Copyright Notice

deeplearning.ai

Face recognition

What is face recognition?

Face recognition

[Courtesy of Baidu]

Face verification vs. face recognition

Face verification is a building block for face recognition and is "easier" than face recognition.

→ Verification

1:1

99%

99.9

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

→ Recognition

1:K

K=100 ←

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

To perform face recognition in a scalable, way the best option is to learn a similarity function between pairs of faces. The idea is:

1 - learn a similarity function on a certain dataset (not test dataset)
2 - at runtime compare the test image with all the faces of the employee and see if one matches.

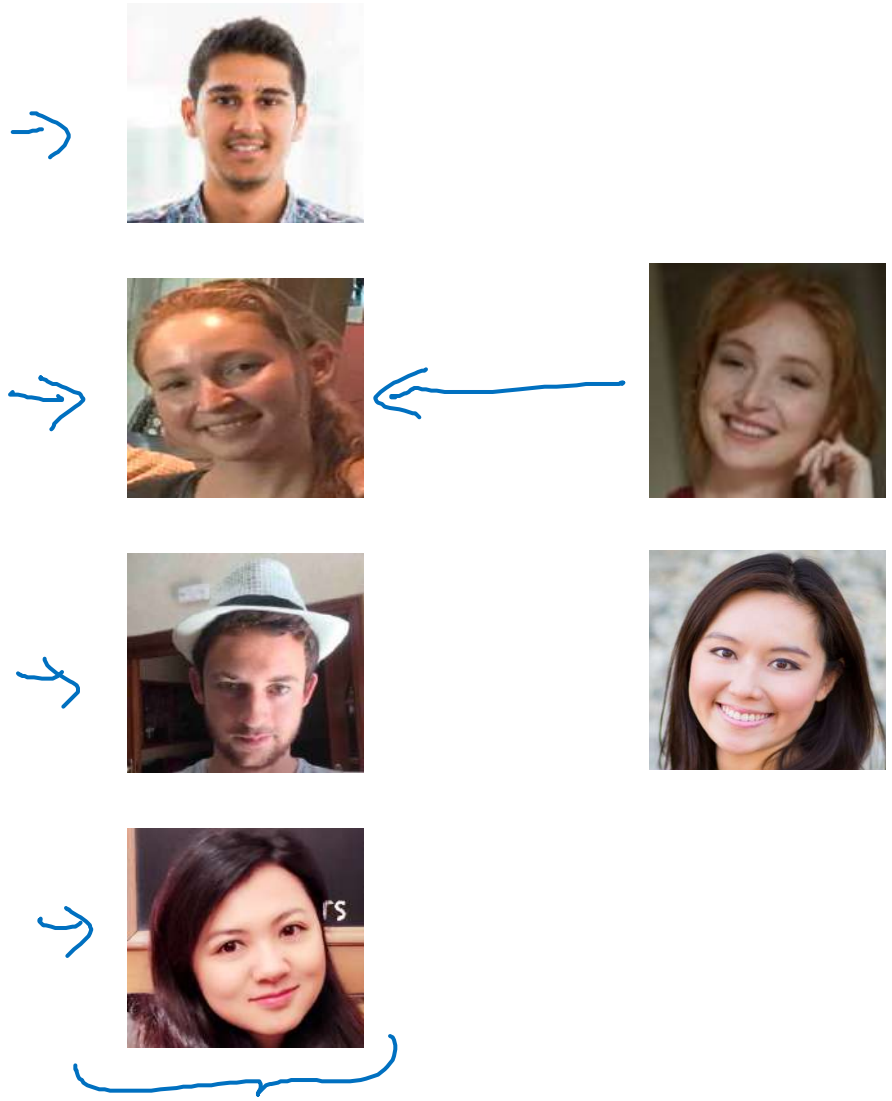
deeplearning.ai

Face recognition

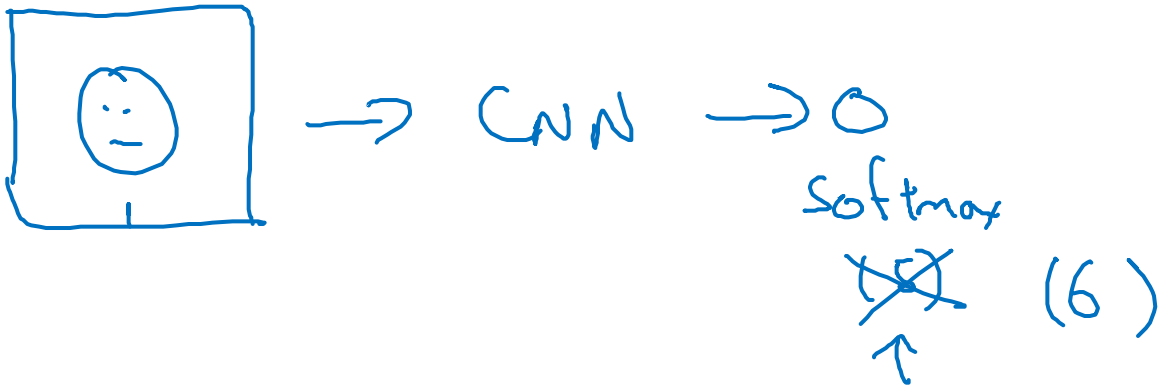
This enables 1-shot learning because i can use my pretrained "similarity network" to compare pairs of my target images.

One-shot learning

One-shot learning



Learning from one example to recognize the person again



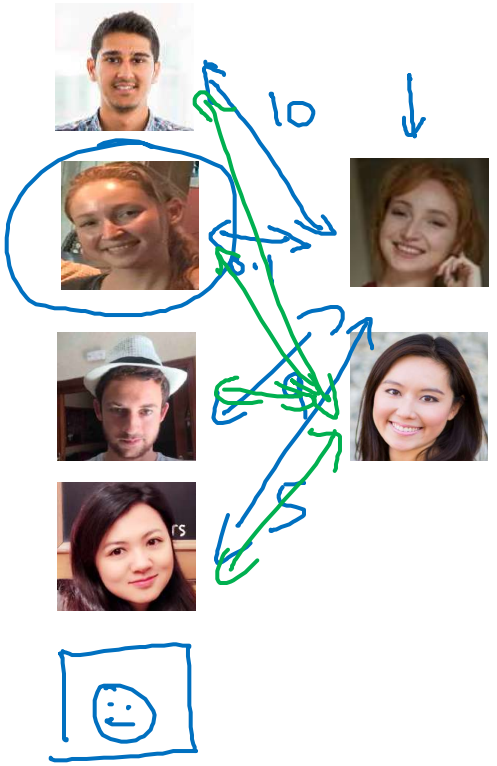
Learning a "similarity" function

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

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

$> \tau$ "different"

} Verification.



d(img1, img2)

To learn a face similarity function one uses Siamese networks. These networks have n replicated branches with shared weights, which are joined at the end by a similarity loss. In case of face recognition

deeplearning.ai

Face recognition

Siamese network

- "triplet loss" is typically used to learn high similarity (small distance) between two images of the same person and low similarity compared to a third image of another person. Of the 3 images:
  - one "anchor" image belongs to the target person
  - one "positive" image belongs to the same person
  - one "negative" image belongs to another person.

Siamese network

NOTE: When using triplet loss you need 3 branches.

$x^{(1)}$

shared weights!

"encoding of $x^{(1)}$"

128

$f(x^{(1)})$

128

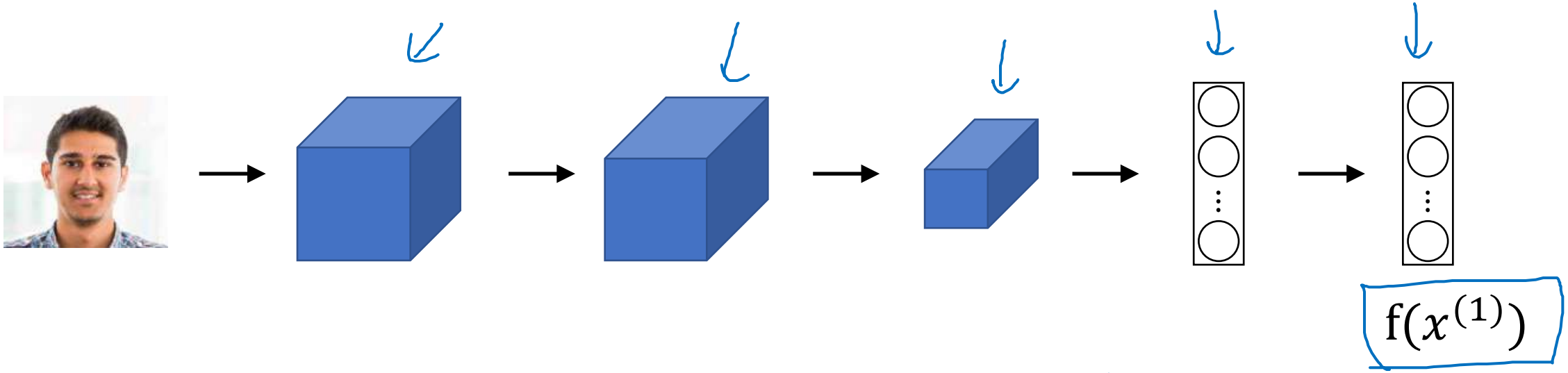
$x^{(2)}$

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

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

[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Goal of learning



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

Learn parameters so that:

