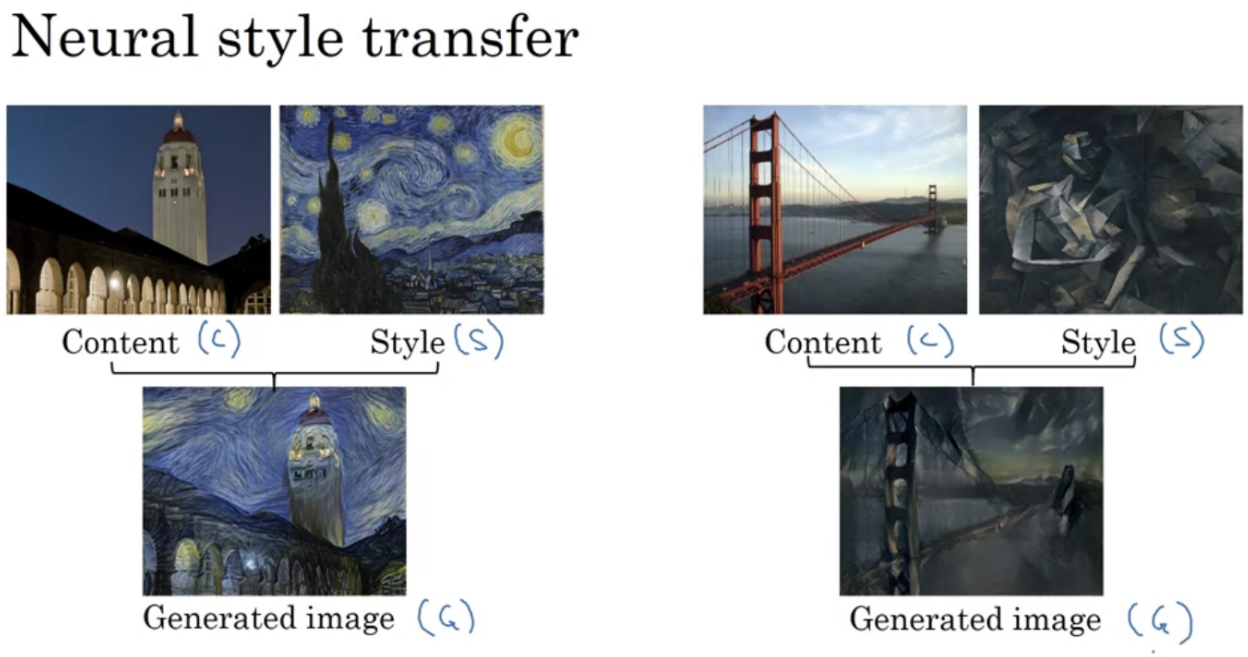
Convolutional Neural Networks DeepLearning AI

Neural Style Transfer

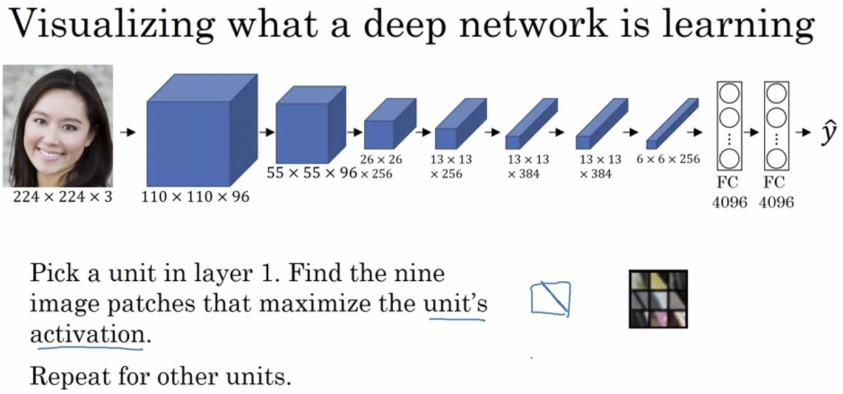
Alec Dewulf

What is Neural Style Transfer?

