WEEK 4 FACE RECOGNITION

live humain

Face verifacation vs. face recognition

Verification

- Input image, name/ID
- Output weather the input iamge is that of the claimed person

Recongnition

- Has a database of K persons
- Get an input iamges
- Output ID if the image is any of the K persons (or "not recognized")
- **Note that Face Recognition is much more hard than Face Verification**

One Shot Learning

Just one simple Learning from one example to recognize the person again

Learning a "similarity" function

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

$$\begin{aligned} \text{If} \quad d(\text{img1},\text{img2}) &\le \tau \quad \rightarrow \text{"same"} \\ &> \tau \quad \rightarrow \text{"different"} \end{aligned}$$

take input of a pair of iamges

Siamese network

First introduced by Taigman et. al.,2014

Use the same parameters to calculate two iamges in order to compare with each other

How can we define a good loss funtion for this target?

Triplet loss

Anchor image

First introduced by Schroff et al.,2015

Set a margin to make sure that we will not choose all zeros as parameters

Loss function

A,P,N stand for three pictures which we define as a triplet of input of our loss function, which is:

- A : anchor
- P : positive *(that is, `high` similarity or the `same` person)*
- N : negative *(that is, `Low` similarity and normally `different` person)*

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

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

ex. of training set
Training set is made up with $10K$ pictures of $1K$ different persons

- we need in fact multiple pictures of the same person so we can create the pair of (A,P)
- en average, each person have about 10 pictures

How we choose 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

We tend to choose those triplets that are `"hard"` to train on. That is :

$$\begin{aligned} & d(A,P)+\alpha \leq d(A,N) \\ \text{but,} \qquad & d(A,P)+\alpha \simeq d(A,N) \end{aligned}$$

In this way, our model will tend to make $d(A,P)$ larger and $d(A,N)$ smaller in converse.

Face Verification and Binary Classification

Another way using binary classification to do the face varification that works also for our but.

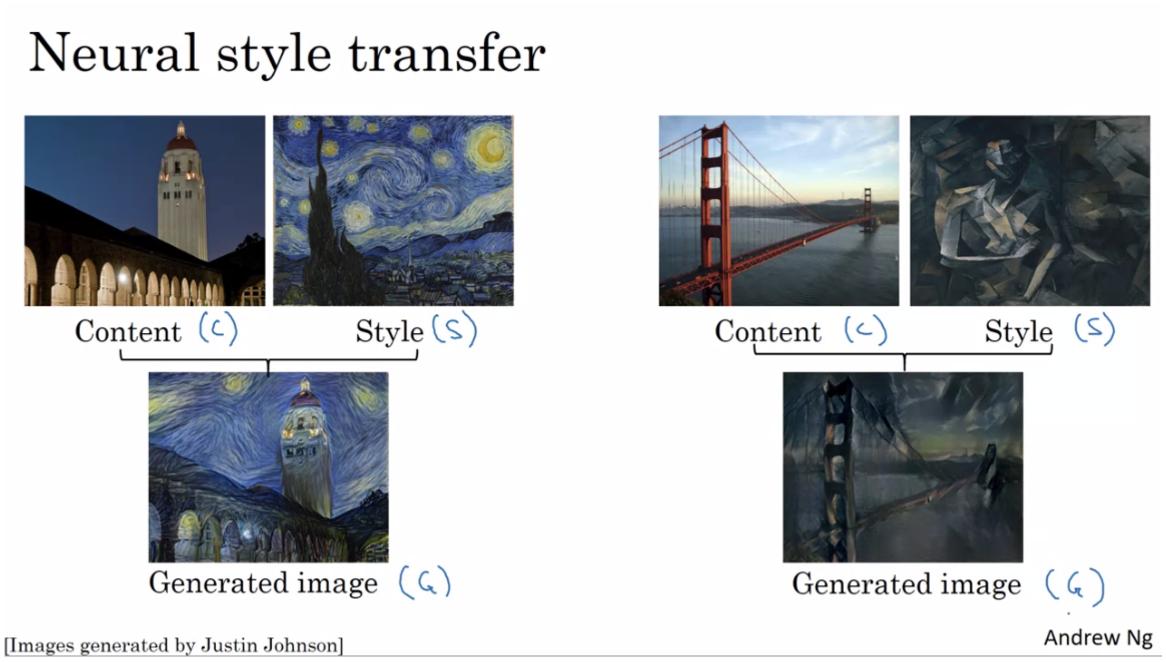
Logistic regression unite to learning the similarity function Cite: Taigman et al., 2014)

Face verification supervised learning: Create a pair of pictures and predict wether these two pictures are belong to the same person.

Neural Style Tranfer

- Content(C)
- Style(S)

- Generated image(G)



Deep ConvNets Learning

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

1. Pick a unit *(that is, a neuron of this layer, or a filter of this layers)* in layer 1. Find the `nine` images patches that maximize the unit's activation
2. Repeat for other units
3. continue in deeper layers

- It comes to us that deeper the layer is, more complex will be the image it specifie.

