Face recognition

FPT Education

FPT UNIVERSITY

Learning Objectives:

- Differentiate between face recognition and face verification
- Implement one-shot learning to solve a face recognition problem
- Apply the triplet loss function to learn a network's parameters in the context of face recognition
- Explain how to pose face recognition as a binary classification problem
- Map face images into 128-dimensional encodings using a pretrained model

Face recognition

Learning Objectives:

- Perform face verification and face recognition with these encodings
- Implement the Neural Style Transfer algorithm
- Generate novel artistic images using Neural Style Transfer
- Define the style cost function for Neural Style Transfer
- Define the content cost function for Neural Style Transfer

Face recognition

FPT Education

FPT UNIVERSITY

FPT Education

FPT UNIVERSITY

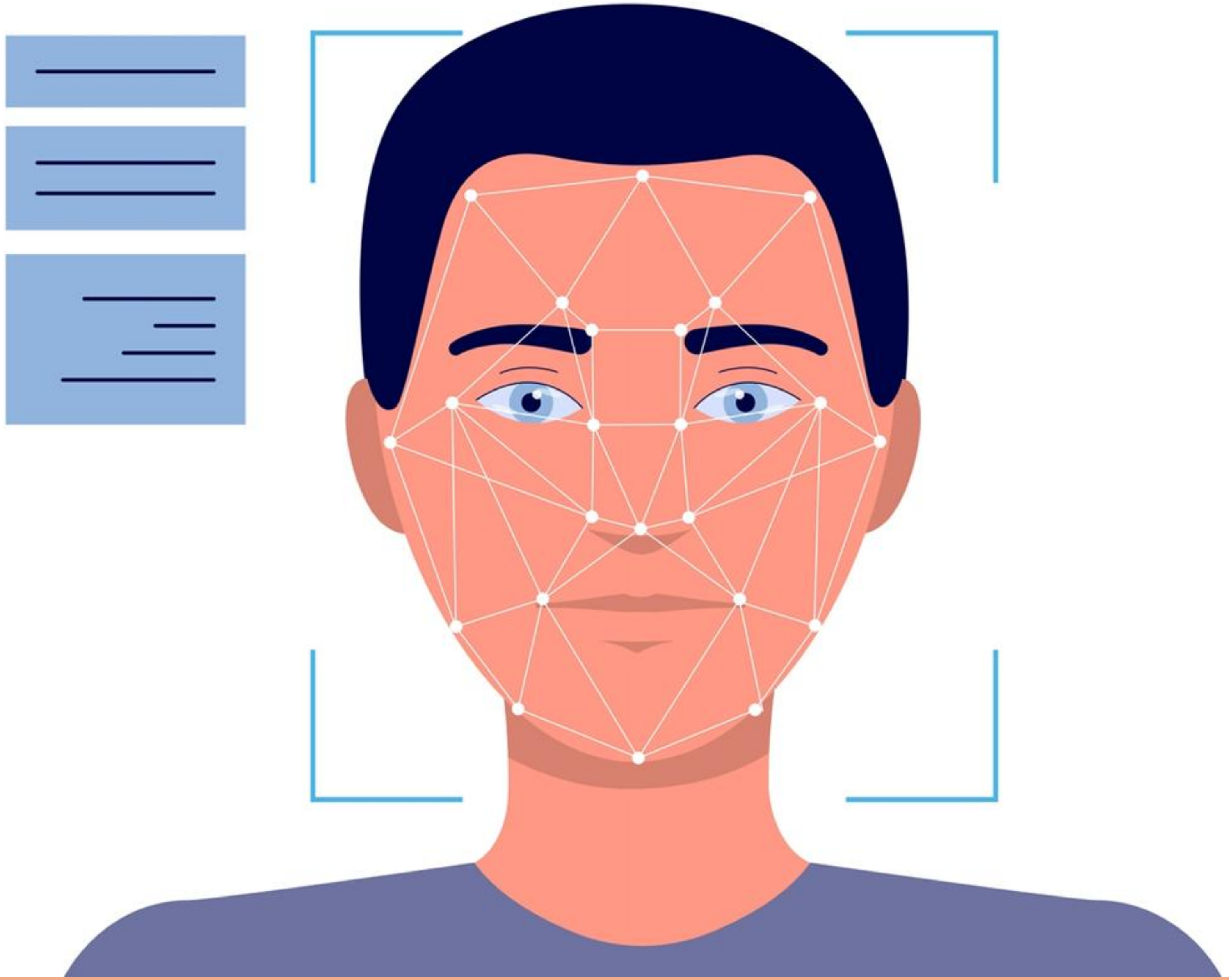
What is face recognition?

Face recognition

Face recognition

- Face recognition is a special application of convolutional neural networks.
- Face verification is a one-to-one problem where the system verifies whether an input image matches the claimed identity of a person.
- Face recognition is a much harder problem that involves identifying a person from a database of many people.
- Face recognition requires high accuracy, which is difficult to achieve due to the one-shot learning problem.
- The next section will discuss how to build a face verification system, which can be used as a building block for a face recognition system.

Face recognition



Face verification vs. face recognition

- Verification
- Input image, name/ID
- Output whether the input image is that of the claimed person
- Recognition
- Has a database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or “not recognized”)

One-shot learning

Face recognition

One-shot learning

- The one-shot learning problem with face recognition is recognizing a person with just one image. A ConvNet with a softmax unit can output a label, but it doesn't work well for a small training set and requires retraining for each new person.
- A similarity function, denoted as d, is learned using a neural network that inputs two images and outputs the degree of difference between them. If the difference is less than a threshold, the two images are predicted to be of the same person. This function d solves the one-shot learning problem and allows adding new people to the database without retraining the network.

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) $\leq \tau$

$> \tau$

Siamese network

Face recognition

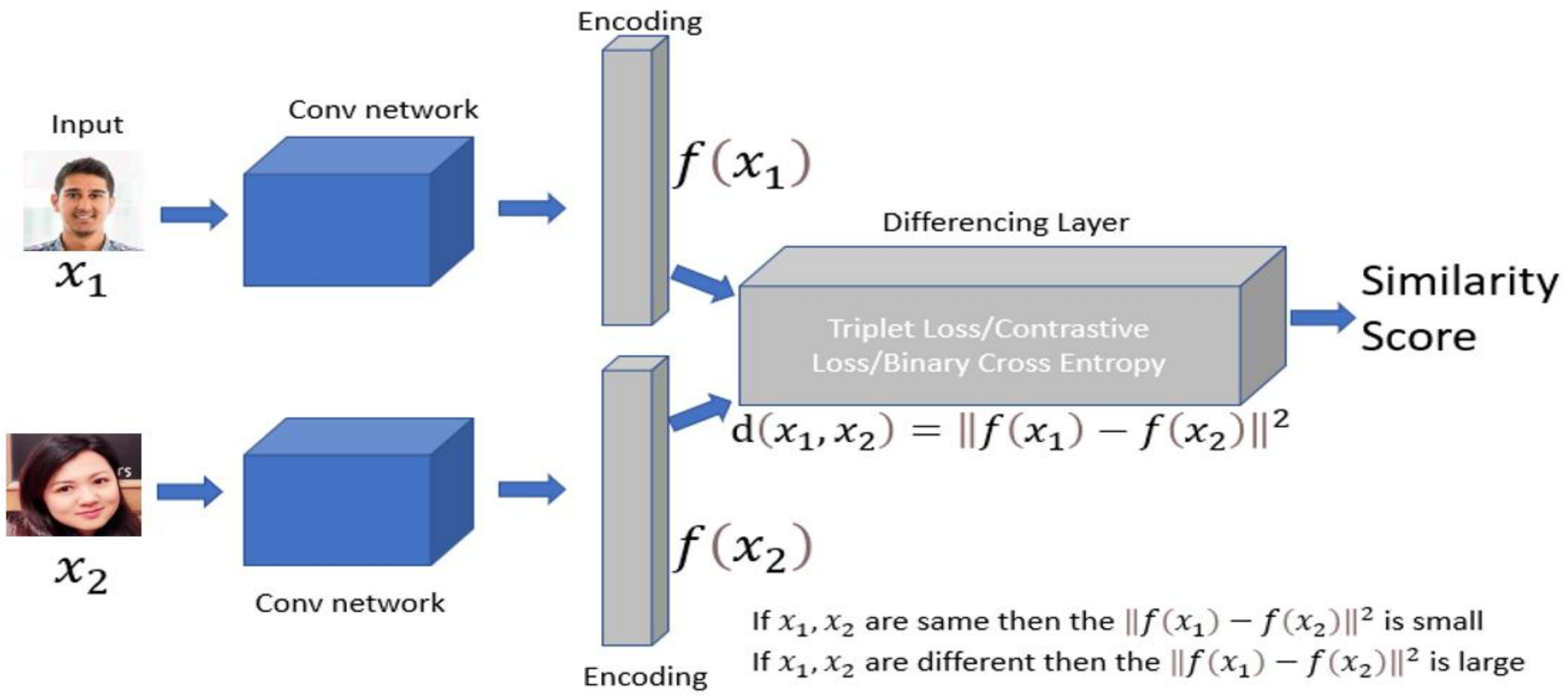
Siamese network

- A face recognition system needs to be able to recognize a person based on just one single image or example of that person's face.
- This is known as the one-shot learning problem, and it is challenging because deep learning algorithms historically do not work well with only one training example.
- One approach to solving this problem is to learn a similarity function, denoted as d, that inputs two images and outputs the degree of difference between them.

Siamese network

- During recognition, if two images have a degree of difference below a certain threshold called tau, they are considered to be of the same person.
- The Siamese network architecture is used to train a neural network to learn the similarity function. This involves running two identical convolutional neural networks on two different inputs and comparing them.
- The objective function is to learn parameters that make the distance between the encodings of images of the same person small and that of images of different persons large. The triplet loss function is used to define this objective function.

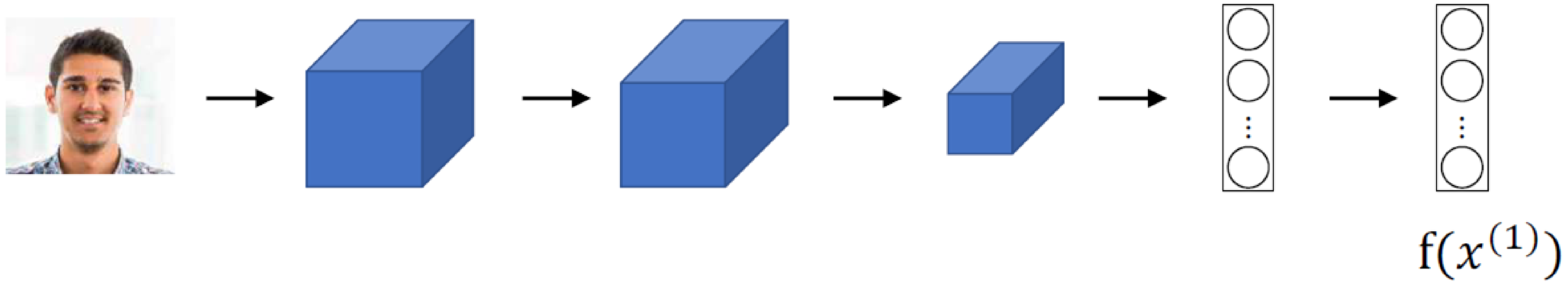
Siamese network



Input
x_1
x_2
Conv network
Conv network
Encoding
Encoding
$f(x_1)$
$f(x_2)$
Differencing Layer
Triplet Loss/Contrastive Loss/Binary Cross Entropy
$d(x_1, x_2) = \|f(x_1) - f(x_2)\|^2$
Similarity Score
If x_1, x_2 are same then the $\|f(x_1) - f(x_2)\|^2$ is small
If x_1, x_2 are different then the $\|f(x_1) - f(x_2)\|^2$ is large

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

Goal of learning



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

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.

Triplet loss

Triplet loss

- The triplet loss is a loss function used to train a neural network to encode images such that the encoding of images of the same person is closer together in some sense than the encoding of images of different people.
- The triplet loss is defined by looking at triplets of images: an anchor image (A), a positive image (P), and a negative image (N).
- The positive image is an image of the same person as the anchor, while the negative image is an image of a different person than the anchor.
- The goal of the triplet loss is to minimize the distance between the anchor and positive images while maximizing the distance between the anchor and negative images.

Learning Objective

- Formally, the triplet loss is defined as

  L=max(||f(A)-f(P)||^2-||f(A)-f(N)||^2+α,0),

  - where f(.) is the encoding function, ||.|| is the L2-norm, and α is a margin parameter that is used to make sure the neural network does not output trivial solutions

Learning Objective

Anchor
A

Positive
P

Anchor
A

Negative
N

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

$$L(A,P,N) = \max(0, || F(A) - F(P) ||^2 < || F(A) - F(N) ||^2 + m)$$

Loss function

- To train the neural network using the triplet loss, a dataset of triplets must be constructed.
- However, choosing triplets randomly can be computationally inefficient, as too many triplets may be too easy for the neural network to learn. Instead, it is recommended to choose "hard" triplets, where the distance between the anchor and positive images is close to the distance between the anchor and negative images.