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

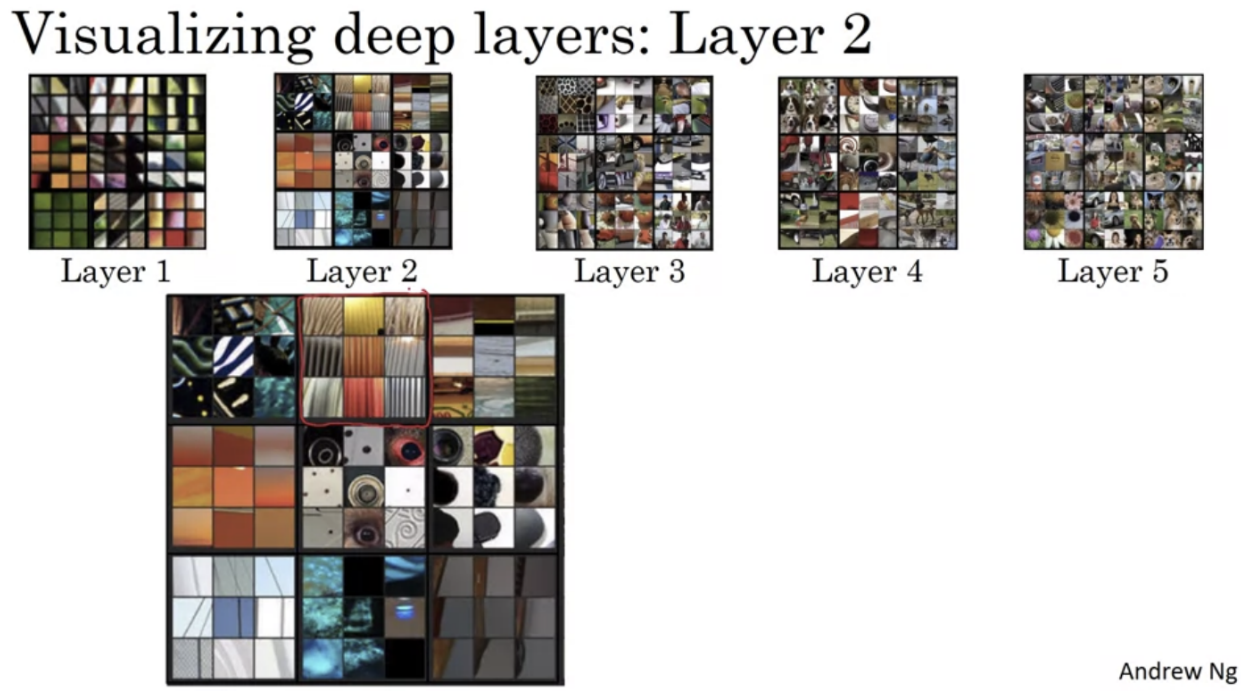


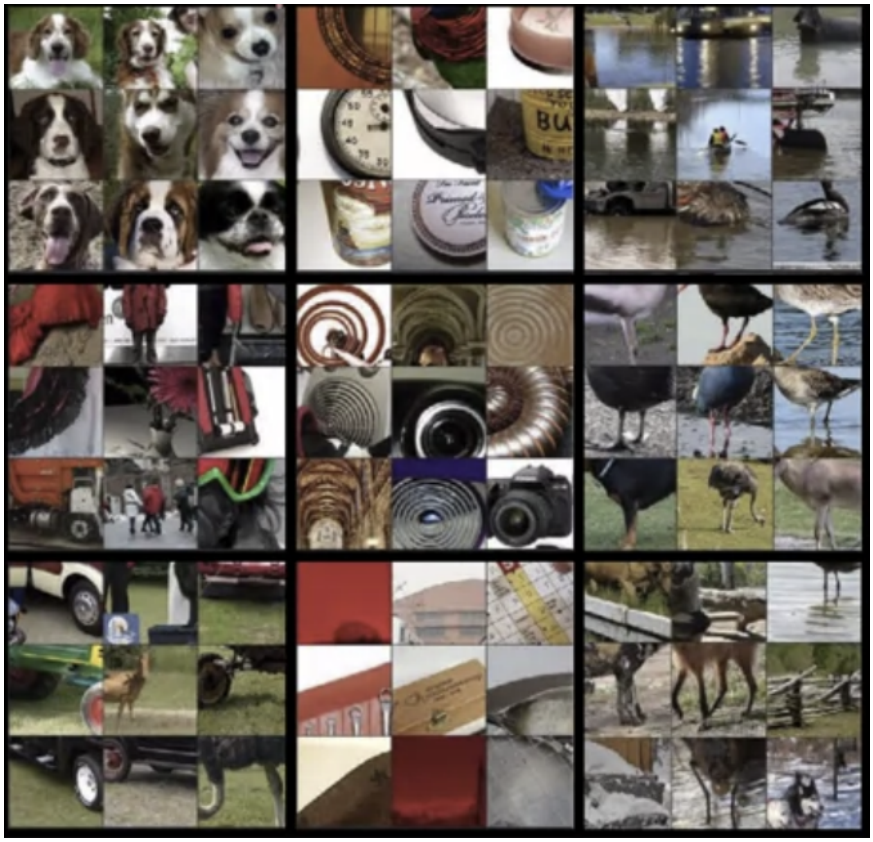
What are Deep ConvNets Learning?

