Convolutional Neural Networks

Week 4

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

99.9

→ Recognition

1:K

- Has a database of K persons
- Get an input image

K=100

- Output ID if the image is any of the K persons (or "not recognized")

One-shot learning

Learning from one example to recognize the person again

Learning a "similarity" function

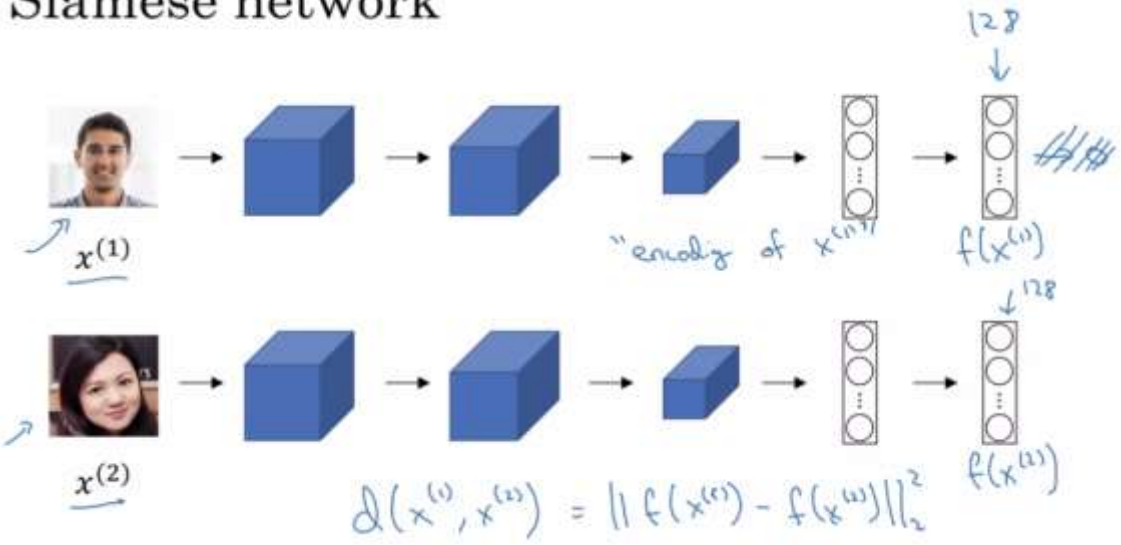
→ d(img1,img2) = degree of difference between images

If d(img1,img2) $\leq \tau$ "same"

$> \tau$ "different"

} Verification.

Siamese network



Parameters of NN define an encoding $f(x^{(i)})$ — 128

Learn parameters so that:

If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

Ar

Triplet Loss

Learning Objective

Anchor (A) | Positive (P) | Anchor (A) | Negative (N)

Want: $\underbrace{\|f(A)-f(P)\|^2}_{d(A,P)} + \alpha \leq \underbrace{\|f(A)-f(N)\|^2}_{d(A,N)}$

$\underbrace{\|f(A)-f(P)\|^2}_{0} - \underbrace{\|f(A)-f(N)\|^2}_{0} + \alpha \leq 0$ (margin) $\quad f(\text{img}) = \vec{0}$

[Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering]

Andrew Ng

Loss function

Given 3 images A, P, N:

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

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

Training set: 10k pictures of 1k persons

Choosing the triplets A,P,N

During training, if A,P,N are chosen randomly, $d(A,P) + \alpha \leq d(A,N)$ is easily satisfied.

$\|f(A) - f(P)\|^2 + \alpha \leq \|f(A) - f(N)\|^2$

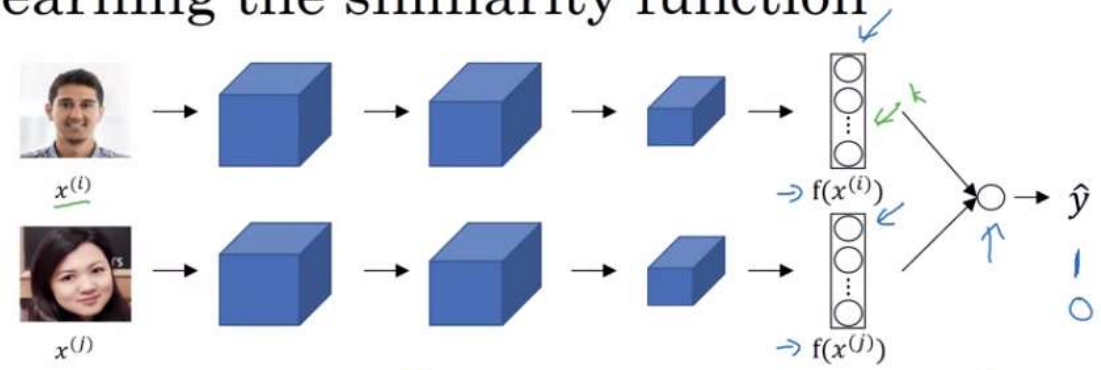
Choose triplets that're "hard" to train on.

