dec 1: Face Recognition:

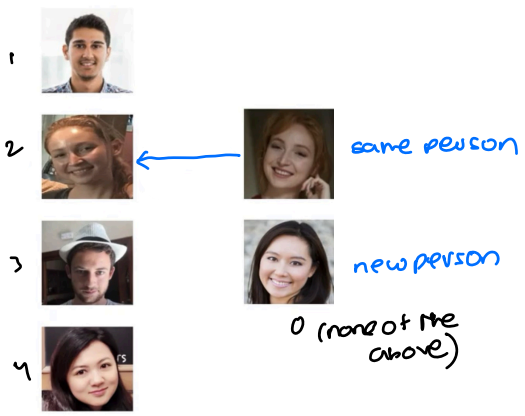
Verification: (1:1)

- input image, name ID
- output whether the input image is that of claimed person.

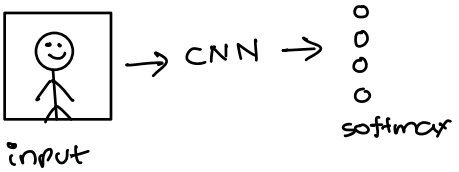
Recognition: (1:K)

- Has a database of K person
- get an input image
- output ID if the image is any of K persons (or not recognized)

dec 2: One shot dearning:



dearning from one example to recognibe the person again



problem: → small training set to learn
→ retrain everytime a new person join.

Learning a similarity function:

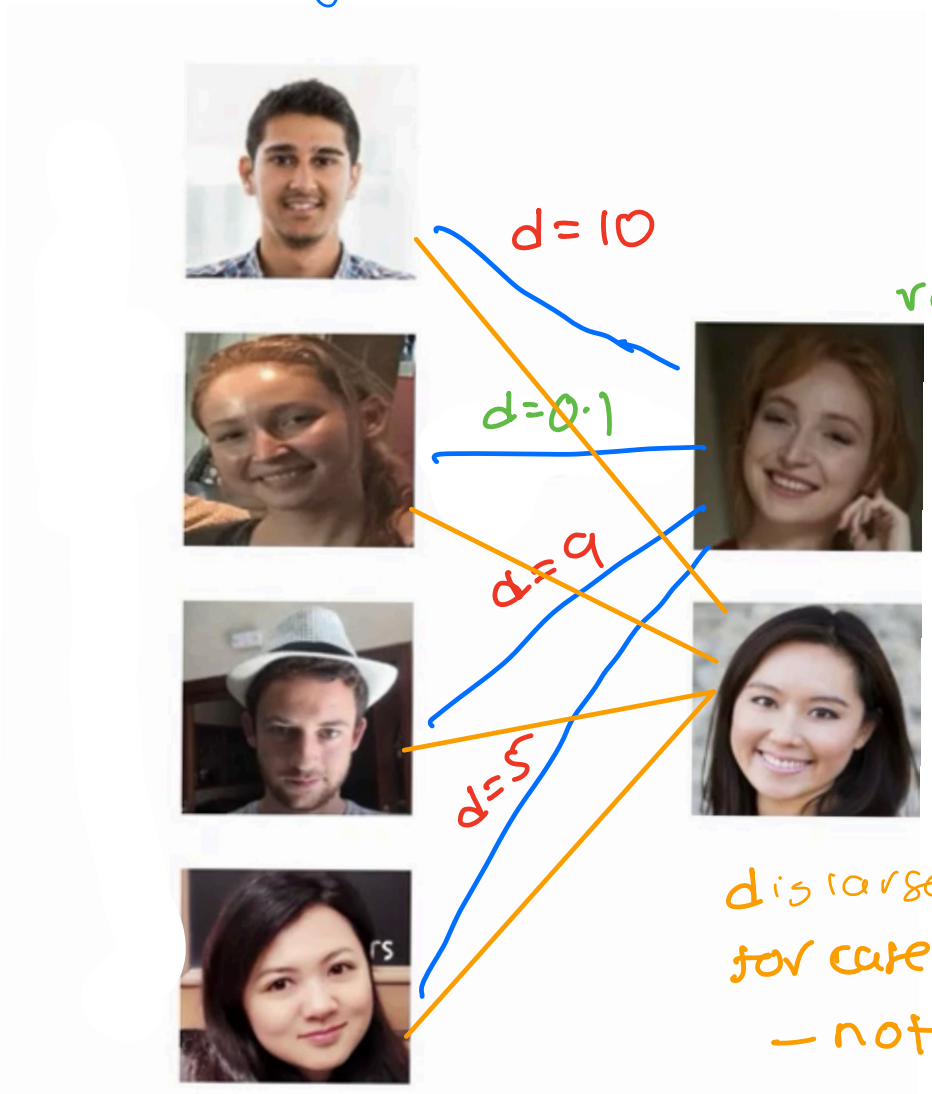
$d(img1, img2)$ = degree of difference between images