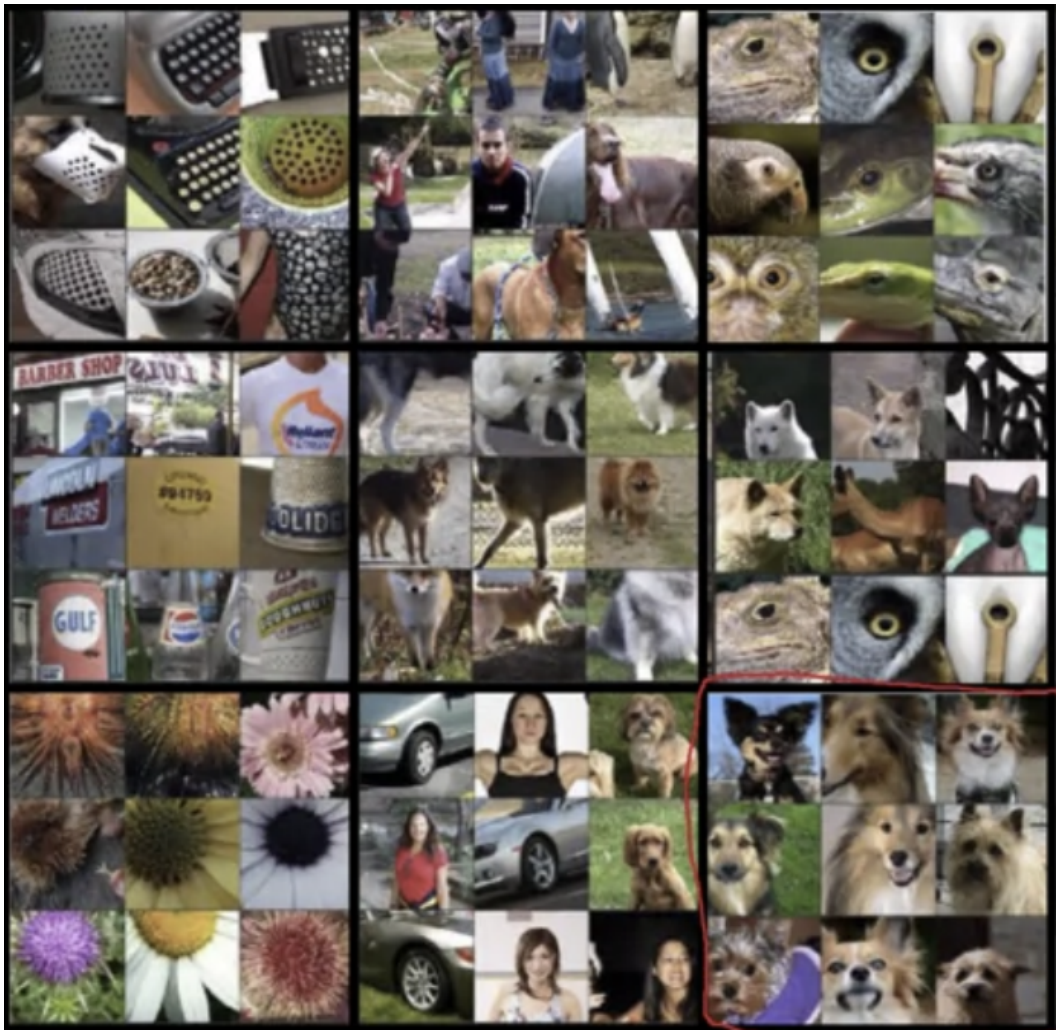


You can check what different hidden units are doing by seeing what maximizes their activations. The earlier hidden units tend to be looking for simpler features like edges or particular colors.

