate vector w 는 synthesis network g 의 모든 layer에 적용될 style을 표현하도록 학습하게 된다. StyleGAN의 generator architecture를

보면, 각 layer(해상도)별로 intermediate vector w가 사용되는데, layer별로 w(style)를 넣어주기 때문에 각 layer가 넣어준 w에 대해서만 style과 correlate되게 된다. 따라서 각 layer에 대해 style이 correlate되는 문제를 해결하기 위해 style mixing 기법을 사용할 수 있다. 낮은 해상도의 이미지는 얼굴의 포즈, 머리 모양, 안경 여부와 같은 대략적인 요소를 표현하고, 이미지의 해상도가 올라갈수록 얼굴 특징, 눈의 특징과 같은 세부적인 요소를 표현하게 된다. StyleGAN은 PGGAN과 같이 해상도를 올려가며 학습하는 구조이기 때문에, w의 해상도를 어느 정도로 할지, 즉 적용하고자 하는 스타일의 해상도를 어느 정도로 할지를 결정함으로써 스타일이 생성될 이미지에 어느 정도까지 영향을 미칠 지를 결정할 수 있게 된다.

2. StyleGAN2 [3-4]

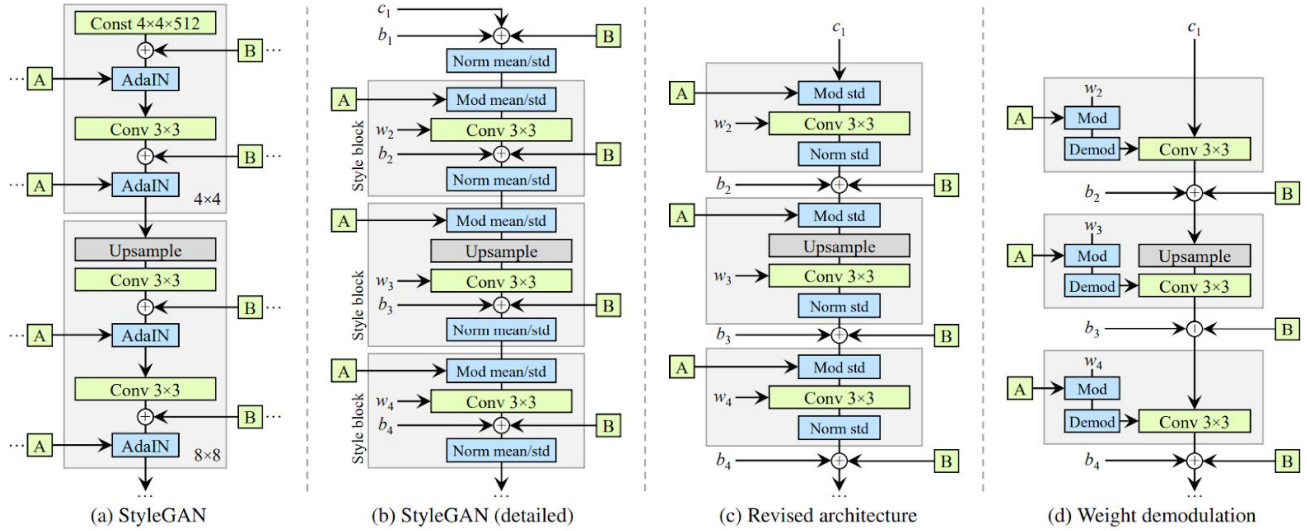
StyleGAN은 고해상도의 이미지 생성에 좋은 성능을 보이지만, 몇 가지 문제점이 있다. [그림 5]와 같이 물방울 모양의 노이즈가 64x64의 모든 feature map에서 발생하는 것과, [그림 6]와 같이 입, 눈과 같은 일부 feature가 얼굴의 움직임에 따른 배열을 따르지 않아 부자연스러운 이미지가 생성되는 것이다. StyleGAN2에서는 이런 문제점들과 latent space를 확장하여 생성하는 이미지의 품질을 개선했다.

[그림 5] StyleGAN의 64x64 feature map에서 물방울 모양의 노이즈가 관찰되었다[3].

[그림 6] StyleGAN으로 생성한 이미지에서 입이 얼굴 배열을 따라가지 못하는 현상이 관찰되었다[3].

2.1 Normalization method instead of AdaIN

StyleGAN에서 사용되는 정규화 기법인 AdaIN은 실제로 넣은 데이터의 통계를 사용하여 정규화(normalize)하는데, 이는 물방울 모양의 노이즈를 발생시킨다[그림 5]. StyleGAN2에서는 실제 데이터의 통계보다, 추정 통계(estimated statics)를 사용하여 합성곱(convolution) 가중치를 정규화하여 물방울 노이즈를 예방하였다.



