4주차 face recognition

Face verification vs. face recognition

→ Verification 1:1 99%
- Input image, name/ID
- Output whether the input image is that of the claimed person

face verification: 사람의 이름 또는 id와 이미지가 주어졌을 때 입력이미지가 그 사람인지 검증하는 것. 1:1문제

→ Recognition 1:K K=100
- Has a database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or "not recognized")

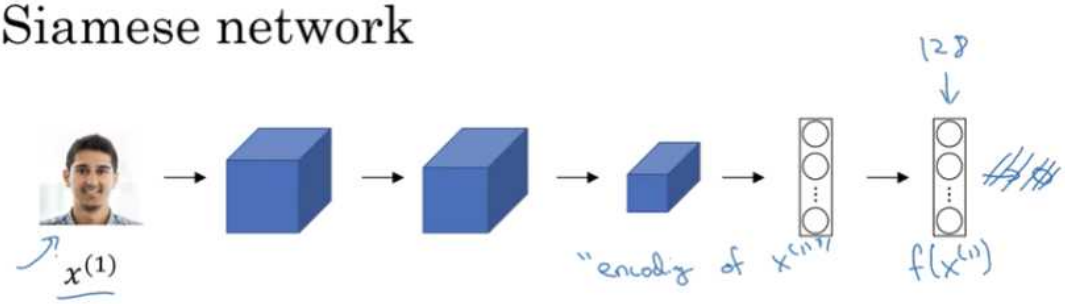
face recognition: 1:K 문제로 K명의 데이터베이스가 있으면 주어진 이미지를 통해서 K명의 데이터베이스에 속하는 사람인지를 판단

4-2 One Shot Learning

- One Shot Learning: 소량의 데이터(1개) 샘플만으로 새로운 클래스를 학습하는 방법
- 샘플의 개수가 적어 학습이 어려운 상황에서 얼굴인식과 원샷학습을 수행하기 위해서, 유사도 함수를 학습시킴
- 유사도 함수 학습: 신경망에서 두 이미지간 차이 정도를 반환하는 함수(d)를 학습 - 두 이미지가 같은 사람이라면 작은 숫자를 반환, 다르다면 큰 숫자를 반환 => 임계값보다 작으면 같은 사람, 크다면 다른 사람이라고 예측

4-3 Siamese Network

Siamese network



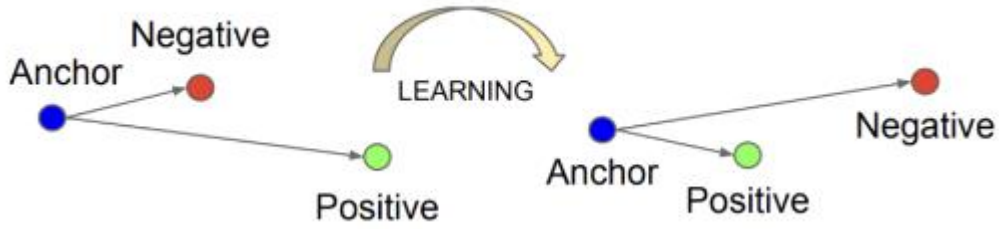
- 일반적인 합성곱 신경망: 이미지(x^1)를 입력하면 합성곱, 풀링, FC층을 순차적으로 지나 feature vector로 마무리되고 softmax unit으로 분류
- 여기에서는 softmax를 사용하지 않고 그 전의 벡터에 집중 => 이미지 x^1를 입력받아 신경망은 128개의 숫자로 된(128차원의) 벡터 반환(이미지를 잘 표현하는 벡터 인코딩)
- Siamese Network: 두 개의 입력 이미지(x^1, x^2)를 비교하고 싶다면, 같은 신경망에 같은 변수와 두 번째 사진을 넣어서, 두 번째 사진을 인코딩하는 또다른 128개의 숫자로 된 특징 벡터를 얻어, 들의 차이를 계산해서 유사도를 측정
- 입력되는 두 이미지가 비슷하다면 유사도는 작은 값, 다르다면 유사도는 큰 값

4-4 Triplet loss

얼굴 이미지에 대한 좋은 인코딩을 얻기 위해서 신경망의 변수들을 훈련시키는 방법 중 하나는 Triplet loss 함수에 경사하강법을 정의하고 적용하는 것

Triplet Loss: 입력 데이터인 Anchor(기준 데이터), Positive(같은 사람), Negative(다른 사람)이미지를 항상 동시에 보면서, 구분해 나누어 처리하는 손실 함수

α: anchor와 positive 쌍, anchor와 negative 쌍 사이의 거리 차이를 보정하기 위한 margin: 그 둘의 차이가 하이퍼파라미터 α 이상이 되도록 함(둘의 사이가 멀어지도록 함)



$$\mathcal{L}(A,P,N) = \max\left(\|f(A)-f(P)\|^2 - \|f(A)-f(N)\|^2 + \alpha, 0\right)$$