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

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

The triplet loss can be used as a loss function in siamese networks to push the networks to learn a similarity function between couples of images. This loss takes 3 images at a time : - an anchor = the person to be recognized
- a positive example = same person
- a negative example = other person

deeplearning.ai

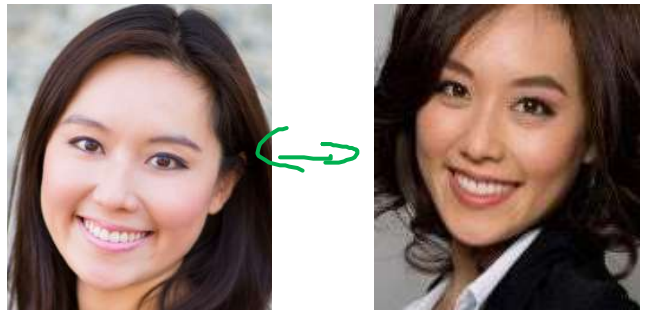
Face recognition

Triplet loss

Learning Objective

idea: you want that $d(A,P) \leq d(A,N)$.

- to avoid the trivial solution (all encodings = 0) actually impose a margin between the two values $d(A,P) + \alpha \leq d(A,N)$.



Anchor (A) Positive (P)

Anchor (A) Negative (N)

$d(A,P) = 0.5$ (encoding)

$d(A,N) = 0.7$

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

$\alpha = 0.2$

$\|f(A) - f(P)\|^2 = d(A,P)$

$\|f(A) - f(N)\|^2 = d(A,N)$

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

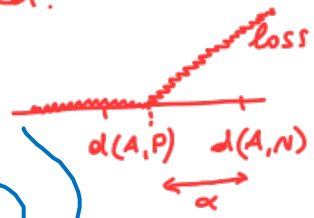
$\|f(A) - f(P)\|^2 \geq 0$, $\|f(A) - f(N)\|^2 \geq 0$, α = margin

$f(\text{img}) = \vec{0}$

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

Loss function

The triplet loss uses a max operation to only "consider" cases where $d(A,P)$ is less than a margin away from $d(A,N)$ and ignore cases when the margin is already respected.



Given 3 images A, P, N:

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

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

NOTE: The SIAMESE NETWORK HAS 3 INPUT BRANCHES.

A, P

Training set: 10k pictures of 1k persons

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

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.

because $d(A,N)$ is more likely to be much greater than $d(A,P)$

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

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

for which $d(A,N) \approx d(A,P)$

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

$d(A,P) \approx d(A,N)$

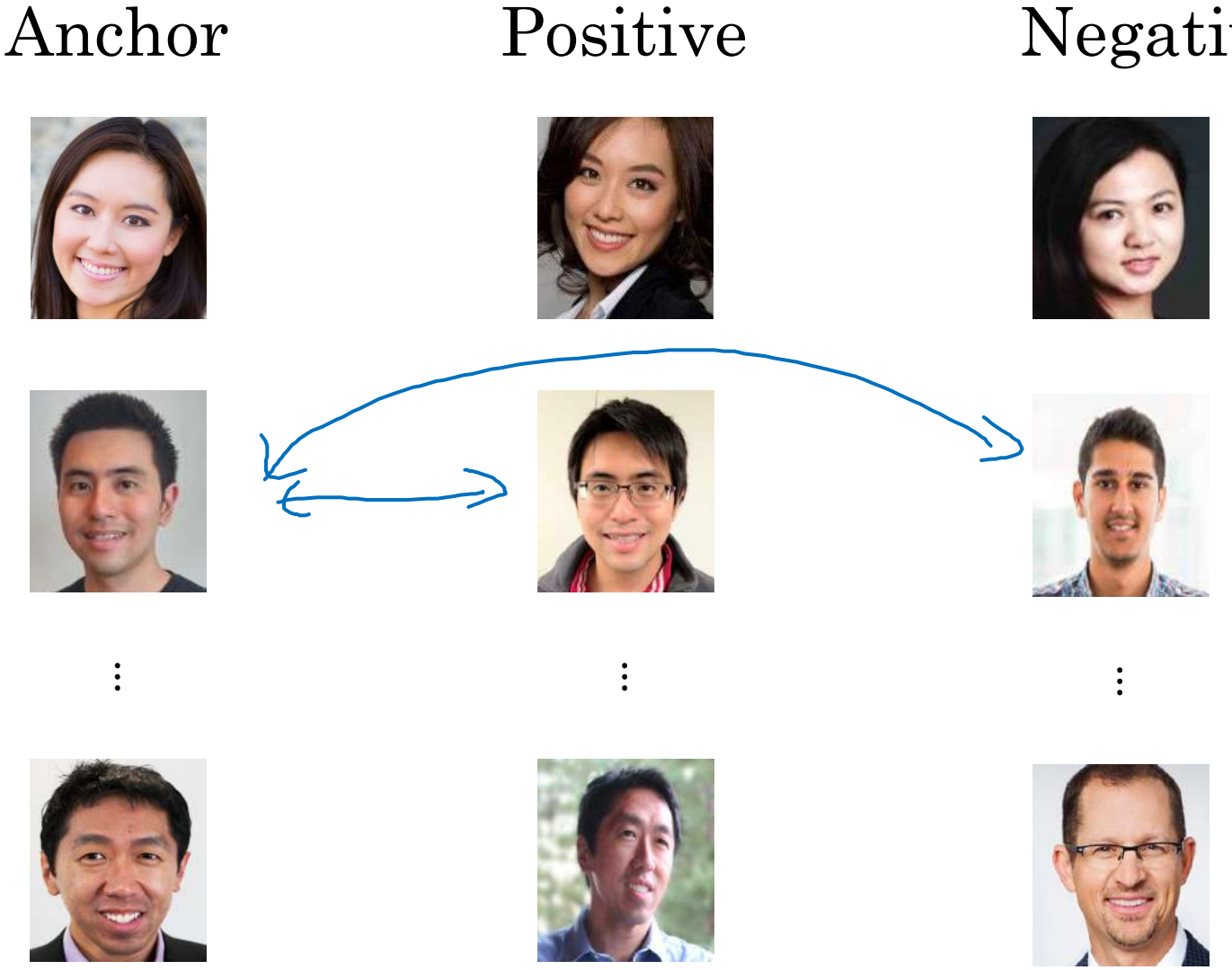
details in this paper

FaceNet

DeepFace

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

Training set using triplet loss



J

$d(x^{(i)}, x^{(j)})$

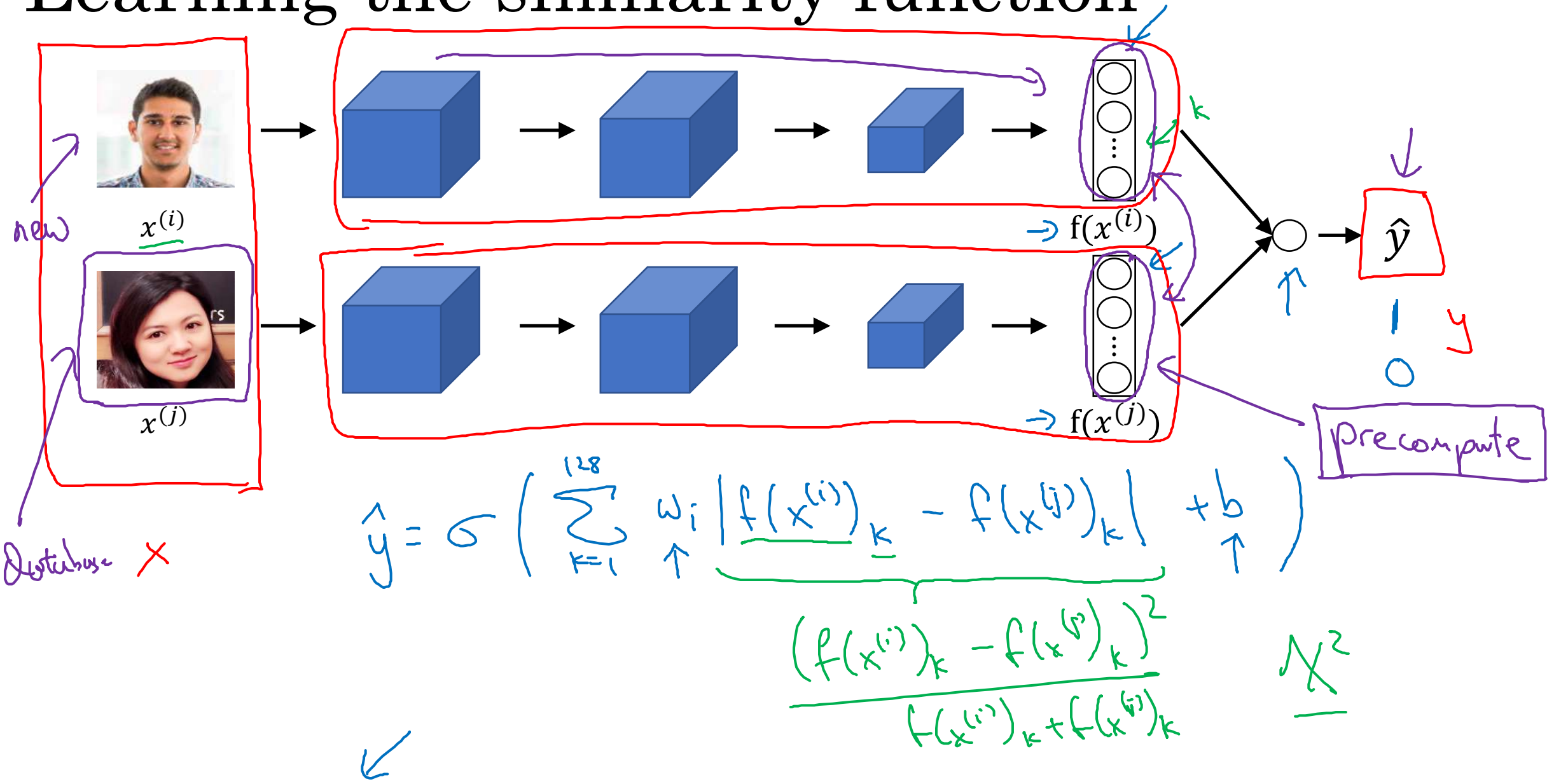
Alternatively to using a triplet loss, one can add a final logistic layer to the network that treats the outputs of two FCs as features for a binary classification problem (same person / different person).