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

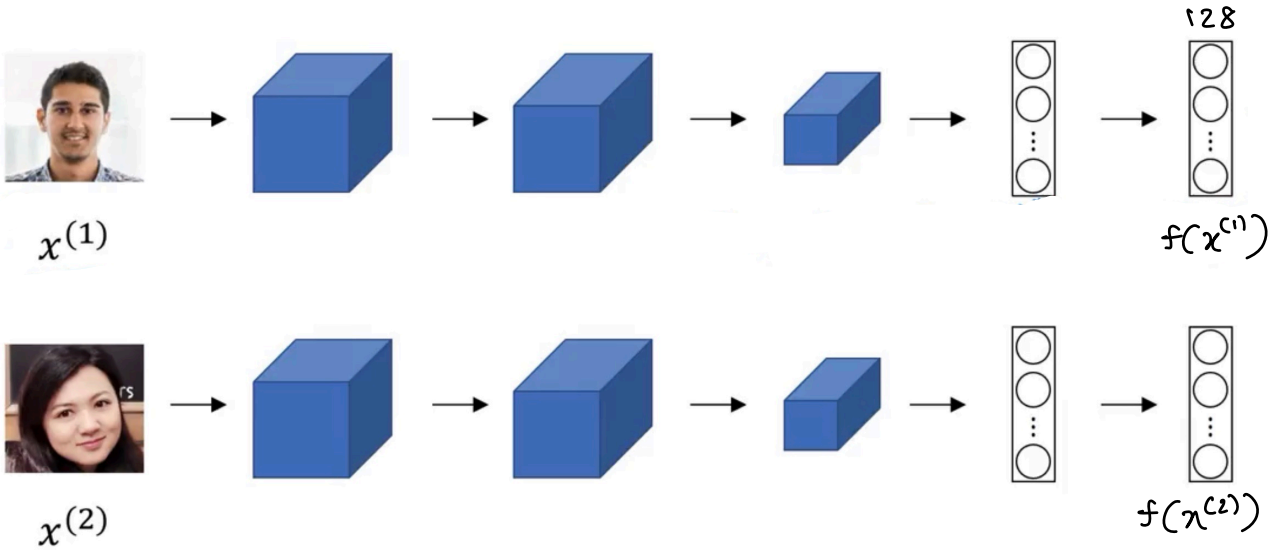
if $d(img1, img2) > \tau$ "different"

} face verification.

Face Recognition:



Lec 3: Siamese Network: (DeepFace 2014)



$$d(x^{(1)}, x^{(2)}) = \left\| f(x^{(1)}) - f(x^{(2)}) \right\|_2^2$$ norm between encodings.

parameters of NN define an encoding $f(x^{(i)})$ (128 dim)

Learn parameters so that:

if $x^{(i)}, x^{(j)}$ are the same person, $\left\| f(x^{(i)}) - f(x^{(j)}) \right\|_2^2$ is small

if $x^{(i)}, x^{(j)}$ are different persons, $\left\| f(x^{(i)}) - f(x^{(j)}) \right\|_2^2$ is large.

Dec 4: Triplet loss: (anchor, positive, negative)

Anchor
A

Positive
P

Anchor
A

Negative
N

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

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

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

α - margin (SVM?) (Pushes $d(A,P) \downarrow$ $d(A,N) \uparrow$)

if $\alpha = 0.2$, for above example

$d(A,P) = 0.5$

$d(A,N) = 0.51$

$d(A,P) + \alpha \leq d(A,N)$ ($d(A,N) \uparrow \geq 0.7$)

$0.5 + 0.2 \not\leq 0.51$

Loss Function:

given 3 images A, P, N:

$$\text{min } \alpha(A,P,N) = \max\left(\underbrace{||f(A) - f(P)||^2 - ||f(A) - f(N)||^2}_{\leq 0} + \alpha,\ 0 \right)$$

cost function: $J = \sum_{i=1}^{m} \alpha(A^{(i)}, P^{(i)}, N^{(i)})$

Training set: 10K pictures of 1K persons. (multi pictures of single person).