[그림 7] StyleGAN2에서 Redesign된 synthesis network[3].

AdaIN의 단계는 자체 통계로 콘텐츠 정보를 정규화(normalization)하는 것과 스타일 정보를 사용하여 정규화된 콘텐츠 정보의 선형 전이(linear transition)하는 것으로 나눌 수 있다. [그림 7 (a)]의 AdaIN이 이에 따라 확장되면 [그림 7 (b)]와 같이 된다. AdaIN의 내부 작업은 콘텐츠의 정규화를 거쳐 스타일 벡터에 의해 선형 변환이 되는 순서로 진행된다. 스타일 블록에서는 스타일 벡터에 의한 선형 변환을 거쳐, 합성곱, 콘텐츠의 정규화 순으로 진행된다[그림 7 (b)]. 여기까지가 StyleGAN에서 기존에 사용하고 있는 AdaIN의 작업 수행 방식이다.

StyleGAN2에서는 이 architecture를 수정하여 사용한다. [그림 7 (c)]를 보면, 평균값으로 작업하지 않고 정규화 작업을 표준 편차로만 나눈다. 스타일의 선형 변환을 계수를 곱하여 수행하고, 특히 노이즈 삽입 부분을 스타일 블록에서 꺼냈다. 또한, 스타일 블록 내부 작업을 기존 방식보다 단순화했는데, 스타일 벡터에 의한 첫 번째 선형 변환은 합성곱 연산 내부에서 처리해 수행하도록 수정하였다[그림 7 (d)]. 스타일 블록에서 스타일 벡터 W 의 선형 변환의 계수 y_s 가 사용된다. 합성곱 가중치 w_{ijk} 와 함께 s 를 곱한 콘텐츠 이미지 처리 작업에서는 콘텐츠 이미지를 가중치 w_{ijk} 와 s 의 곱과 함께 합성곱 연산하는 것과 같다고 할 수 있다. 따라서 이 작업은 [수식 3]과 같이 다시 쓰일 수 있다.

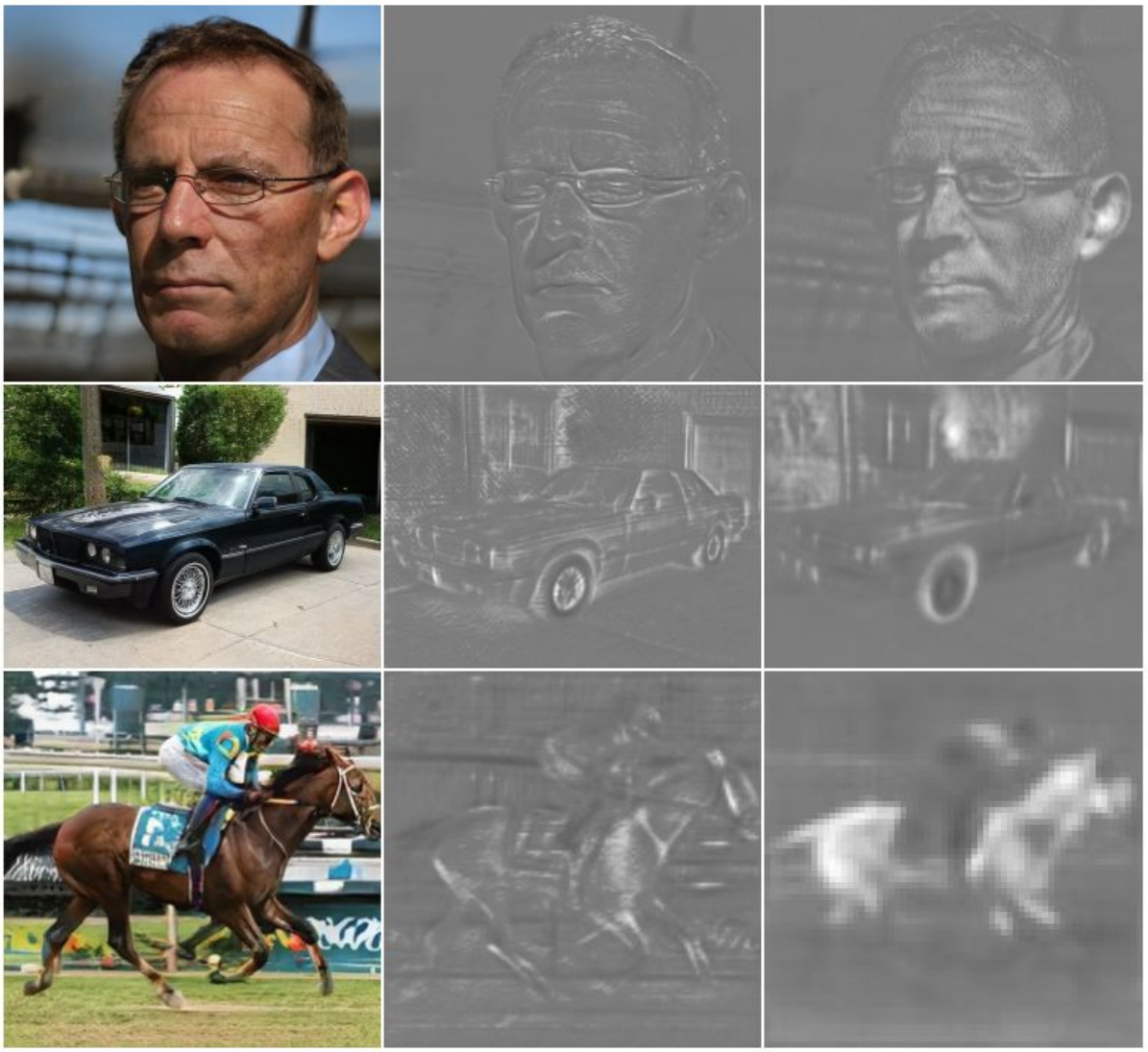
$$w'_{ijk} = s_i \bullet w_{ijk}, \quad \cdots (3)$$