- A,P,N의 손실함수 = 둘 사이의 최댓값을 구함: 식이 0보다 작을 때는, 최댓값이 0이기 때문에 손실은 0임
- 하지만 식이 0보다 클 때는, 즉 Anchor와 Positive 사이 거리가 Anchor와 Negative 사이 거리보다 크다면 손실 발생
- 따라서 손실을 최소화하기 위해서는, 식을 0보다 작거나 같게 만들어야하며, 이게 얼마나 음수인지는 신경망이 신경쓰지 않음

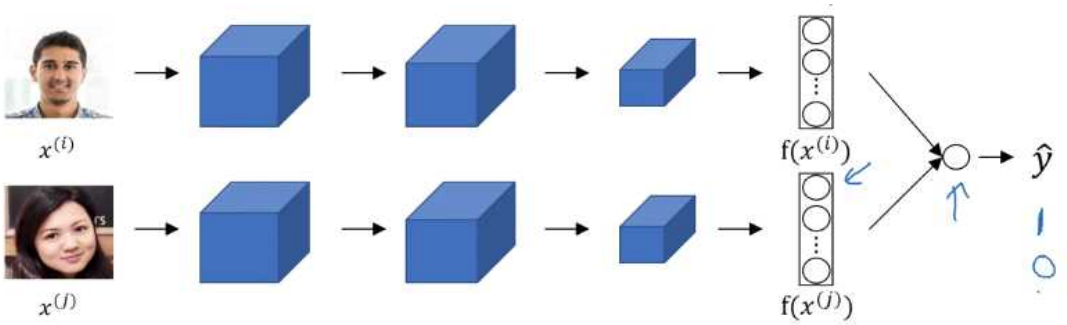
$$J = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

신경망에 대한 전체적인 비용함수는 훈련세트의 각각의 삼중항(A,P,N)들에 대한 손실의 합이 됨

-> 경사하강법을 사용해 비용함수를 최소화시키는 방향으로 신경망을 훈련시킴

- 삼중항 데이터 세트를 정의하기 위해서 A와 P의 쌍들이 필요함 – 시스템을 훈련시키기 위해서 같은 사람에 대한 많은 이미지의 데이터 세트 필요 (-> 여러 사진으로 시스템을 훈련시킨 후에 얼굴 인식 시스템의 원샷 학습 문제에 적용 가능)

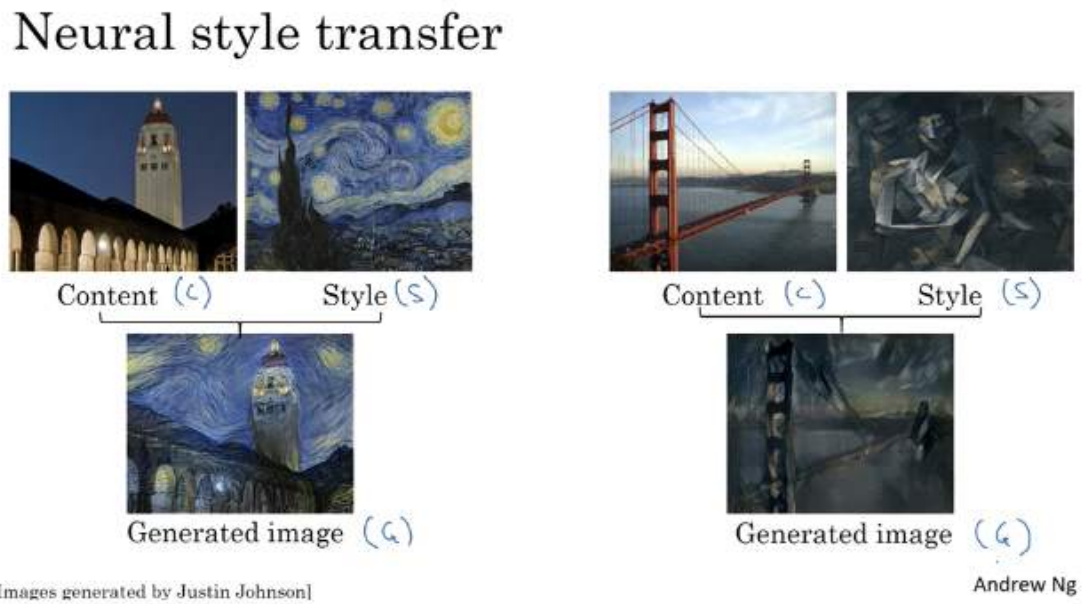
4-5 Face Verification



- 얼굴 인식 문제를 이진 분류 문제로 표현하는 법(삼중항 손실의 대안): 이미지를 128개의 feature vector로 나타내기 위해 ConvNet(샴신경망)을 통과시켜 embeding(encoding)시키고, 128개의 feature vector를 Logistic Regression을 통해 예측하도록 함. target output은 같은 사람일 경우 1을 출력하고, 다른 사람일 경우 0을 출력-> Binary Classification
- 역전파를 사용해 샴신경망을 훈련하기 위해 서로 다른 쌍들을 사용함

