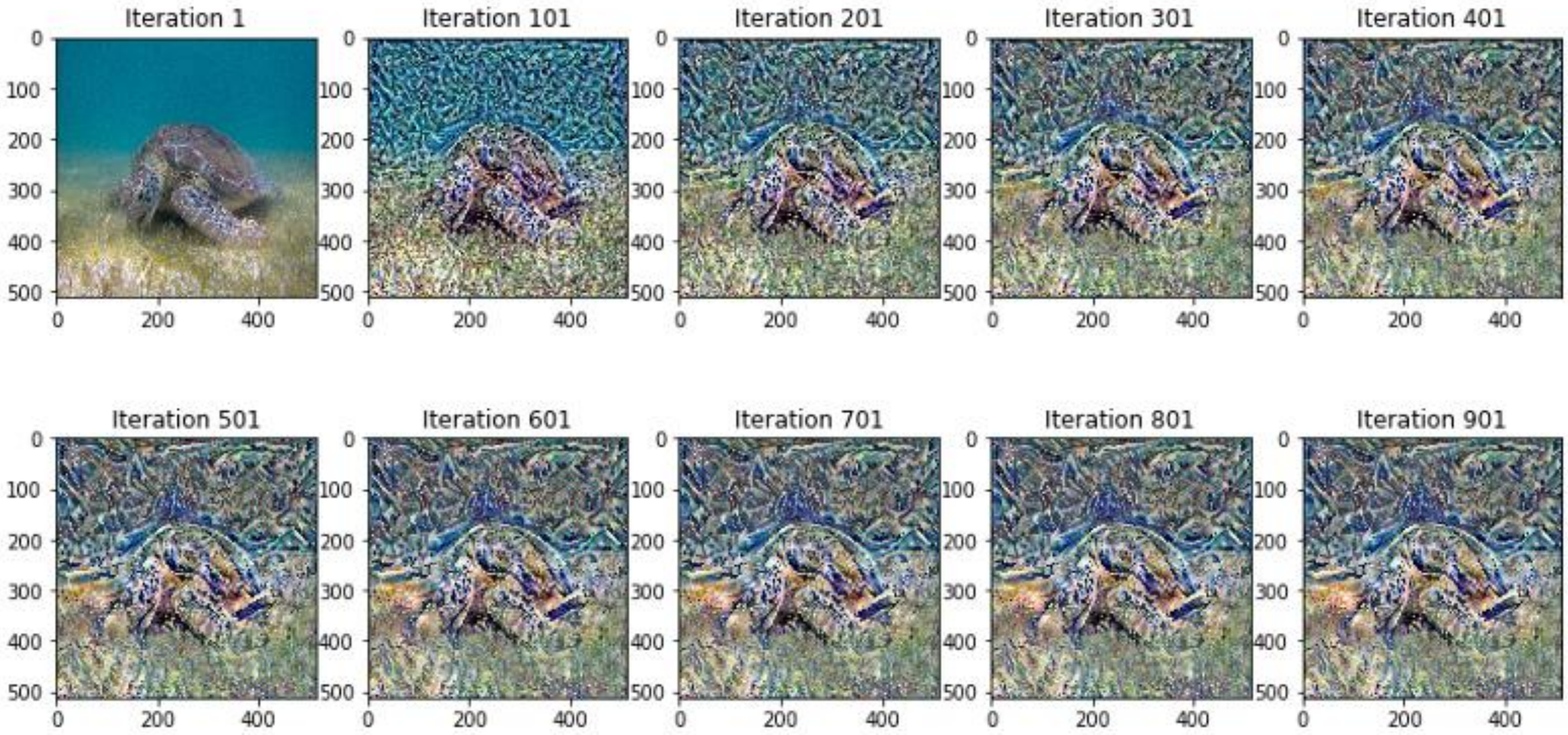


NEURAL STYLE TRANSFER:

CREATING ART WITH DEEP LEARNING

Amjad Mahfoud

AI Research Engineer at Mavericks AI

OVERVIEW

1. Intro style transfer
2. Convolutional Neural Networks
3. Gatys - A Neural Algorithm of Artistic Style

INTRO STYLE TRANSFER:

Neural style transfer is an optimization technique used to take three images, a **content** image, a **style reference** image (such as an artwork by a famous painter), and the **input** image you want to style, and blend them together such that the input image is transformed to look like the content image, but "painted" in the style of the style image.

This is a technique outlined in Leon A. Gatys' paper, A Neural Algorithm of Artistic Style, which is a great read.

Link: https://arxiv.org/abs/1508.06576

Style

Content

Output

EXAMPLE

let's take an image of this turtle and Katsushika Hokusai's *The Great Wave off Kanagawa:*

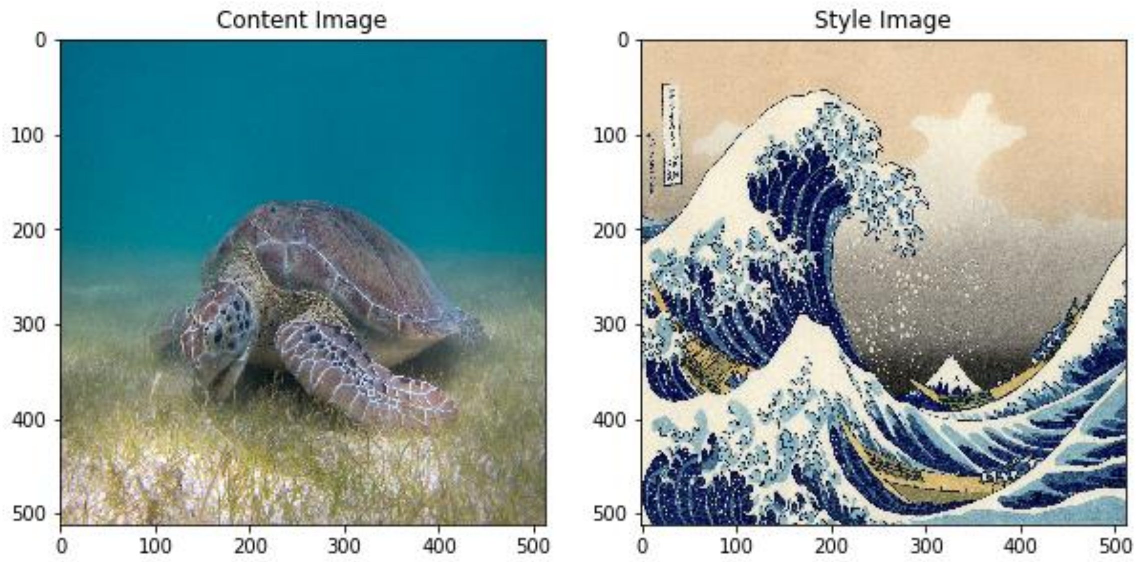


Image of Green Sea Turtle by P. Lindgren, from [Wikimedia Commons]

EXAMPLE CNT'D

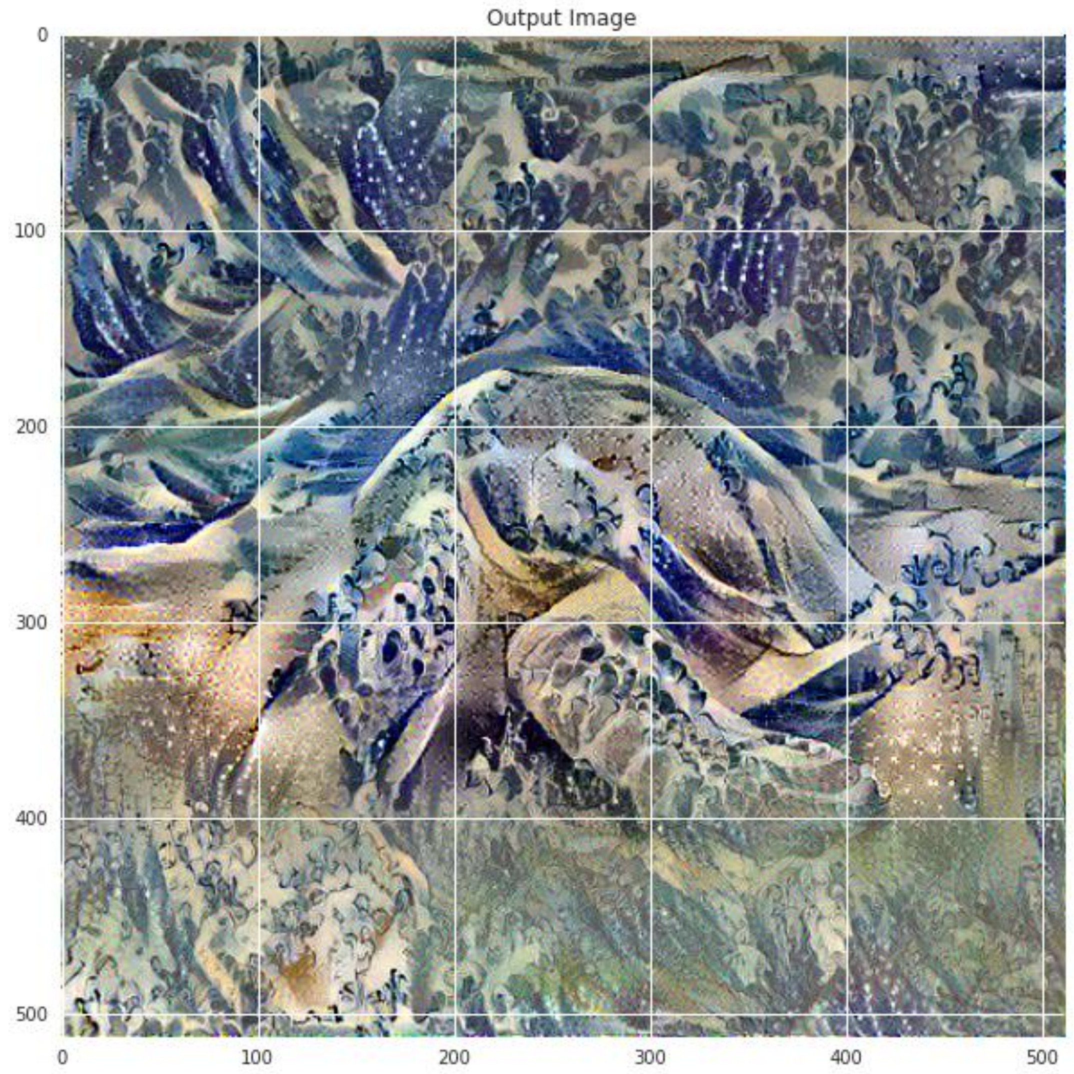
Now how would it look like if we decided to add the texture or style of the waves to the image of the turtle?

Something like this?

Is this magic or just deep learning?

Fortunately, this doesn't involve any magic:

style transfer is a fun and interesting technique that showcases the capabilities and internal representations of neural networks.



HOW?

The principle of neural style transfer is to define two distance functions, one that describes how different the content of two images are, ***<u>Lcontent</u>***, and one that describes the difference between the two images in terms of their style, ***<u>Lstyle</u>***.