입력이 표준 정규분포와 출력의 표준편차를 따르는 것으로 가정할 때, 합성곱 내부 처리에서의 정규화 작업에서 원하는 것은 표준 편차의 역수로 출력을 곱하는 것이라 할 수 있다. 따라서 가중치 w_{ijk} 와 합성곱의 곱셈에 표준 편차의 역수를 곱하는 작업은 표준 편차의 역수에 곱한 가중치 w_{ijk} 와 합성곱하는 것과 같다. 즉, 정규화 작업은 [수식 4]와 같이 수행된다.

$$w''_{ijk} = w'_{ijk} / \sqrt{\sum_{i,k} w'^{2}_{ijk} + \varepsilon}, \quad \cdots (4)$$

이로 인해, 스타일 블록의 작업 순서는 스타일에 의한 선형 변환을 거쳐, 합성곱, 출력 정규화 순서로 이루어지고, 이것은 하나의 합성곱 과정으로 표현이 가능하다. 정규화 부분은

출력이 정규 분포라 가정한 정규화 과정으로, 물방울을 발생시키는 실제 분포를 사용한 정규화는 사용되지 않는다. 수정된 normalization architecture를 사용한 결과물은 [그림 8]과 같다.

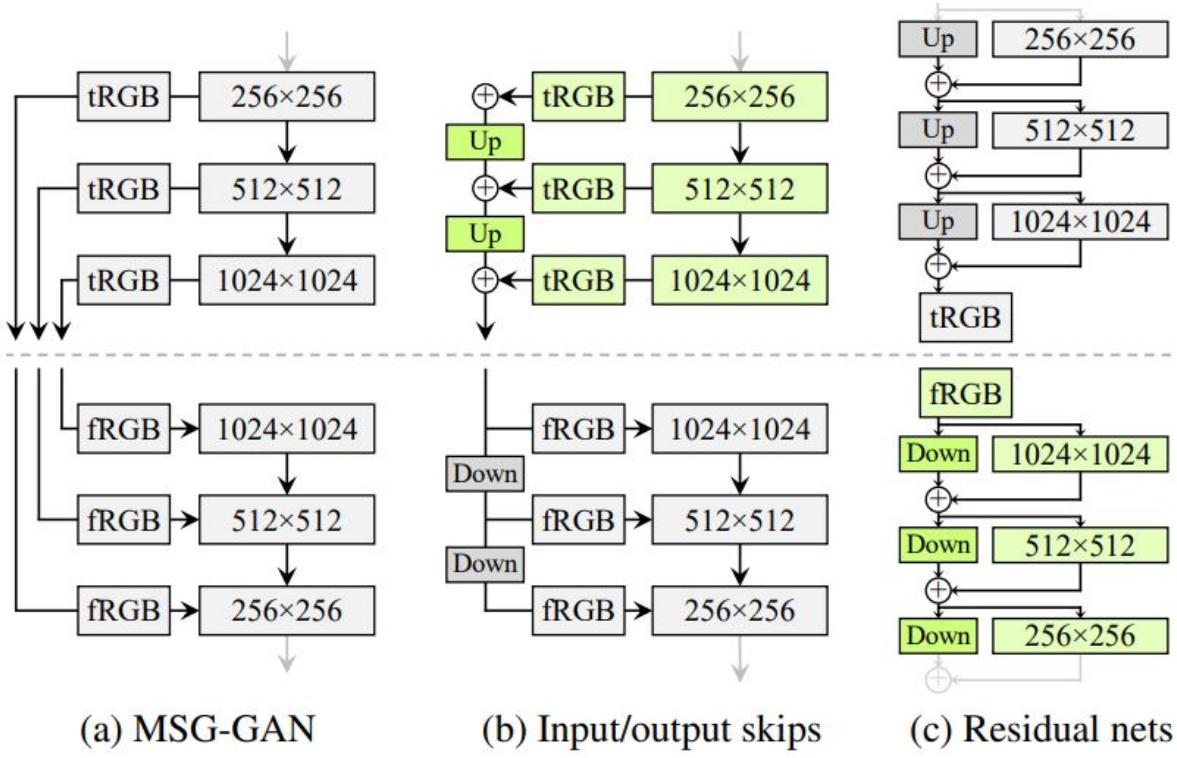


[그림 8] StyleGAN의 synthesis architecture를 수정하여 이미지를 생성하자, 64x64의 feature map에서도 물방울 모양의 노이즈는 발견되지 않았다[3].

2.2 A high-resolution image generation method instead of Progressive Growing

기존의 styleGAN generator는 단순한 구성을 갖고 있어, Progressive Growing 없이 고해상도의 이미지를 생성하는 것이 어려웠다. 그러나 generator와 discriminator의 표현 능력을 향상시켜, 고해상도의 이미지를 progressive growing 없이도 생성할 수 있게 되었다. Progressive Growing은 고해상도를 생성하기 위해 generator와 discriminator를 점차적으로 추가하는 방식이다. 이때, 각 generator가 독립되어 있기 때문에 입매가 얼굴의 방향을