deeplearning.ai

Face recognition

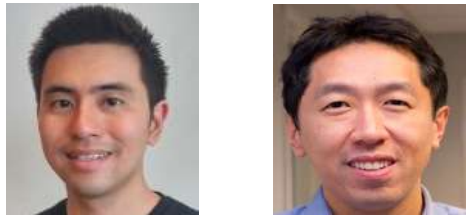
Face verification and binary classification

Learning the similarity function



[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Face verification supervised learning

x	y	
	1	"Same"
	0	"Albert"
	0	
	1	

[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

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)

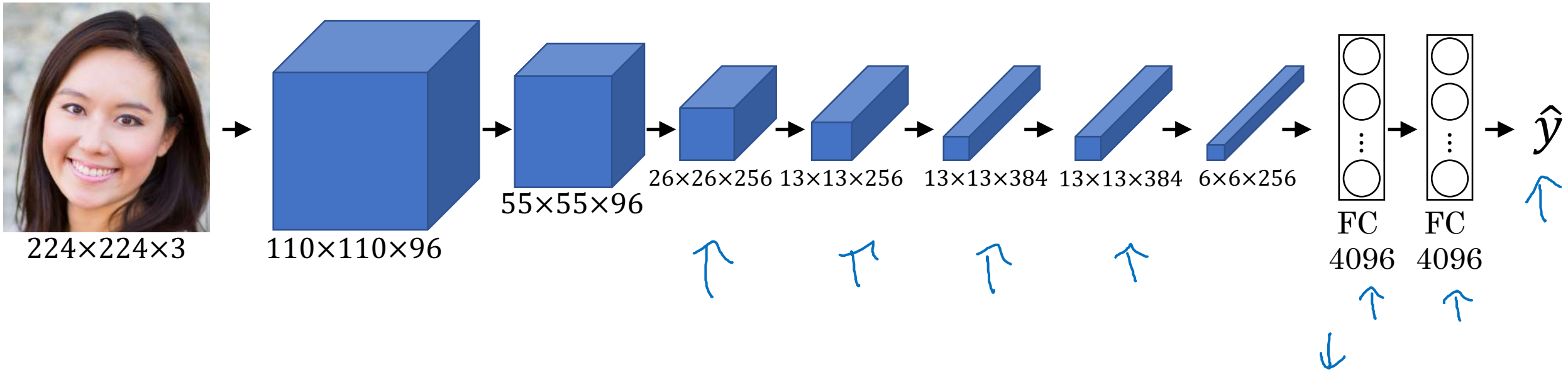
[Images generated by Justin Johnson]

deeplearning.ai

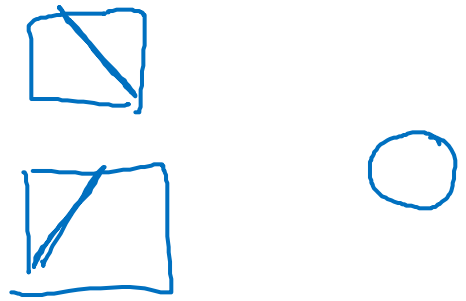
Neural Style Transfer

What are deep ConvNets learning?

Visualizing what a deep network is learning



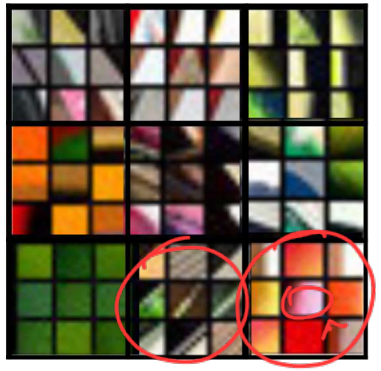
Pick a unit in layer 1. Find the ~~nine~~ ~~image~~ patches of the nine input images in the dataset that maximize the unit's activation. → visualize all 9 image patches.



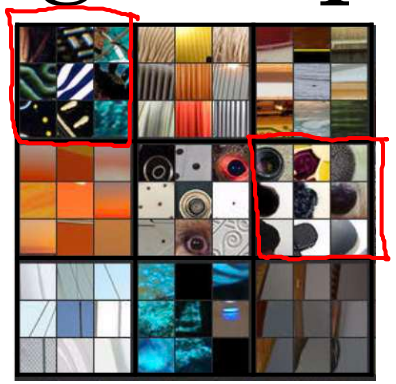
Repeat for other units.

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

Visualizing deep layers

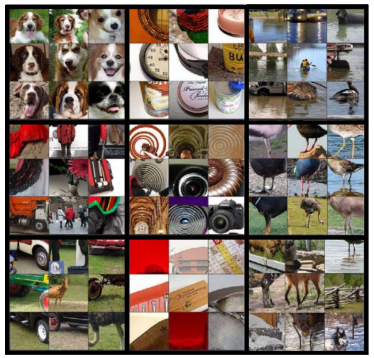


Layer 1

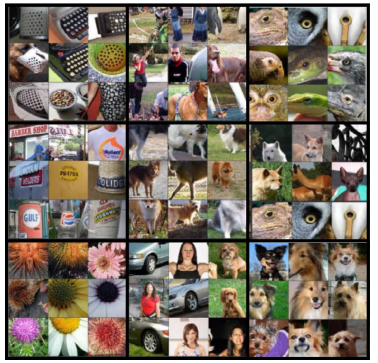


Layer 2

Layer 3



Layer 4

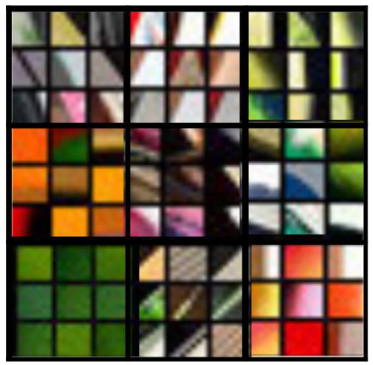


Layer 5

2 neurons of Layer 1

the corresponding patch in one of the 9 images of the dataset that mostly activated that neuron