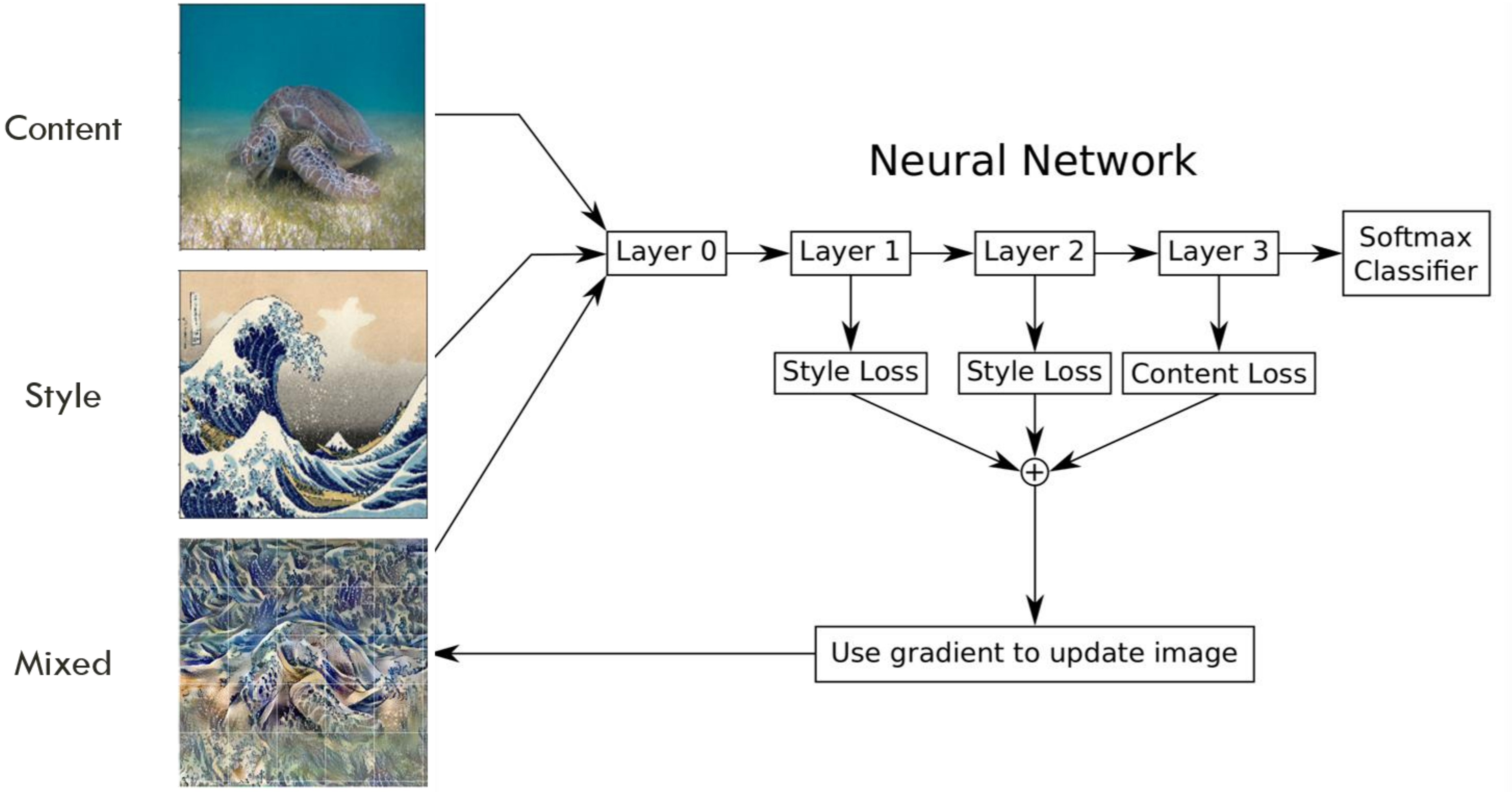
Then, given three images, a desired style image, a desired content image, and the input image (initialized with the content image), we try to transform the input image to minimize the content distance with the content image and its style distance with the style image.

HOW? CNT'D

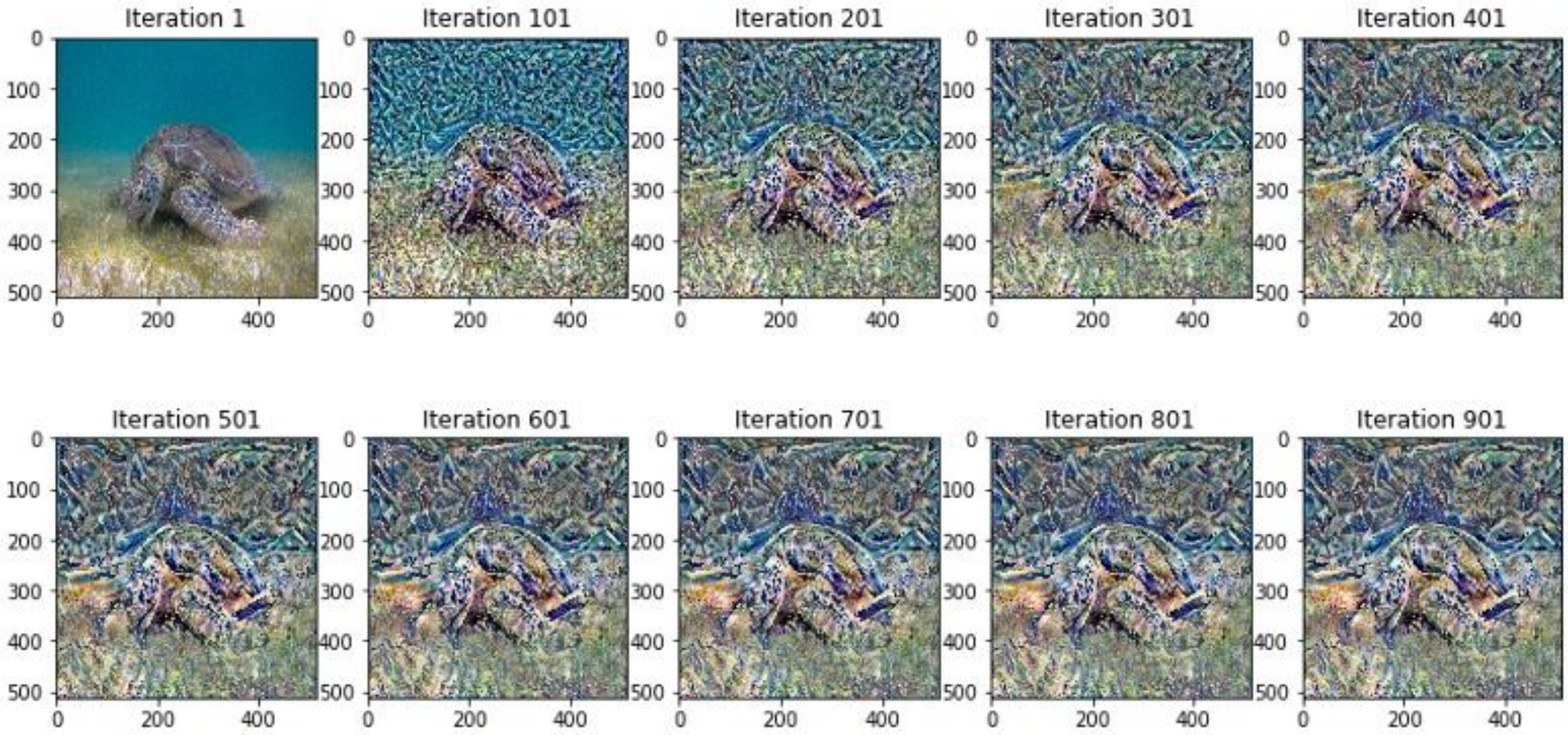
In summary, we'll take the base input image, a content image that we want to match, and the style image that we want to match.

We'll transform the base input image by minimizing the content and style distances (losses) with backpropagation, creating an image that matches the content of the content image and the style of the style image.

HOW? CNT'D



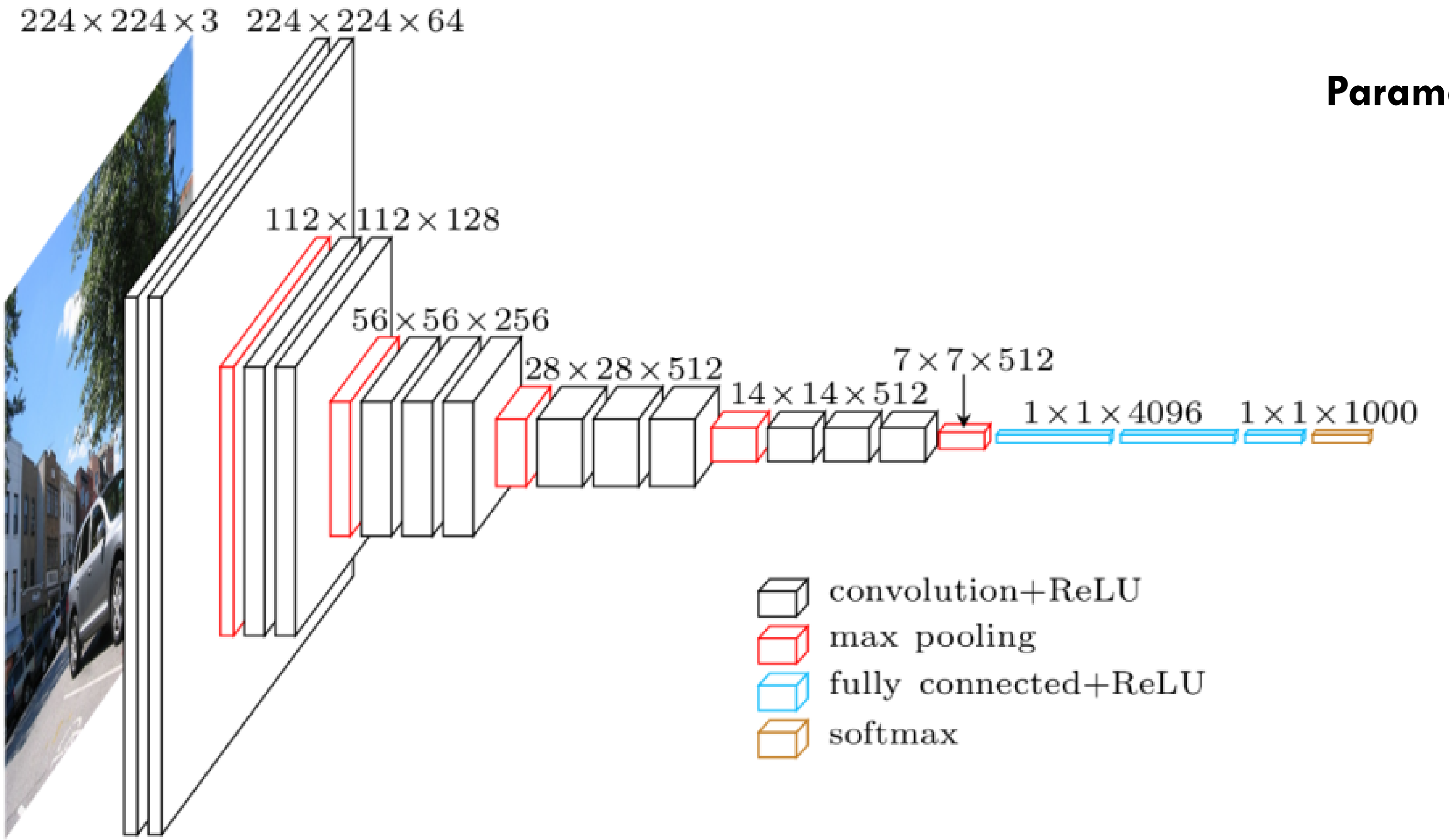
Content
Style
Mixed
Neural Network
Layer 0
Layer 1
Layer 2
Layer 3
Softmax Classifier
Style Loss
Style Loss
Content Loss
Use gradient to update image