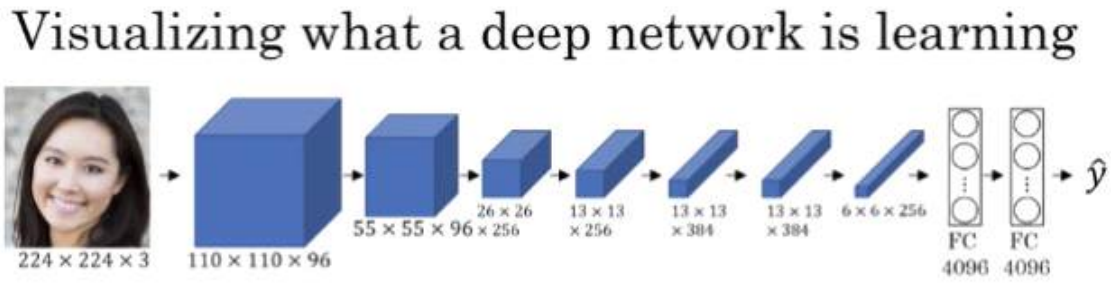
database에서 이미지가 입력으로 사용되면, 매번 embeding할 필요 없이 미리 계산되어 embeding 된 값과 비교해서 예측에 사용할 수 있다. 계산량을 줄여 효율이 높아진다.

4-6 What is neural style transfer?

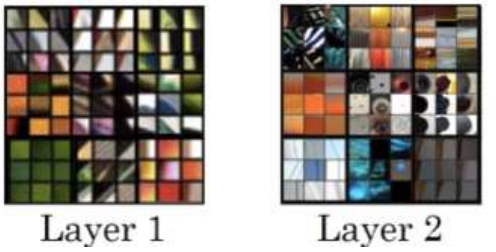


이미지를 새로운 스타일로 변형: 이를 구현하기 위해서는 신경망의 얕은 층과 깊은 층에서 추출된 특성을 살펴보아야함

4-7 What are deep CNs learning? 심층 신경망이 실제로 학습하는 것은 무엇인가?

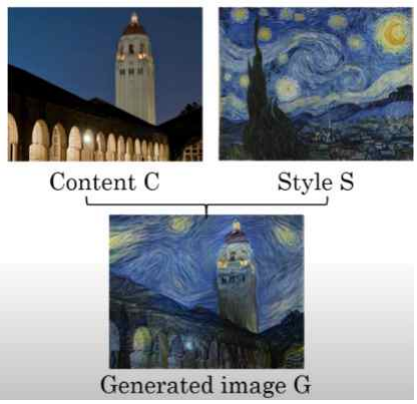


- AlexNet의 은닉 유닛과 다른 층들이 무엇을 계산하는지 시각화해보자
- 훈련세트가 신경망을 통과하도록 하고 어떤 이미지가 그 유닛의 활성값을 최대화하는지(고도로 활성화시키는지) 찾음
- 1층에서 훈련된 은닉유닛들은 모서리 혹은 색깔과 같은 단순한 feature을 찾음
- 깊은 층의 은닉유닛으로 갈수록 이미지의 더 큰 부분을 보고, 더욱 복잡한 패턴을 감지



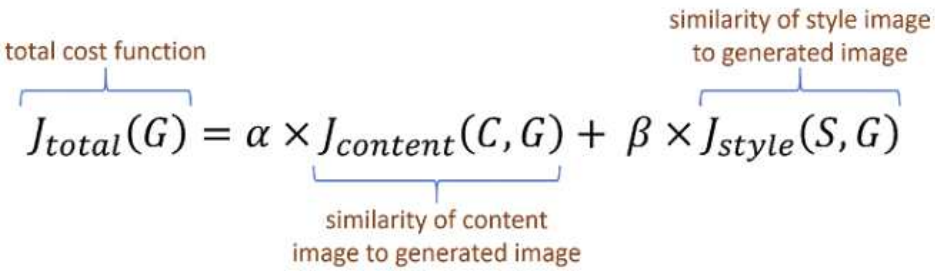
4-8 Cost Function

neural style transfer system을 만들기 위해 생성 이미지(G)에 대한 비용함수를 정의해보자-> 비용함수을 최소화해(경사하강법 이용) 원하는 이미지를 생성할 수 있음



