

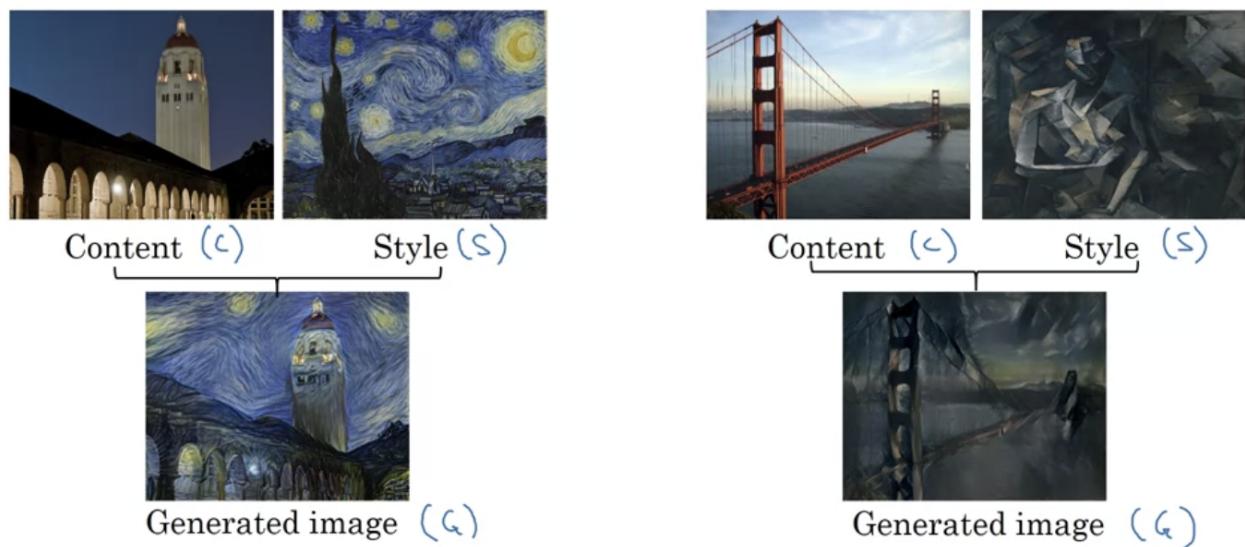
Neural Style Transfer

Alec Dewulf

What is Neural Style Transfer?

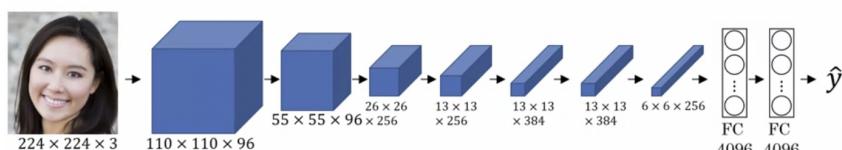
Neural style transfer allows you to generate a new version of an image in the style of another.

Neural style transfer



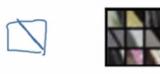
What are Deep ConvNets Learning?