CONVOLUTIONAL NEURAL NETWORKS

Feature Extraction

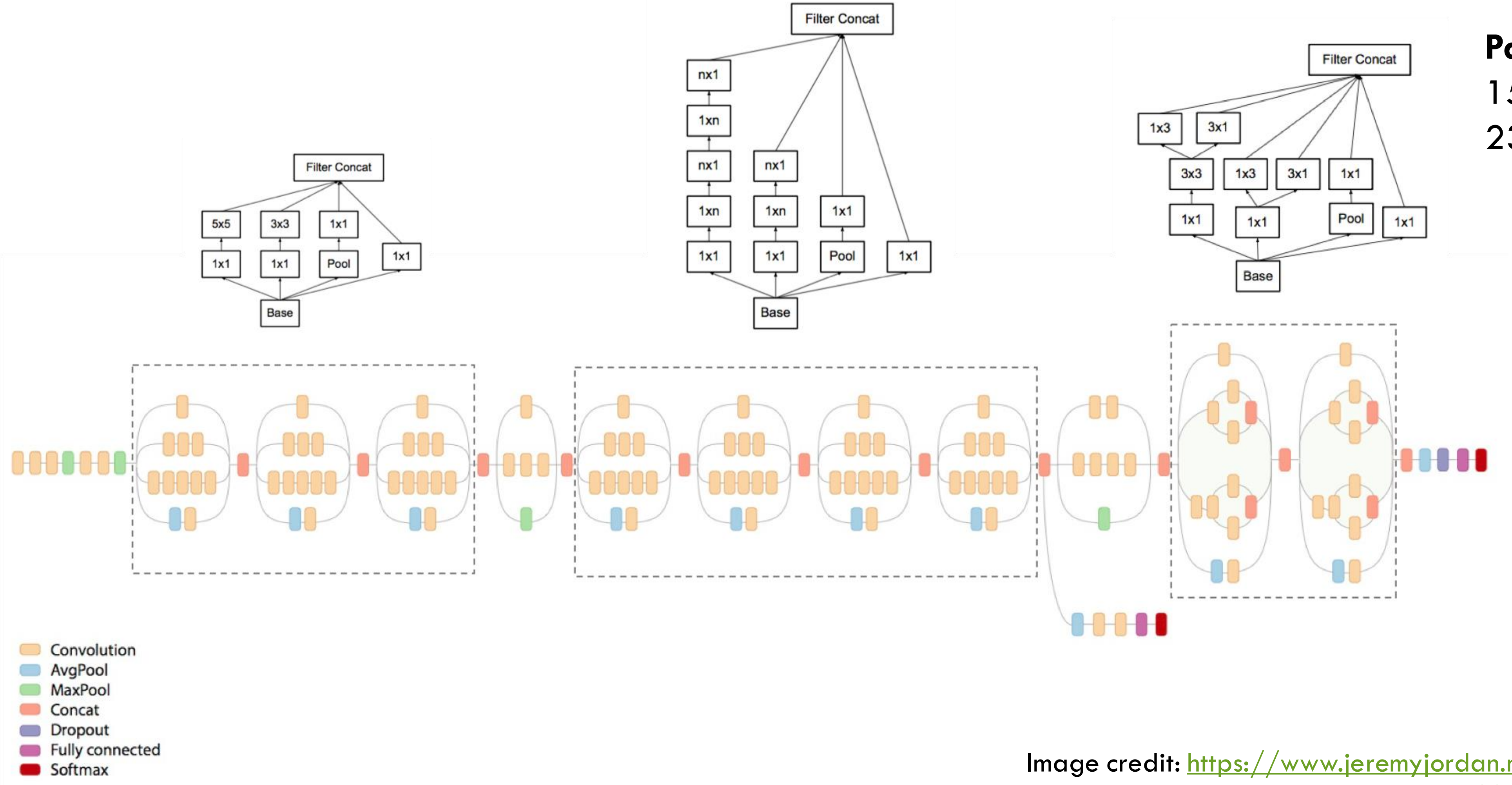
CONVOLUTIONAL NEURAL NETWORKS VGG-16



Parameters: 138 million

Image credit: https://www.jeremyjordan.me/convnet-architectures/
Paper: https://arxiv.org/abs/1409.1556

CONVOLUTIONAL NEURAL NETWORKS INCEPTION



Parameters:

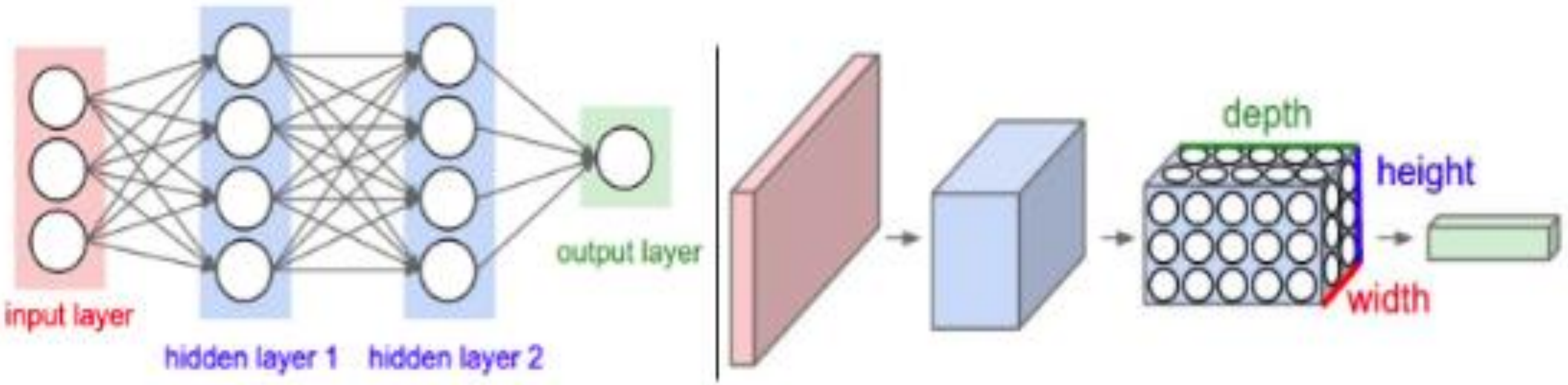
15 million (V1)

23 million (V3)

Image credit: https://www.jeremyjordan.me/convnet-architectures/

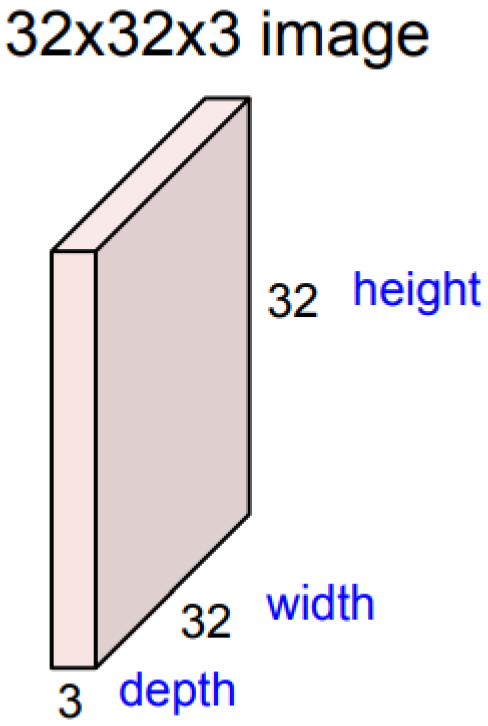
Paper: https://arxiv.org/abs/1409.4842

FULLY CONNECTED NETWORK VS CNN

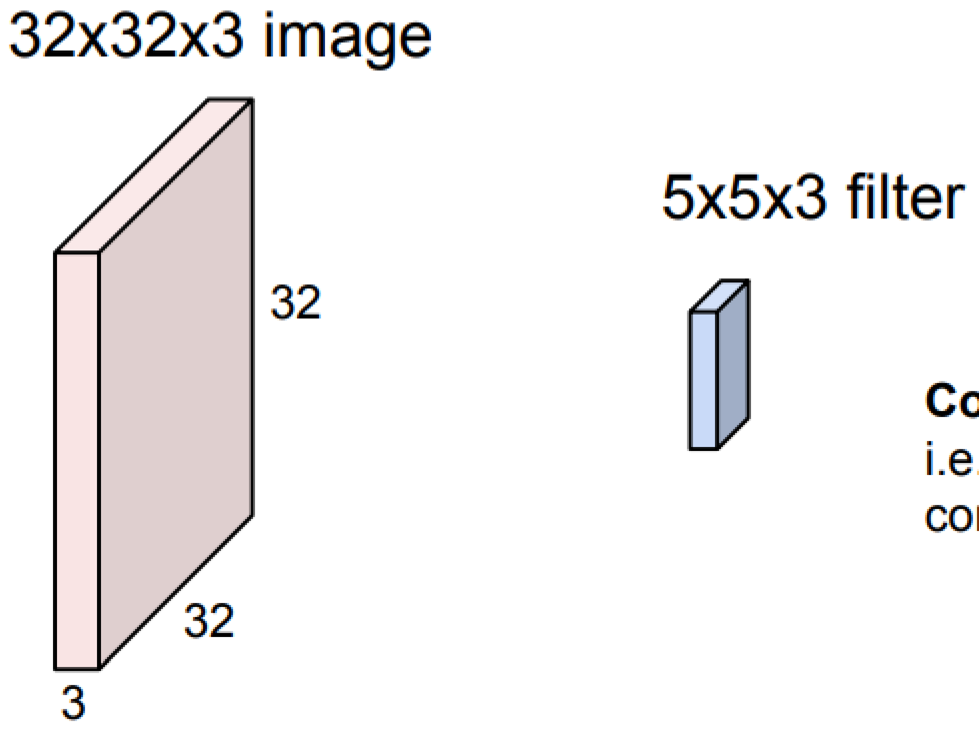


Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

CONVOLUTION LAYER

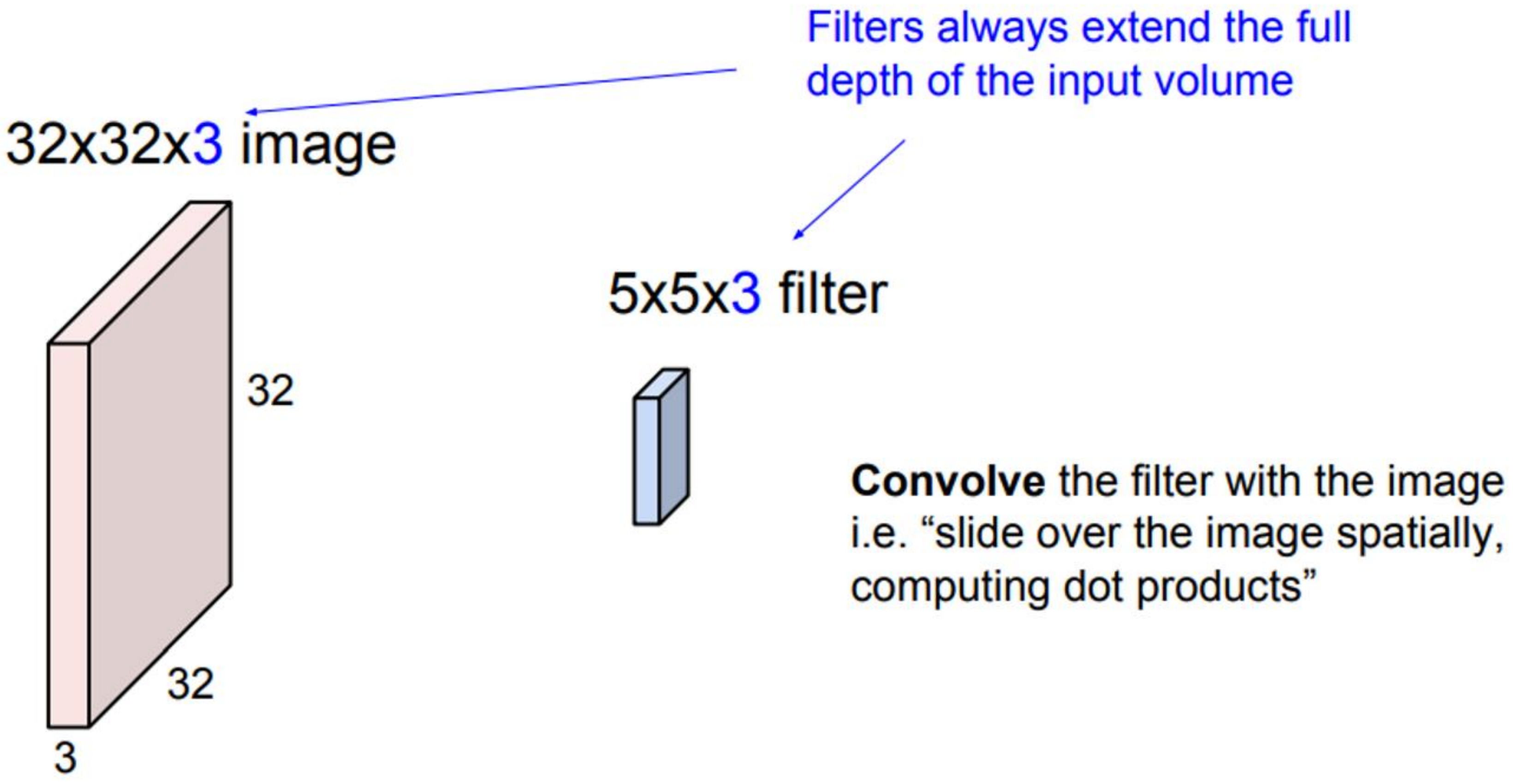


CONVOLUTION LAYER



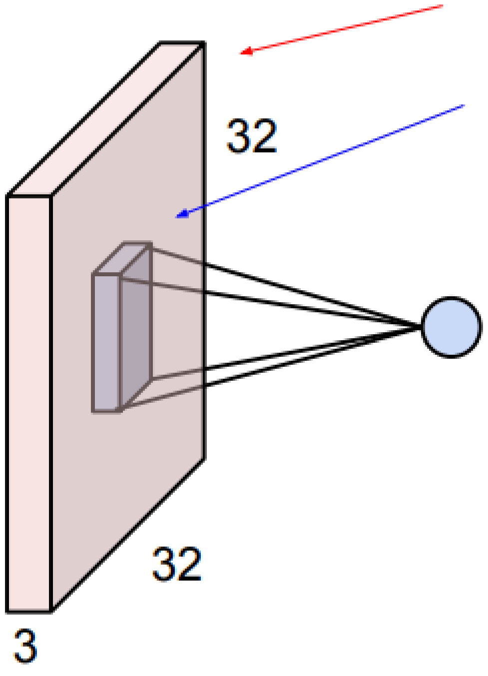
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