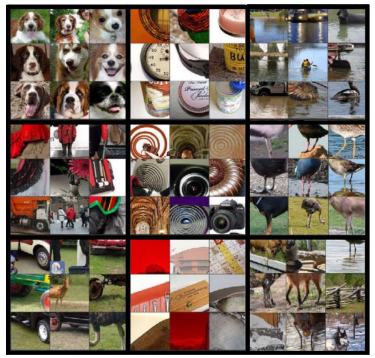
Visualizing deep layers: Layer 1



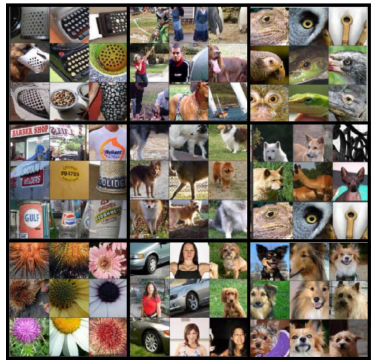
Layer 1

Layer 2

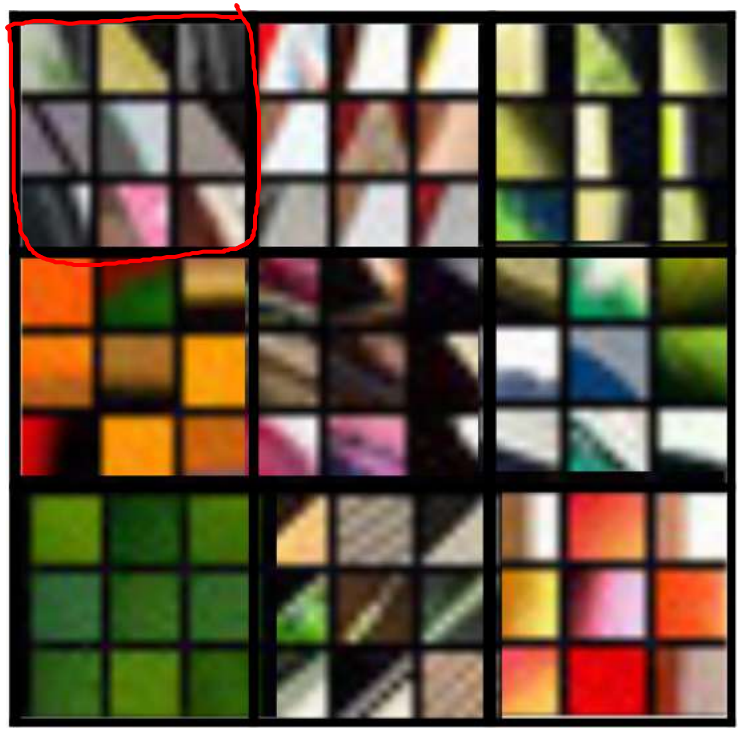
Layer 3



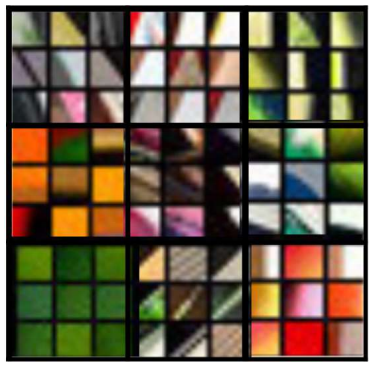
Layer 4



Layer 5



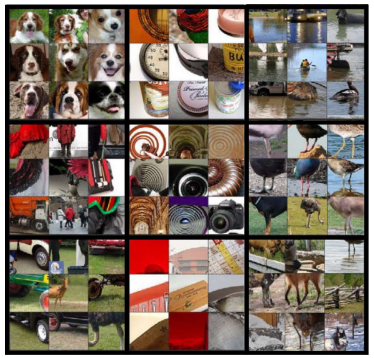
Visualizing deep layers: Layer 2



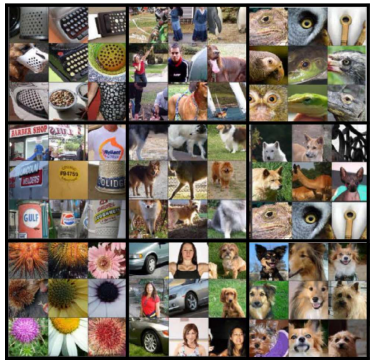
Layer 1

Layer 2

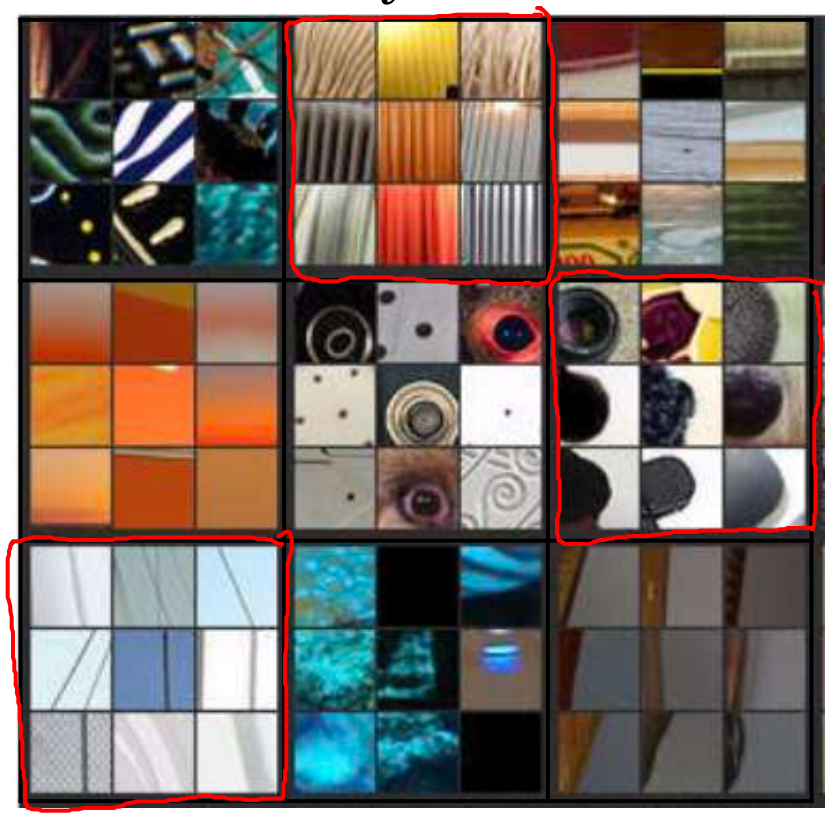
Layer 3



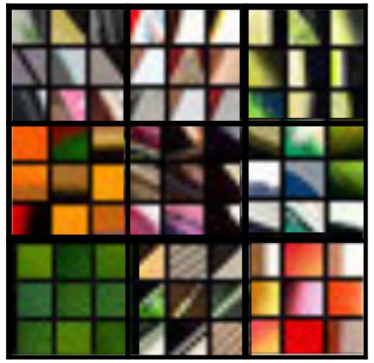
Layer 4



Layer 5



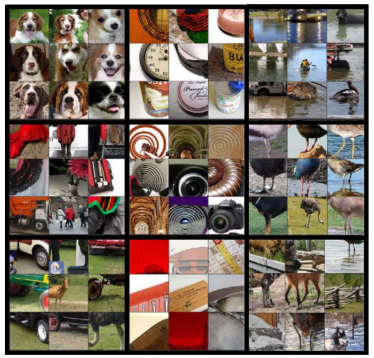
Visualizing deep layers: Layer 3



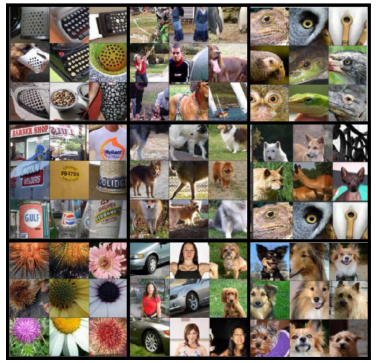
Layer 1

Layer 2

Layer 3

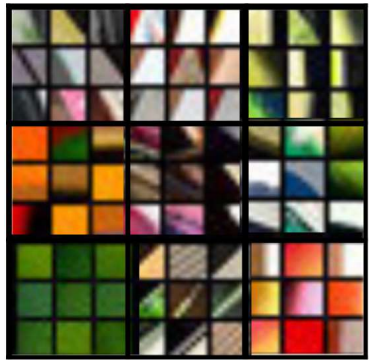


Layer 4

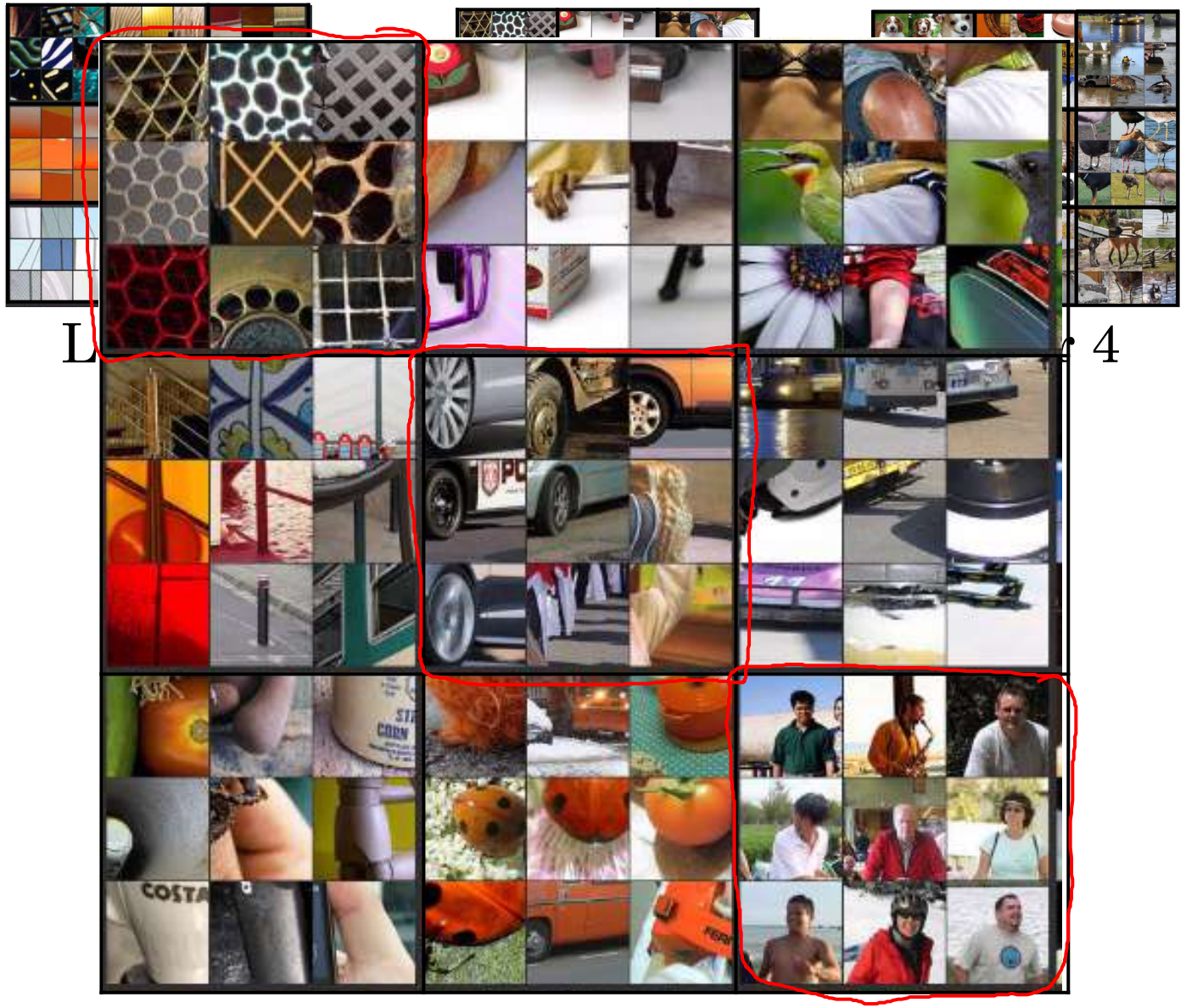


Layer 5

Visualizing deep layers: Layer 3

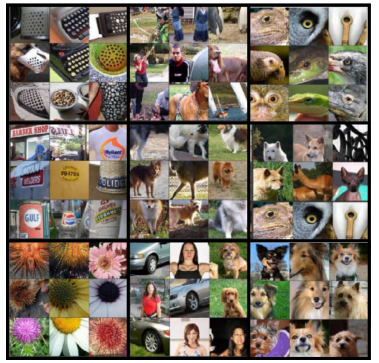


Layer 1