Loss function

Given 3 image A, P, N

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

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

Training set: 10k pictures of 1k persons

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.

$$||f(A) - f(P)||^2 + \boldsymbol{\alpha} \leqslant ||f(A) - f(N)||^2$$

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

$$d(A,P) + \boldsymbol{\alpha} \leqslant d(A,N)$$

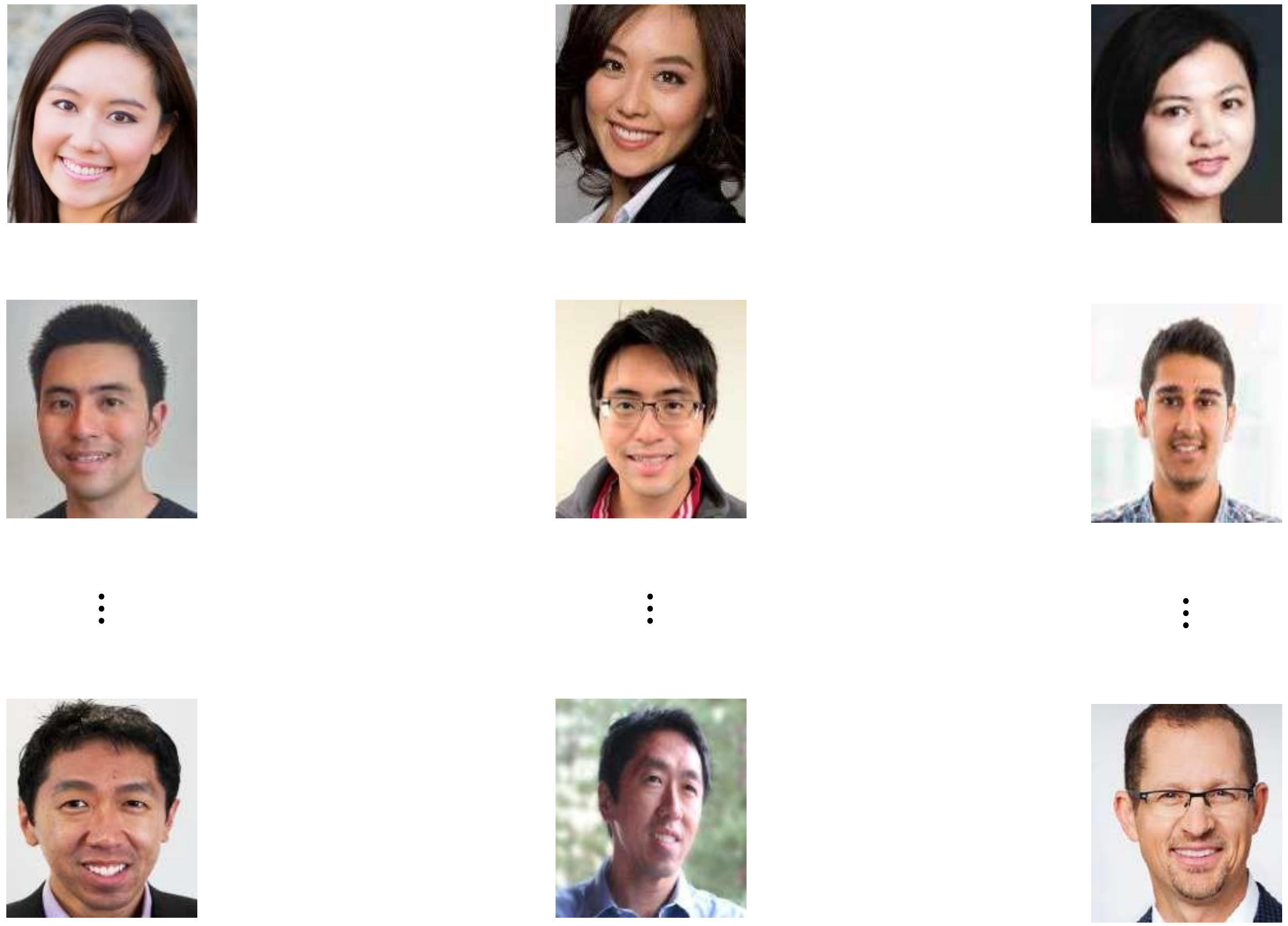
$$d(A,P) \cong d(A,N)$$

Training set using triplet loss

- Large-scale face recognition systems are often trained on datasets with millions or even tens of millions of images, making it difficult to train a model from scratch.
- Therefore, it may be useful to download pre-trained models from companies that have posted their parameters online.
- However, it is still important to understand how these algorithms were trained in case modifications or new applications are needed.

Training set using triplet loss

Anchor Positive Negative



J

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

FPT Education

FPT UNIVERSITY

Face verification and binary classification

Face recognition

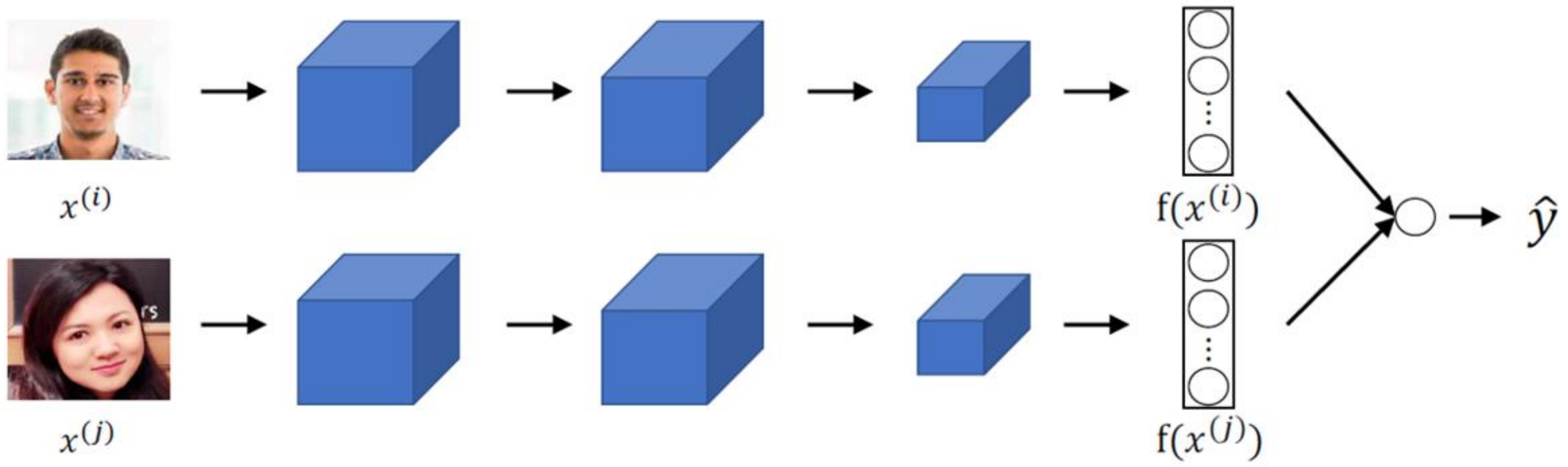
Face verification and binary classification

- The Triplet Loss and the Siamese Network are two methods for training a neural network to perform face recognition.
- The Triplet Loss involves training a neural network to learn a 128-dimensional encoding of a face image that is unique to each individual.
- The Siamese Network, on the other hand, treats face recognition as a binary classification problem, where the network takes in a pair of images and outputs a prediction of whether they depict the same person or not.

Learning the similarity function

- The Siamese Network uses two neural networks to compute the embeddings for each image in a pair, and a logistic regression unit to predict whether the images depict the same person or not based on the absolute value of the differences between the embeddings.
- The network is trained on pairs of images using backpropagation to adjust the weights and improve the accuracy of the predictions.

Learning the similarity function



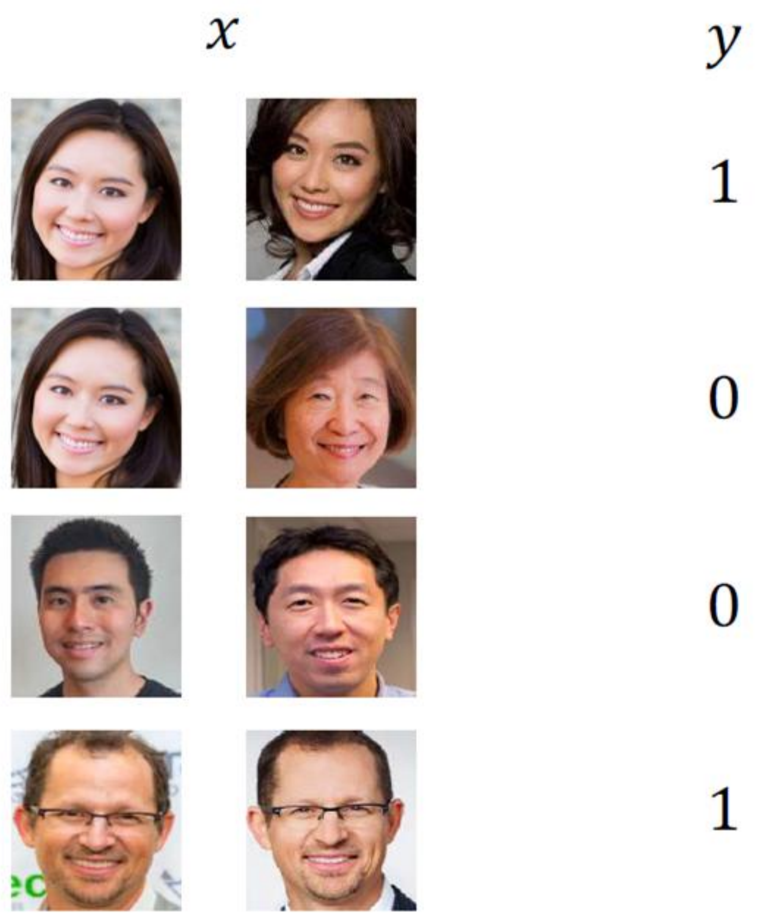
$$\mathcal{J} = \sum_{i=1}^{m} \left[\| f(A^{(i)}) - f(P^{(i)}) \|_2^2 - \| f(A^{(i)}) - f(N^{(i)}) \|_2^2 + \alpha \right]_+$$

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

Face verification supervised learning

- To improve the efficiency of the face recognition system, pre-computed encodings can be used instead of computing the embeddings for each image in real-time. This can save significant computation time, especially for large databases of images.
- Overall, both the Triplet Loss and Siamese Network approaches are effective ways to train a neural network for face recognition, and the choice between the two depends on the specific needs and constraints of the application.

Face verification supervised learning



What is neural style transfer?

Neural Style Transfer

What is neural style transfer?

- Neural Style Transfer is a technique that allows you to generate new images by combining the content of one image with the style of another.
- It works by using a pre-trained Convolutional Neural Network (ConvNet) to extract features from the content and style images.
- The content image represents the subject matter of the new image, while the style image represents the desired style.
- The output image is then generated by iteratively adjusting the pixel values of a random noise image until it matches the content and style representations.

Neural style transfer

- To implement Neural Style Transfer, you need to first select a pre-trained ConvNet and extract features from the content and style images at various layers of the network.

- The content representation is typically extracted from a deeper layer of the ConvNet, while the style representation is extracted from a shallow layer.
- The content representation is then used to guide the overall structure and subject matter of the generated image, while the style representation is used to determine the texture and color of the image.

