What is Face Recognition?

Face Verification: 1:1

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

Face Recognition: 1:K

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

Face recognition is generally harder than face verification. Eg: if the accuracy of face verification is 99%, and there are 100 people (K=100) in our database. Then there will be one person being misrecognized in average.

Face verification is actually a building block of face recognition system.

One-shot Learning

Building a face recognition system is actually not easy. Eg, if I want to build a face recognition system for my company which has 100 employees. They all hand in Only One image of their face. It's impossible to train a neural network with only one image.

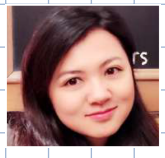
What can I do to solve it? Hint: a face recognition system doesn't need to really "know" a person. It just needs to know if this person is the person in the hand-in image.

Eg: I, as a human, have only seen one Person, my mom, in this world, and I can recognize her. But a computer can't. It needs to see thousands of people first, then can recognize a person.

One-shot learning: Learning from one example to recognize the person again.

In my comany, there are three employees with three images. The system needs to recogize a person via a camera.

Employees' image on ID cards



An employee

Not an employee

Intuitively, we could build a network on the right:

Input X

→ CNN → ○ softmax(4)

3: Three employees
+
1: None of the above

But this won't work well due to two reasons.

First, the training data is too few (only three images).

Second, if there is a new person joining my company, the last layer needs to changed and retrained.

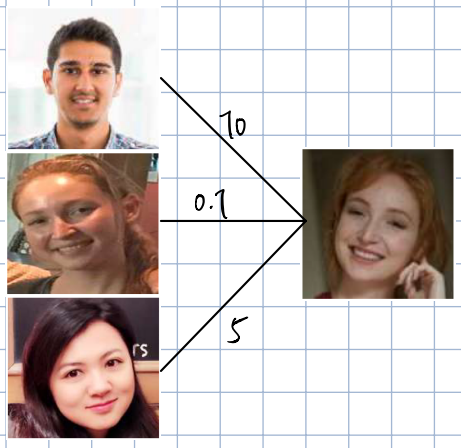
Thus, what we need is to learn a "similarity" function.

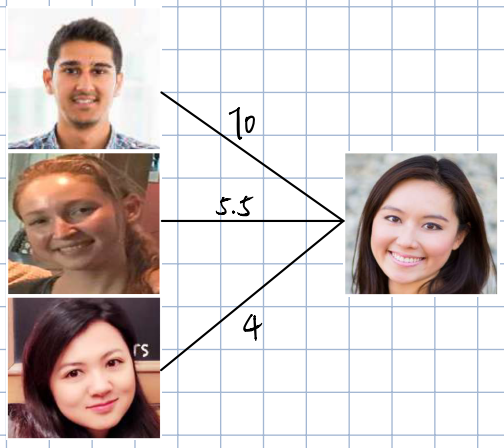
d(img1, img2) = degree of difference between images τ: Predefined Threshold

if d(img1, img2) ≤ τ, "same" person

if d(img1, img2) > τ, "different" person

} To solve Face Verification problem

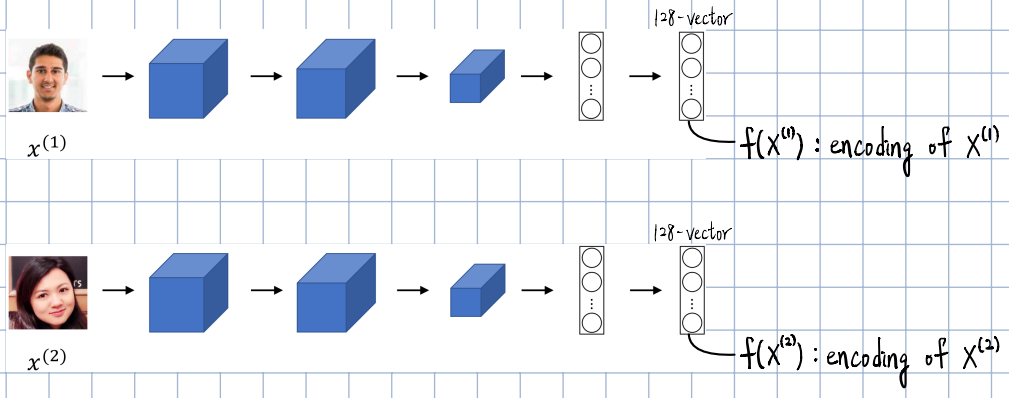




Siamese Network

Now, the question is how to learn a "similarity" function d(img1, img2). Use Siamese network.