따라가지 않는 [그림 6]와 같은 사진이 생성되기도 한다. 따라서 styleGAN2에서는 Progressive Growing을 사용하지 않고, 고해상도의 이미지를 생성하는 방법을 새롭게 제안했고, 네트워크는 MSG-GAN[그림 9 (a)]과 비슷하다. 최선의 네트워크를 선택하기 위해 styleGAN2를 제안한 논문에서는 (b) 형태의 generator/discriminator와 (c) 형태의 generator/discriminator의 모든 조합을 실험하고, 결과가 가장 좋은 것을 선택하였다.



[그림 9] MSG-GAN의 generator & discriminator (a)와 새롭게 구성하여 실험한 (b), (c)의 generator & discriminator의 architecture[3].

FFHQ와 LSUN Car 데이터 셋으로 실험을 한 결과, (b)의 generator를 사용하는 것은 perceptual path length(PPL)을 상당히 개선하였고, (c) 형태의 discriminator를 사용하는 것은 Frechet inception distance(FID)를 개선했다. 따라서 styleGAN2에서는 (b)의 generator와 (c)의 discriminator를 채택하여 기존의 PGGAN 형식의 Progressive Growing 방식에서 나타나는 입매 또는 눈가 불일치 문제를 해결하였다. 이런 새로운 네트워크를 사용하여 생성한 이미지는 [그림 10]와 같다. [그림 6]와 달리, styleGAN2에서는 얼굴 방향의 변화와 함께 자연스럽게 변한 것을 관찰할 수 있다.