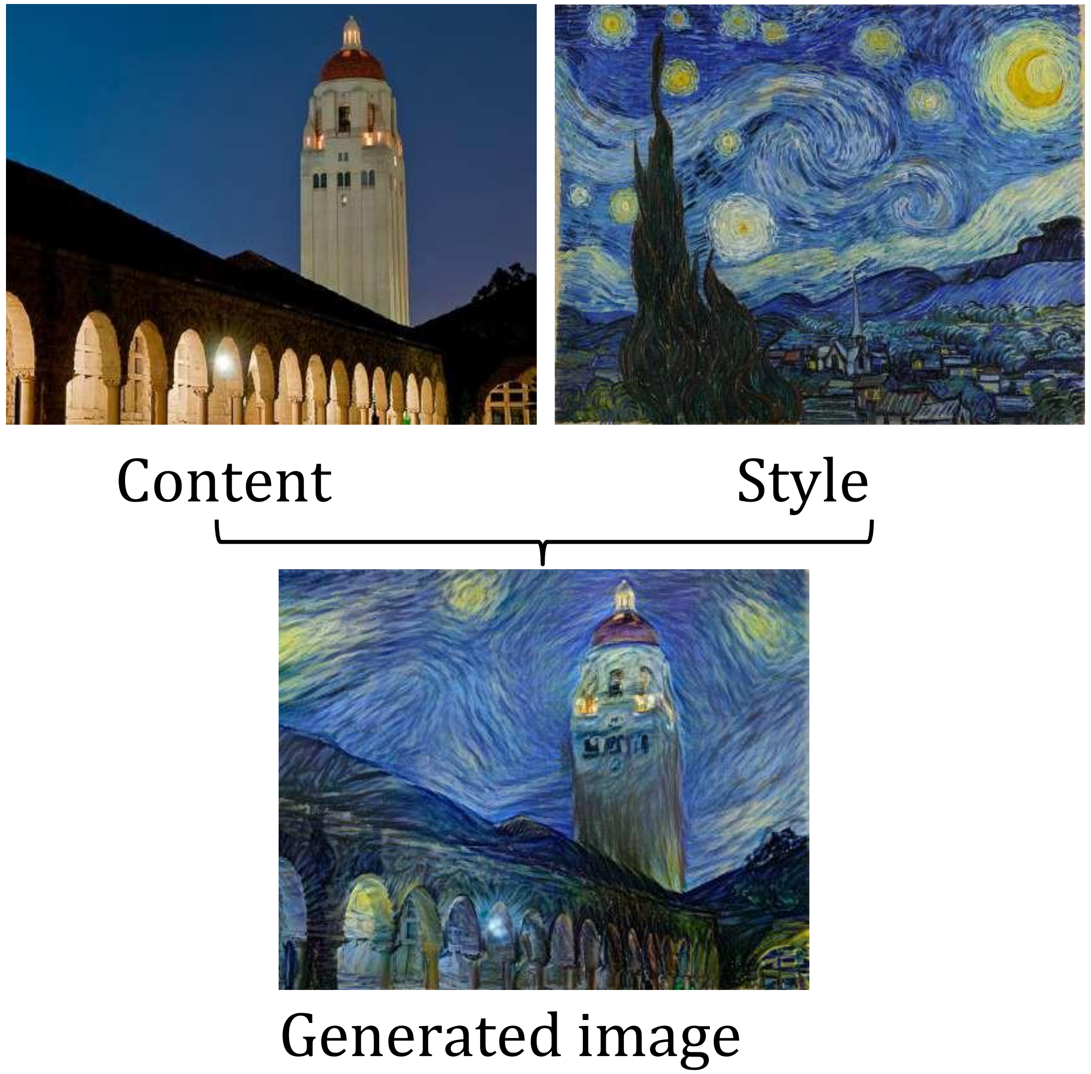
Neural style transfer

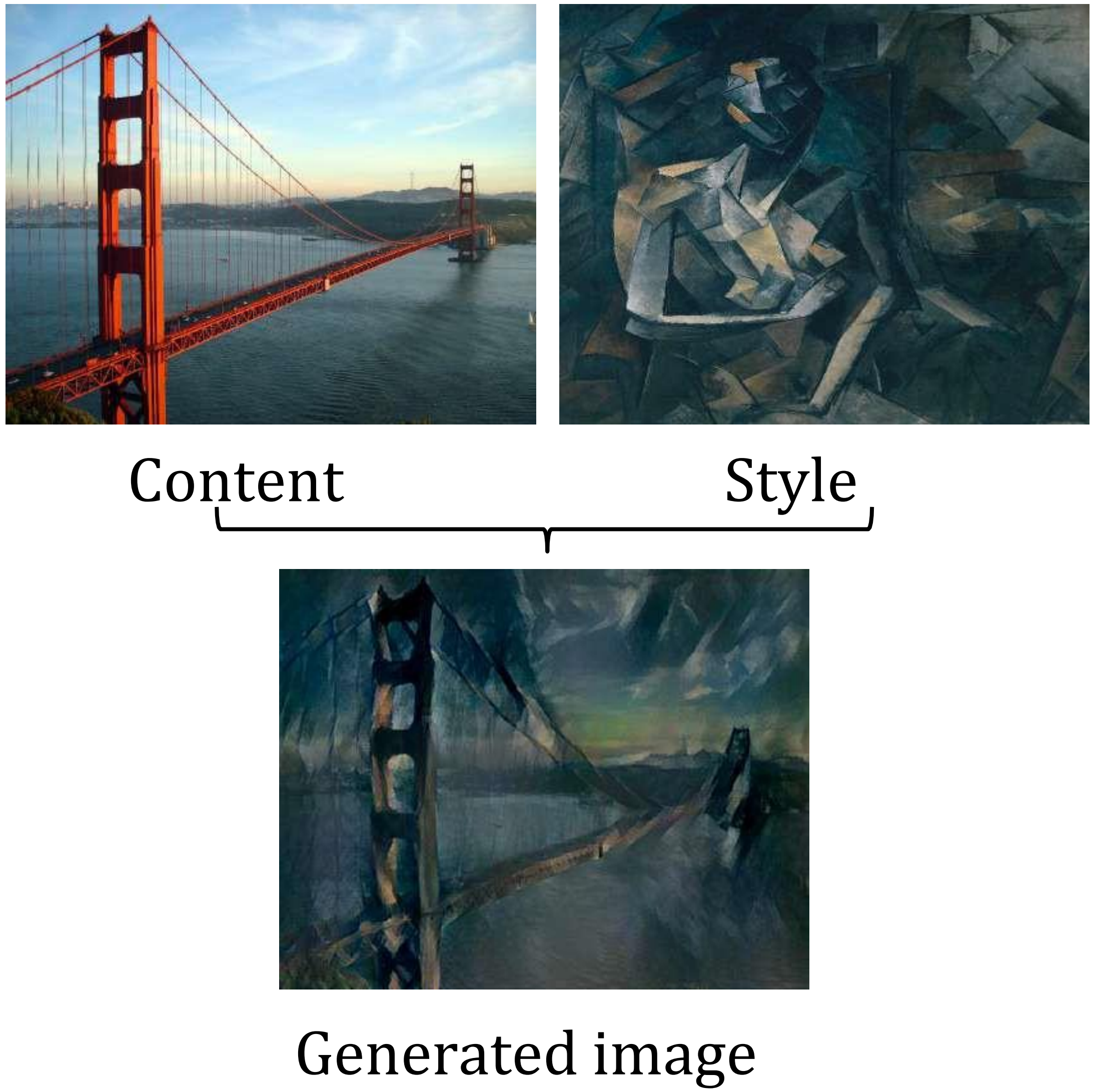
- Neural Style Transfer is a technique that involves adjusting the pixel values of a random noise image using an optimization algorithm to create an output image that combines the content of one image with the style of another.

- This process minimizes a loss function that measures the difference between the features extracted from the content and style images and those extracted from the generated image.

- The technique has become popular in recent years due to its ability to create visually appealing images, and it has been used in various fields like art, design, and advertising.

Neural style transfer





FPT Education

FPT UNIVERSITY

What are deep ConvNets learning?

Neural Style Transfer

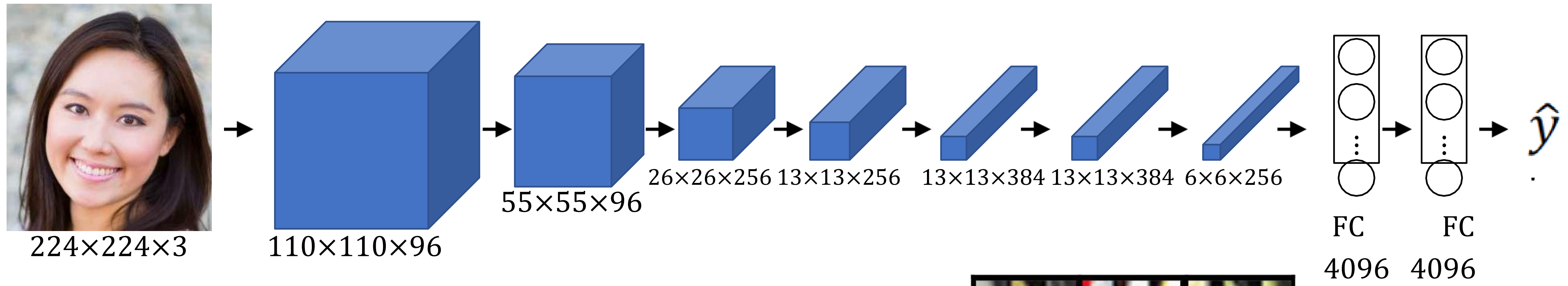
What are deep ConvNets learning?

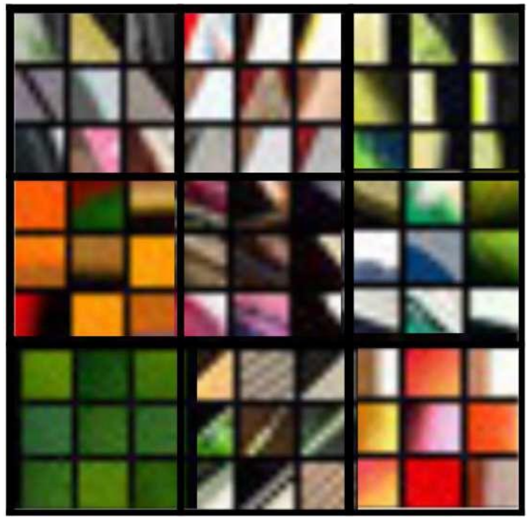
- Visualizations can help understand what deeper layers of ConvNets are computing.
- The visualization technique involves finding the images or patches in the training set that maximize a particular hidden unit's activation.
- Consider examples of visualizations of what different hidden units in various layers of a ConvNet are detecting, starting with layer 1 and moving to deeper layers.

Visualizing what a deep network is learning

- In layer 1, the units are detecting relatively simple features such as edges or a particular shade of color.
- In layer 2, the features detected are more complex shapes and patterns, such as vertical textures with lots of vertical lines or rounder shapes to the left part of the image.
- In layer 3, the features detected are even more complex, such as detecting certain textures like honeycomb or square shapes.
- In layer 4, the features detected become even more sophisticated, such as detecting dogs, water, or bird legs.
- In layer 5, the features detected become even more varied, such as keyboards, text, or flowers.
- This intuition is used to build a neural-style transfer algorithm.

Visualizing what a deep network is learning

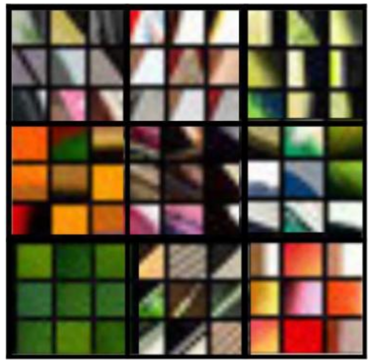




Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

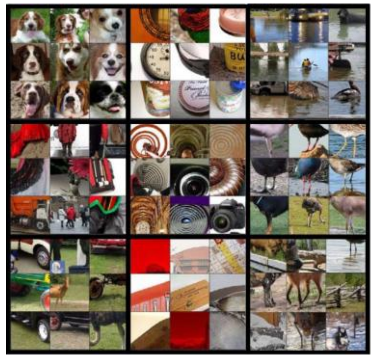
Repeat for other units.