Read 2014 "DeepFace closing the gap to human level performance"



Define the descrepency between $X^{(1)}$ and $X^{(2)}$ as $d(X^{(1)}, X^{(2)}) = \| f(x^{(1)}) - f(x^{(2)}) \|_2^2$

Remember that the above two networks have the SAME parameters.

Formally, the goal of learning is:

- Parameters of neural network define an encoding $f(X^{(i)})$
- Learn parameters so that:

  If two arbitrary pictures $X^{(i)}$ and $X^{(j)}$ are the same person, $\| f(x^{(i)}) - f(x^{(j)}) \|_2^2$ is small.

  If two arbitrary pictures $X^{(i)}$ and $X^{(j)}$ are different person, $\| f(x^{(i)}) - f(x^{(j)}) \|_2^2$ is large.

Now, the question is that how to define the loss function so we can have a good encoding of pictures.

Use Triplet Loss.

Triplet Loss

Read "FaceNet: A unified embedding for face recognition and clustering"


Anchor (A)


Positive (P)


Anchor (A)


Negative (N)

Goal: $\underbrace{\|f(A)-f(P)\|_2^2}_{d(A,P)} \leq \underbrace{\|f(A)-f(N)\|_2^2}_{d(A,N)}$

$\Rightarrow \|f(A)-f(P)\|_2^2 - \|f(A)-f(N)\|_2^2 \leq 0$ → One trivial solution for this equation is function f outputs 0. $(0-0)-(0-0) \leq 0$ will always be easily learned. To avoid this, margin α is added.

Thus, the goal becomes $\|f(A)-f(P)\|_2^2 + \alpha \leq \|f(A)-f(N)\|_2^2$

$\Rightarrow \|f(A)-f(P)\|_2^2 - \|f(A)-f(N)\|_2^2 + \alpha \leq 0$

An intuitive explanation: when $d(A,P)=0.5$, $\alpha=0.2$, $d(A,N)=0.6$, the system should try harder to make either $d(A,P)$ smaller or $d(A,N)$ larger so that $d(A,P)+\alpha \leq d(A,N)$ is fulfilled.

Now, we can formally define triplet loss function.

Given three images, Anchor (A) Positive (P) Negative (N)

$$\mathcal{L}(A,P,N) = \max\left(\|f(A)-f(P)\|_2^2 - \|f(A)-f(N)\|_2^2 + \alpha, 0\right)$$

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

Given the Training set: 10k pictures of 1k different persons (i.e., 10 pictures for each person)

How to carefully select a triplet pair?

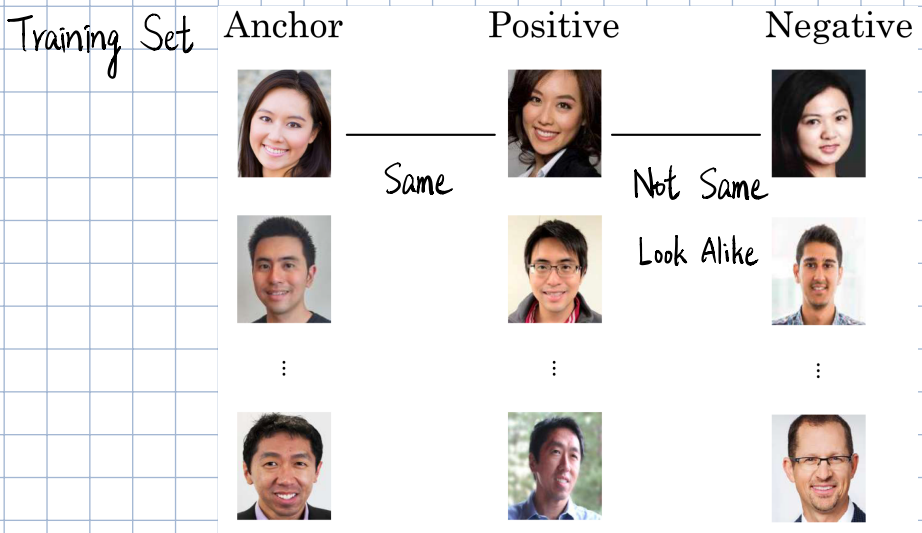
During training, if A, P, N are chosen casually and randomly,

$d(A,P)+\alpha \leq d(A,N)$ will be easily satisfied, because d(A,N) will be easily larger than d(A,P).
(I.e., there is not much training going on)

Therefore, we need to choose triplets that are HARD to train on.

