Week 4

Face recognition vs. face verification:

→ Verification: 1:1 matching problem
verifies if its "rather"

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

→ Recognition: 1:K matching problem
checks and compares with all K employees

- Has database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or 'not recognised')

One Shot Learning:

The problem with face recognition is that the model only sees the person's face once (in the database only 1 picture is there). Also if someone new join the team, we can't retrain the network. So rather than face recognition, we use a similarity function.

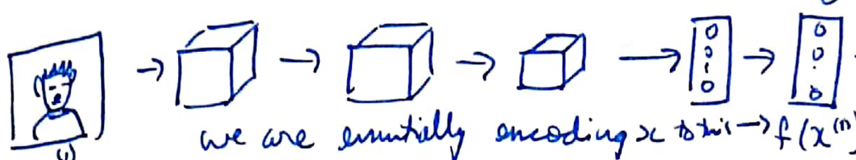
$d(\text{img}1, \text{img}2)$ = degree of difference between image

If $d(\text{img}1, \text{img}2)$ $\leq \tau$ "same"
$> \tau$ "different" } verification

(So we don't use the earlier method of training) on a large set of pictures of a person

Siamese Network:

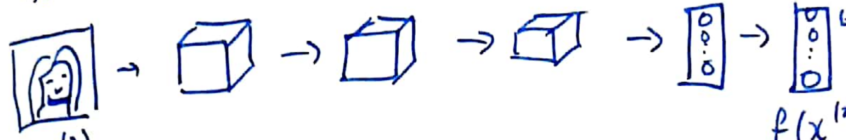
lets say 128 outputs



remove the softmax

$x^{(1)}$

we are essentially encoding x to this $\rightarrow f(x^{(1)})$



$\leftarrow 128$

$x^{(2)}$ $f(x^{(2)})$

Then $d(x^{(1)}, x^{(2)}) = \| f(x^{(1)}) - f(x^{(2)}) \|_2^2$

→ Parameters ~~of~~ of NN define an encoding $f(x^{(i)})$ – the 128 parameters in this case

→ 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

we can learn using the Triplet loss Function.

Triplet Loss Function:

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

we want:

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

$$= \| f(A) - f(P) \|^2 - \| f(A) - f(N) \|^2 \leq 0$$

But here the algo might give both value N so $N - N \leq 0$, therefore we add a margin α

$\therefore \ \|f(A)-f(P)\|^2 - \|f(A)-f(N)\|^2 + \alpha \leq 0$

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

So for the example if we got,

$d(A,P) = 0.5$ and $d(A,N) = 0.51$

then to maintain margin (if $\alpha = 0.2$), it'll ~~push~~ either increase $d(A,N)$ to 0.7 or decrease $d(A,P)$ to 0.31 so $\alpha = 0.2$ is maintained.

Loss Function:

Given 3 images A, P, N,

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

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

So if we have a training set of 10K pictures of 1K persons, we need to divide them into A, P and A, N.

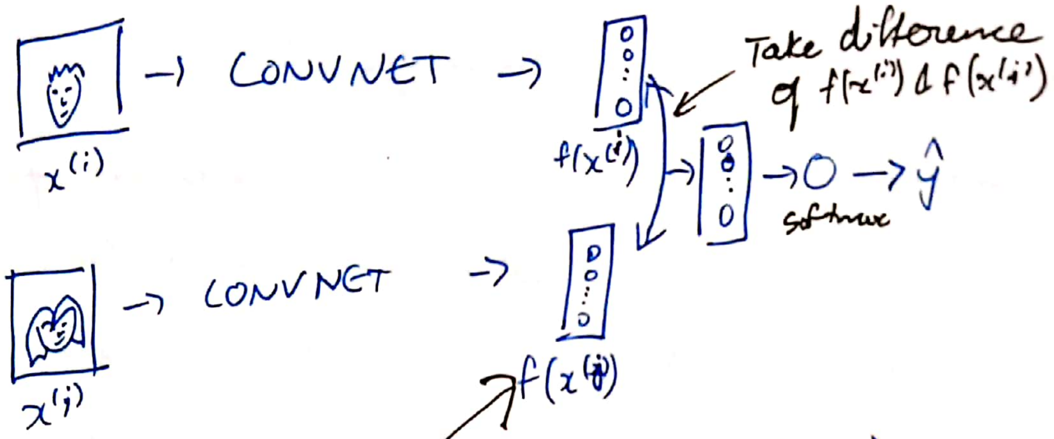
Choosing the triplets A, P, N,

→ If we choose A, P, N randomly, then $d(A,P) + \alpha \leq d(A,N)$ is easily satisfied

→ Choose triplets that're "hard" to train on. i.e $d(A,P) \approx d(A,N)$ - similar people

Binary classification:

Another way to compute the output is by using a binary classification unit.



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

we can even use other difference functions like Kaiser function $\chi^2 = \frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$

For the employee images in the database, rather than storing the raw images, we can precompute this vector and store them instead. So we compare the vector of the new image with the vectors in the database.

Neural Style Transfer

Content (C)　　　Style (S)

Generated Image (G)

Visualising what ConvNets are learning,
In layer 1, they learn simple features like edges, and go onto detecting more complicated features in subsequent layers. Check out the video to see them.

Cost function:

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

Steps:

1. Initialise G randomly
   $G: 100 \times 100 \times 3$
2. Use Gradient Descent to minimize $J(G)$
   $$G := G - \frac{\alpha}{\partial G} J(G)$$
   derivative not alpha

Choosing Content cost function $J_{content}(C,G)$:

→ Use a hidden layer l to compute the content cost such that it is not too shallow or deep.

→ Use pre-trained ConvNet (Eg. 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

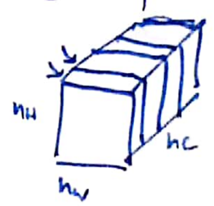
$$J_{content}(C,G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

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 channel i.e how correlated are the activations across different channels

Two ~~feature~~ ~~parameter~~ activations are correlated if they have something in common (like edge detecting ~~parameter~~ activation has reddish tinge like the red detecting ~~parameter~~ activation)

So we check the correlation between activations across different channels



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

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

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

$$J^{[l]}_{style}(S, G) = \frac{1}{(2n_H^{[l]} n_W^{[l]} n_C^{[l]})} \| G^{[l](S)} - G^{[l](G)} \|_F^2$$

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

So,

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

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

and we use Gradient Descent or any other optimising algorithm to minimise J.

We can perform convolution on 1D and 3D, not just 2D.