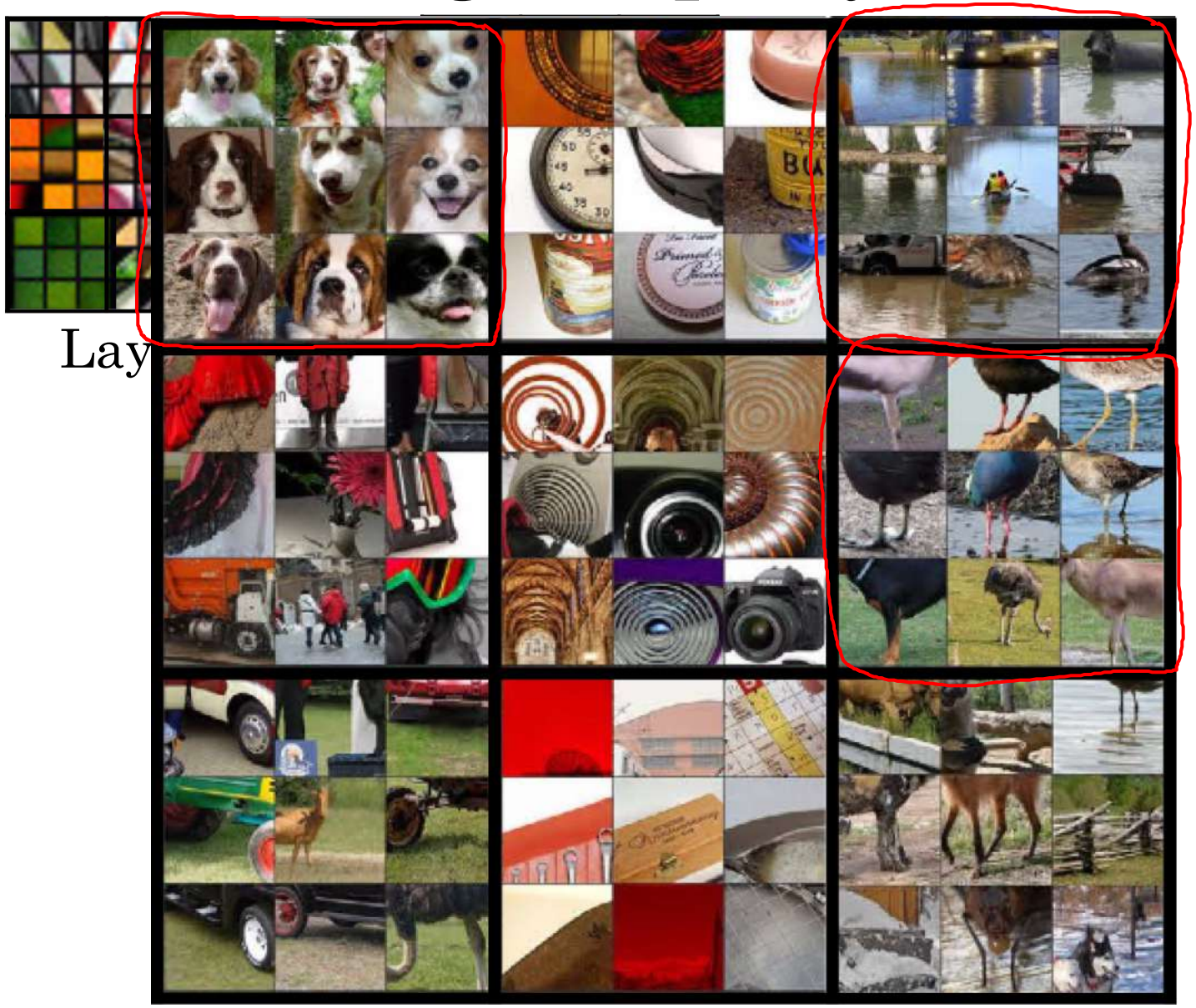
L

4

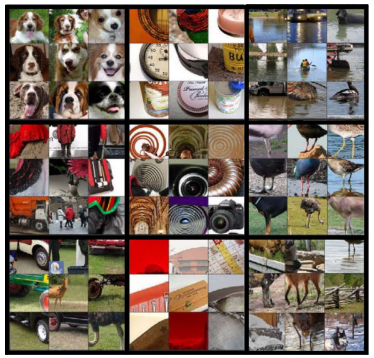


Layer 5

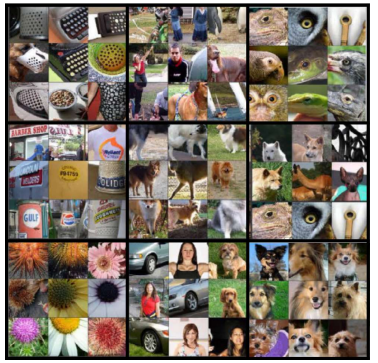
Visualizing deep layers: Layer 4



Lay

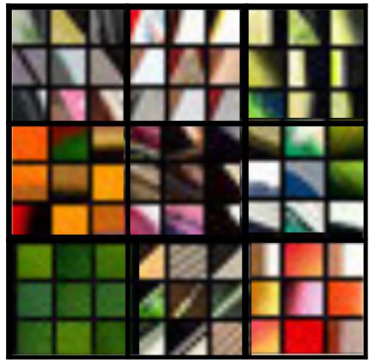


Layer 4

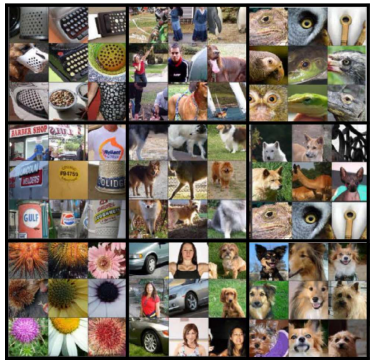


Layer 5

Visualizing deep layers: Layer 5



Layer 1



Layer 5

Andrew Ng

These are the steps for neural style transfer:

- Take content (C), style (S) images and generate a stylized image randomly (G).
- Pass C, S, G into a pretrained CNN up to a certain intermediate layer l and compute the difference in contents between the representations of C and G and the difference in style between S and G. The difference in style requires the preliminary computation of the style matrices of S and G.
- Based on the content and style differences, compute the loss associated with the current G.