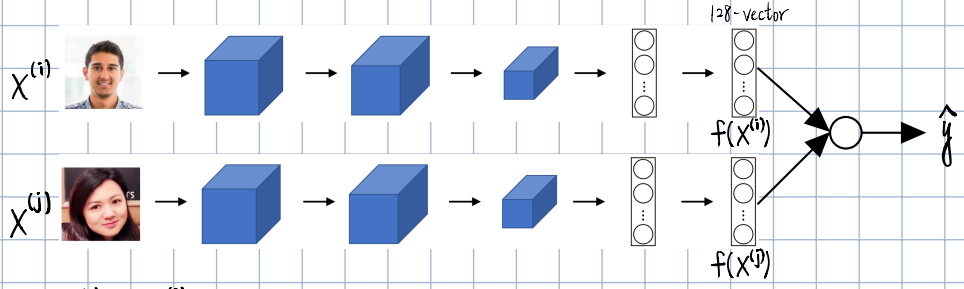
$d(A,P)+\alpha \leq d(A,N)$

$d(A,P) \approx d(A,N)$ Now, the NN needs to try its best to find the subtle differences between A and N.



Lastly, we can precompute the encoding of images in databases to save memory and time in inference.

Another way to train the network above is to reformat it into a Binary Classification Problem.



if $x^{(i)}$ and $x^{(j)}$ are the same person, $\hat{y}=1$. $\hat{y}=0$, if otherwise.

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

element-wise difference

sigmoid function

the k-th component of $f(x^{(i)})$

the normal logistic regression unit $\hat{y}=wx+b$

Alternative equation χ^2 to calculate the difference

$$\frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$$

Training Set

x		y
$x^{(i)}$	$x^{(j)}$	
		1
		0
		0
		1

deeplearning.ai

Neural Style Transfer

What is neural style transfer?

Neural style transfer


Content (C) Style (S)

Generated image (G)


Content (C) Style (S)

Generated image (G)

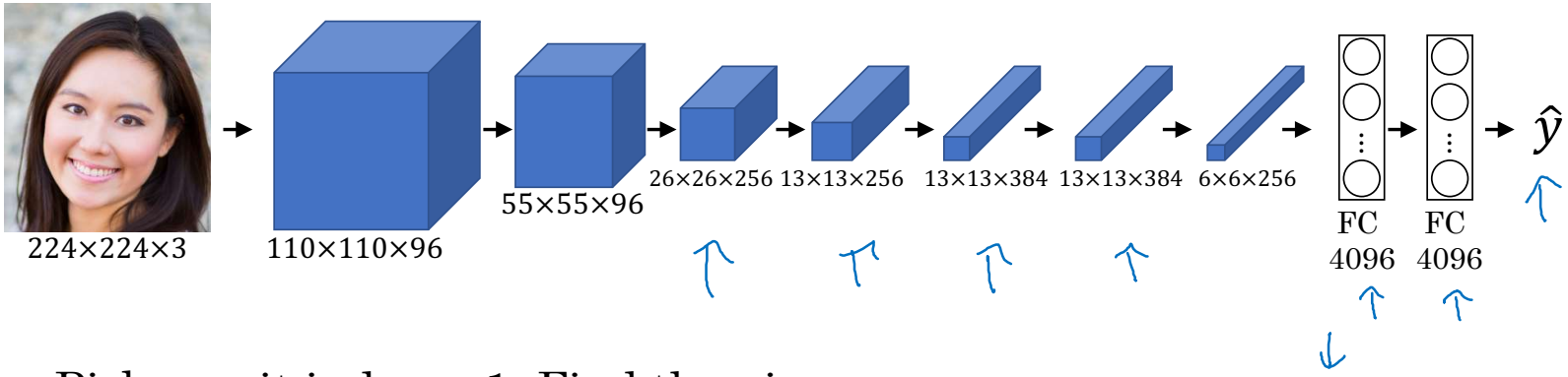
[Images generated by Justin Johnson]

deeplearning.ai

Neural Style Transfer

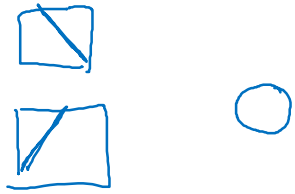
What are deep ConvNets learning?