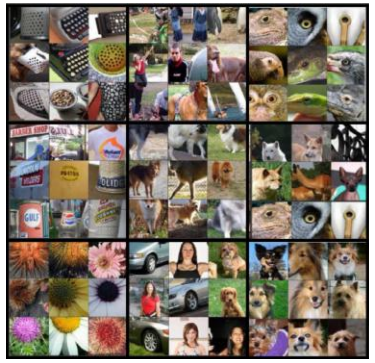
Visualizing deep layers


Layer 1

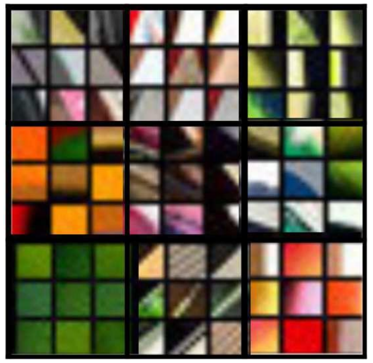
Layer 2

Layer 3


Layer 4

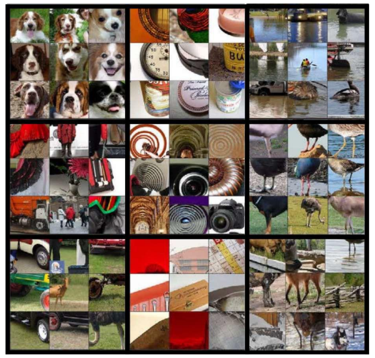

Layer 5

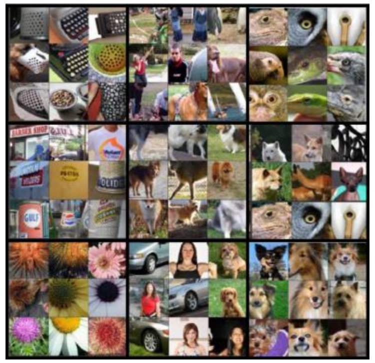
Visualizing deep layers: Layer 1

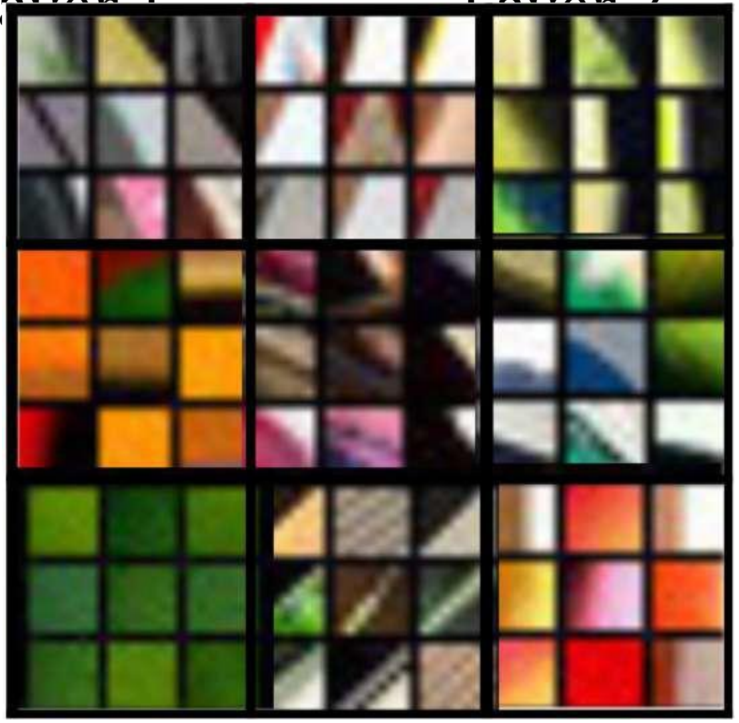

Layer 1

Layer 2

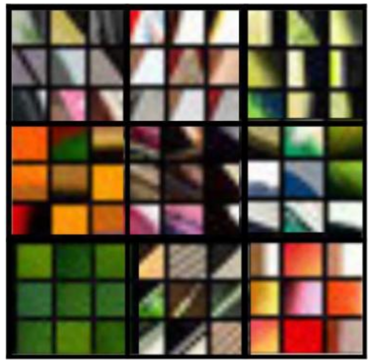
Layer 3


Layer 4


Layer 5

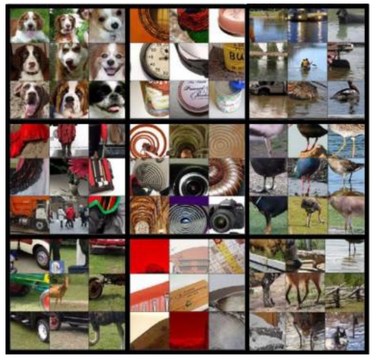


Visualizing deep layers: Layer 2

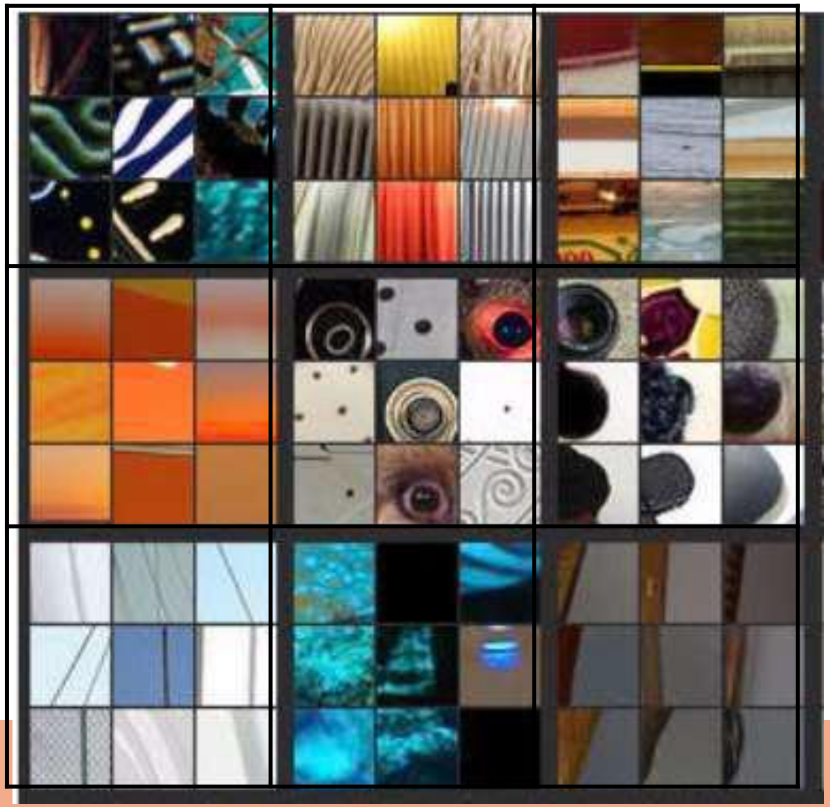

Layer 1

Layer 2

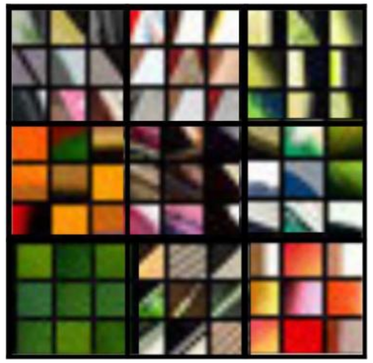
Layer 3


Layer 4

Layer 5

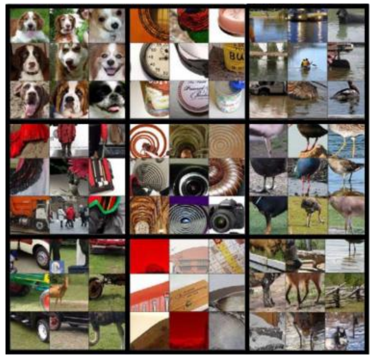


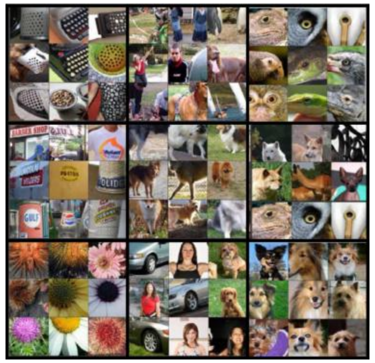
Visualizing deep layers: Layer 3

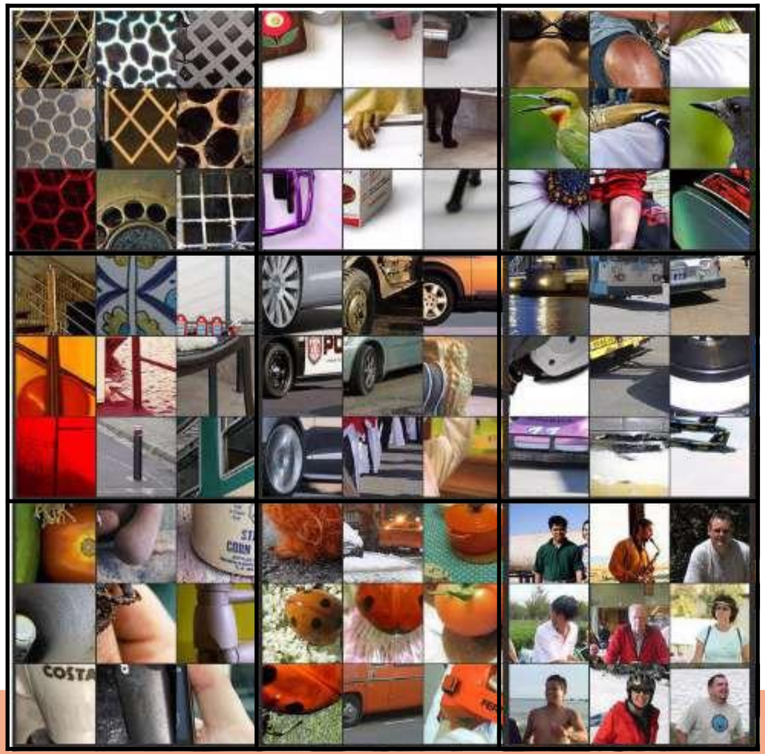

Layer 1

Layer 2

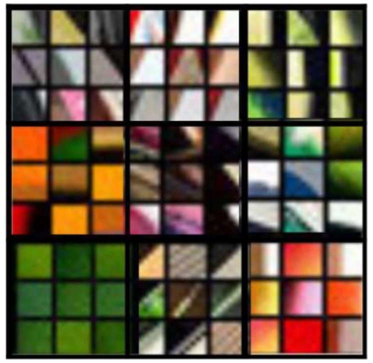
Layer 3

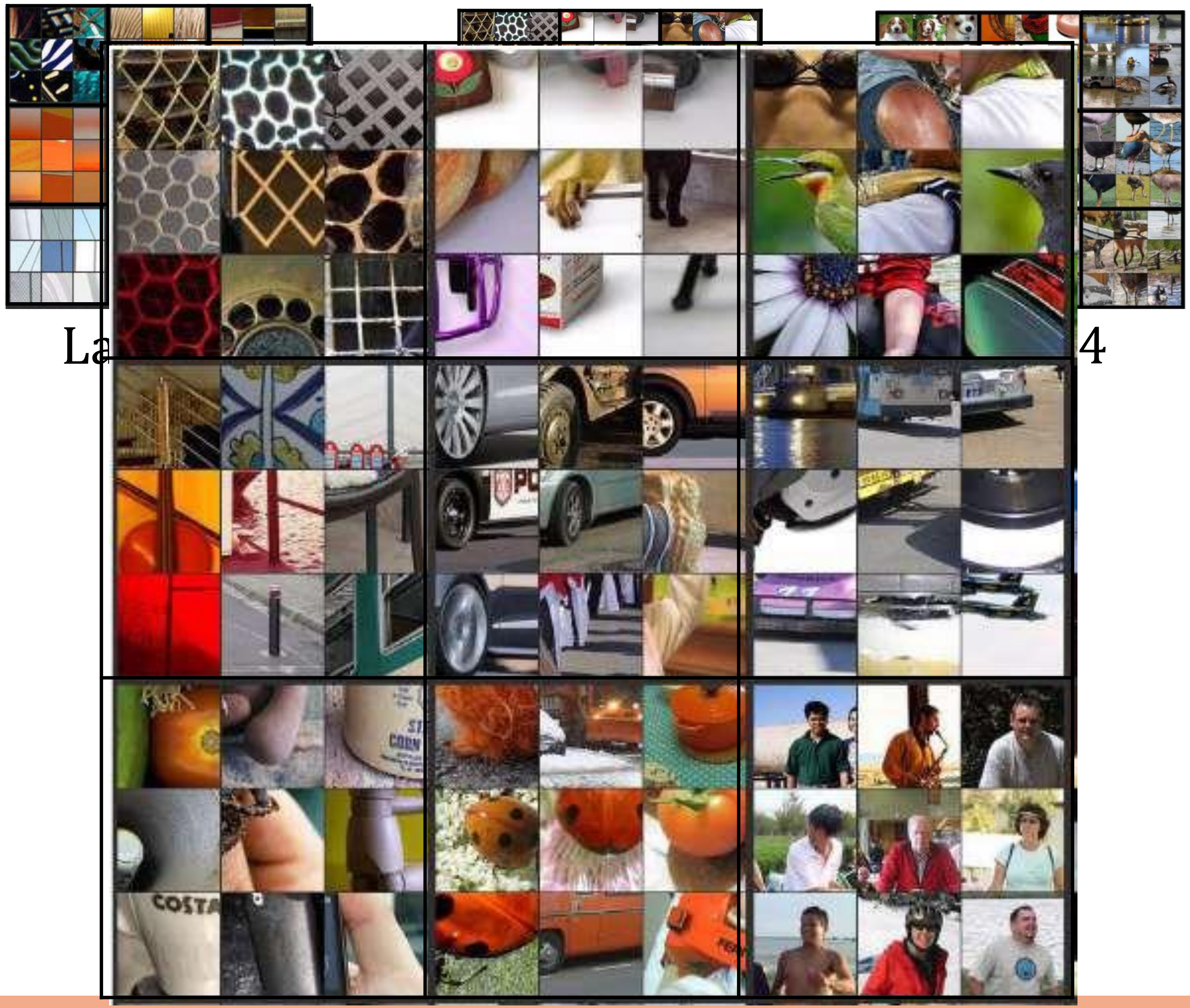

Layer 4


Layer 5



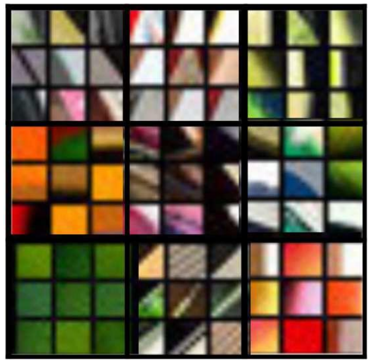
Visualizing deep layers: Layer 3

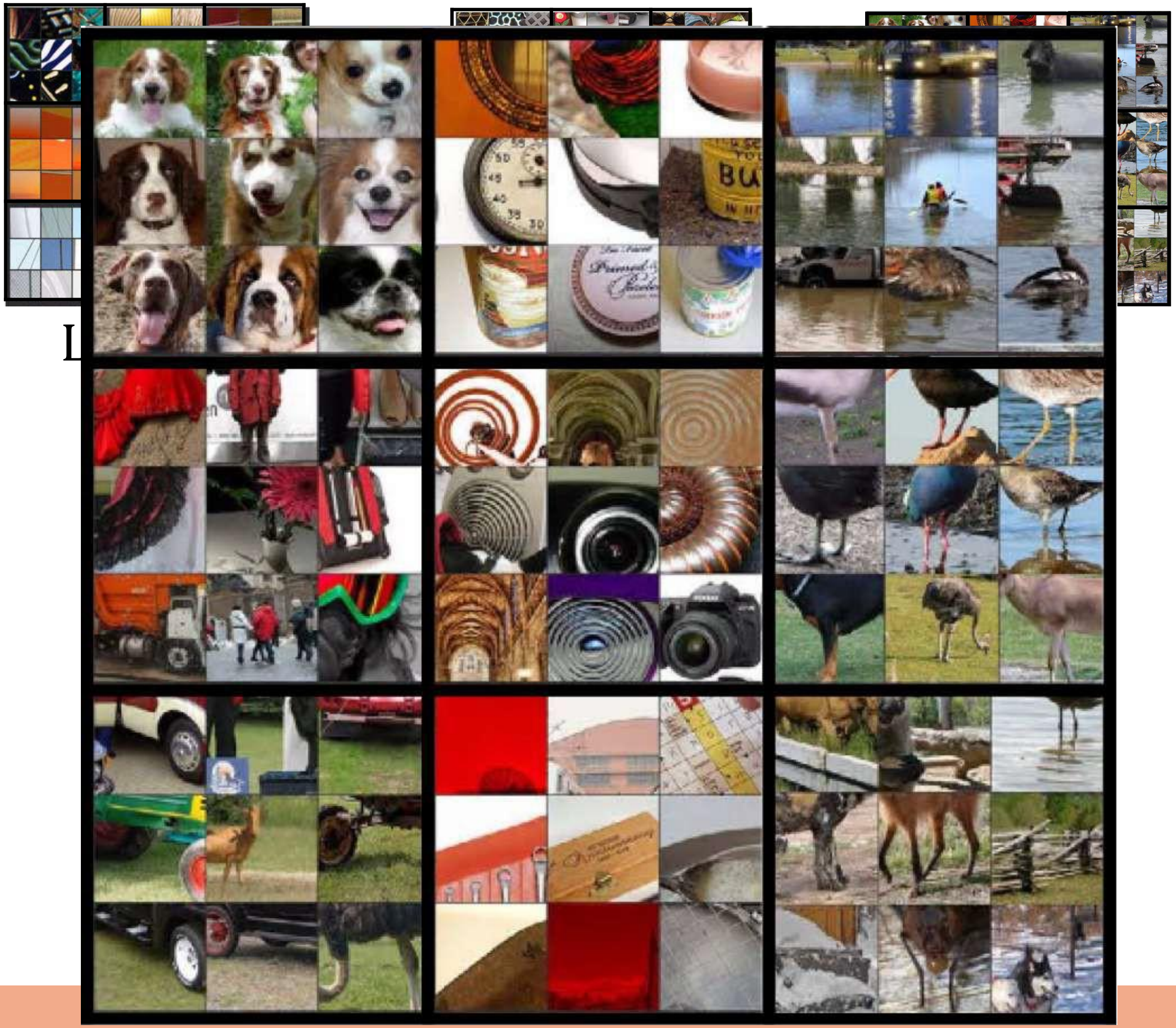

Layer 1



Layer 5

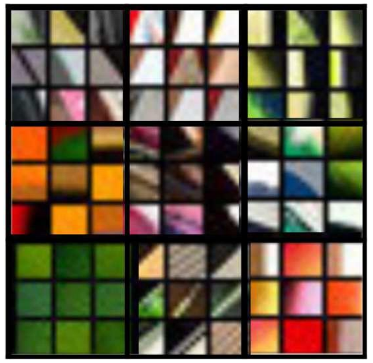
Visualizing deep layers: Layer 4

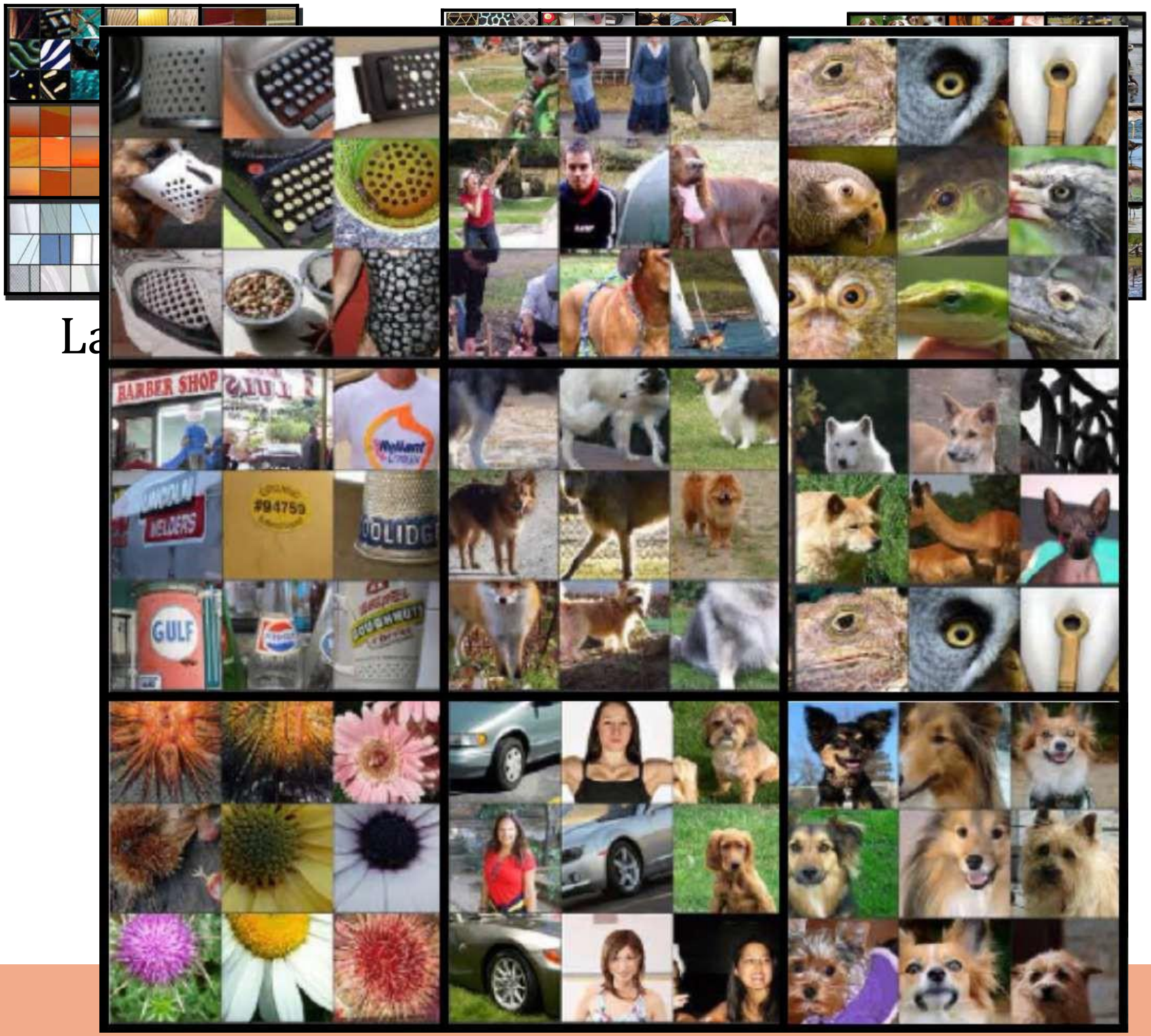

Layer 1

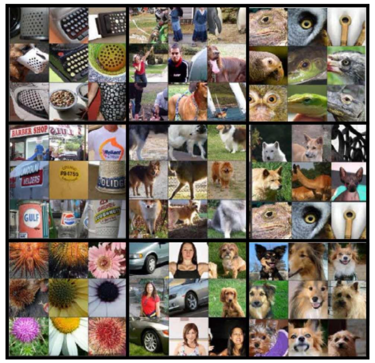

L

Layer 5

Visualizing deep layers: Layer 5


Layer 1


La

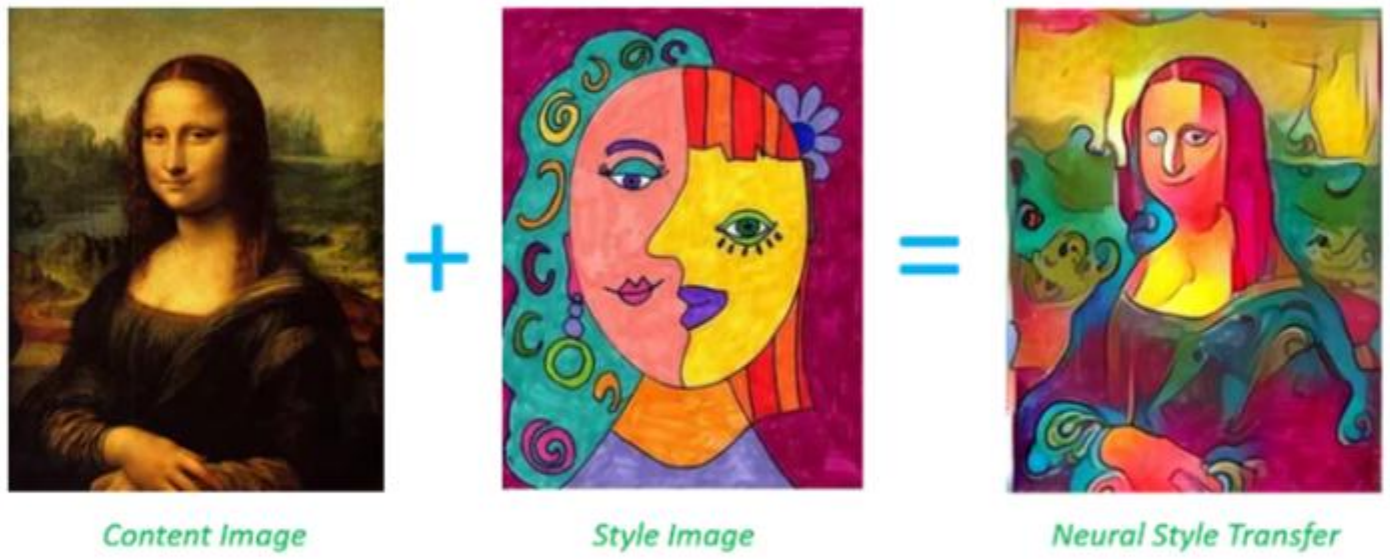

Layer 5

Cost function

Neural Style Transfer

Cost function

- Neural style transfer is one of the application of Conv nets.
- Neural Style Transfer is the process of using a convolutional neural network to transfer the style of one image (style image S) to another image (content image C) and generate a new image G.



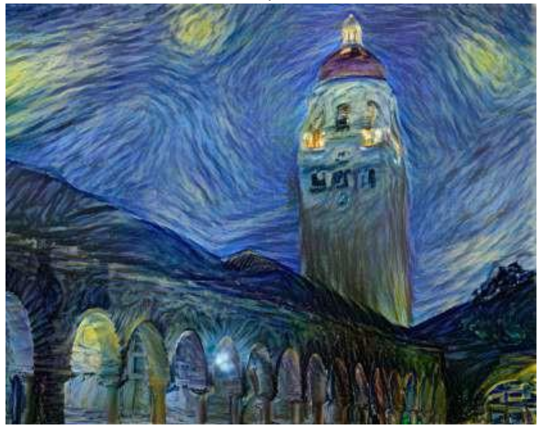