Neural Style Transfer Cost Function

* J_content(C,G): 내용(C), 생성(G)이미지의 함수: C와 G의 내용이 얼마나 비슷한지 측정
* J_style(S,G): 스타일 비용 함수: S와 G의 스타일이 얼마나 비슷한지 측정
* 하이퍼파라미터) α: content weight , β: style weight
- 하이퍼파라미터를 사용해 내용비용과 스타일 비용사이의 가중치를 둠



새로운 이미지를 생성하기 위해 해야할 것

1. Initiate G randomly

   G: $100 \times 100 \times 3$

2. Use gradient descent to minimize $J(G)$

무작위로 초기화해서 백색 노이지의 이미지 G-> 점점 비슷해짐

4-9 Content Cost Function

Neural style transfer의 비용함수에는 Content Cost Function, Style Cost Function이 있음

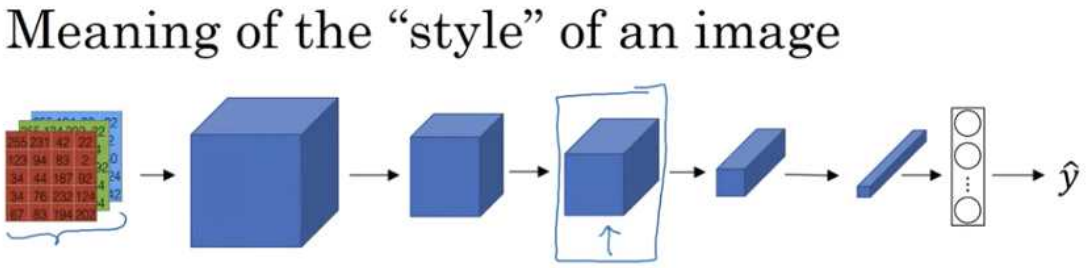
$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

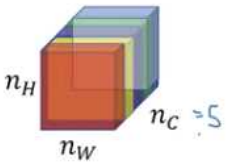
- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

Content Cost Function(J_content(C,G)): content image와 generated image의 유사도 (값이 작을수록 두 이미지는 유사하다는 의미)

$$J_{\text{content}}(C, G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

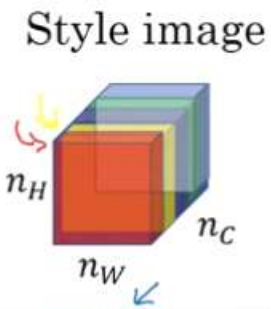
4-10 Style Cost Function

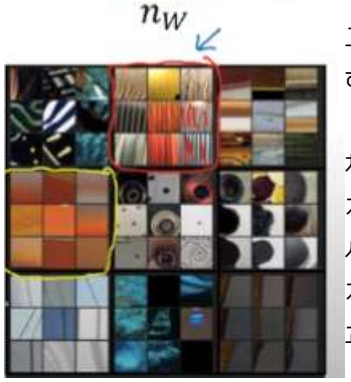




How correlated are the activations across different channels?

style의 의미: image style의 측정을 정의하기 위해, 선택한 l층에서의 서로 다른 채널들의 활성값 사이의 상관관계





채널 간 활성 값이 어떻게 상관되어 있는지 보자.
빨간색 채널과 노란색 채널이 있을 때 빨간색 채널은 세로선 텍스쳐를 찾으려고 하고 노란색 채널은 주황색을 찾으려고 한다고 가정.
두 채널이 높은 상관관계를 가진다는 것은 이미지 조각이 세로선을 가질 때 주황색을 가질 것이라는 것
상관관계가 없다면 세로선 텍스쳐가 있어도 주황색이 아님.

고차원 특성들이 서로 다른 이미지 부분에서 얼마나 자주 함께 발생하는지 여부를 측정

채널 사이의 상관관계 정도를 스타일의 측정으로 사용하면 생성이미지(G)에서 빨간색 채널과 노란색 채널이 상관되는지 정도를 측정하면, 세로선 텍스쳐와 주황색 색조가 얼마나 자주 함께 발생하거나 발생하지 않는지를 알려줌 => 생성이미지의 스타일이 입력이미지의 스타일과 얼마나 비슷한지에 대한 측정을 말함

Style matrix

Let $a^{[l]}_{i,j,k}$ = activation at (i, j, k). $G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

$$G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{ijk} \, a^{[l](S)}_{ijk'}$$

$$G^{[l](G)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](G)}_{ijk} \, a^{[l](G)}_{ijk'}$$

"Gram matrix"

Style cost function

$$J^{[l]}_{style}(S, G) = \frac{1}{\left(2 n_H^{[l]} n_W^{[l]} n_C^{[l]}\right)^2} \sum_k \sum_{k'} \left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'}\right)$$

$$J_{style}(S, G) = \sum_l \lambda^{[l]} J^{[l]}_{style}(S, G)$$