Visualizing what a deep network is learning



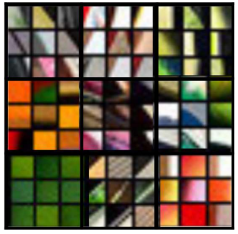
Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

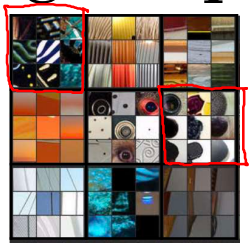


[Zeiler and Fergus., 2013, Visualizing and understanding convolutional networks]

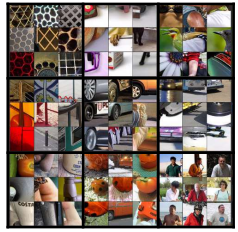
Visualizing deep layers



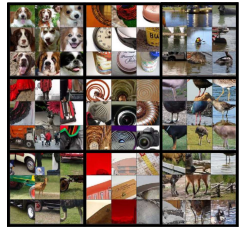
Layer 1



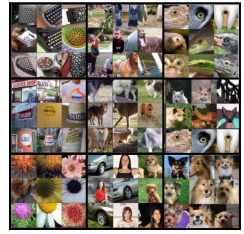
Layer 2



Layer 3



Layer 4

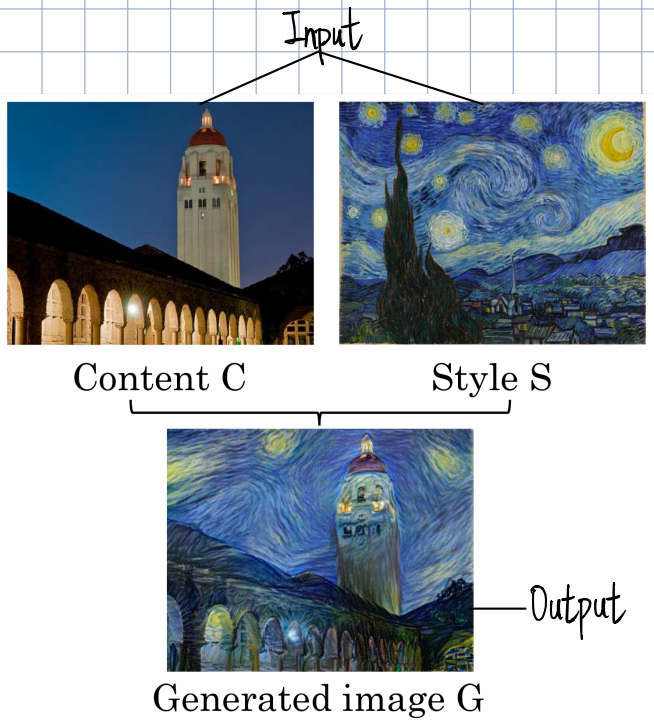


Layer 5

Andrew Ng

The Cost Function of Neural Style Transfer

Read 2015 "A Neural Algorithm of Artistic Style"



Goal: Create a generated image G.

I can define a cost function $J(G)$ that measures how good a particular generated image is and use gradient descent to minimize $J(G)$ in order to generate this image G.

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

$J_{content}(C, G)$: It measures how similar the content of the Generated image G to the content of the Content image C is.

$J_{style}(S, G)$: It measures how similar the content of the Generated image G to the content of the Style image S is.

After defining the cost function $J(G)$, the generated image G can be found as the following:

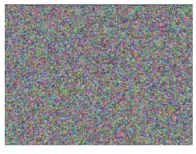
Step 1: Initiate G randomly.