Neural Style Transfer

deeplearning.ai

- At this point

Cost function

idea: you want similarity between:

- generated image and content image
- style of the generated image and content image

Neural style transfer cost function

Content C

Style S

Generated image G

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

cost function

Similarity between C and G

Similarity between the style of S and G

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

Find the generated image G

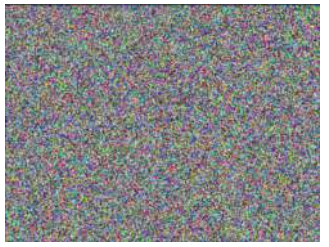
1. Initiate G randomly

   G: 100×100×3

   ↑ RGB

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

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



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

deeplearning.ai

Neural Style Transfer

Content cost function

Content cost function

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

mid-level (not too shallow, not too deep)

- 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

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

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

Neural Style Transfer

deeplearning.ai

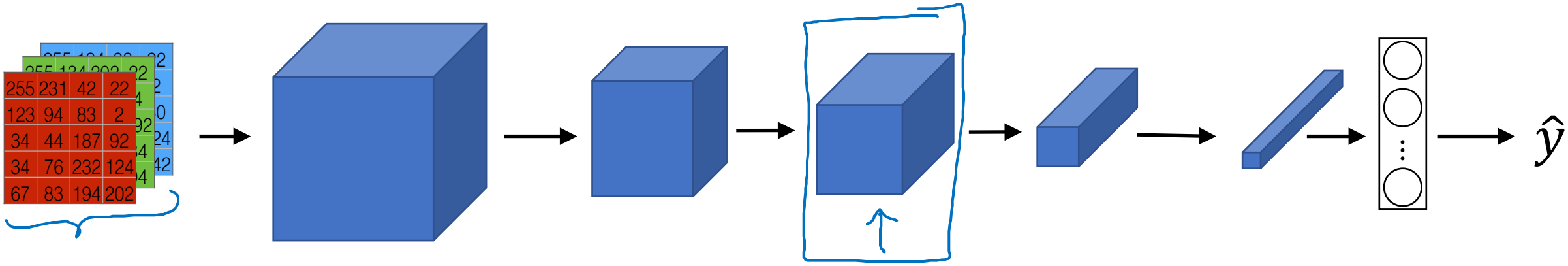
Style cost function

IDEA: The difference in style between two images is measured as the difference of their respective "style matrices". These matrices are computed on the feature representation of an image at an hidden layer of a pretrained CNN (3D feature representation $W \times h \times c$).

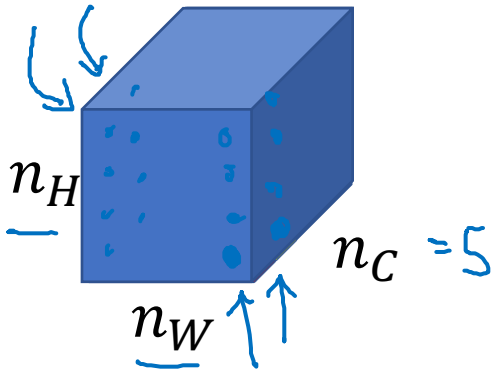
The style of an image (feature representation) is measured depending on the correlation between its channels. The intuition is that the correlation between two channels is telling you how likely two high-dimensional features are occuring together in the original image. In essence:

- take images (S)tyle and (G)enerated, pass them through a CNN and take their representation at layer l.
- for each of them compute the style matrix $G^{(S)}$ and $G^{(G)}$ (G stands for GRAM matrix actually). The style matrix is of size channel x channel and on the element kk' they have the correlation of channels k and k' (that is G_k linearized $\cdot G_{k'}$ linearized).
- The loss is based on the Frobenius norm between the style and generated images.
- The loss is backpropagated through the network (with the G as input), keeping all the weights fixed except for those leading back to the input pixels => change G.

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.

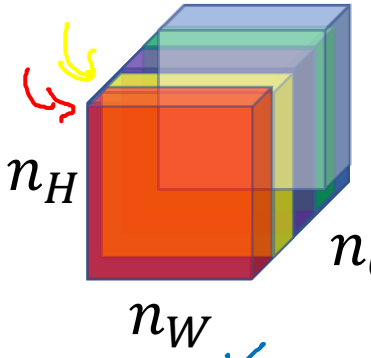


How correlated are the activations across different channels?

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

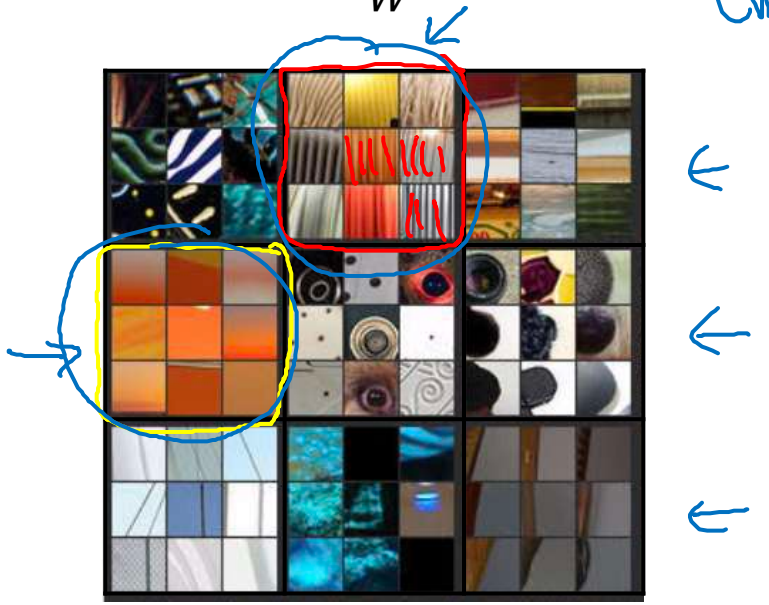
Intuition about style of an image

Style image

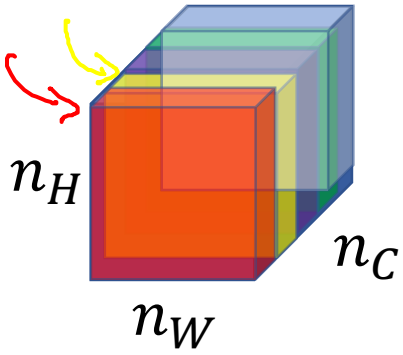


Correlated?

Uncorrelated



Generated Image



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

Style matrix

= GRAM MATRIX

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

n_c

$G^{[l]}_{kk'}$

$k = 1, \ldots, n_c^{[l]}$

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

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

"Gram matrix"

$$J^{[l]}_{style}(S, G) = \frac{1}{(\ldots)} \left\| G^{[l](S)} - G^{[l](G)} \right\|_F^2$$

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

β

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

Style cost function

$$\left\| G^{[l](S)} - G^{[l](G)} \right\|_F^2$$

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

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

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

G

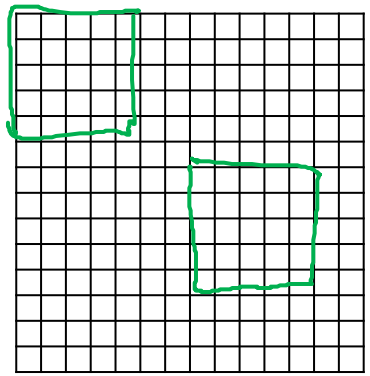
[Gatys et al., 2015. A neural algorithm of artistic style]