Cost function

- To implement this technique, we need to define a cost function that measures how good a generated image is. This cost function will be minimized using gradient descent to generate the desired image.
- The cost function consists of two parts: the content cost and the style cost.
  - The content cost measures how similar the content of the generated image is to the content of the content image.
  - The style cost measures how similar the style of the generated image is to the style of the style image.
- We weight these costs using hyperparameters alpha and beta to control the relative importance of each.

Cost function

Content C

Style S



Generated image G

$$\underbrace{J_{total}(G)}_{\text{total cost function}} = \alpha \times \underbrace{J_{content}(C,G)}_{\text{similarity of content image to generated image}} + \beta \times \underbrace{J_{style}(S,G)}_{\text{similarity of style image to generated image}}$$

Here α and β are hyperparameters to specify the relative weighting between the content cost and the style cost.

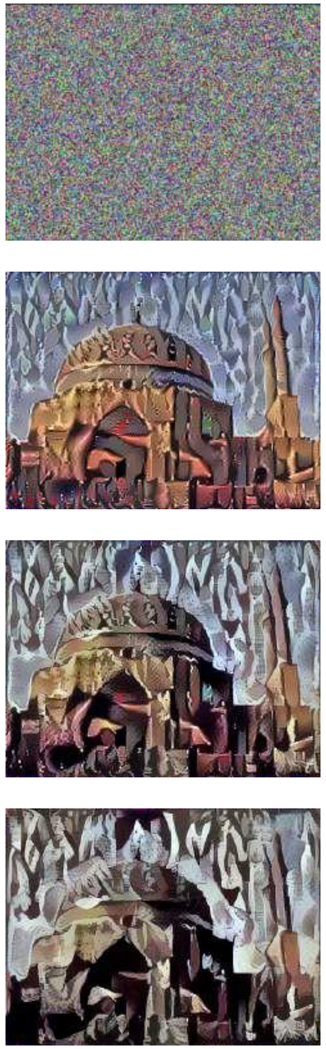
Find the generated image G

- The Neural Style Transfer Algorithm was originally proposed by Leon Gatys, Alexander Ecker, and Matthias Bethge. Their paper is not too difficult to read, so it is recommended to check it out for more details on the algorithm.
- To implement the algorithm, we start by randomly initializing the generated image. Then we minimize the cost function using gradient descent to update the pixel values of the generated image until we obtain an image that combines the content of the content image with the style of the style image.