G: $100 \times 100 \times 3$

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

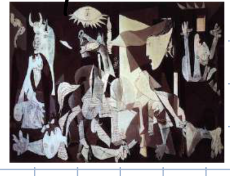
$$G: G - \frac{\partial J(G)}{\partial G}$$

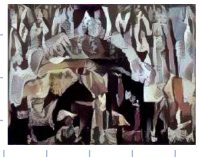
An example: Initialized G



Content

Style





Final Output G

The Content Cost function: $J_{content}(C, G)$

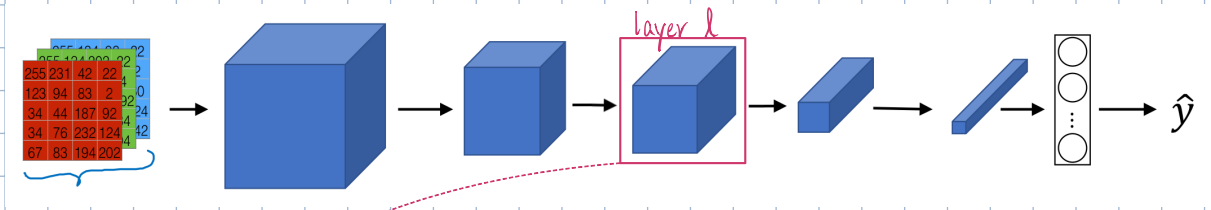
- Use the hidden layer l to compute content cost. The hidden layer l can't be neither too shallow nor too deep. Explain why.
- Use pre-trained ConvNet (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activiation of layer l on the Content image C and the generated image G.
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content.

Define $J_{content}(C, G) = \frac{1}{2} \underbrace{\| a^{[l](C)} - a^{[l](G)} \|_2^2}$

L2 norm of two vectors

The Style Cost function: $J_{style}(S, G)$

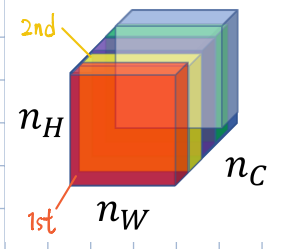
Question: What is the meaning of the "style" of an image?



Answer:

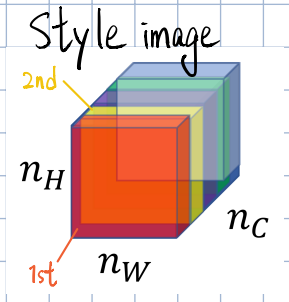
Step 1: Choose a layer l and use l's activiation to measure "style".

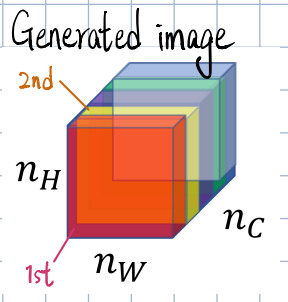
Step 2: Define the "style" as correlation between activations across channels.



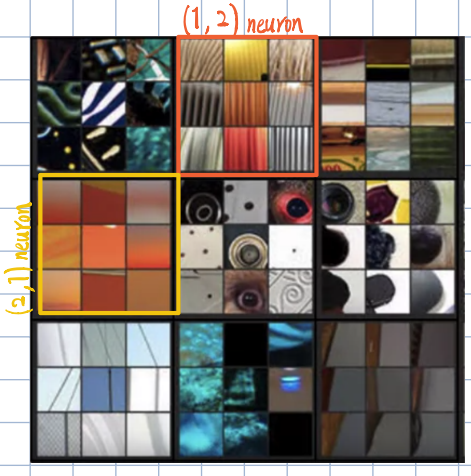
How "correlated" are the activations across different channels? E.g., $n_c=5$. We want to know how correlated the 1st channel and the 2nd channel is.

Intuition about style of an image.





Let's say the 1st channel corresponds to (1, 2) neuron, and the 2nd channel corresponds to (2, 1) neuron.



Degree of correlation between channels can be used as a measure of the style.
I can measure the degree to which in the generated image, the 1st channel is correlated or uncorrelated with the 2nd channel and that will tell, in the generated image, how often this type of vertical texture occurs or doesn't occur with this orange-ish tint and this gives a measure of how similar is the style of the generated image to the style of the input style image.

What does it mean for the 1st channel and the 2nd channel to be highly correlated?

It means that, for an input image, whatever part of the image has this type of subtle vertical texture, that part of the image will probably have these orangeish tint.

What does it mean for the 1st channel and the 2nd channel to be uncorrelated?

It means that whenever there is this vertical texture, it probably won't have that orangeish tint.

Style Matrix:

Let $a^{[l]}_{i,j,k}$ = activation at the position (i, j, k) in the hidden layer l.

Height, Width, Channel

"gram matrix"

$n_c^{[l]}$: the number of channels in the layer l

The style matrix: $G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$. $G^{[l]}_{kk'}$, $k = 1, \dots, n_c^{[l]}$, $k' = 1, \dots, n_c^{[l]}$.