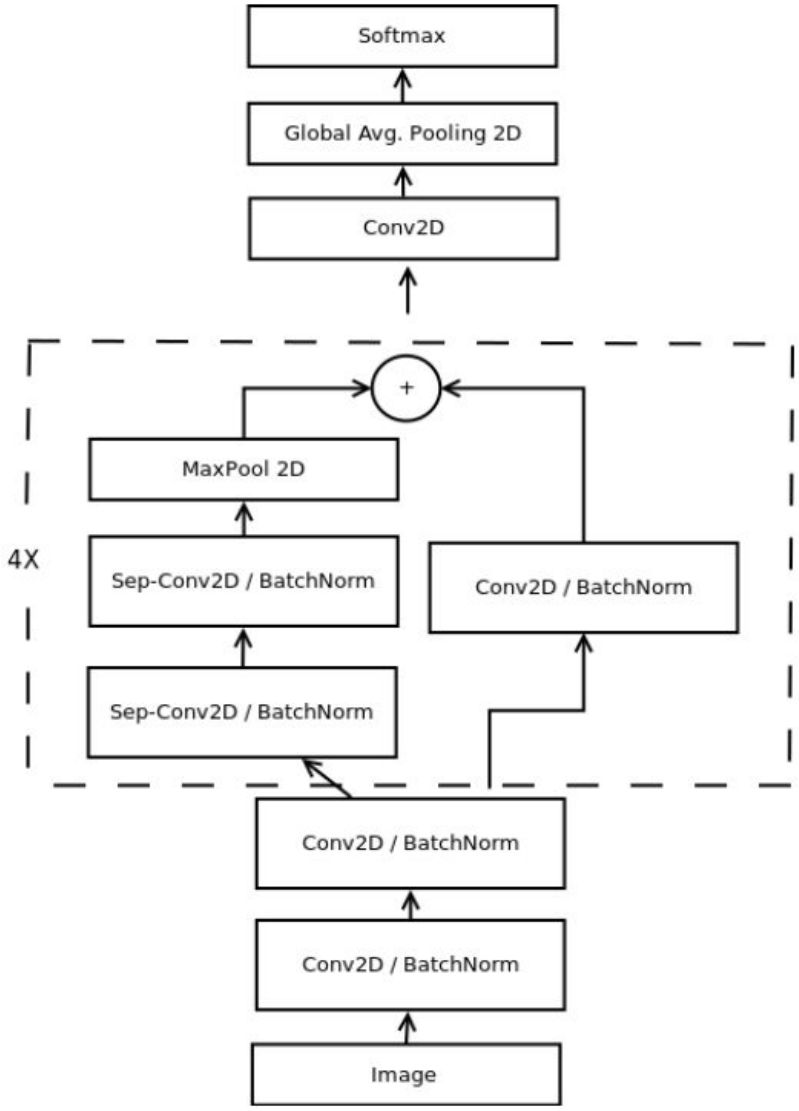
[그림 10] StyleGAN2에서 생성한 사람 얼굴의 이미지[3]. 얼굴 방향의 변화에 따른 눈가와 입매가 자연스럽다.

3. Emotion Classification

Object detection을 위한 Convolutional Neural Networks(CNNs)는 일반적으로 수백만 개의 파라미터를 가지는 구조이다. 이런 CNN architecture를 실시간으로 import하여 object detection 작업을 수행하는 것은 불가능에 가깝다. 따라서 Arriaga, O. et al.은 real-time으로 emotion과 gender classification이 가능한 CNN 모델을 제안하였다. 파라미터의 수를 줄임으로써 1) hardware에서의 느린 수행 속도를 개선하고, 2) residual 모듈과 결합된 depth-wise separable convolutions과 fully connected layer의 deletion을 결합해준다.

해당 연구에서는 'mini-Xception'이라는 architecture를 제안하고 있다. 이 모델의 구조는 4개의 residual depth-wise separable convolution을 포함하는 fully-convolutional neural network이다. 각각의 convolution은 ReLU의 activation function과 batch normalization operation을 따르고 있다(followed by). 마지막 layer에는 예측 결과를 생성하기 위해 global average pooling과 soft-max activation function을 적용한다. 이 구조는 대략 6만 여 개의 파라미터를 포함하고 있고, 이는 원래 CNN 구조에 비해 80배 감소된 수치이다.

[그림 11]은 'mini-Xception'의 architecture를 보여준다. 이 architecture는 FER-2013(감정 표현별로 사람의 얼굴을 모아 놓은 데이터 셋)의 emotion classification 작업에서 66%의 정확도를 보인다. 파라미터의 감소로 인해 architecture의 계산 비용이 줄고, real-time에서 사용이 가능하다.



[그림 11] mini-Xception의 architecture[4].

V. 구현

1. Requirements

1.1 StyleGAN2

Google Colab에서 tensorflow 1.x 버전(GPU 지원)을 사용해서 빌드하였다. NVLab에서 배포한 StyleGAN2을 dvschultz가 fork한 StyleGAN2를 이용하여 model을 학습시켰다. 미리 학습된 network는 NVLab에서 배포한 1024x1024에 최적화된 FFHQ styleGAN2 모델을 사용하였다. 사용한 StyleGAN2 코드는 아래의 링크에서 자세히 확인할 수 있다.

https://github.com/dvschultz/stylegan2.git

1.2 Emotion Classification

python 3.7 환경에서 GPU는 NVIDIA gtx 1650 super를 사용하였고, CUDA 10.1, CuDNN 7.6, tensorflow-gpu 2.1.0을 이용하였다. 소스코드로는 mini-Xception을 사용하여 real-time으로 성별과 표정을 인식한 연구[4]의 소스코드에서 표정 인식 부분만을 사용하여 구현한 GitHub 프로젝트를 사용했다. 자세한 코드의 내용은 아래 링크에서 확인 가능하다.

https://github.com/omar178/Emotion-recognition.git

2. 진행 상황

2.1 Generate personal emoji from face image

[그림 12] kimg = 10,277인 emoji styleGAN2 model의 스냅샷.