$d(A,P) + \alpha \leq d(A,N)$

$d(A,P) \approx d(A,N)$ (↓ ≈ ↑)

Face Verification and Binary Classification

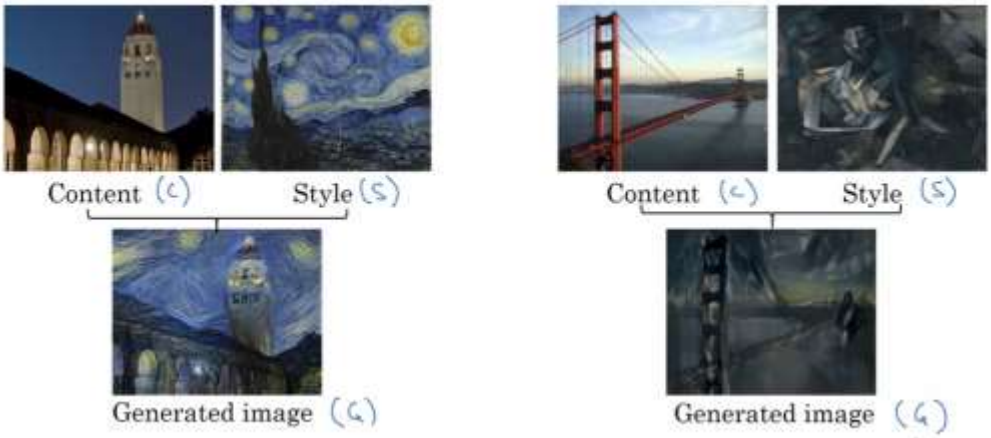
Learning the similarity function



$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_i \left|f(x^{(i)})_k - f(x^{(j)})_k\right| + b\right)$

Neural Style Transfer

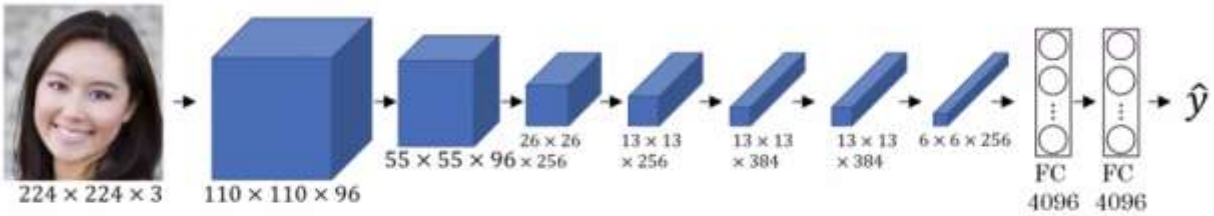
Neural style transfer



Images generated by Justin Johnson]

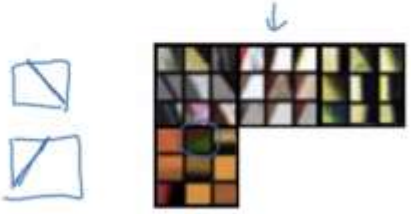
Andrew Ng

Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.



Zeiler and Fergus., 2013, Visualizing and understanding convolutional networks]

Andrew Ng

Visualizing deep layers


Layer 1


Layer 2


Layer 3


Layer 4


Layer 5

Cost Function

Neural style transfer cost function


Content C


Style S


Generated image G

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

[Gatys et al., 2015. A neural algorithm of artistic style. Images on slide generated by Justin Johnson] Andrew

Find the generated image G

1. Initiate G randomly

   G: $100 \times 100 \times 3$

   ↑ RGB

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

$$G := G - \frac{\partial}{\partial G} J(G)$$

Content cost function

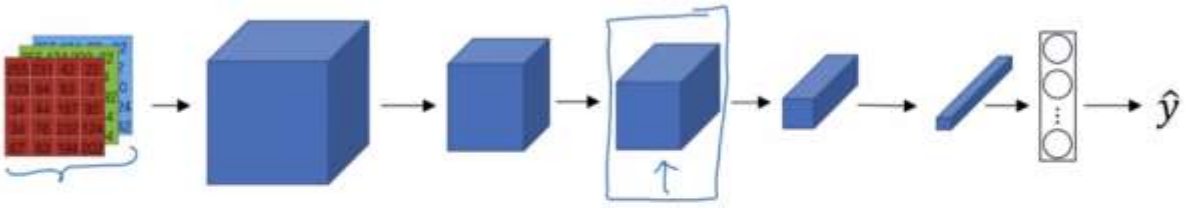
$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

- 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

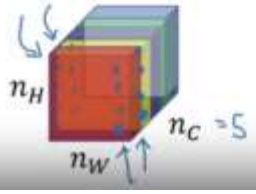
$$J_{content}(C, G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

Style Cost Function

Meaning of the "style" of an image



Say you are using layer l's activation to measure "style."
Define style as correlation between activations across channels.