CONVOLUTION LAYER



CONVOLUTION LAYER

32x32x3 image
5x5x3 filter w



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

1	1	1	0	0
0	1	1	1	0
0	0	1x1	1x0	1x1
0	0	1x0	1x1	0x0
0	1	1x1	0x0	0x1

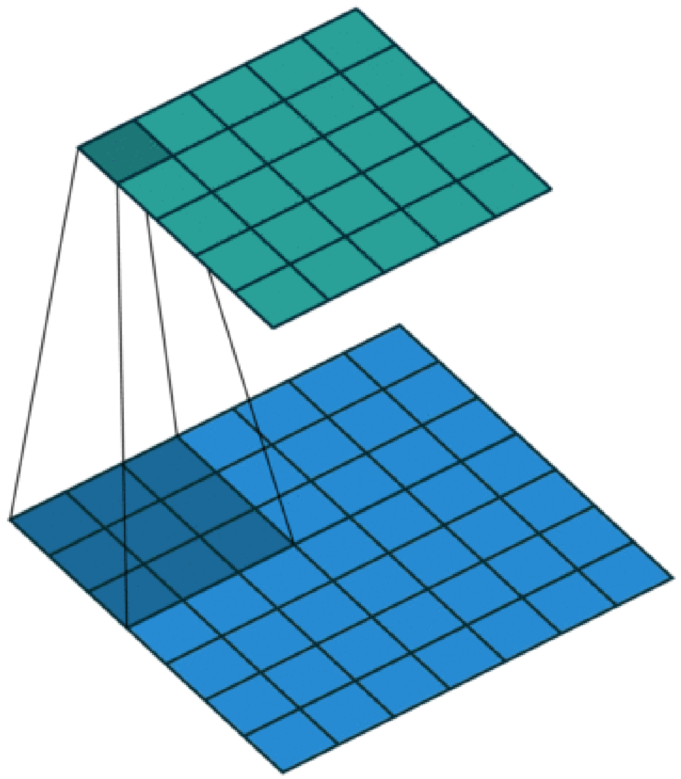
4	3	4
2	4	3
2	3	4

For the sake of explaining, we have shown you the operation in 2D, but in reality convolutions are performed in 3D. Each image is namely represented as a 3D matrix with a dimension for width, height, and depth. Depth is a dimension because of the colors channels used in an image (RGB)

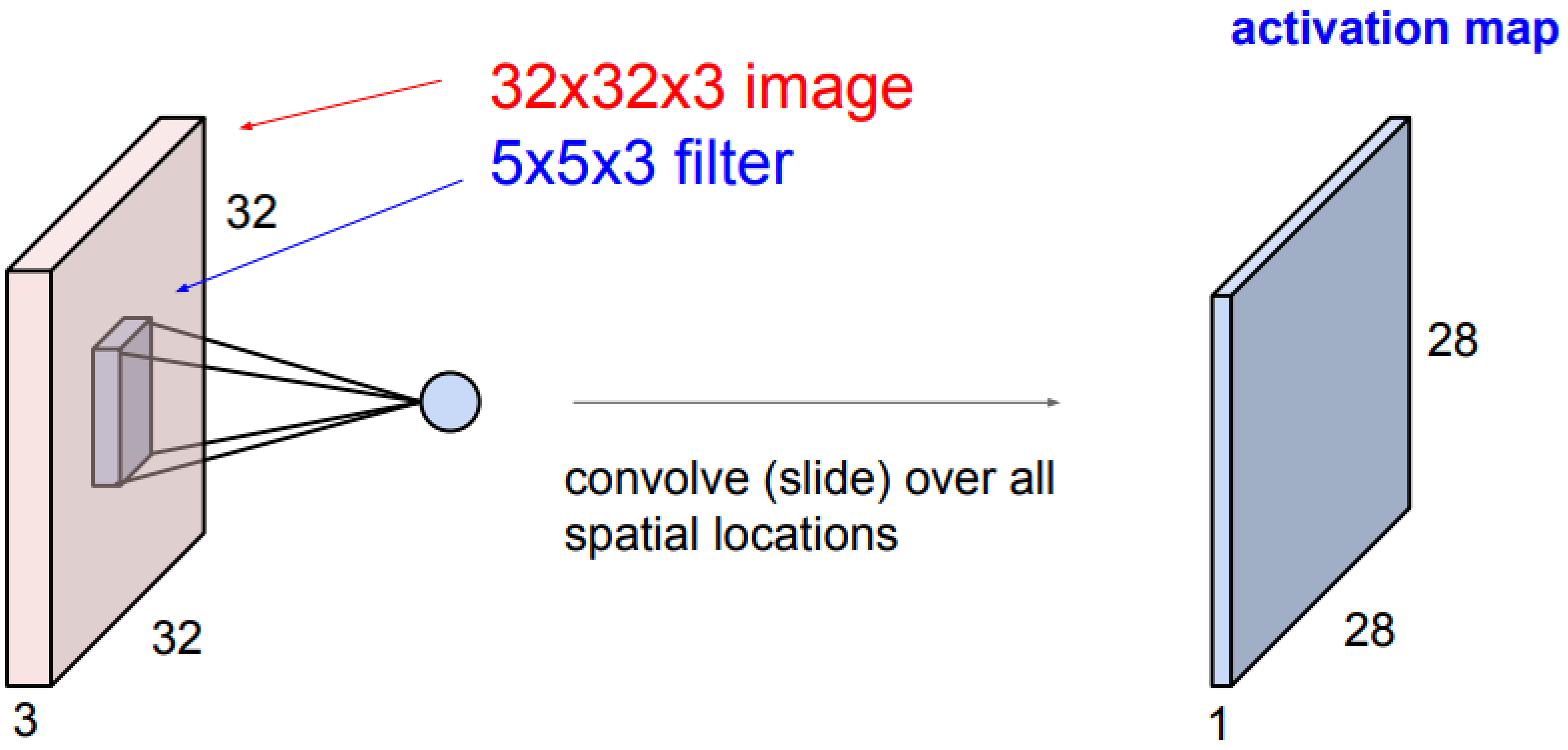
In the figure above, you can see the convolution operation. The **filter** (the green square) slides over our **input** (the blue square) and the sum of the convolution goes into the **feature map** (the red square).

The area of our filter is also called the receptive field, named after the neuron cells! The size of this filter is 3x3.