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







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

Content C Style S

Generated image G

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

[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

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

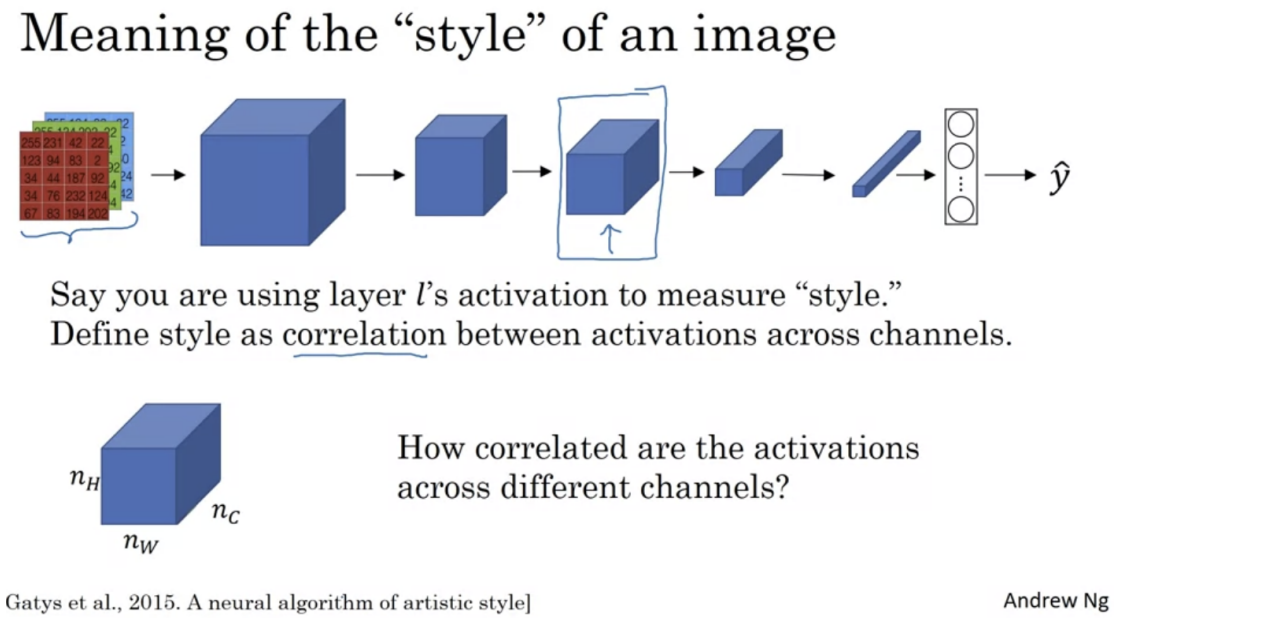
- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