- select triplets of images (A, P, N)
- train algorithm with G.D

choosing the triplets A, P, N: (FaceNet: 2015)

During training, if A, P, N are chooren randomly,

$d(A,P) + \alpha \leq d(A,N)$ is easily satisfied.

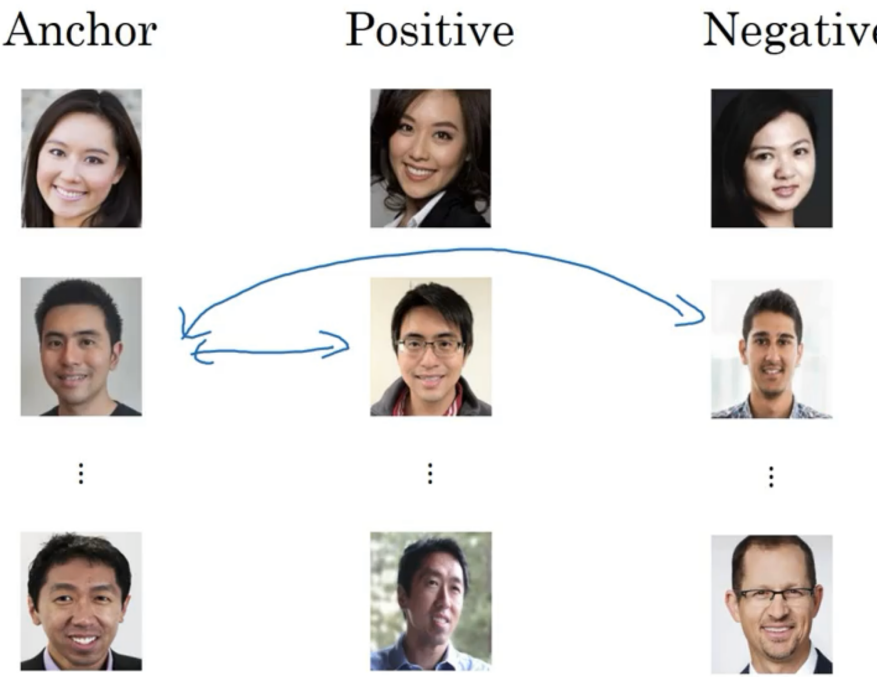
since, most of them are not similar.

Choose triplets that're "hard" to train to

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

$$d(A,P) \simeq d(A,N) \Rightarrow (A,P,N).$$

Training set using triple loss



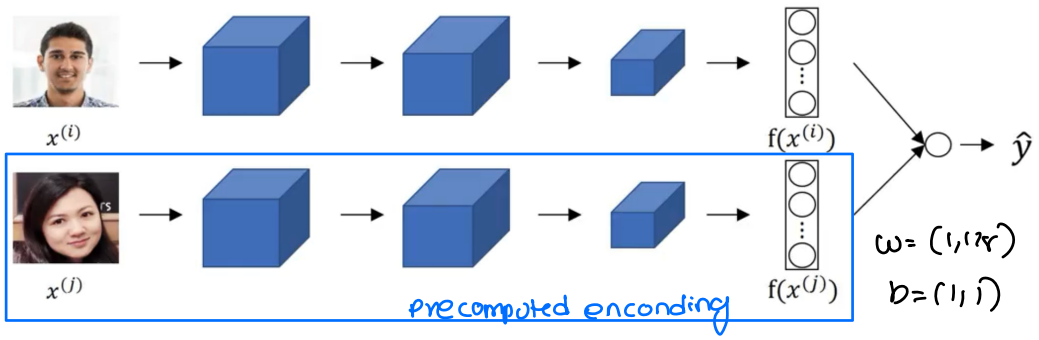
use GD,

and minimize cost function J and learn encoding such that

$d(x^{(i)}, x^{(j)})$ - small for same person

$d(x^{(i)}, x^{(j)})$ - large for different person.

dec: learning the similarity function: (DeepFace 2014).



face recognition → binary classification.