How correlated are the activations across different channels?

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"

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

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

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

We can also apply CNNs to 1D and 3D data.

Choose a "middle" activation layer $a^{[l]}$

We would like the "generated" image G to have similar content as the input image C. Suppose you have chosen some layer's activations to represent the content of an image.

- In practice, you'll get the most visually pleasing results if you choose a layer in the **middle** of the network--neither too shallow nor too deep.

Content Cost Function $J_{content}(C, G)$

We will define the content cost function as:

$$J_{content}(C,G) = \frac{1}{4 \times n_H \times n_W \times n_C} \sum_{\text{all entries}} (a^{(C)} - a^{(G)})^2$$

Instructions: The 3 steps to implement this function are:

1. Retrieve dimensions from `a_G`:
   - To retrieve dimensions from a tensor `X`, use: `X.get_shape().as_list()`
2. Unroll `a_C` and `a_G` as explained in the picture above
   - You'll likey want to use these functions: tf.transpose and tf.reshape.
3. Compute the content cost:
   - You'll likely want to use these functions: tf.reduce_sum, tf.square and tf.subtract.

You can combine the style costs for different layers as follows:

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

where the values for $\lambda^{[l]}$ are given in STYLE_LAYERS.

- The style of an image can be represented using the Gram matrix of a hidden layer's activations.
- We get even better results by combining this representation from multiple different layers.
- This is in contrast to the content representation, where usually using just a single hidden layer is sufficie
- Minimizing the style cost will cause the image G to follow the style of the image S.

- To do so, your program has to reset the graph and use an "Interactive Session".
- Unlike a regular session, the "Interactive Session" installs itself as the default session to build a graph.
- This allows you to run variables without constantly needing to refer to the session object (calling "sess.run()"), which simplifies the code.

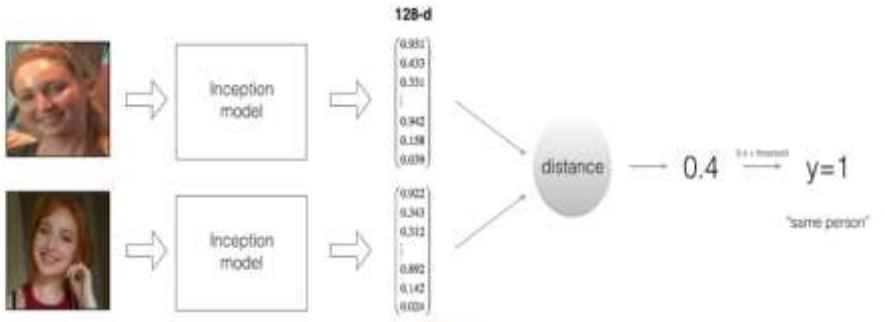


Figure 2:
By computing the distance between two encodings and thresholding, you can determine if the two pictures represent the same person

So, an encoding is a good one if:

- The encodings of two images of the same person are quite similar to each other.
- The encodings of two images of different persons are very different.

You would thus like to minimize the following "triplet cost":

$$\mathcal{J} = \sum_{i=1}^{m} \big[\underbrace{|| f(A^{(i)}) - f(P^{(i)}) ||_2^2}_{(1)} - \underbrace{|| f(A^{(i)}) - f(N^{(i)}) ||_2^2}_{(2)} + \alpha \big]_+$$

Here, we are using the notation "$[z]_+$" to denote $max(z,0)$.

Ways to improve your facial recognition model

Although we won't implement it here, here are some ways to further improve the algorithm:

- Put more images of each person (under different lighting conditions, taken on different days, etc.) into the database. Then given a new image, compare the new face to multiple pictures of the person. This would increase accuracy.
- Crop the images to just contain the face, and less of the "border" region around the face. This preprocessing removes some of the irrelevant pixels around the face, and also makes the algorithm more robust.

Key points to remember

- Face verification solves an easier 1:1 matching problem; face recognition addresses a harder 1:K matching problem.
- The triplet loss is an effective loss function for training a neural network to learn an encoding of a face image.
- The same encoding can be used for verification and recognition. Measuring distances between two images' encodings allows you to determine whether they are pictures of the same person.