Visualizing what a deep network is learning



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

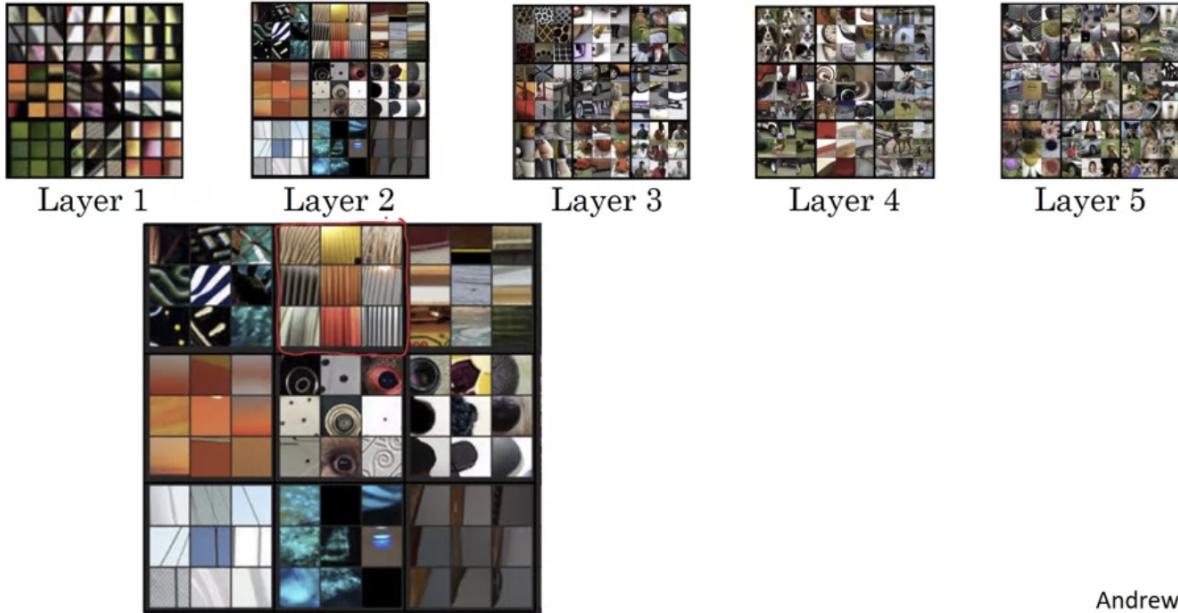
Repeat for other units.



You can check what different hidden units are doing by seeing what maximizes their activations. The earlier hidden units tend to be looking for simpler features like edges or particular colors.

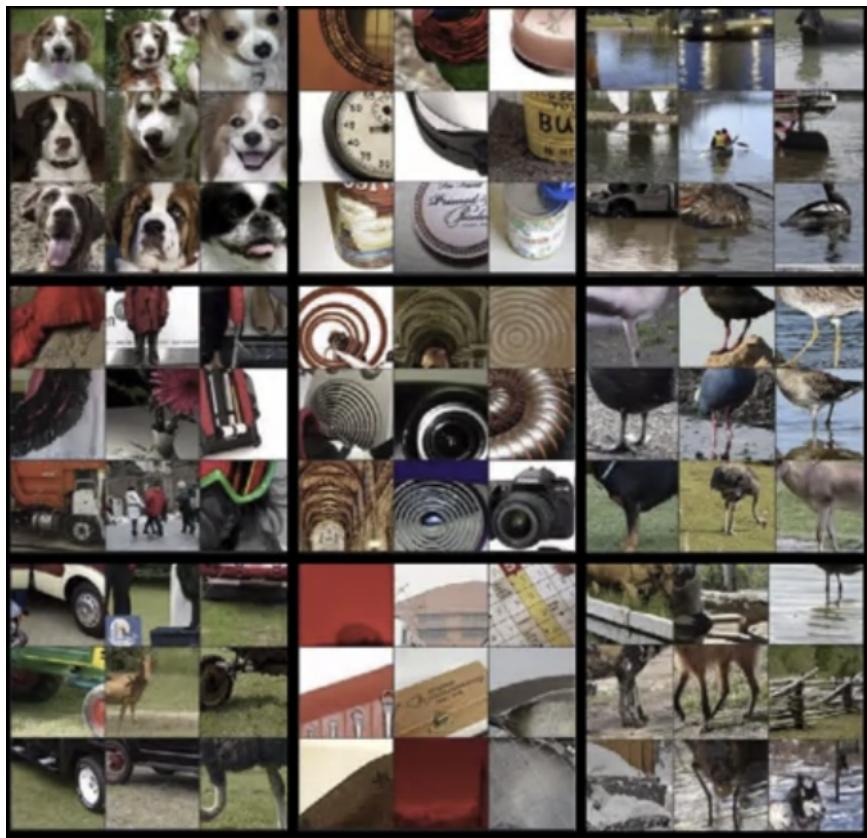
Each section is 9 image patches that cause a hidden unit to have a very high activation.