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

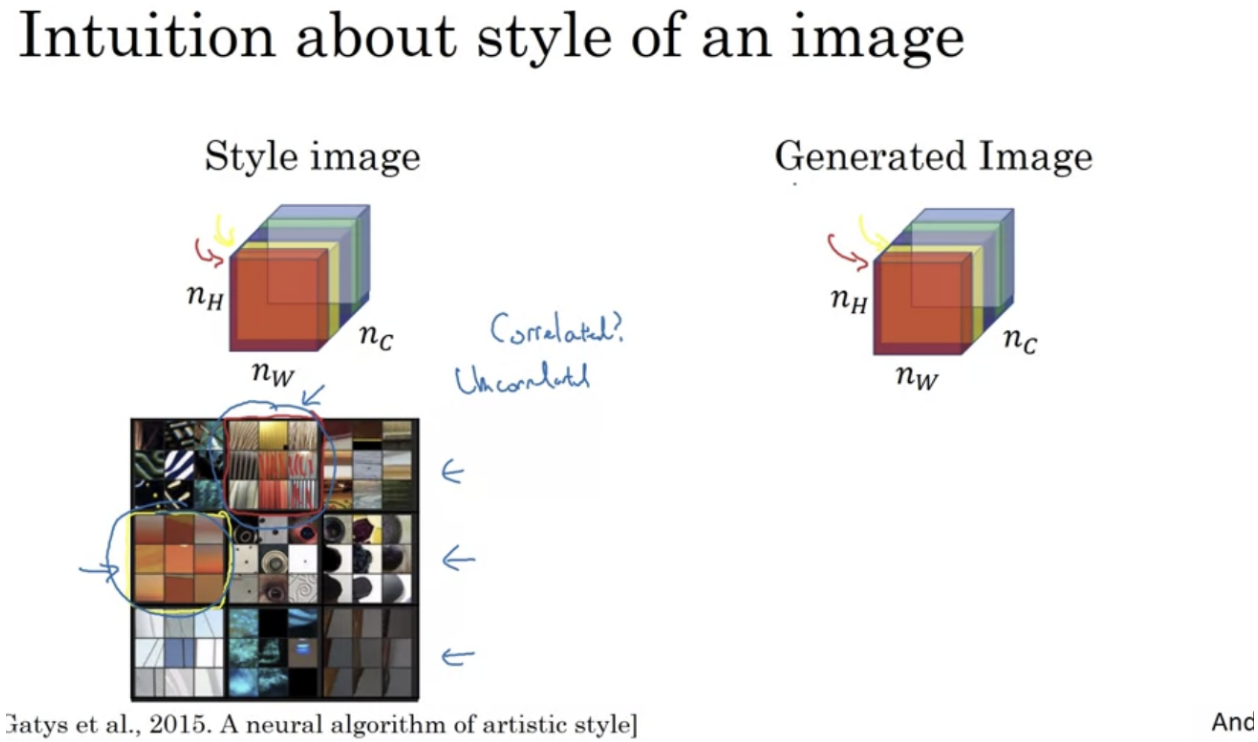
$$J_{content}(C, G) = \frac{1}{2} \| a^{[l](C)} - a^{[l](G)} \|^2$$

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

Style Cost Function



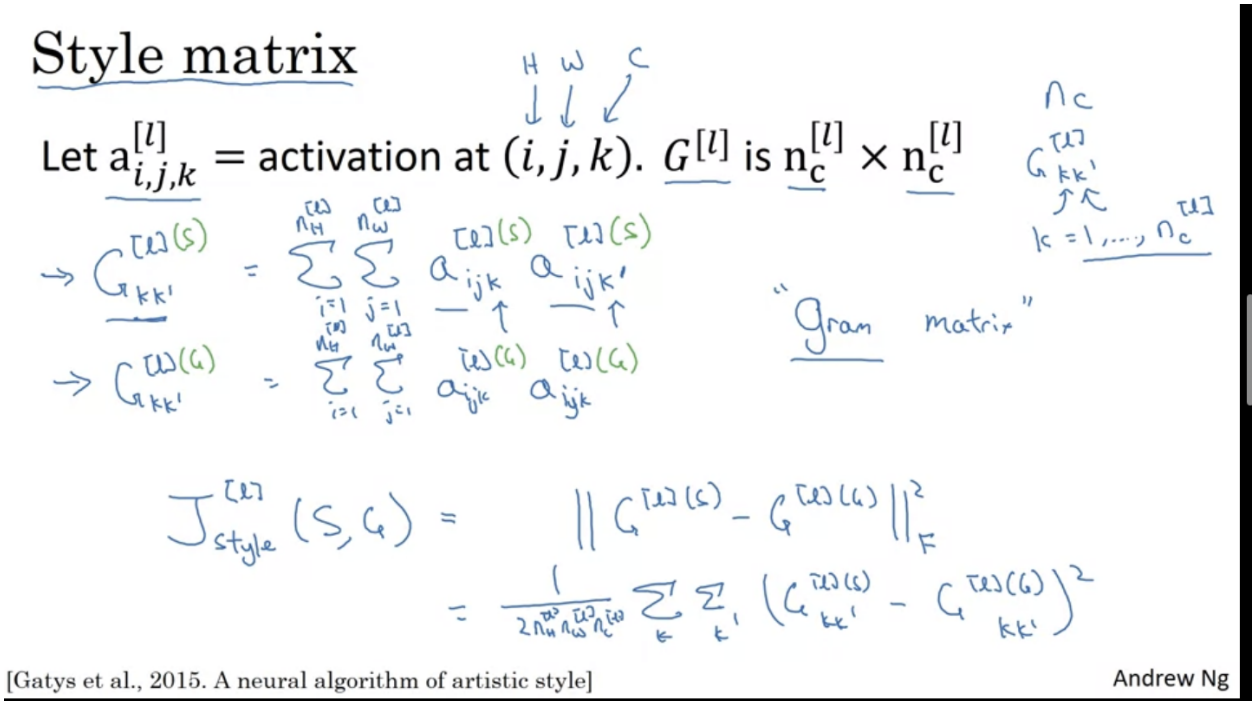
The correlation tells you what high level style components come together. For example, looking at the activations in a layer of the style image tells you what is correlated.