Cost function

> Gatys et al.,2015 A Neural Algorithm of Artistic Style

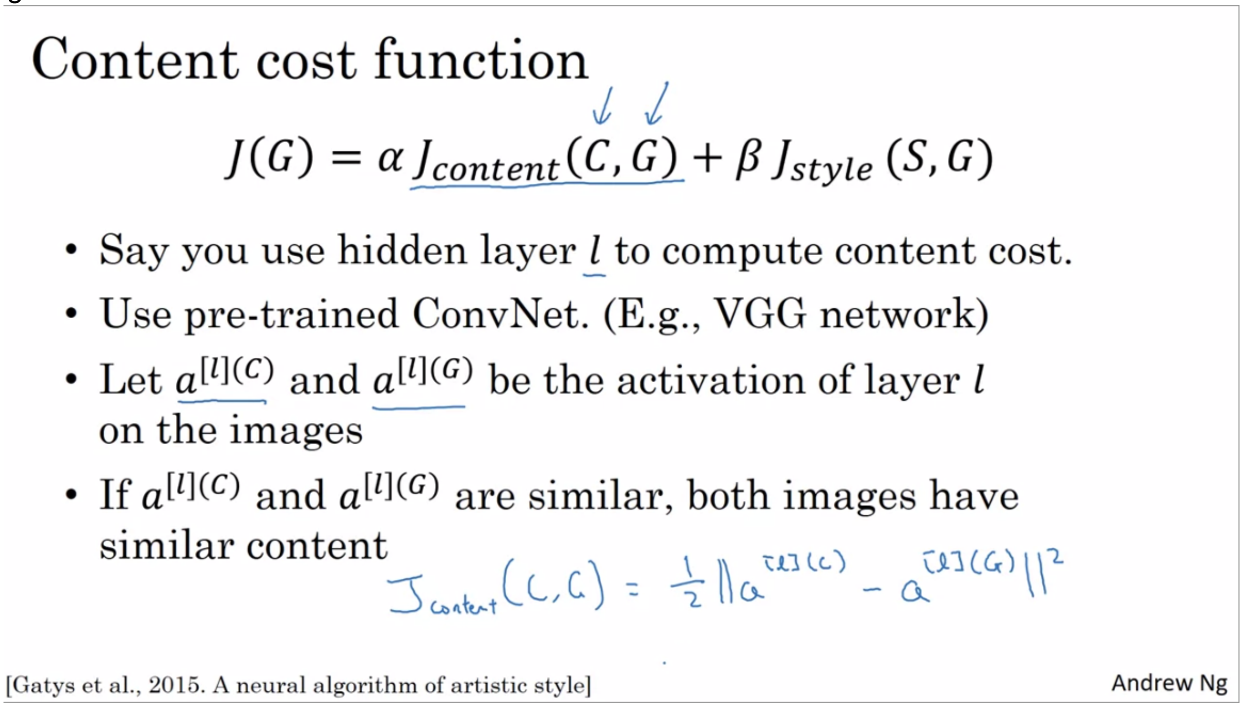
We define a cost funtion like this:

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

Find the generated image G

1. Initiate G randomly
   $G : 100 \times 100 \times 3$
2. Use gradient descent to minimize $J(G)$
   $G := G - \frac{\partial}{\partial G} J(G)$

Content cost function



Content cost function

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

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

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

[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

The hidden layer chosen here is nether too shalow non too deep

Style cost function

We don't care too much of of content of different images, in revenche, the style. We denote the style by detecting the high-light features of different channels of certain layers of activations for different images (S,G). Style is the combine of these high-light features of all channels. We consider both their absolute relation and relative relation.

Mraning

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

The problem comes that how can we measure the degree of correlation?

Correlation

After finishing our pre-step job, we have already all the parameters for each layer of our basic `ConvNet`, we deteste the degree of correletion between two images by choosing on layer and calculate their activation in correspond.

Each layer of activations of the `style image` will give us slices in series in the sense of the style. That is, different slice of activation in layer l, gives us a serie of features that different channel corresponds.

So when we look at the activations of the same layer but of our `generated image`, we will find also a series of features of different channels. If the two series of features can well correspond, we say that these two image are correleted and uncorreleted if not.

EX.
if we choose the first hidden layer, that is, the first Conv layer normally, we will have a very clear feature of every channel because we know already how it looks like each filter.

While in deeper layer, different channels come from some series of acitivation that is usually much more complex. Two different style picture, even though of the same subject, will result in perhaps totally different features after being passed to the model. Because they donnot have the same style for the activations.

Style matrix

Let $a^{[l]}_{i,j,k}$

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

$G^{[l]}$ of $n^{[l]}_c \times n^{[l]}_c$ is called a style metrix of layer l defined by:

$$G^{[l](S)}_{kk'} = \sum_{i=1}^{n^{[l]}_H} \sum_{j=1}^{n^{[l]}_W} a^{[l](S)}_{ijk} a^{[l](S)}_{ijk'}$$

$$G^{[l](G)}_{kk'} = \sum_{i=1}^{n^{[l]}_H} \sum_{j=1}^{n^{[l]}_W} a^{[l](G)}_{ijk} a^{[l](G)}_{ijk'}$$

$$\begin{aligned} \mathrm{J}^{[l]}_{\text{style}}(S, G) &= \frac{1}{(2n^{[l]}_H n^{[l]}_W n^{[l]}_C)^2} \left\| G^{[l](S)} - G^{[l](G)} \right\|^2_F \\ &= \frac{1}{(2n^{[l]}_H n^{[l]}_W n^{[l]}_C)^2} \sum_k \sum_{k'} (G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'})^2 \end{aligned}$$

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

$$\begin{aligned} \mathrm{J}(G) = \alpha \mathrm{J}_{content}(C, G) \\ + \beta \mathrm{J}_{style}(S, G) \end{aligned}$$