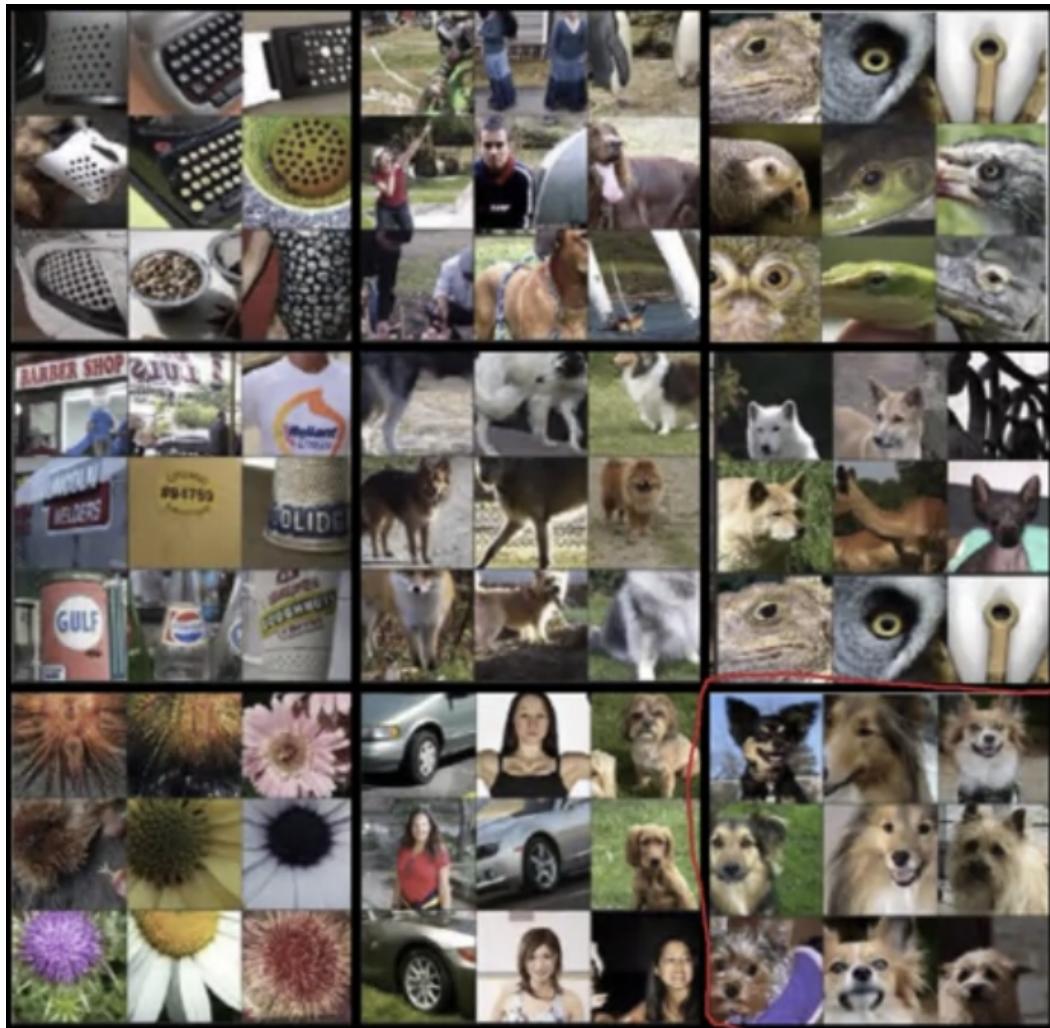
Visualizing deep layers: Layer 2



Andrew Ng



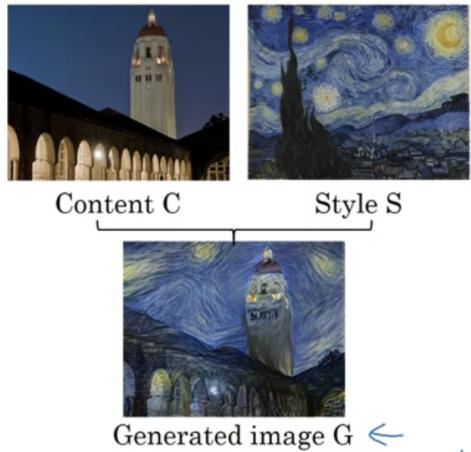




Cost Function

J_Content measures how similar the content of C is to G and J_style measures how similar the styles are of the input images S and G.