[그림 13] [그림 12]의 모델에 왼쪽 샘플 사진을 넣은 결과.

[그림 14] kimg = 10,016인 emoji styleGAN2 model의 스냅샷.

[그림 15] [그림 14]의 모델에 왼쪽 샘플 사진을 넣은 결과.

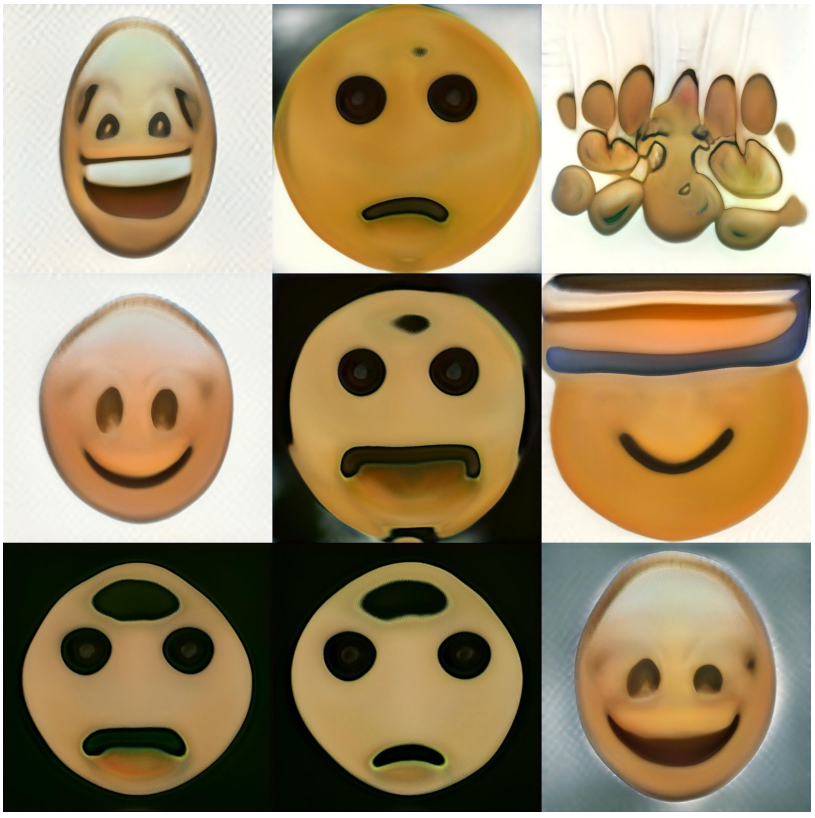
현재 training은 kimg(사진 1,000장단 학습 길이. Epoch와 유사한 개념.)의 절반 정도 수행이 완료 되었고, 그 결과 생성되는 스냅샷은 [그림 12]와 같다. 생성된 이미지에서 사람의 얼굴 특징을 거의 찾을 수 없는 [그림 12]에 사람 얼굴의 샘플 이미지를 넣으면 결과물이 [그림 13]과 같이 나온다. 그러나 사람 얼굴의 흔적이 많이 남아 있는 [그림 14]의 모델에 샘플 이미지를 넣으면 [그림 15]와 같이 옷과 얼굴 부분들의 흔적이 남아 있음을 볼 수 있다.

StyleGAN2에서 pretained network로 FFHQ(사람 얼굴) model이 사용되었는데, kimg가 커질수록 사람 얼굴의 특징들이 사라져 가는 것을 관찰할 수 있다. 향후, kimg가 낮고 사람 얼굴의 특징이 많이 드러나는 모델을 적용한 샘플 이미지[그림 15]와, 추후 완성된 모델에 FFHQ model을 blending 해주어 생성한 모델에 적용한 샘플 이미지 중 더 좋은 성능의 모델을 선택할 계획이다.