For the style image: $G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{i,j,k} a^{[l](S)}_{i,j,k'}$

For the generated image: $G^{[l](G)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](G)}_{i,j,k} a^{[l](G)}_{i,j,k'}$

$$J^{[l]}_{style}(S, G) = \left\| G^{[l](S)} - G^{[l](G)} \right\|^2_{\text{Frobenius norm}}$$

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

The style cost function from multiple different layers. $J_{style}(S, G) = \sum_l \lambda^{[l]} J^{[l]}_{style}(S, G)$

Recall the overall cost function $J(G)$

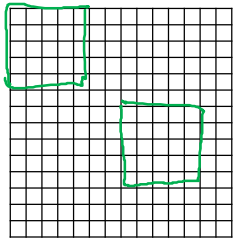
$$J(G) = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$

deeplearning.ai

Convolutional Networks in 1D or 3D

1D and 3D generalizations of models

Convolutions in 2D and 1D



2D input image

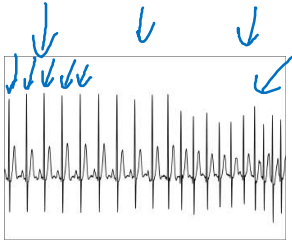
14×14

*

2D filter

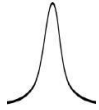
5×5

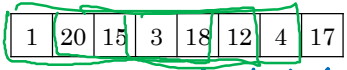
16 filters

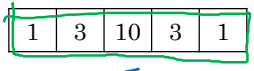


EKG signals

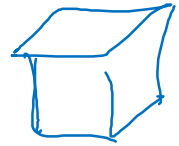
*







16 filters



$14 \times 14 \times 3 \quad * \quad 5 \times 5 \times 3$

$\rightarrow 10 \times 10 \times 16$

$10 \times 10 \times 16 \quad * \quad 5 \times 5 \times 16$

$\rightarrow 6 \times 6 \times 32$

$14 \times 1 \quad * \quad 5 \times 1$

$\rightarrow 10 \times 16$

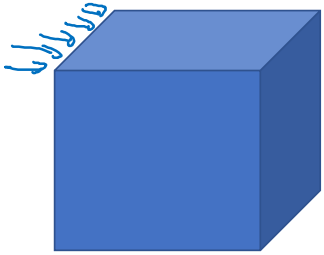
$10 \times 16 \quad * \quad 5 \times 16$

32 filters

$\rightarrow 6 * 32$

Andrew Ng

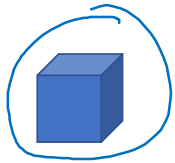
3D convolution



3D volume

14 x 14 x 14

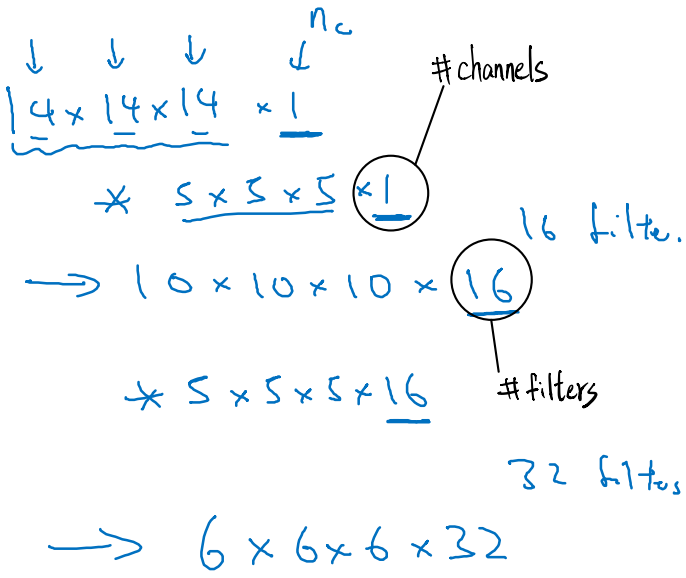
*



3D filter

5 x 5 x 5

16 filters



Everything I learn in ConvNet/CNN can be applied to 1D and 3D data. 3D ConvNet/CNN can be used to detect features from 3D data. E.g., movie, CT scan, point cloud, etc.

Andrew Ng