Neural style transfer cost function



$$J(G) = \alpha J_{\text{Content}}(C, G) + \beta J_{\text{Style}}(S, G)$$

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

In gradient descent you're actually updating the pixel values of the image G.

Find the generated image G

1. Initiate G randomly

G: $100 \times 100 \times 3$
↑
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]

Andrew

Content Cost Function

Content cost function

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

- 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

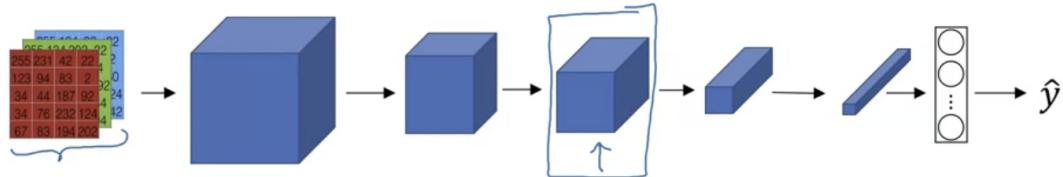
[atys et al., 2015. A neural algorithm of artistic style]

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

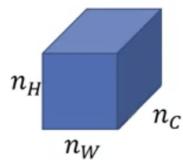
It's really just an element-wise squared difference between the similarities of the images.

Style Cost Function

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.



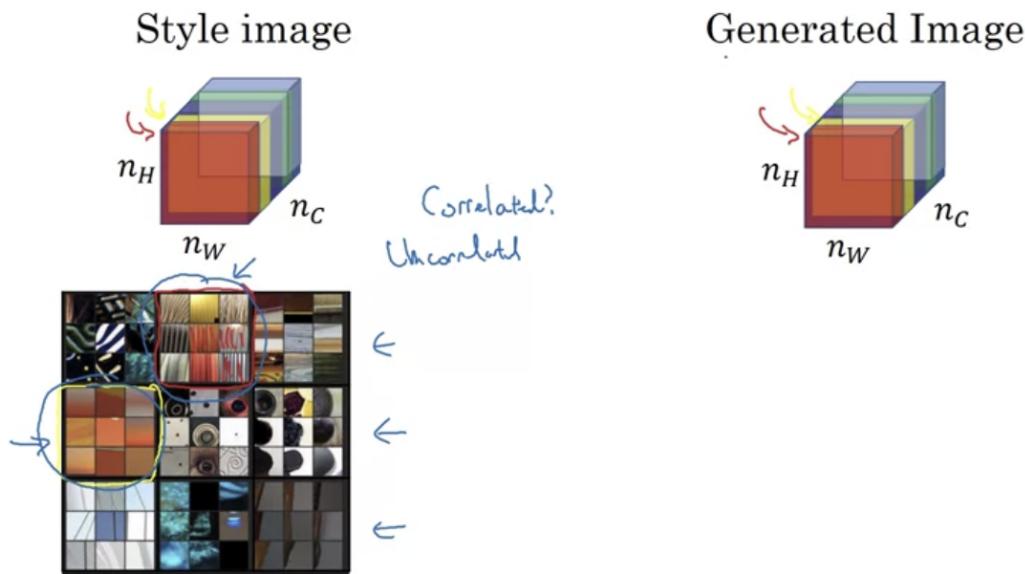
How correlated are the activations
across different channels?

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

Andrew Ng

The correlation tells you what high level style components come together. For example, looking at the activations in a layer of the style image tells you what is correlated.

Intuition about style of an image



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

And

Here the orange detecting neuron is associated with the vertical line neuron.

G is the style matrix. It is defined as a pairwise multiplication of the activations in a layer. If activations have high correlation (the styles are similar) then the elements of G will be large. You would compute G for the style image and the generated image.

The cost is just the difference between the two matrices.