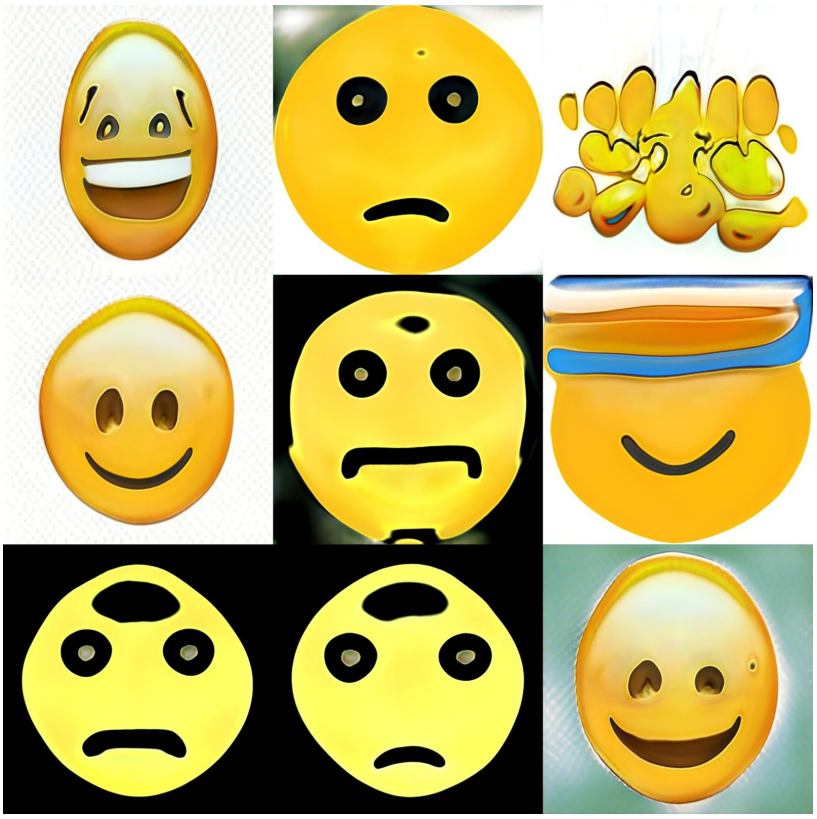
[그림 16]부터 [그림 19]까지는 현재까지 완성된 모델[그림 12]과 FFHQ model을 blending하여 해상도별 스냅샷을 도출한 결과이다. 아직 emoji model이 완벽히 만들어지지 않아 결과물의 질이 좋지 않지만, emoji model의 해상도를 높여가며 적용할수록 특징이 더 많이 드러나는 것을 관찰할 수 있다. Network blending은 layer swapping 기술을 적용한 모듈로, 저해상도부터 고해상도까지 원하는 만큼의 스타일을 적용시킬 수 있도록 구현되어 있다. 스타일을 가져오기를 원하는 모델과 적용시키기를 원하는 모델의 배치 사이즈가 동일하고, styleGAN2 모델이라면 서로 다른 모델들을 blending하는 것이 가능하다.

[그림 16] FFHQ styleGAN2 model과 [그림 12]의 모델을 8x8로 blending한 결과.

[그림 17] FFHQ styleGAN2 model과 [그림 12]의 모델을 64x64로 blending한 결과.

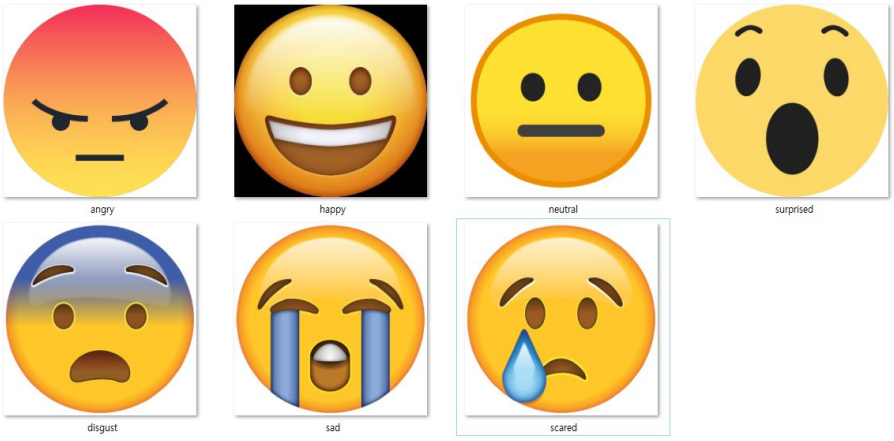


[그림 18] FFHQ styleGAN2 model과 [그림 12]의 모델을 256x256로 blending한 결과.



[그림 19] FFHQ styleGAN2 model과 [그림 12]의 모델을 1024x1024로 blending한 결과.