Find the generated image G

1. Initialize the generated image G randomly say 100*100*3 or 500*500 *3 or whatever dimension we want it to be.
2. Use gradient descent to minimize cost function defined above to minimize *J(G):*

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



FPT Education

FPT UNIVERSITY

Content cost function

Neural Style Transfer

Content cost function

- The content cost function in the neural style transfer algorithm measures the similarity between the content of the content image and the generated image.
- It is defined by using a pre-trained ConvNet, such as a VGG network, and choosing a hidden layer "l" somewhere in the middle of the layers of the neural network.

Content cost function

Say we use a hidden layer l (of a pre-trained CNN) to compute the content cost. (l is usually one of the middle hidden layers; neither too shallow nor too deep).

If $a^{[l](C)}$ and $a^{[l](G)}$ denote the activations of layer l for the images C and G respectively, we conclude that the images C and G have similar content if $a^{[l](C)}$ and $a^{[l](G)}$are similar.

The content cost function is given by:

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

FPT Education

FPT UNIVERSITY

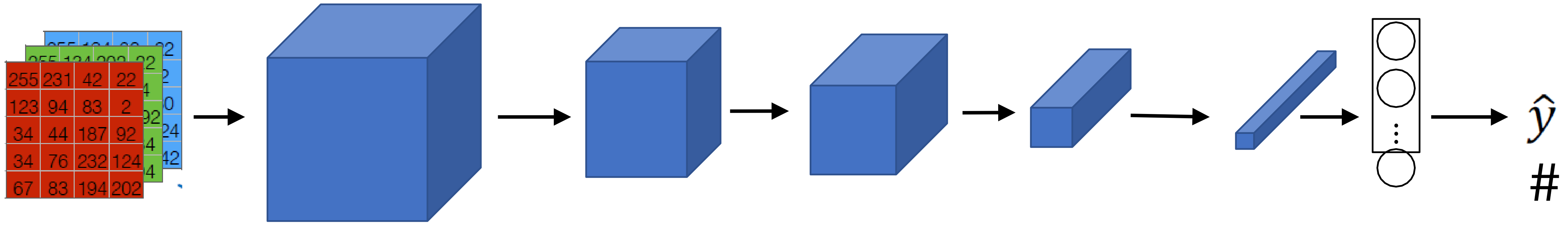
Style cost function

Neural Style Transfer

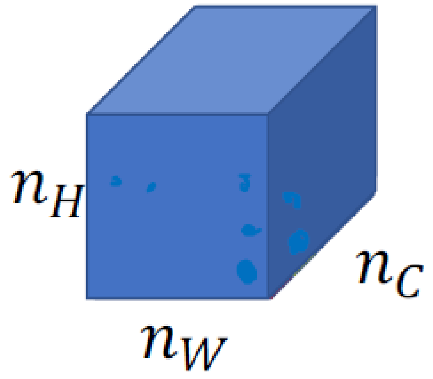
Meaning of the “style” of an image

- In the previous section, the focus was on defining the style cost function for the neural style transfer algorithm.
- The style of an image is defined as the correlation between activations across different channels in a chosen layer, which is typically computed using a pre-trained convolutional neural network such as VGG.
- The style matrix, denoted as G, is computed by taking the sum of products of activations for all pairs of channels in a given layer, and then normalizing it by the number of positions (height, width) and number of channels in that layer.
- The style cost function is defined as the Frobenius norm of the element-wise difference between the style matrix of the style image and the generated image, squared and normalized by some hyperparameters.

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.



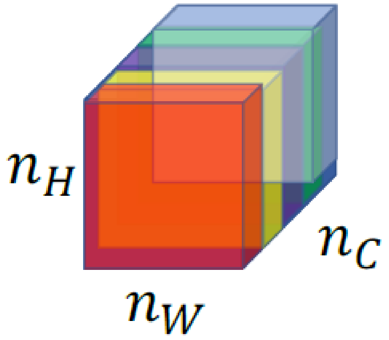
How correlated are the activations across different channels?

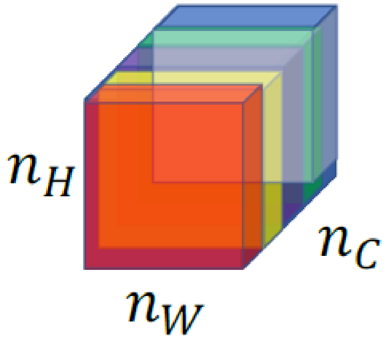
Intuition about style of an image

- Neural style transfer involves computing a style cost function using multiple layers in a neural network to measure low-level and high-level features of an image.
- The style cost function is combined with a content cost function using hyperparameters alpha and beta to define an overall cost function, and the goal is to minimize this cost function using optimization algorithms like gradient descent to generate a visually pleasing output.

Intuition about style of an image

Style image Generated Image





Style cost function

Let $a^{[l]}_{i,j,k}$ be the activation at (i, j, k). We compute a $n_c^{[L]} \times n_c^{[L]}$ style matrix $G^{[L]}$ for each image (S and G) with elements denoting the correlations of activations across channels. (k, k'=1, 2, ..., $n_c^{[l]}$).

$$G^{[l]}_{kk'} = \sum_{i=1}^{n_h^{[l]}} \sum_{j=1}^{n_w^{[l]}} a^{[l]}_{i,j,k} a^{[l]}_{i,j,k'}$$

Now, we will have two style matrices $G^{[l](S)}$and $G^{[l](G)}$. The style cost function for layer l is given by:

$$J^{[l]}_{style}(S, G) = \frac{1}{(2n_h^{[l]} n_w^{[l]} n_c^{[l]})^2} ||G^{[l](S)} - G^{[l](G)}||^2_F = \frac{1}{(2n_h^{[l]} n_w^{[l]} n_c^{[l]})^2} \sum_{k=1}^{n_c^{[l]}} \sum_{k'=1}^{n_c^{[l]}} (G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'})^2$$

The overall style cost function is as follows:

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

where $\lambda^{[l]}$ is a hyperparameter for layer l.

Style cost function

- How to perform convolutions over 1D or 3D data, which are commonly found in applications such as natural language processing and video processing.
  - Convolutions over 1D data are typically used for tasks such as text classification, where the input is a sequence of words.
  - Convolutions over 3D data are used for tasks such as video classification, where the input is a sequence of frames. The process of convolution is similar to that of 2D convolutions, but with some differences in the dimensions of the input, kernel, and output.

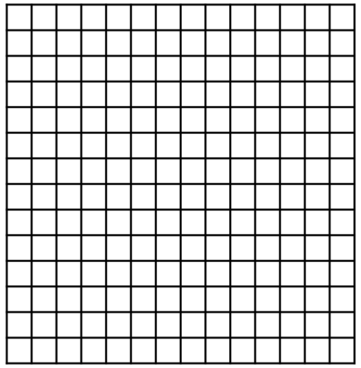
1D and 3D generalizations of model

Convolutional Networks in 1D or 3D

1D and 3D generalizations of models

- How ConvNets can be generalized from 2D data to 1D and 3D data?
  - Consider the example of an EKG signal as an example of 1D data and explain how a 1D filter can be used to detect features in different positions along the time series.
  - They also mention that recurrent neural networks are typically used for sequence data, but ConvNets can be applied to some problems as well.

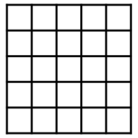
Convolutions in 2D and 1D



2D input image

14×14

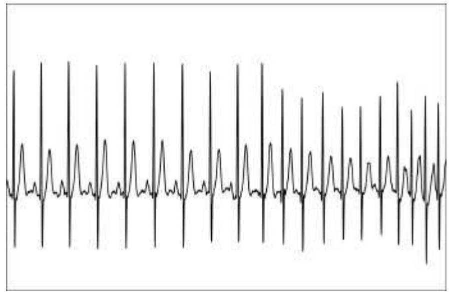
*



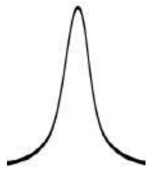
2D filter

5×5

- 14 x 14 x 3 * 5 x 5 x 3 >>> 10 x 10 x 16
- 10 x 10 x 16 * 5 x 5 x 16 >>> 6 x 6 x 32