CONVOLUTION LAYER

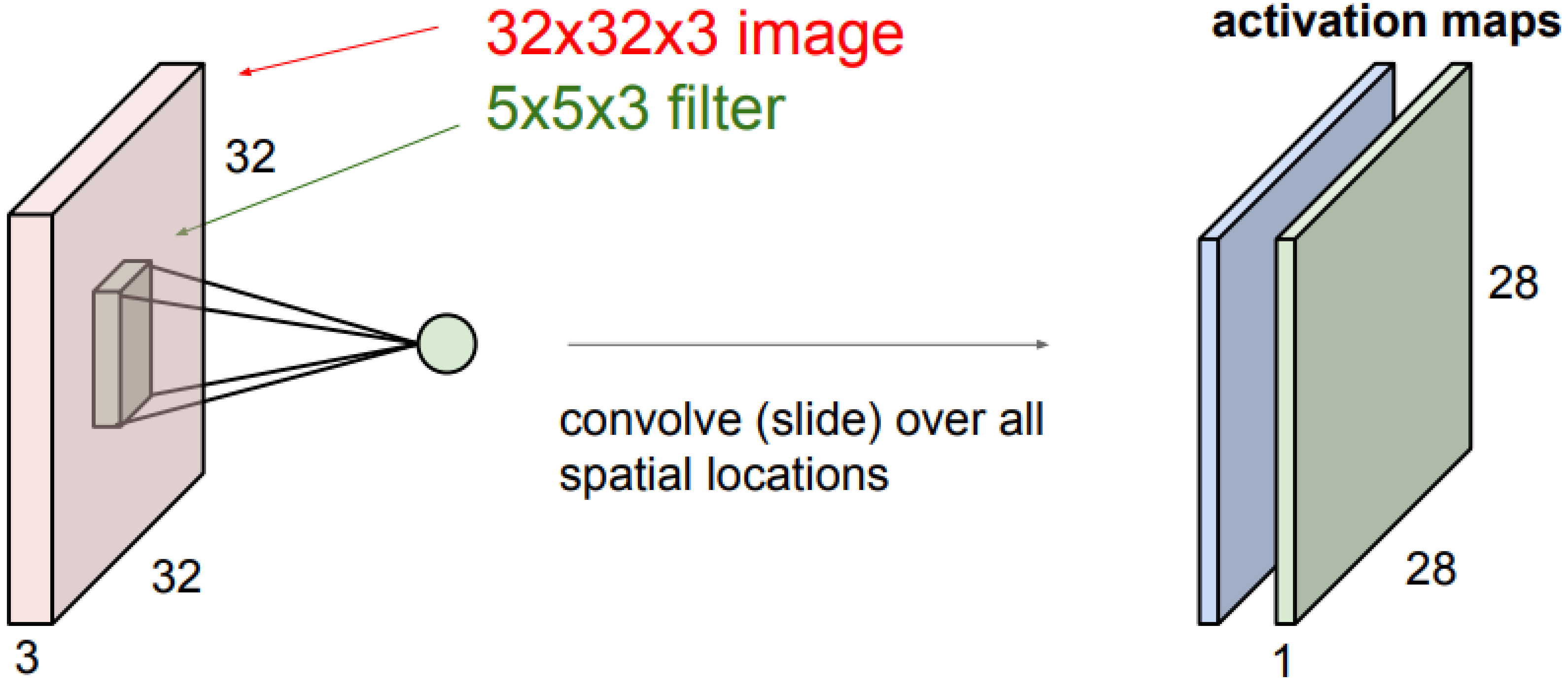


CONVOLUTION LAYER



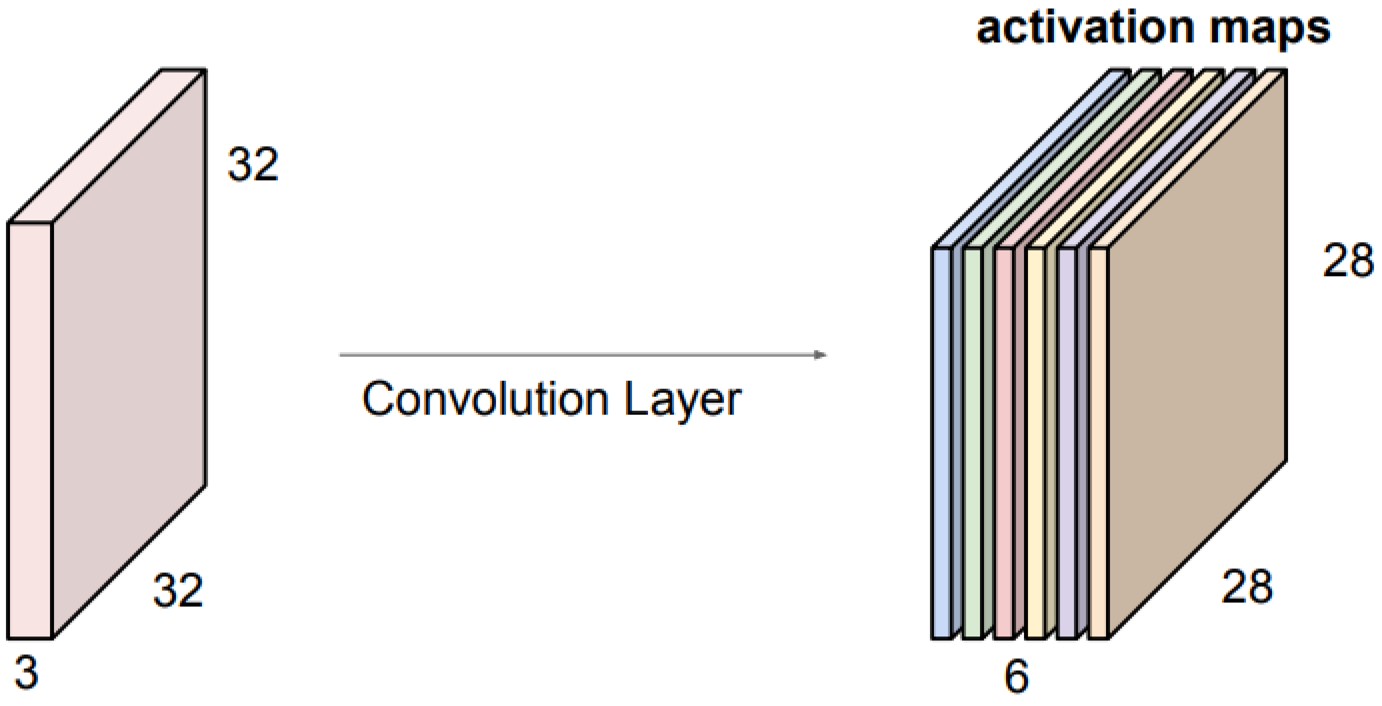
CONVOLUTION LAYER

Consider another filter, green one



CONVOLUTION LAYER

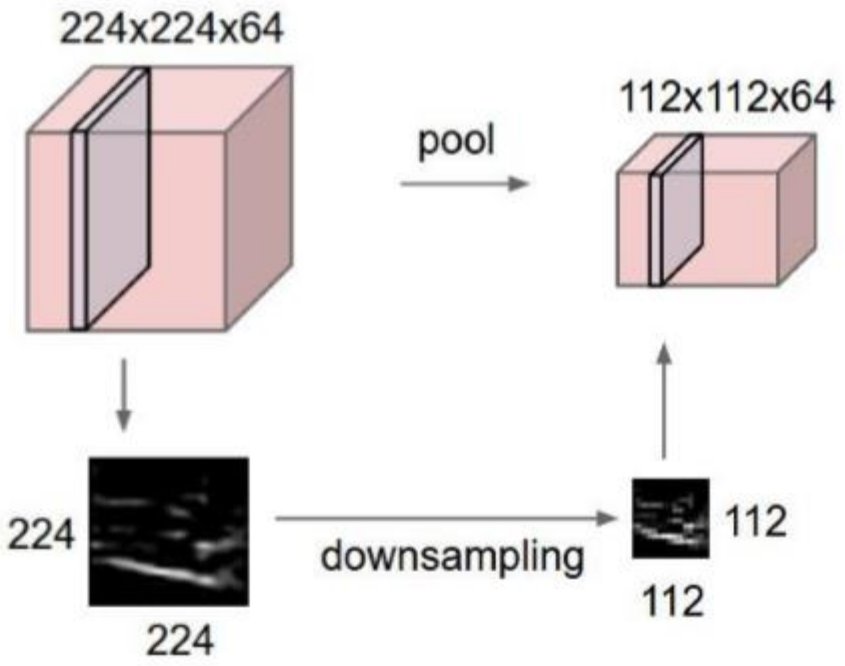
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



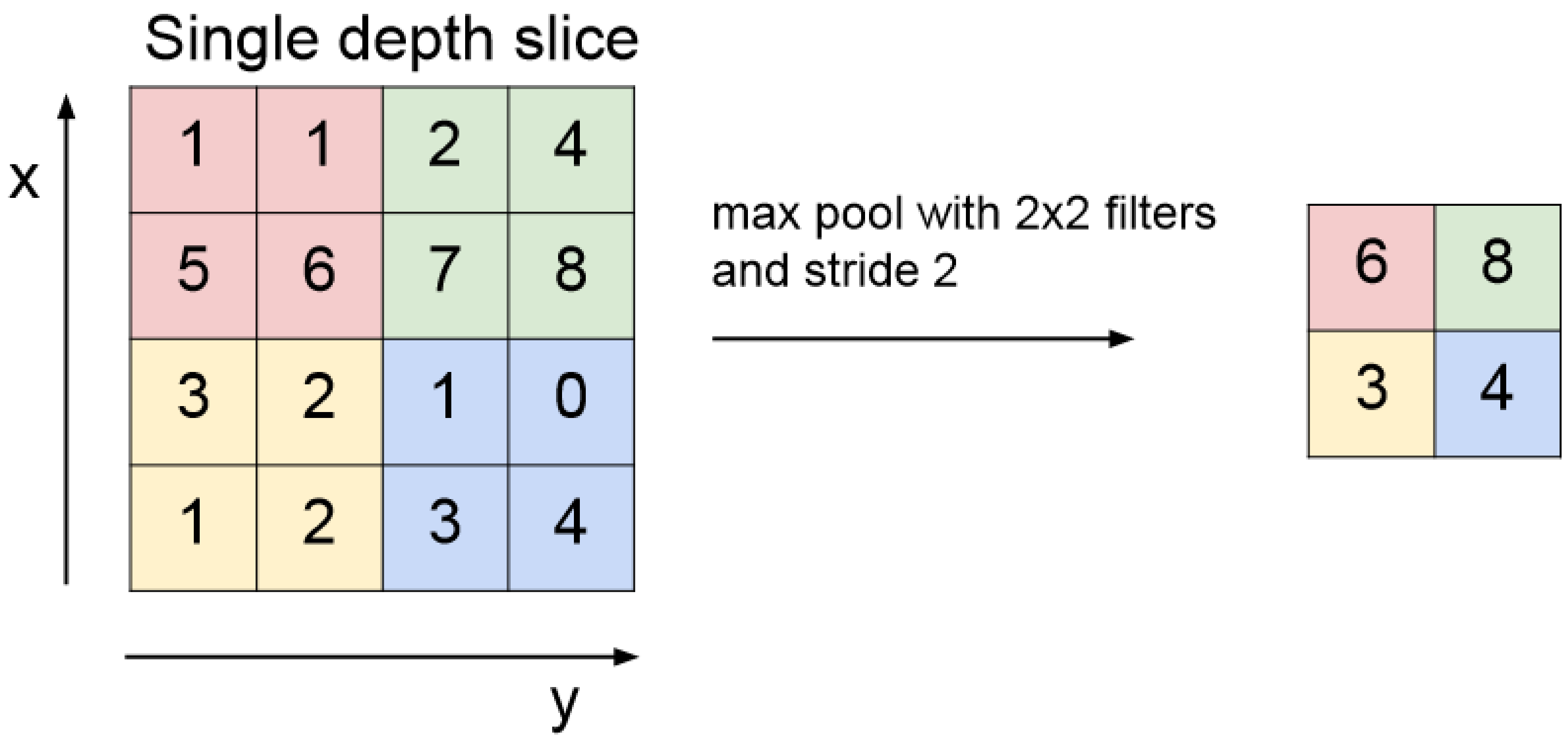
We stack these up to get a "new image" of size 28x28x6!

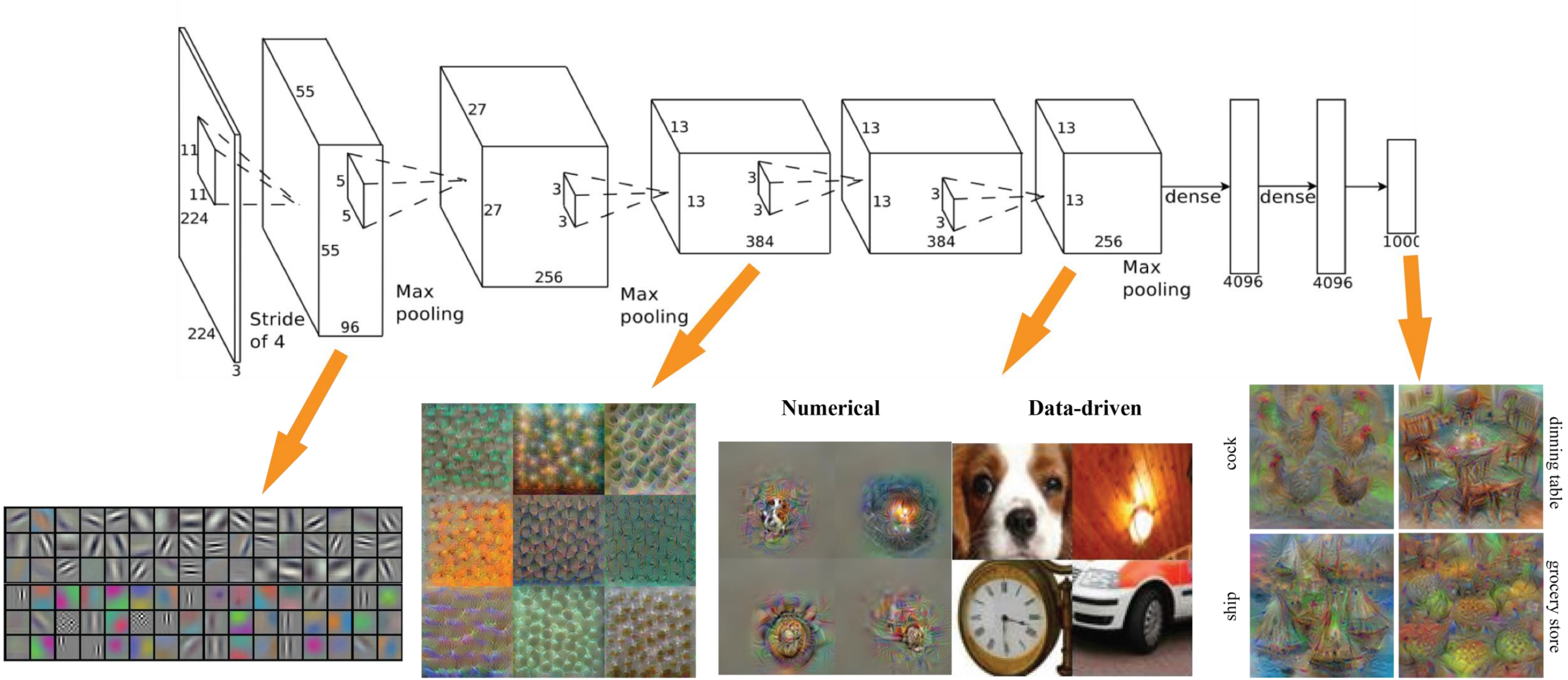
POOLING LAYER

- makes the representations smaller and more manageable
- operates over each activation map independently



MAX POOLING


Single depth slice
x
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
y
max pool with 2x2 filters
and stride 2
6 8
3 4