deeplearning.ai

Convolutional Networks in 1D or 3D

1D and 3D generalizations of models

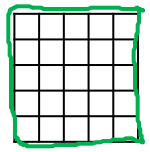
Convolutions in 2D and 1D



2D input image

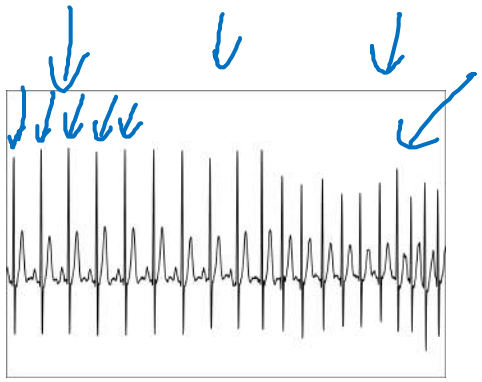
14×14

*

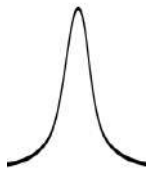


2D filter

5×5



*

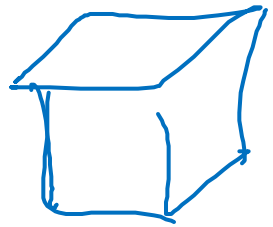


1	20	15	3	18	12	4	17

14

1	3	10	3	1

5



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

$\rightarrow 10 \times 10 \times 16$

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

$\rightarrow 6 \times 6 \times 32$

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

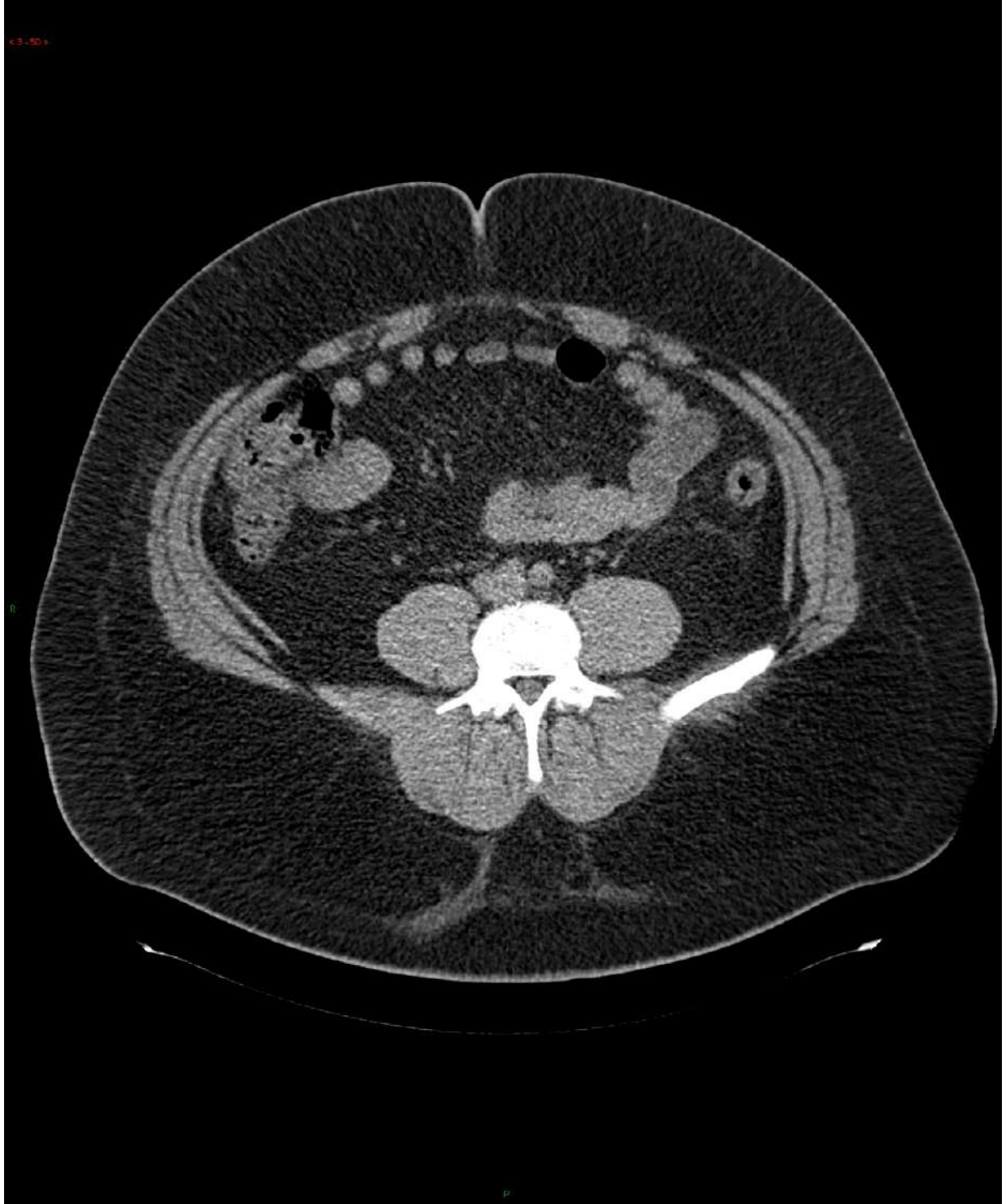
$\rightarrow 10 \times 16$

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

$\rightarrow 6 \times 32$