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.



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

Style cost function

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

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

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

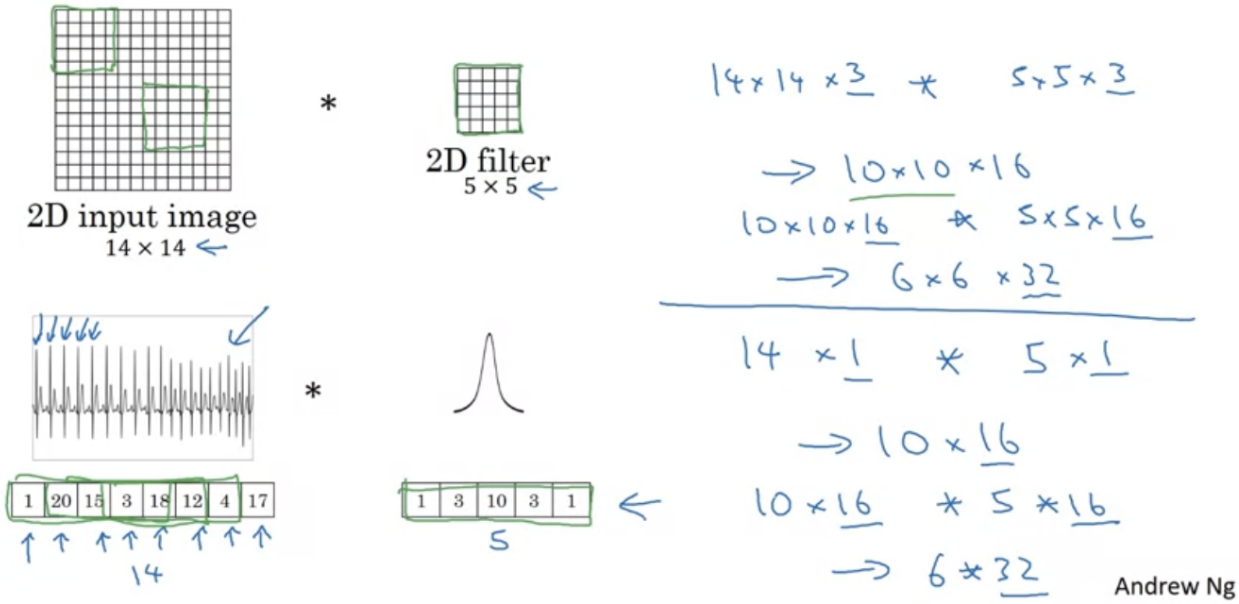
Each layer is weighted by a lambda. The overall cost function is therefore:

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

$$\min_G$$

1D and 3D Generalizations

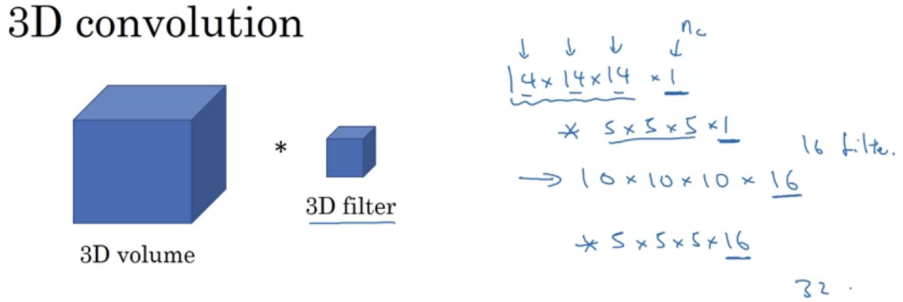
You can do convolutions in 1D as well to reduce dimension.



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.



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