Conv 1: Edge+Blob

Conv 3: Texture

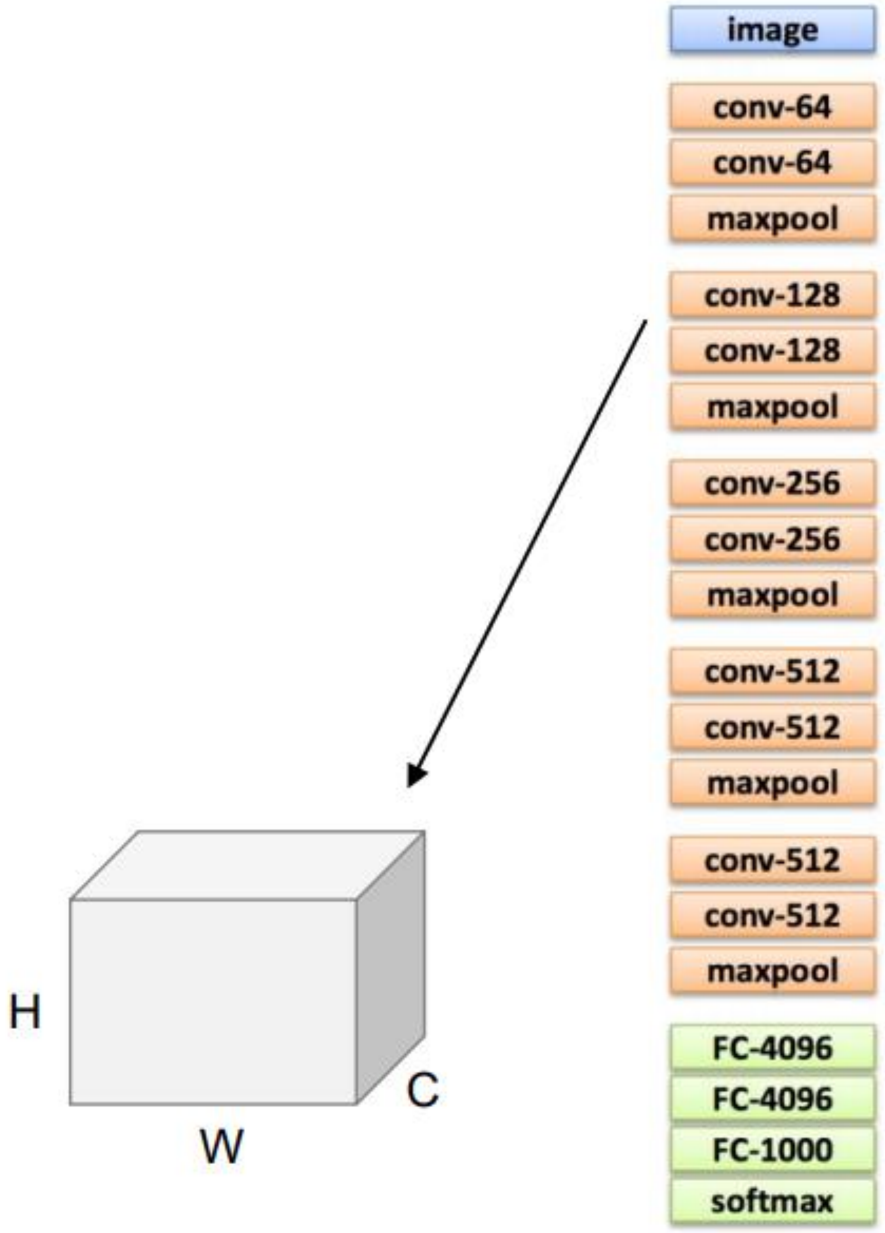
Conv 5: Object Parts

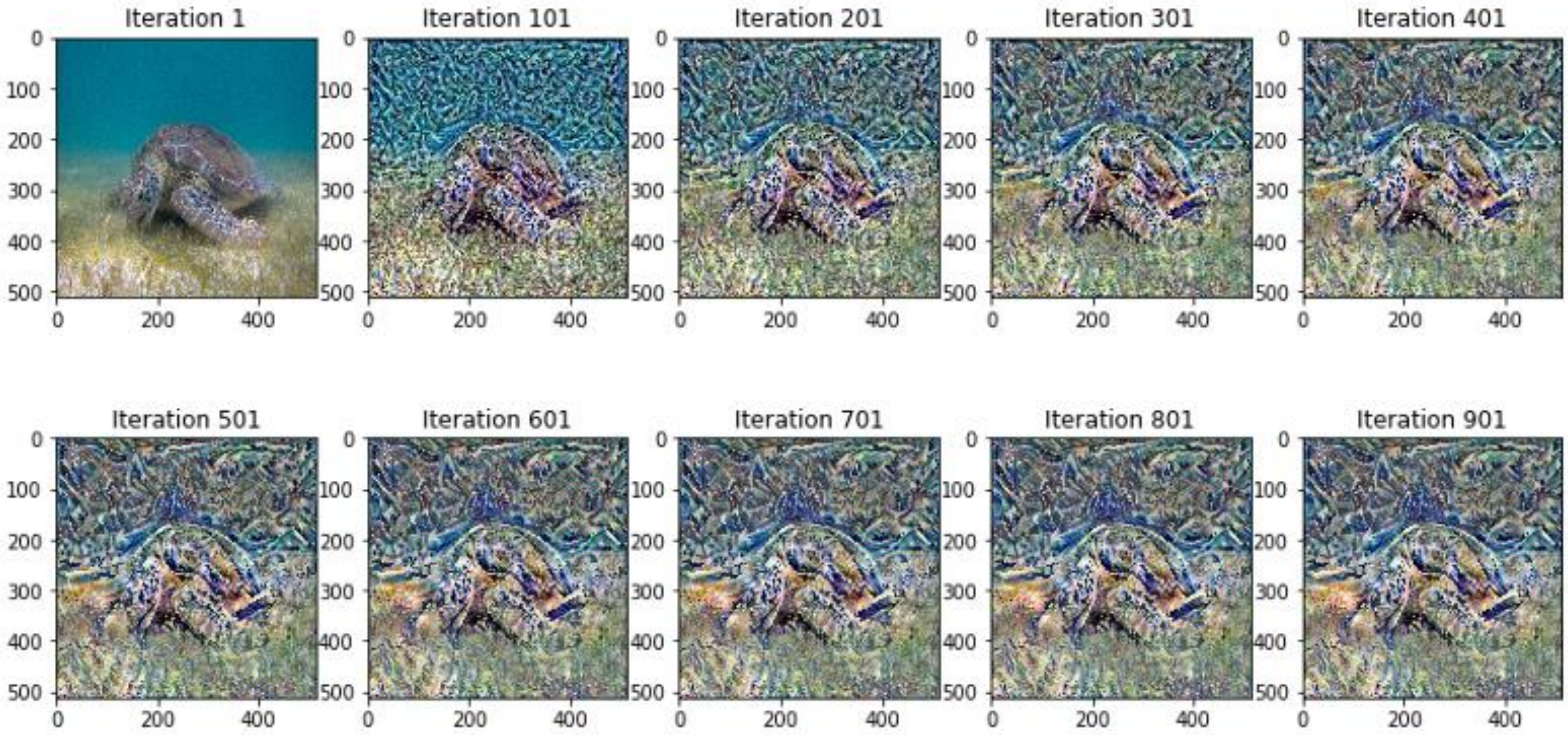
Fc8: Object Classes

VGG-STYLE NETWORKS

Consist of repeated

1. Convolutions
2. ReLU
3. MaxPool
4. FC + Softmax at the end





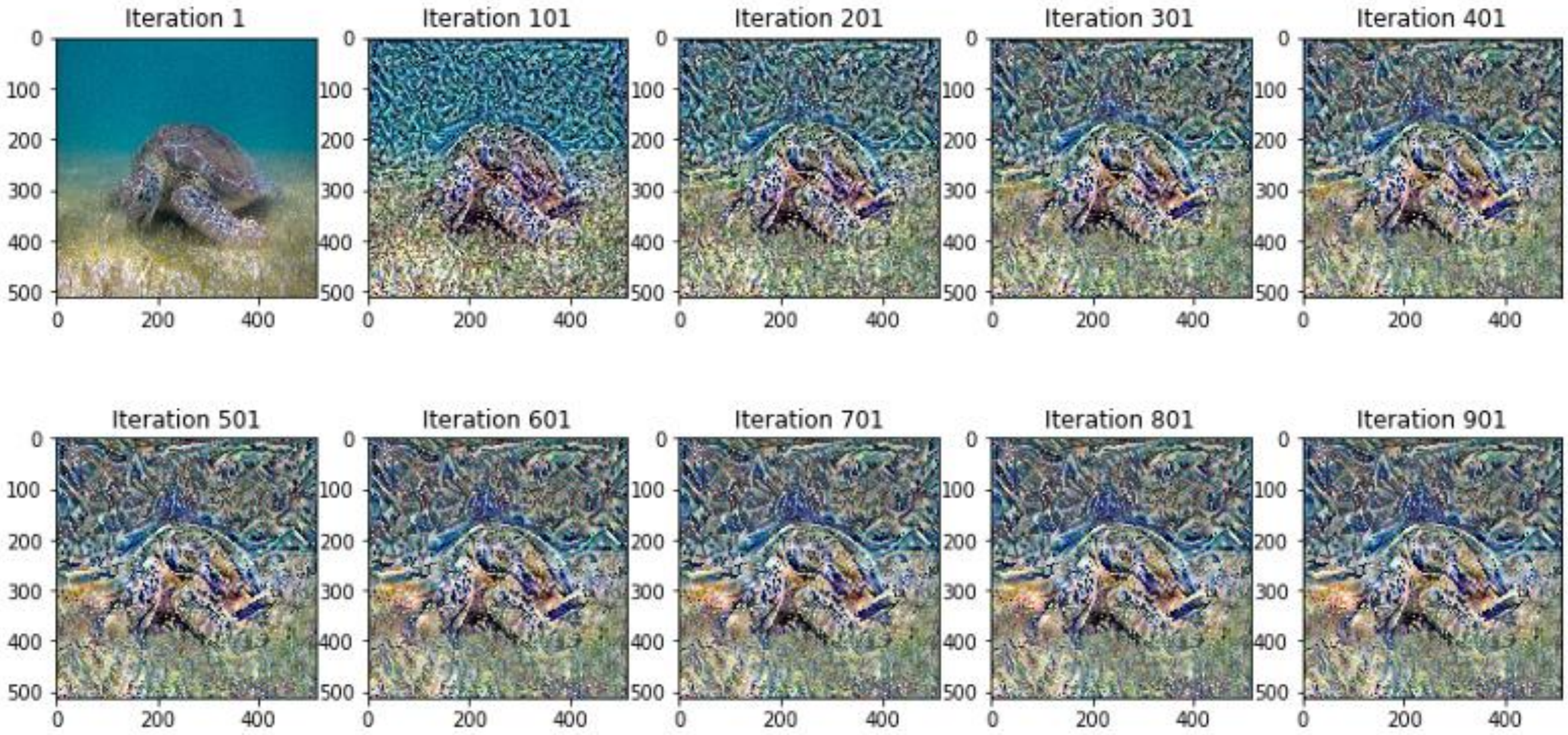
A NEURAL ALGORITHM OF ARTISTIC STYLE

Gatys – et al

A NEURAL ALGORITHM OF ARTISTIC STYLE

The principle of neural style transfer is to define two distance functions, one that describes how different the content of two images are, Lcontent, and one that describes the difference between the two images in terms of their style, Lstyle. Then, given three images, a desired style image, a desired content image, and the input image (initialized with the content image), we try to transform the input image to minimize the content distance with the content image and its style distance with the style image.

In summary, we'll take the base input image, a content image that we want to match, and the style image that we want to match. We'll transform the base input image by minimizing the content and style distances (losses) with backpropagation, creating an image that matches the content of the content image and the style of the style image.