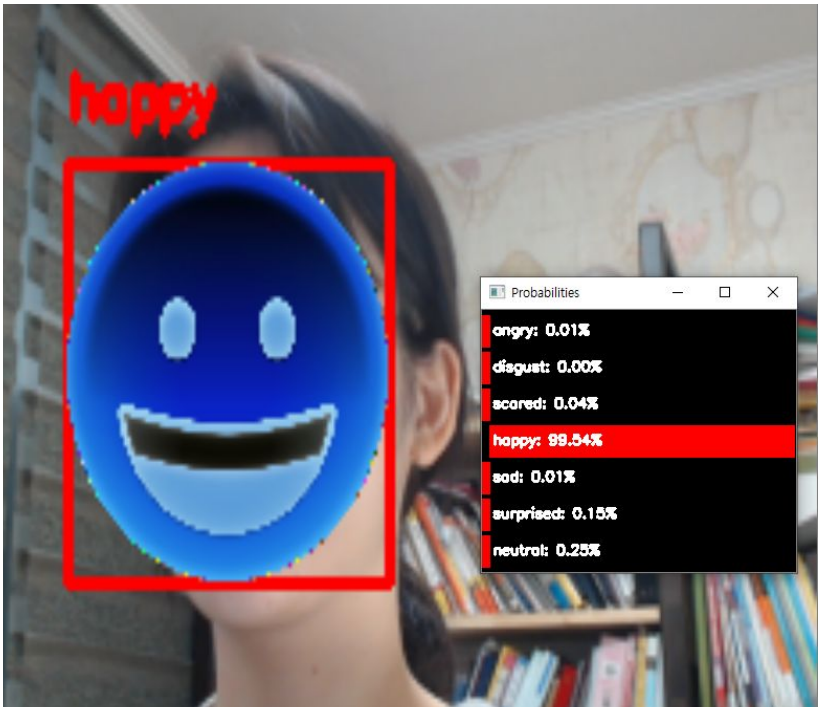
2.2 Emotion recognition in real-time



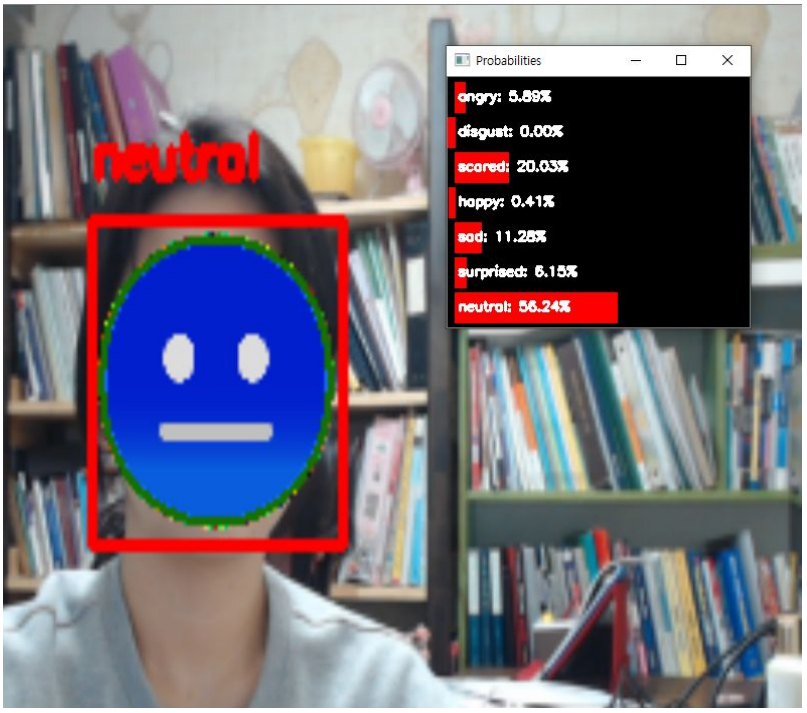
[그림 20] 사용한 emoji의 png 파일.

[그림 20]의 emoji들을 검출한 얼굴 영역 위에 렌더링하도록 구현하였다. 구현 결과는 [그림 21]과 [그림 22]와 같다. 테스트를 해본 결과, “angry”, “disgust”, “scared”는 잘 인식하지 못하는 것을 느꼈다. 성능 향상을 위해서 데이터 셋을 추가하거나, 감정 목록을 조정하고,

mini-Xception의 구조에서 기여할 수 있는 부분을 찾아야 할 것 같다. [그림 21]과 [그림 22]를 보면, 렌더링한 이모티콘의 색상이 RGB로 표현되지 않는 것을 볼 수 있다. 렌더링 과정에서 문제가 생긴 것으로 추정되는데, 추후 수정할 계획이다.



[그림 21] 웃으면 'happy'라는 감정으로 인식이 된다.



[그림 22] 무표정일 때에는 'neutral'로 인식이 된다.

VI. References

[1] Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4401-4410). https://arxiv.org/abs/1812.04948v2

[2] Huang, X., & Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1501-1510). https://arxiv.org/abs/1703.06868

[3] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8110-8119). http://arxiv.org/abs/1912.04958

[4] Arriaga, O., Valdenegro-Toro, M., & Plöger, P. (2017). Real-time convolutional neural networks for emotion and gender classification. arXiv preprint arXiv:1710.07557. https://arxiv.org/abs/1710.07557