*



- 14 x 1 * 5 x 1 >>> 10 x 16
- 10 x 16 * 5 * 16 >>> 6 * 32

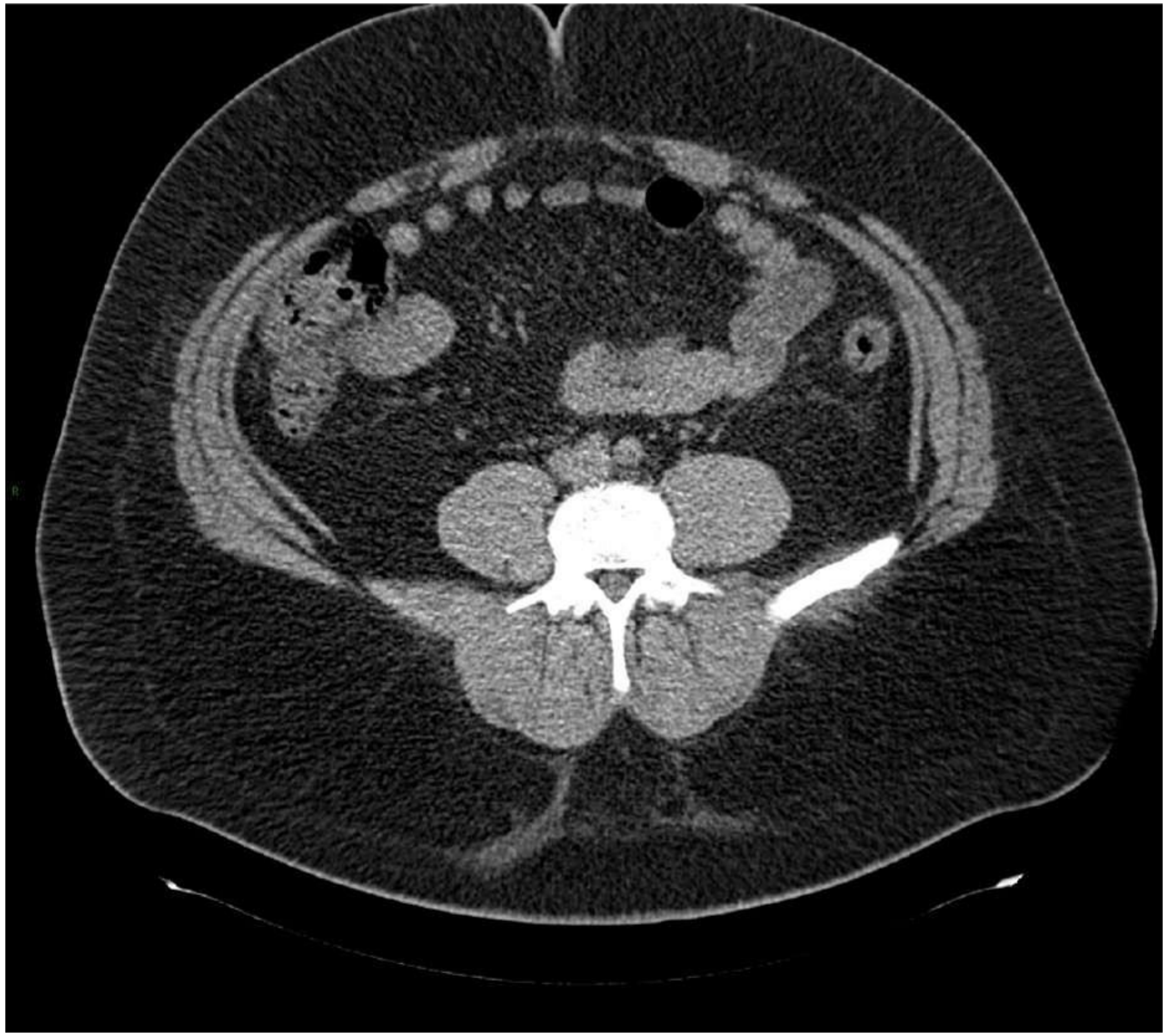
1	20	15	3	18	12	4	17

1	3	10	3	1

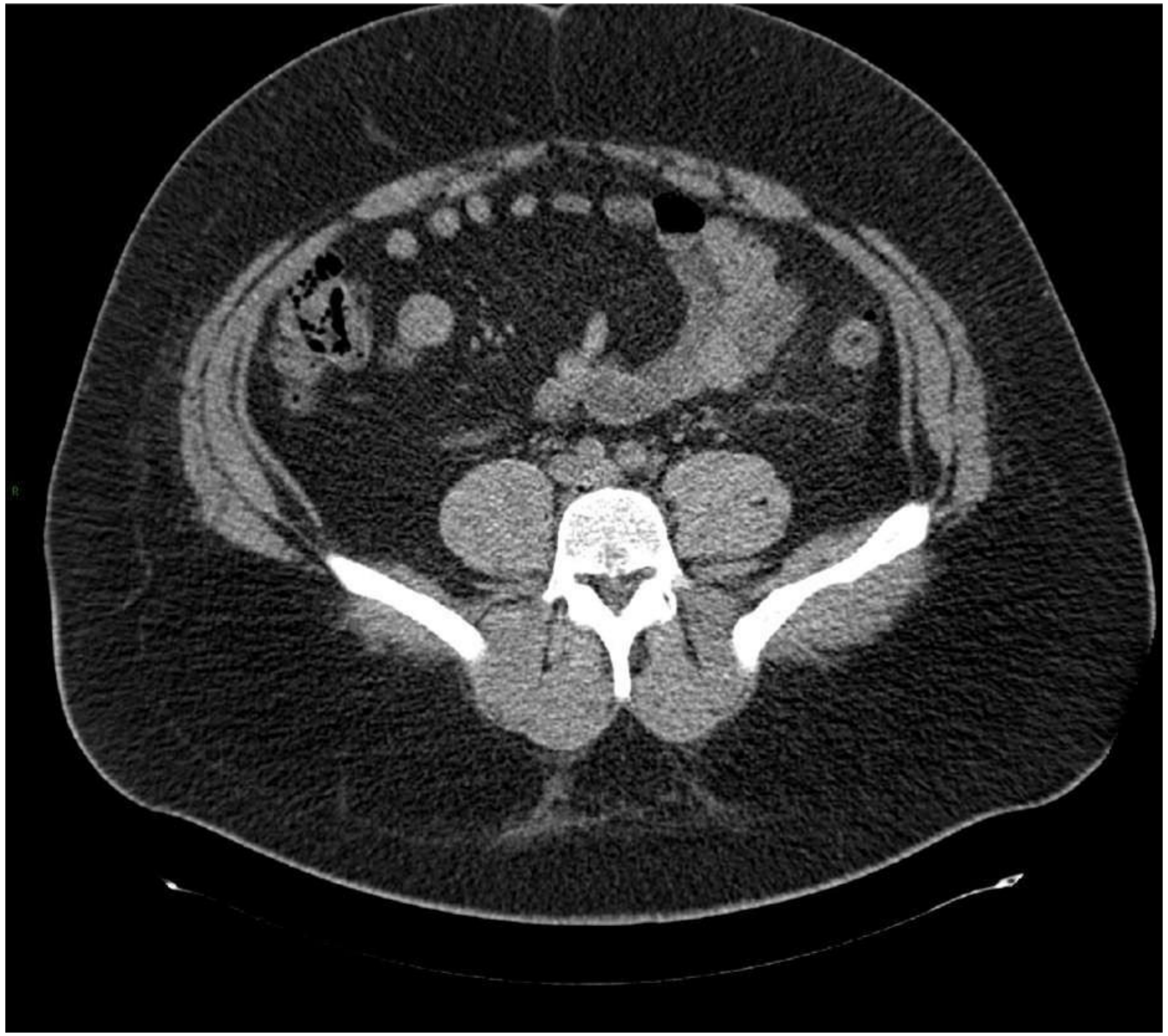
3D data

- For 3D data, consider the example of a CT scan, which is a type of medical scan that gives a 3D model of the body.
- 3D volume can be convolved with a 3D filter, and use an example of a 14 x 14 x 14 volume convolved with a 5 x 5 x 5 filter, which gives a 10 x 10 x 10 output.
- Number of channels in the input and output volumes must match, and use an example of using 16 filters to get a 10 x 10 x 10 x 16 output.