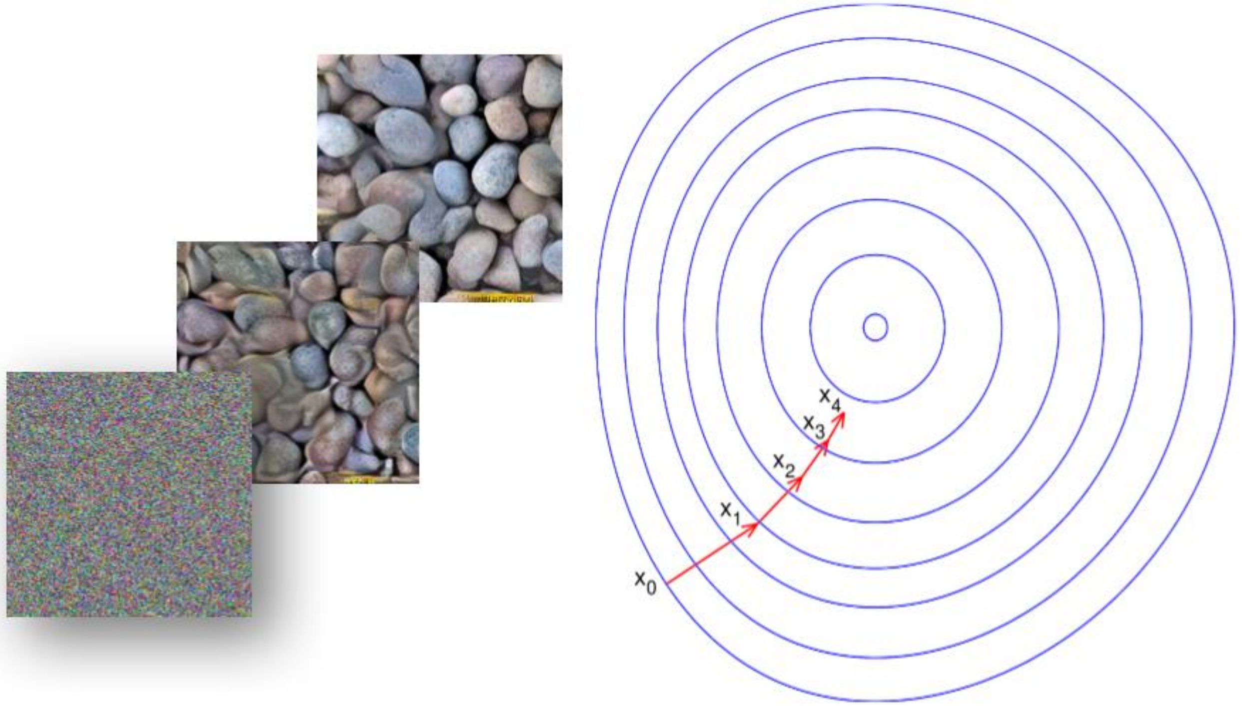
CONTENT TRANSFER

RECONSTRUCTING CONTENT

Given image, how can we find a new one with the same content?

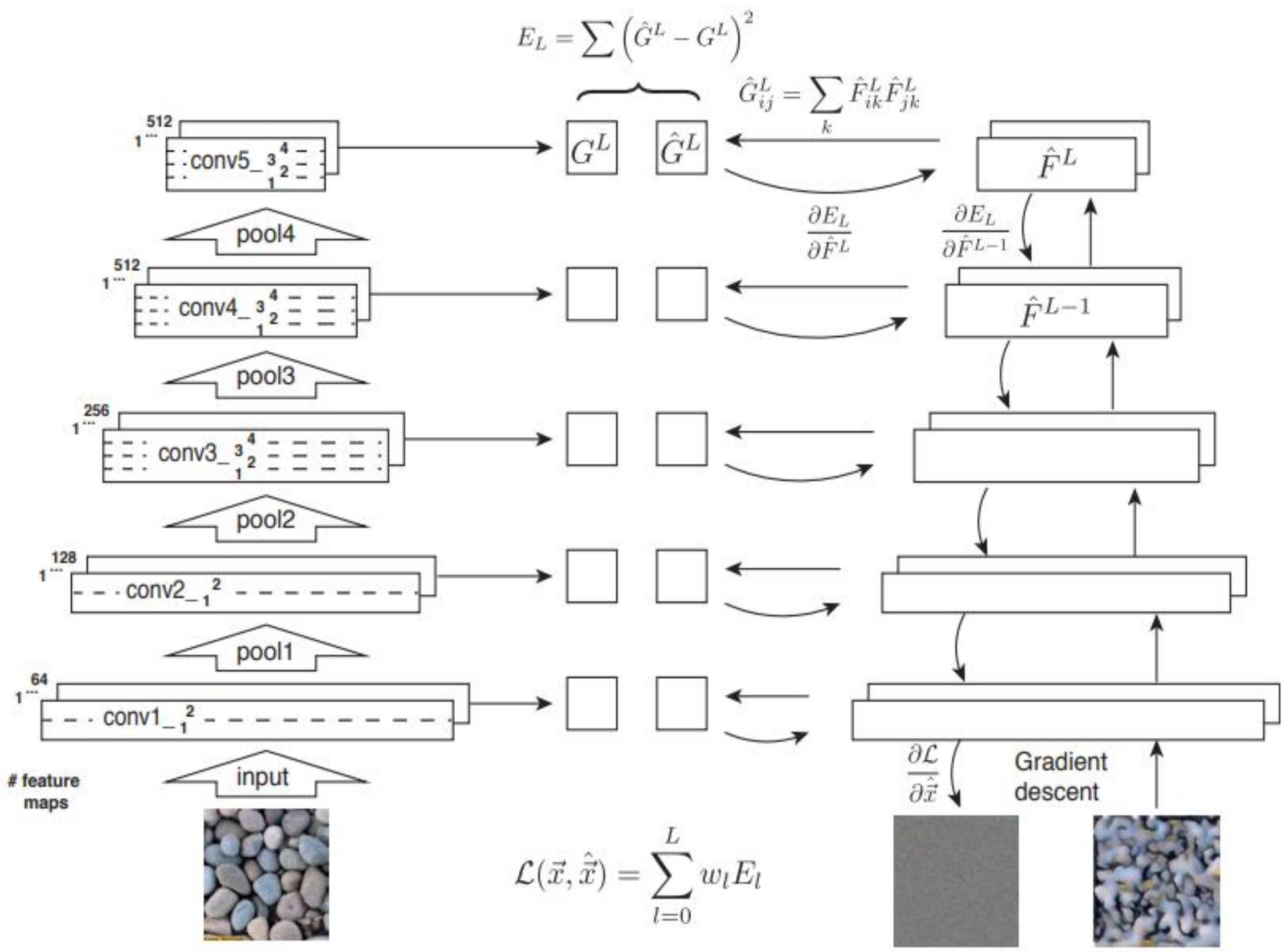
1. Find content distance measure between images
2. Start from random noise image
3. Minimize distance through iteration

$$x^* = \arg\min_x \mathcal{L}(x)$$



CONTENT DISTANCE M

1. Load a pre-trained CNN (e.g. VGG19)
2. Pass image #1 through the net
3. Save activation maps from conv-layers
4. Pass image #2 through the net
5. Save activation maps from conv-layers
6. Calculate Euclidean distance bet activation maps from image #1 a #2 and sum up for all layers



CONTENT DISTANCE MEASURE

1. Load a pre-trained CNN (e.g. VGG19)
2. Pass image #1 through the net
3. Save activation maps from conv-layers
4. Pass image #2 through the net
5. Save activation maps from conv-layers
6. Calculate Euclidean distance between activation maps from image #1 and #2 and sum up for all layers

$$L_{content}(x,\hat{x}) = \frac{1}{2}\sum_{l} w_l (A_l(x) - A_l(\hat{x}))^2$$

RECONSTRUCTING CONTENT

1. Start from random image
2. Update it using gradient descent

$$L_{content}(x,\hat{x}) = \frac{1}{2}\sum_{l} w_l (A_l(x) - A_l(\hat{x}))^2$$

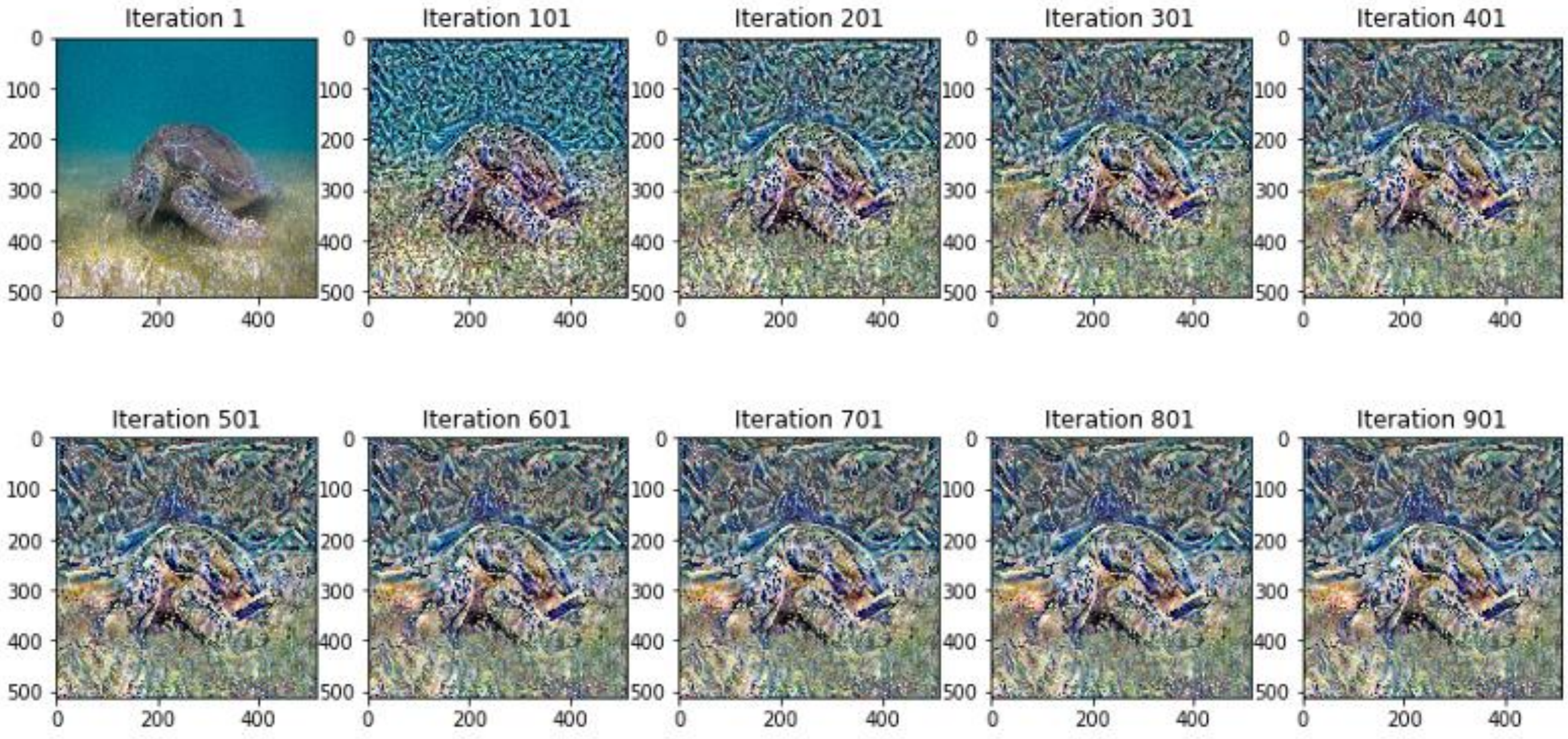
$$\hat{x}_{t+1} = \hat{x}_t - \varepsilon \frac{\partial L_{content}}{\partial \hat{x}}$$

RECONSTRUCTING CONTENT

Reconstructions from intermediate layers Higher layers are less sensitive to changes in color, texture, and shape

RECONSTRUCTING CONTENT

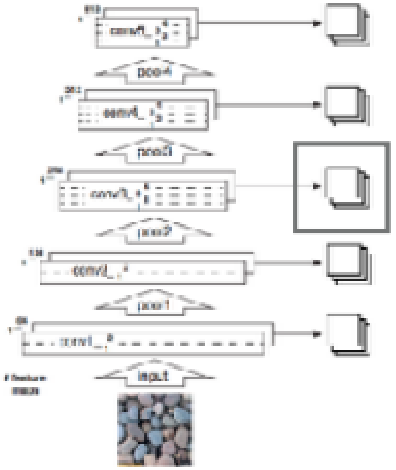
Reconstructions from the representation after last last pooling layer (immediately before the first Fully Connected layer)