$\hat{y} = 1$ (same person)

$\hat{y} = 0$ (different person)

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

$$d(x^{(i)}, x^{(j)}) = \left| f(x^{(i)}) - f(x^{(j)}) \right| \quad \text{(abs element wise difference)}$$

$$d(x^{(i)}, x^{(j)}) = \frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k} \quad \chi^2$$

$$\hat{y} = \sigma\left(\underset{(1,128)}{w}\, \underset{(128,1)}{d(x^{(i)}, x^{(j)})} + \underset{(1,1)}{b}\right)$$

$(1,1)$

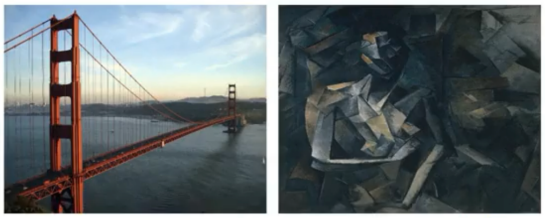
Face Verification supervised learning:

x	y	
 	1	"same"
 	0	"different"
 	0	different.
 	1	same

dec1: Neural Style Transfer:

Content (C) Style (S)

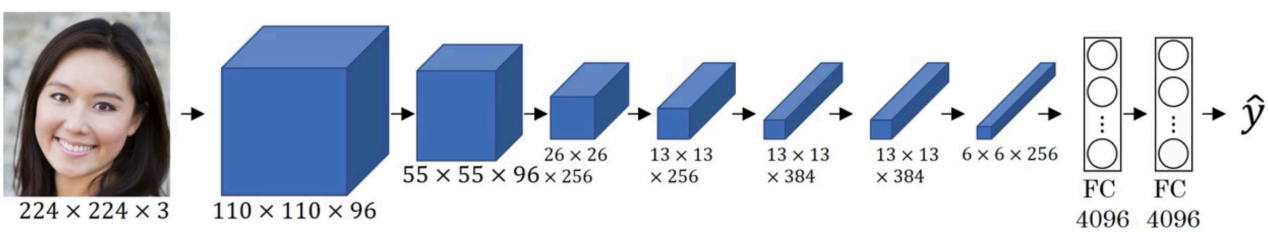
Generated image (G)



Content (C) Style (S)

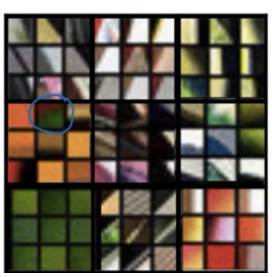
Generated image (G)

dec2: What deep network is learning:

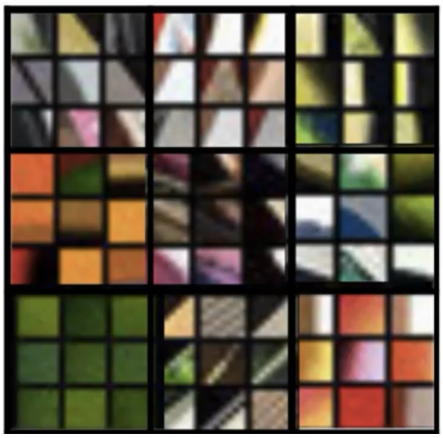


Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

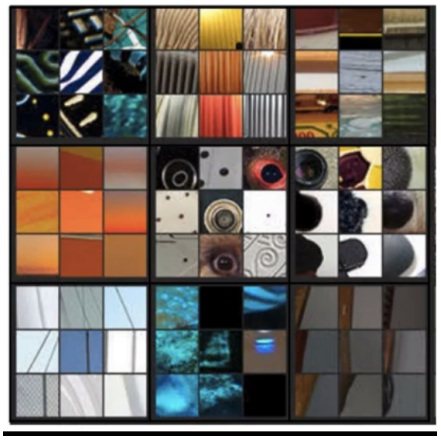
Repeat for other units.