Style matrix

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

$\downarrow H \downarrow W \downarrow C$

n_c
 $G_{kk'}^{[l]}$
 $k = 1, \dots, n_c$

$$\rightarrow G_{kk'}^{[l](S)} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_W} a_{ijk}^{[l](S)} a_{ijk'}^{[l](S)}$$

"Gram matrix"

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

$$J_{style}^{[l]}(S, G) = \| G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)} \|_F^2$$

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

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

Andrew Ng

The term out front is a normalization constant that doesn't really do much because the result is multiplied by the hyperparameter beta anyway.

Style cost function

$$\| G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)} \|_F^2$$

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

It turns out you get better results if you use the style cost function from different layers. Hence:

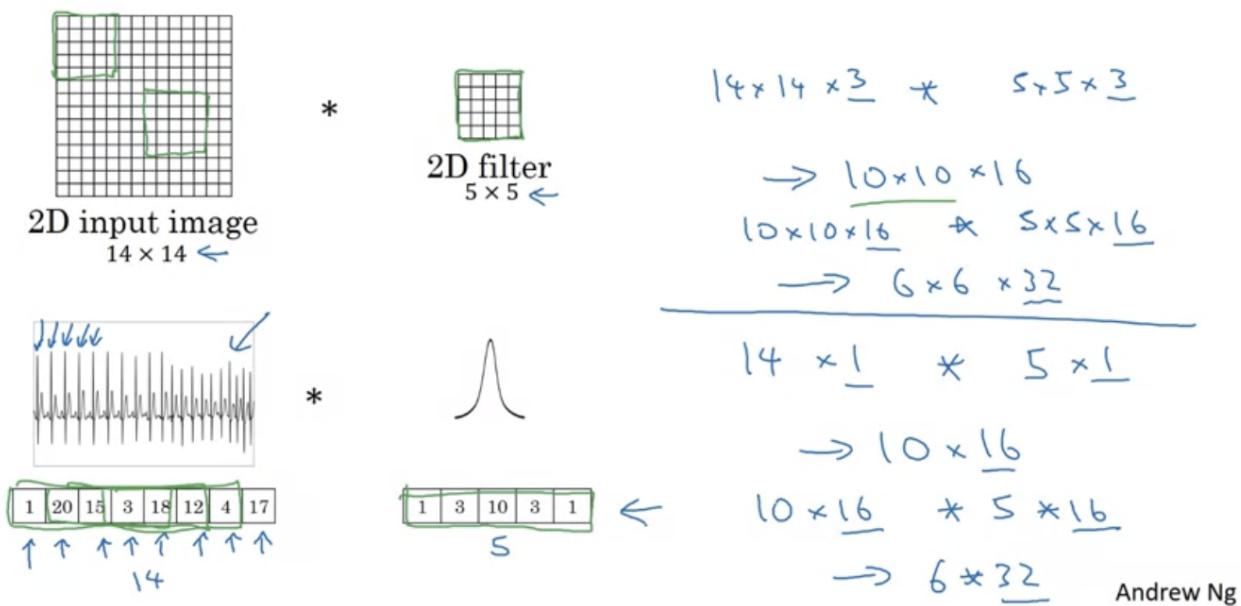
$$J_{style}(S, G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S, G)$$

Each layer is weighted by a lambda. The overall cost function is therefore:

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

1D and 3D Generalizations

You can do convolutions in 1D as well to reduce dimension.

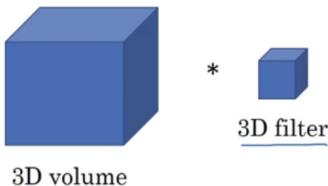


The bottom is EKG data looking at someone's heartbeat.

For a lot of 1D applications you use a recurrent neural network most of the time.

You can also convolve 3D data to produce a 4D output volume.

3D convolution



$$\begin{array}{c}
 \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
 \overbrace{14 \times 14 \times 14}^{\text{---}} \times \underbrace{1}_{n_c} \\
 * \underbrace{s \times s \times s \times 1}_{\text{16 filter.}} \\
 \rightarrow \underbrace{10 \times 10 \times 10 \times 16}_{* s \times s \times s \times 16} \\
 \end{array}$$

32 .

An example of 3D data may be movie data, where each slice would be the image at a moment in time.