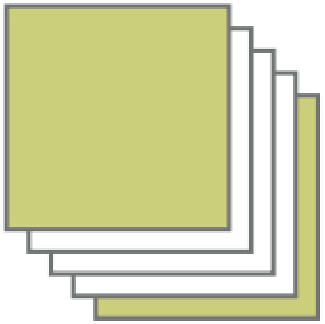


STYLE TRANSFER

STYLE DISTANCE MEASURE

1. Represent style by **Gram matrix** - pairwise covariance of activation maps
2. Just the uncentered covariance matrix between vectorized activation maps





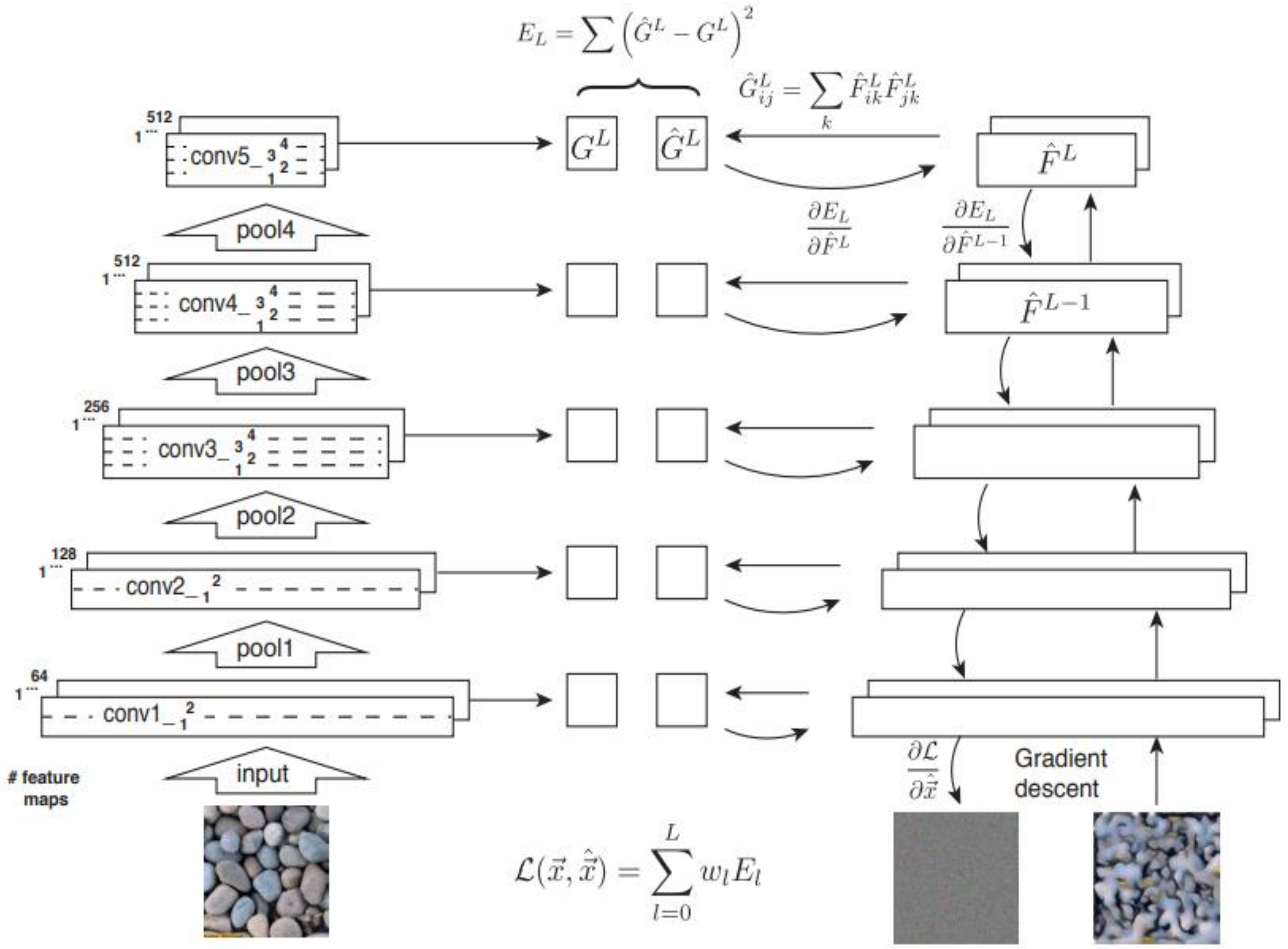
$$G_{ij}^{l}(x) = \vec{A}_i^{l}(x) \bullet \vec{A}_j^{l}(x)$$

$$\begin{pmatrix} G(A_1, A_1) & \cdots & G(A_1, A_n) \\ \vdots & \ddots & \vdots \\ G(A_n, A_1) & \cdots & G(A_n, A_n) \end{pmatrix}$$

STYLE DISTANCE MEA

Style loss - Euclidean distance betwe
Gram matrices from two images

$$L_{style}(x,\hat{x}) = \frac{1}{2}\sum_{l} w_l (G^l(x) - G^l(\hat{x}))^2$$

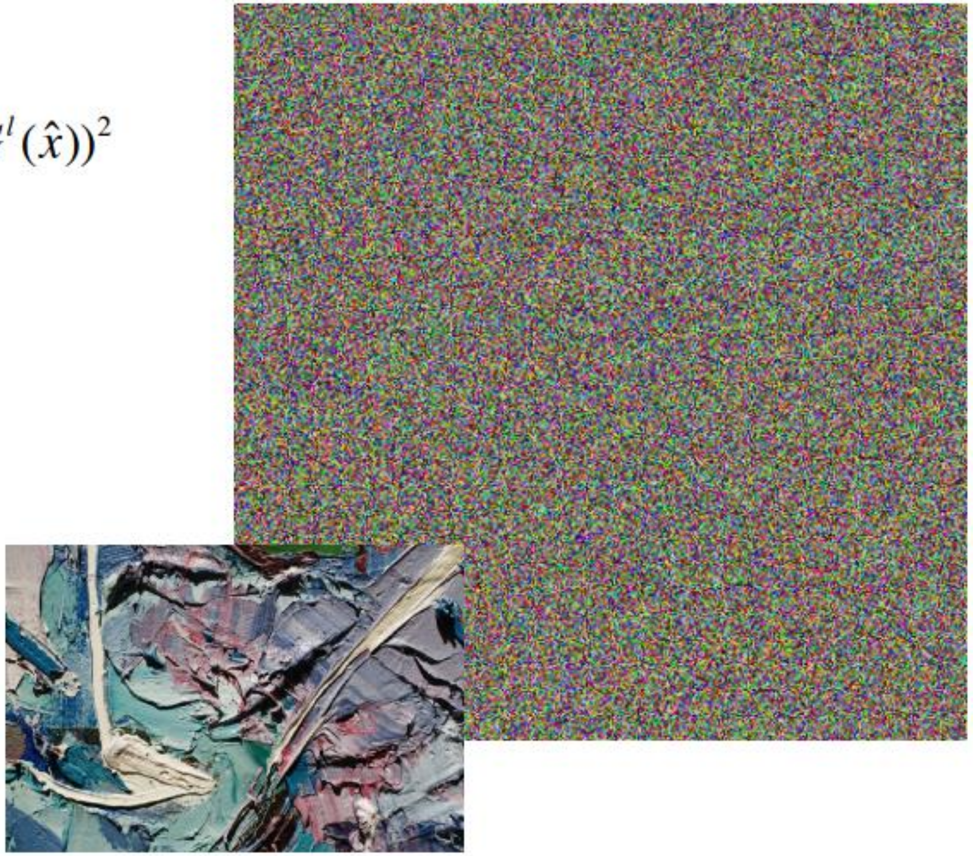


RECONSTRUCTING STYLE

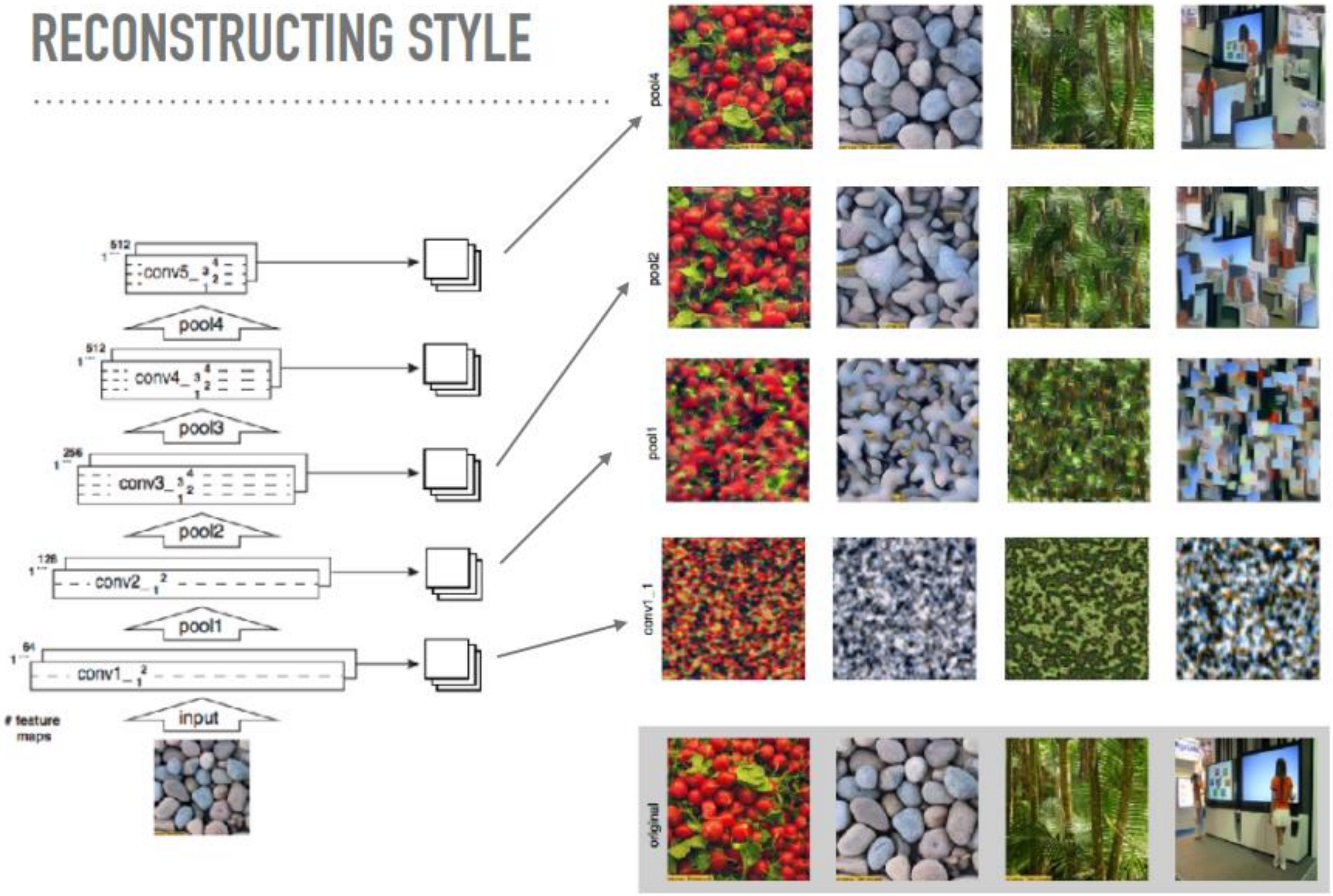
1. Start from random image
2. Update it using gradient descent

$$L_{style}(x,\hat{x}) = \frac{1}{2}\sum_{l} w_l (G^l(x) - G^l(\hat{x}))^2$$

$$\hat{x}_{t+1} = \hat{x}_t - \varepsilon \frac{\partial L_{style}}{\partial \hat{x}}$$



RECONSTRUCTING STYLE



THANKS FOY YOUR ATTENTION
ANY QUESTIONS?

AMJAD MAHFOUD
AI RESEARCH ENGINEER AT MAVERICKS AI | amjadoof@gmail.com