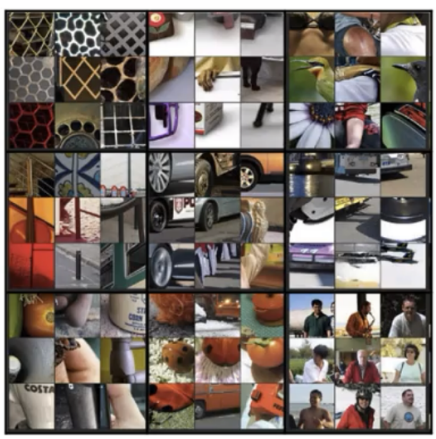
layer 1



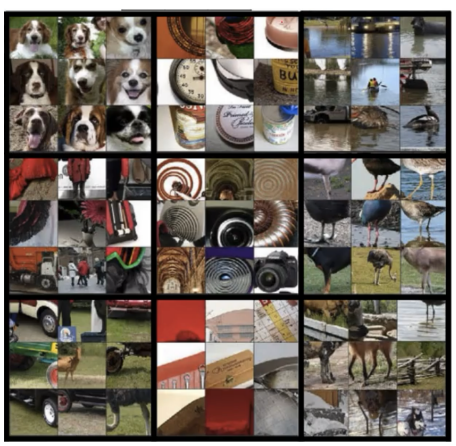
layer1: edges



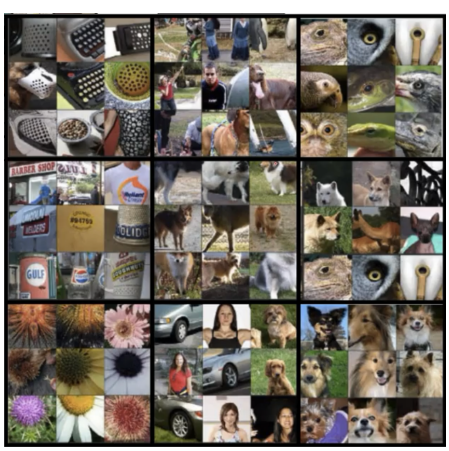
layer2: complex edges



layer3: special shapes

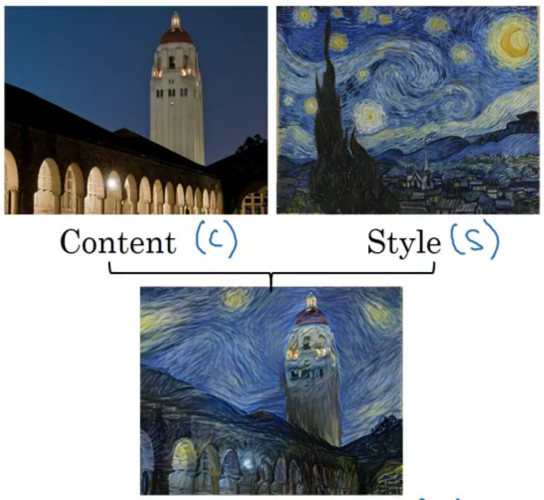


layer4: complex objects, dogs.



layer5: different persons, dogs, flowers ...

dec3: cost function:

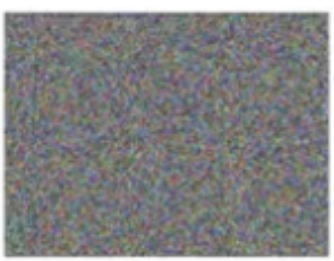


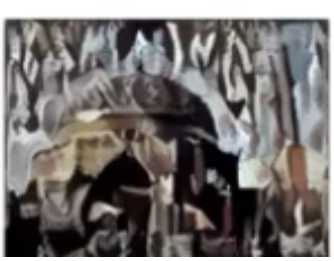
$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

Algorithm to find generated image G

1. initiate G randomly G: 100x100x3
2. use gradient descent to minimize J(G)

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





dec4: Content cost function:

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

1. say you use hidden layer l to compute content cost
   - if l is small:
     force G image match C, pixel to pixel wise
   - if l is large:
     if there is dog in C, make sure to dog in G somewhere.

   choose l is between.