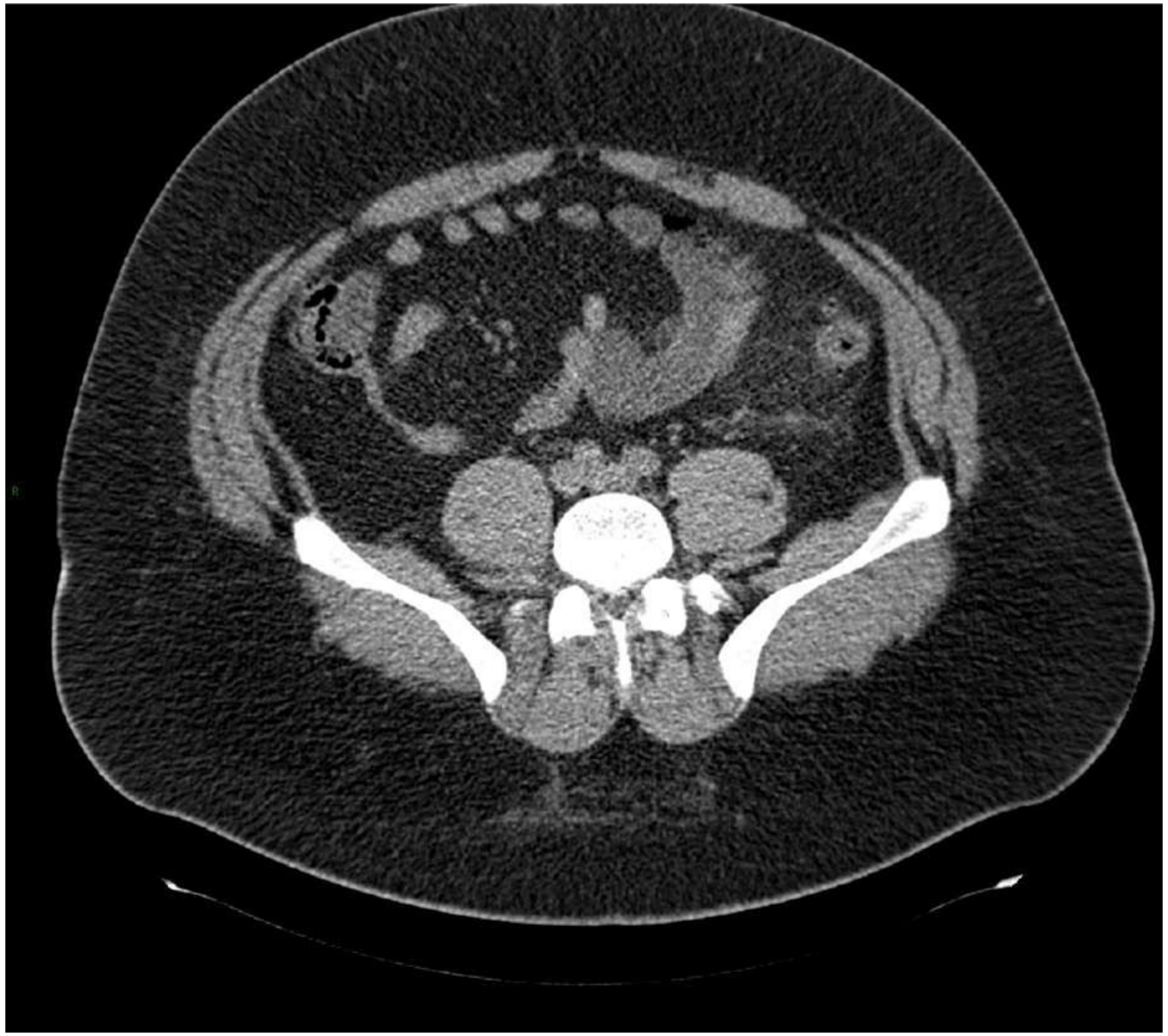
3D data



3D data

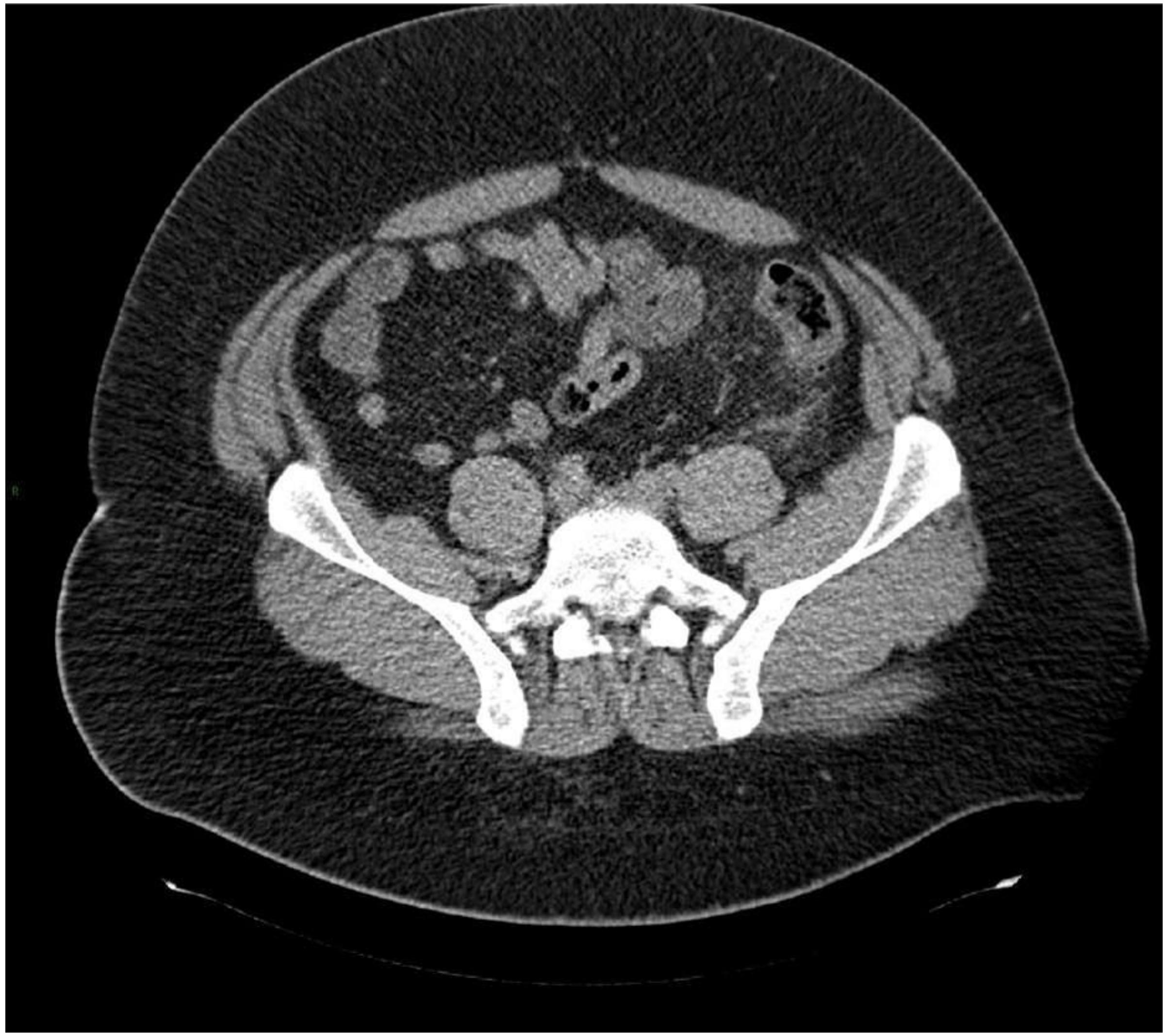


3D data

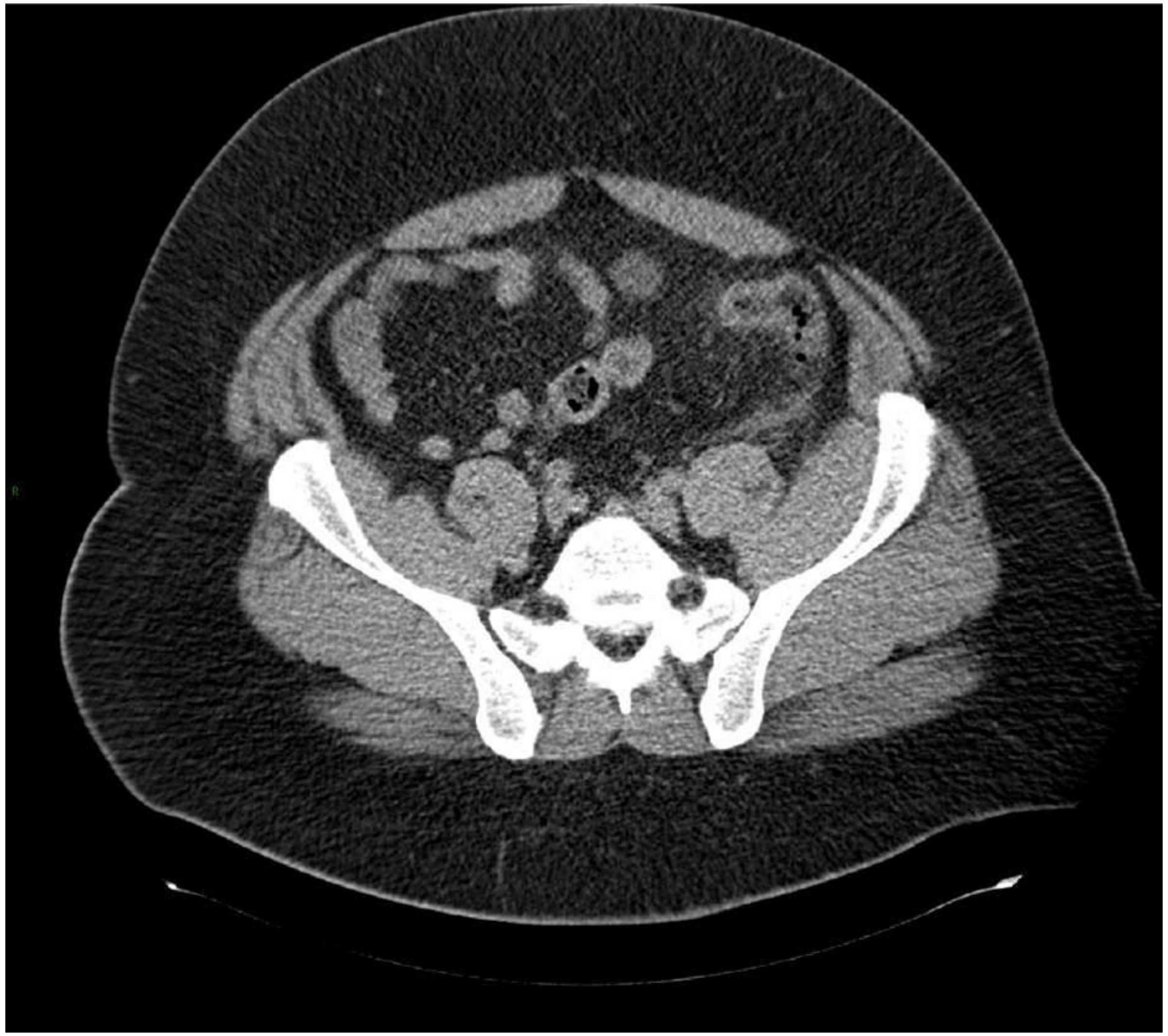


3D data

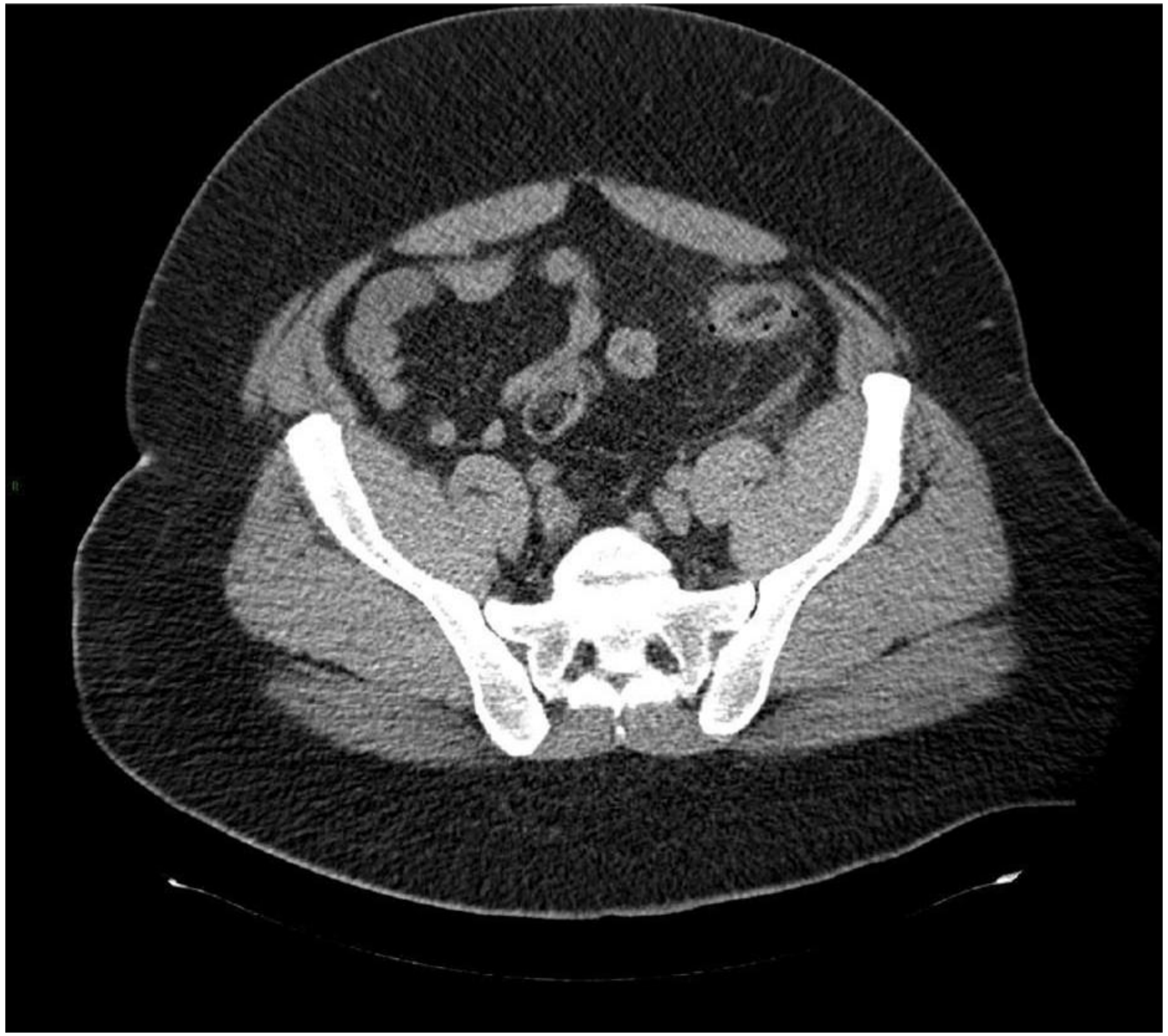
3D data



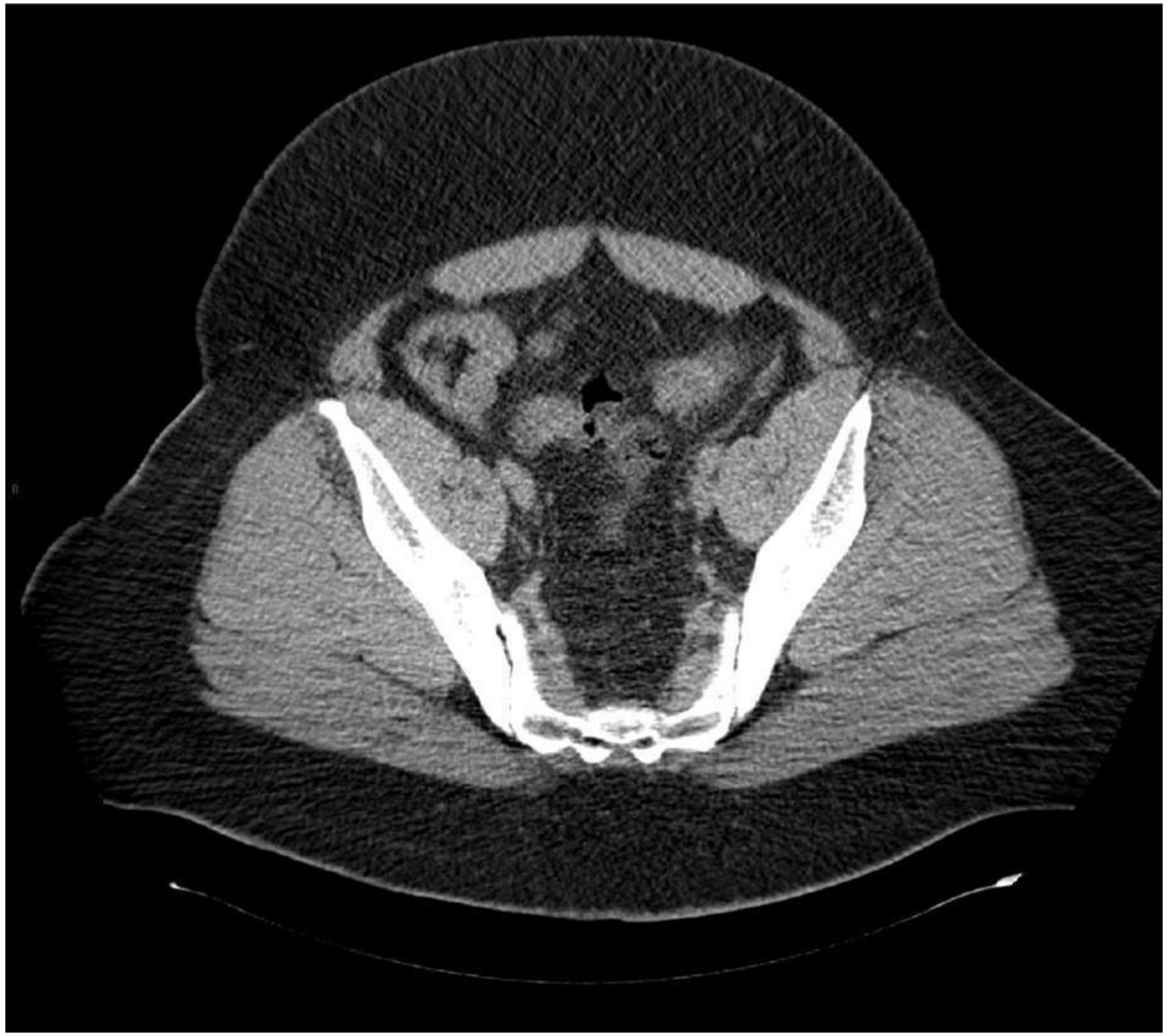
3D data



3D data



3D data

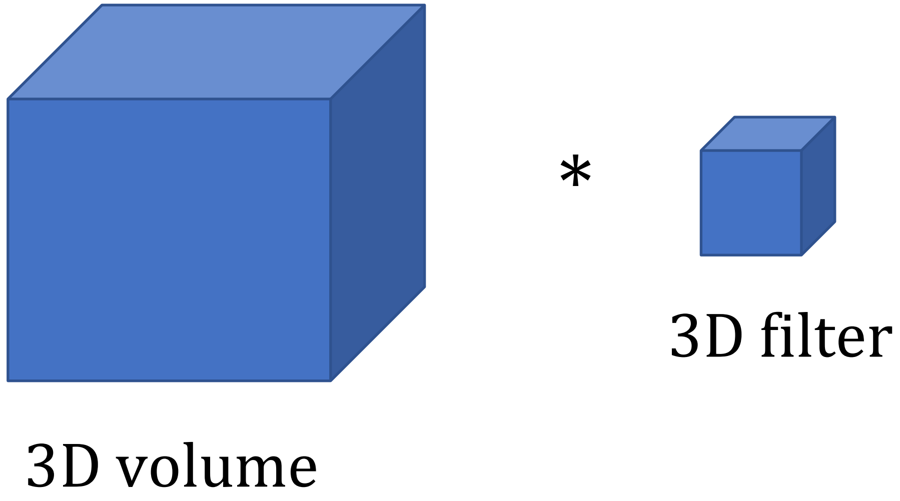


3D data

3D convolution

- Movie data can be treated as a 3D volume, where the different slices are different slices in time through the movie, and ConvNets can be used to detect motion or people taking actions in movies.

3D convolution



- 14 x 14 x 14 x 1 * 5 x 5 x 5 x 1
- >>>>>> 10 x 10 x 10 x 16 (16 filter)
- 5 x 5 x 5 x 16
- >>>>>> 6 x 6 x 6 x 32 (32 filter)

Summarization

- Face recognition identifies individuals based on facial features.
- One-shot learning classifies objects with minimal examples.
- Siamese networks learn similarity metrics, often used in face verification and one-shot learning.
- Triplet loss minimizes distances between similar pairs and maximizes distances between dissimilar pairs in Siamese networks.
- Neural style transfer merges content and style for artistic images.
- Deep ConvNets learn hierarchical features, capturing basic to abstract features.
- Cost functions measure dissimilarity during neural network training. Content cost preserves content features in style transfer. Style cost transfers artistic styles to generated images.
- Neural network architectures can generalize to 1D sequences or 3D data, enabling diverse applications.

Question

1. Does face verification compare one face, while recognition compares against K faces?
2. Why do we learn a function d(img1, img2) for face verification?
3. Is it reasonable to train on 100,000 images of 100,000 different people?
4. What is the correct definition of the triplet loss?
5. In a Siamese network, do both branches have different inputs but identical parameters?
6. Are you more likely to find a “cat detector” unit in deeper (e.g., layer 4) rather than shallow layers?
7. Is neural style transfer trained as a supervised learning task?
8. Does the style matrix G[l] measure how feature detector activations in layer l correlate?
9. In neural style transfer, what is updated in each optimization step?
10. For a 3D input volume 32×32×32×16 with 32 filters of size 3×3×3 (stride 1, no padding), what is the output volume?
11. What is the output volume for 3D input 64×64×64×3 using 16 filters of size 4×4×4, stride 2, zero padding?
12. In neural style transfer, which gradients are used?
13. What losses are used to generate the image in neural style transfer?
14. When is triplet loss large?
15. If you want to verify if a face belongs to a workgroup with changing members, what approach is best?
16. Does face recognition require K comparisons?
17. To train a triplet loss system for face verification, must data come only from current team members?
18. Do the upper and lower networks in a Siamese network share parameters?
19. In neural style transfer, do we optimize image pixels instead of network weights?
20. Does the style matrix G[l] in deep layers measure feature detector correlation?