Andrew Ng

3D data

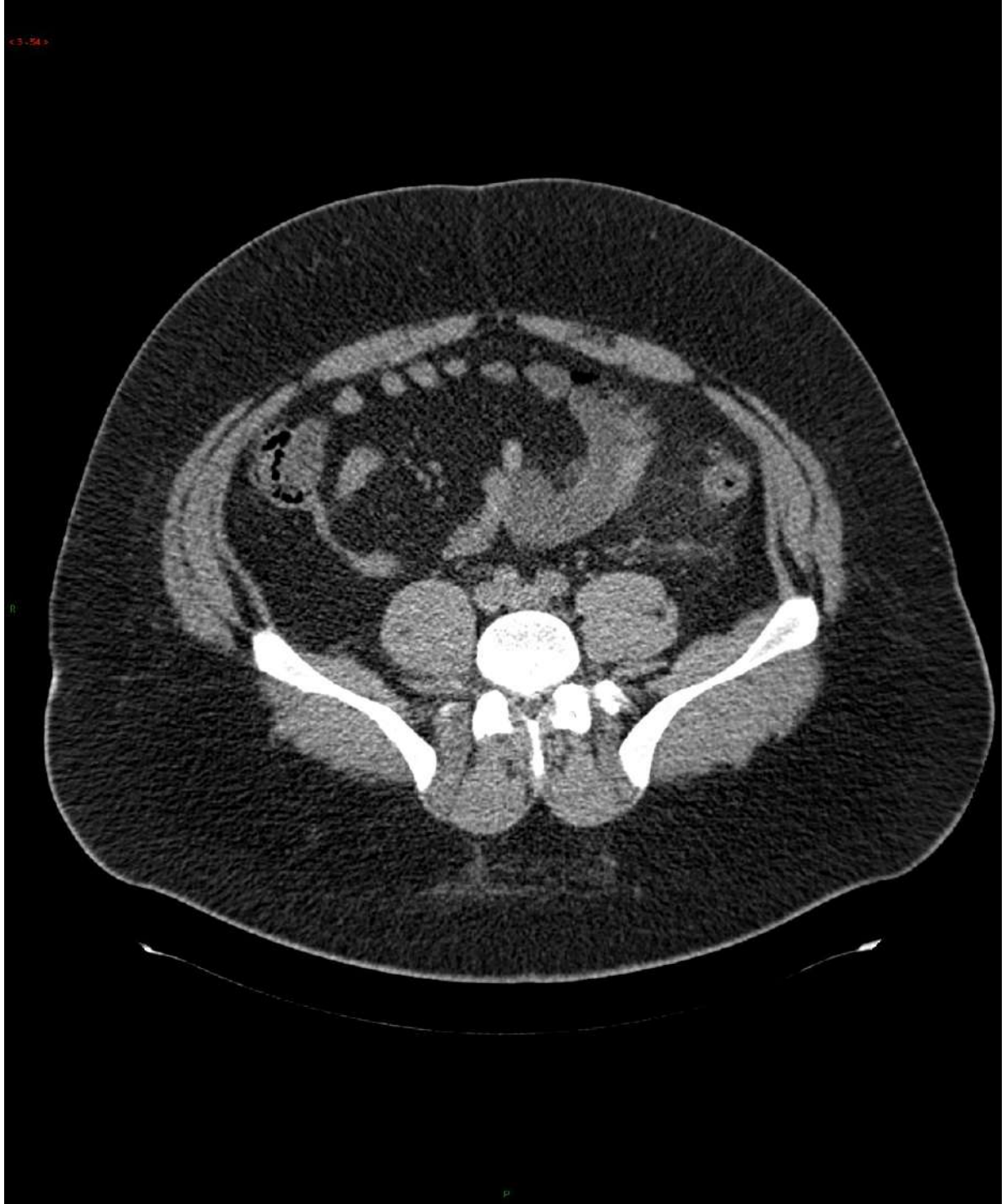


Andrew Ng

3D data

Andrew Ng

3D data

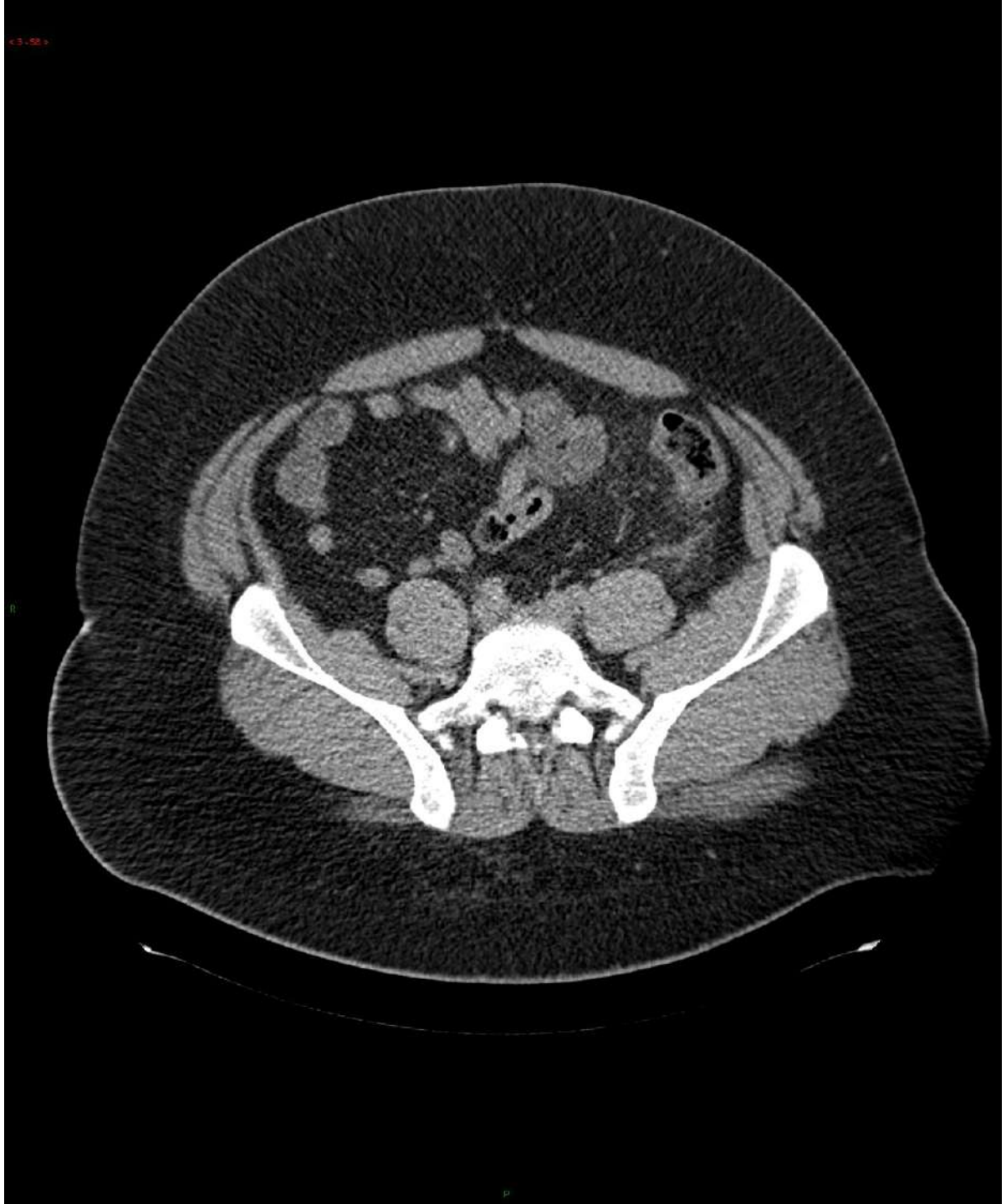


Andrew Ng

3D data

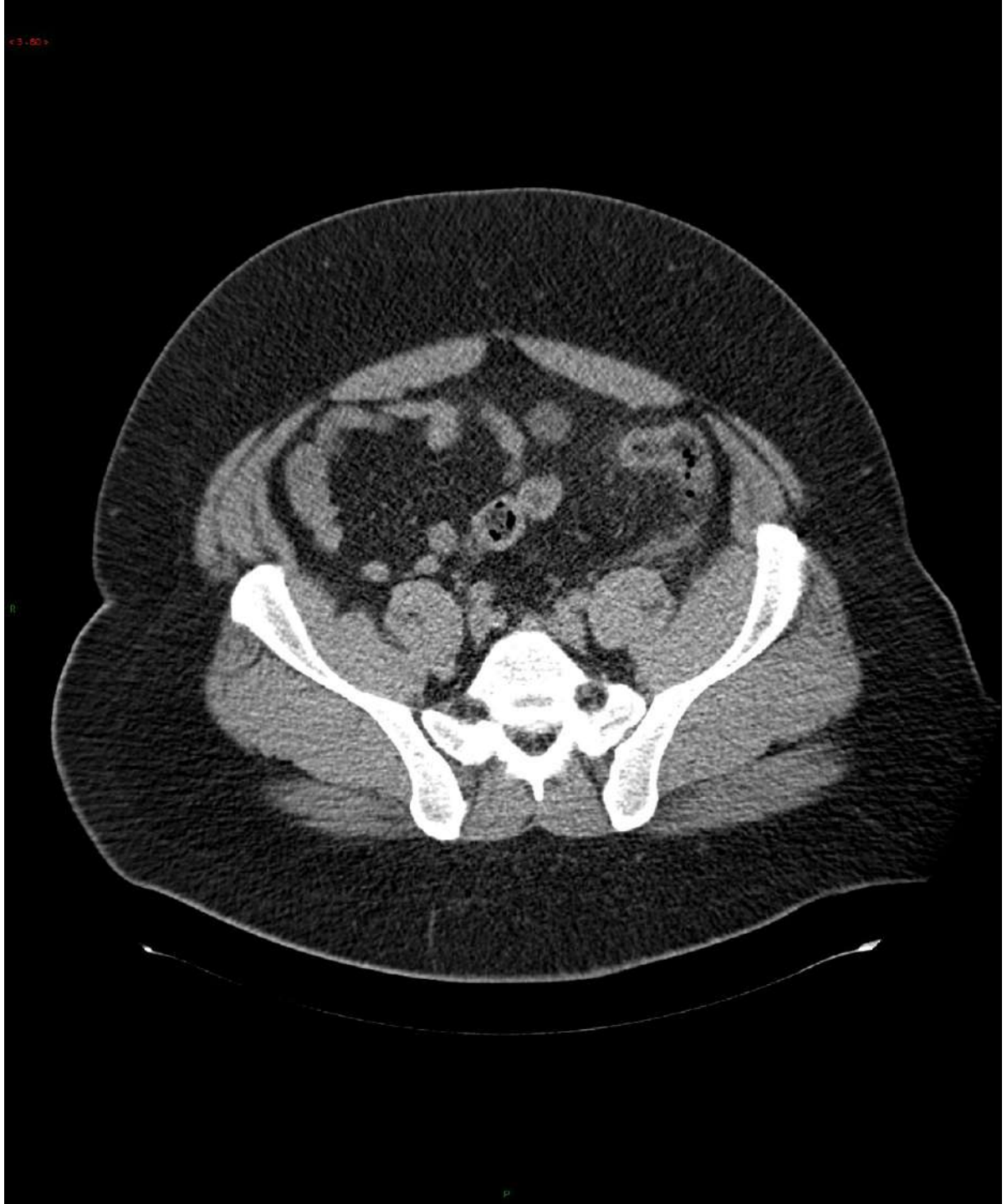
Andrew Ng

3D data



Andrew Ng

3D data



Andrew Ng

3D data

Andrew Ng

3D data

Andrew Ng

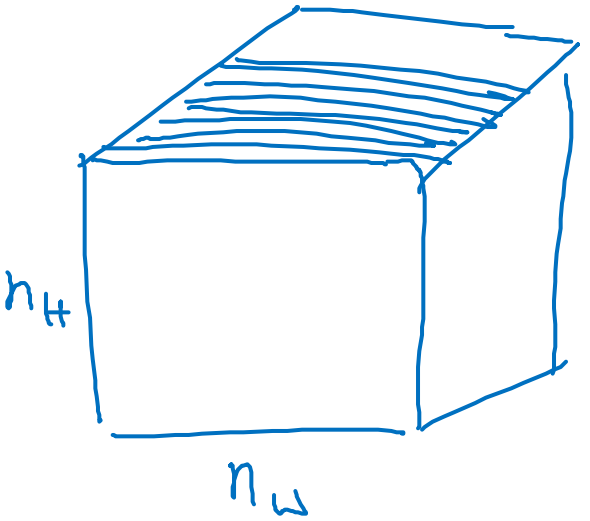
3D data

Andrew Ng

3D data

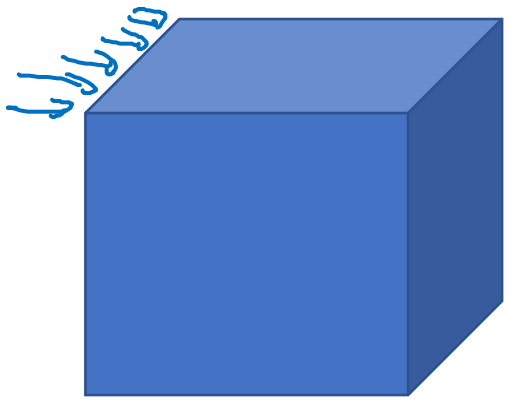
Andrew Ng

3D data



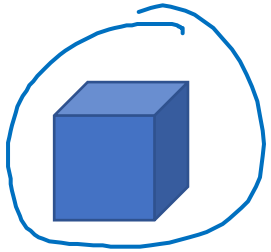
Andrew Ng

3D convolution



3D volume

*



3D filter

$14 \times 14 \times 14 \times 1$ (n_c)

$* \; 5 \times 5 \times 5 \times 1$

16 filters.

$\rightarrow 10 \times 10 \times 10 \times 16$

$* \; 5 \times 5 \times 5 \times 16$

32 filters

$\rightarrow 6 \times 6 \times 6 \times 32$