2. Use pretrained ConvNet (Eg. VGG Network)

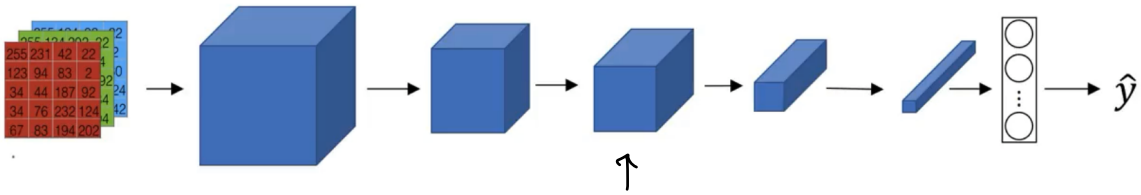
3. let $a^{[l](C)}$ and $a^{[l](G)}$ be activations of layer l on in C, G images.

4. if $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

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

element-wise sum of square of differences between $a^{[l](C)}$ and $a^{[l](G)}$

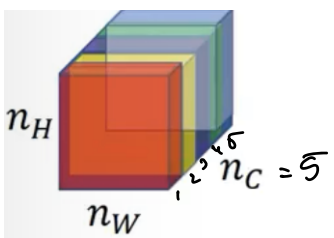
Lec 5: Style of an image:



Say you are using layer l's activation to measure "style"

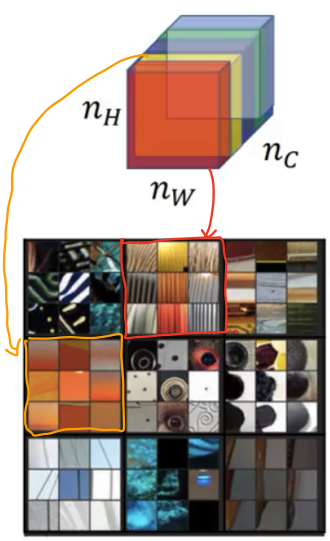
Style: correlation between activations across channels.

layer l activation

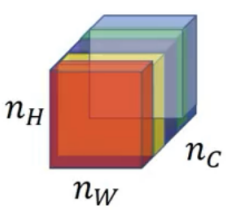


How correlated are the activations across different channels?

Style image



Generated Image



nc[1], nc[2] are highly correlated means
- part image which has vertical edges, also contains organe-ish part aswell

nc[1], nc[1] are not correlated means
- those parts are not connected to each other.

"correlation tells you which of texture components tend to occur or not occur together".

Style Matrix:

let $a^{[l]}_{i,j,k}$ = activation at (i, j, k) $(i-H, j-W, k-C)$

style matrix $G^{[l]}$ is $n_C \times n_C$

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

style image. $G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} a^{[l](S)}_{ijk} a^{[l](S)}_{ijk'}$ $\quad k = 1, \ldots, n_C^{[l]}$

if kk' are correlated $G_{kk'}$ will large, if not correlated $G_{kk'}$ will be small.

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

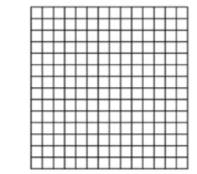
$$J_{style}(S, G) = \sum_l \lambda^{[l]} J^{[l]}_{style}(S, G)$$

↓

take into both low level and highly features.

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

dec6: 1D and 3D convolution:



2D input image
14 × 14

*

2D filter
5 × 5

$14 \times 14 \times 3 \ * \ 5 \times 5 \times 3 \times 16 = 10 \times 10 \times 16$

1D: (heart ECG)

1	20	15	3	18	12	4	17

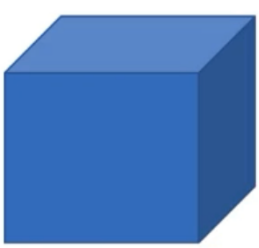
*

1	3	10	3	1

layer1: $14 \times 1 \ * \ 5 \times 1 \times 16 = 10 \times 16$

layer2: $10 \times 16 \ * \ 5 \times 16 \times 32 = 6 \times 32$

3D: (video, 3D scan)



3D volume

*

3D filter

$14 \times 14 \times 14 \times 1 \ * \ 5 \times 5 \times 5 \times 1 \times 16 = 10 \times 10 \times 10 \times 16$

$10 \times 10 \times 10 \times 16 \ * \ 5 \times 5 \times 5 \times 16 \times 32 = 6 \times 6 \times 6 \times 32$

$(n \times n) * (f \times f)$ with p, s

$$\left[\frac{n+2p-f}{s}+1\right] \times \left[\frac{n+2p-f}{s}+1\right]$$

$$\frac{64+0-4}{2}+1$$

31×31