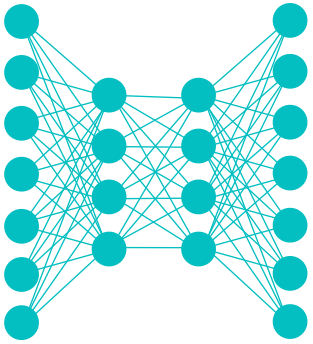
Lecture Notes for

Neural Networks and Machine Learning

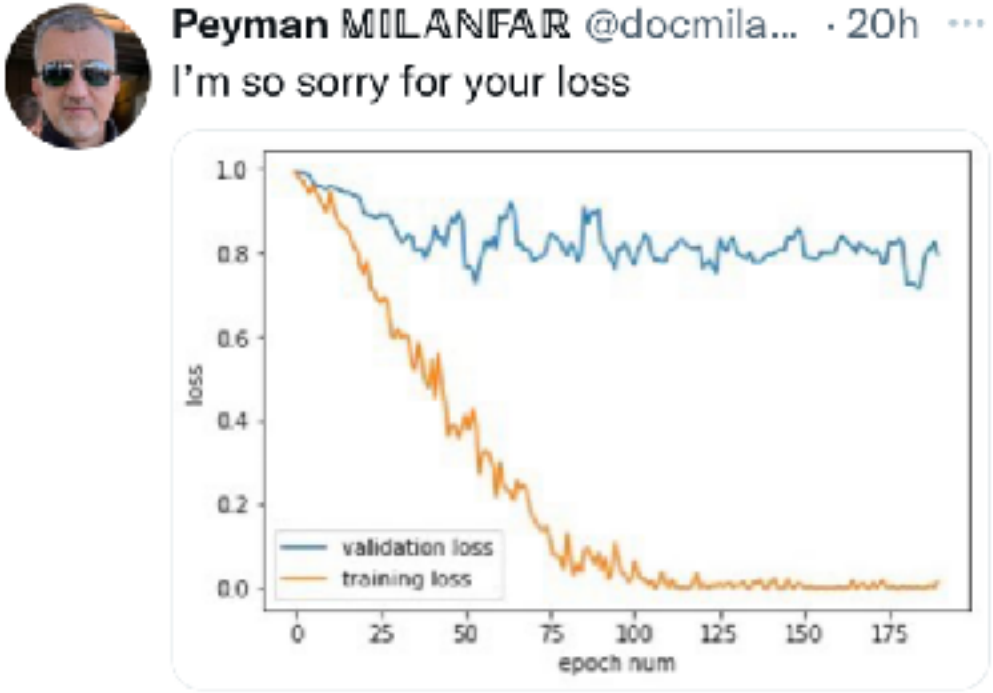
Neural Style Transfer



Logistics and Agenda

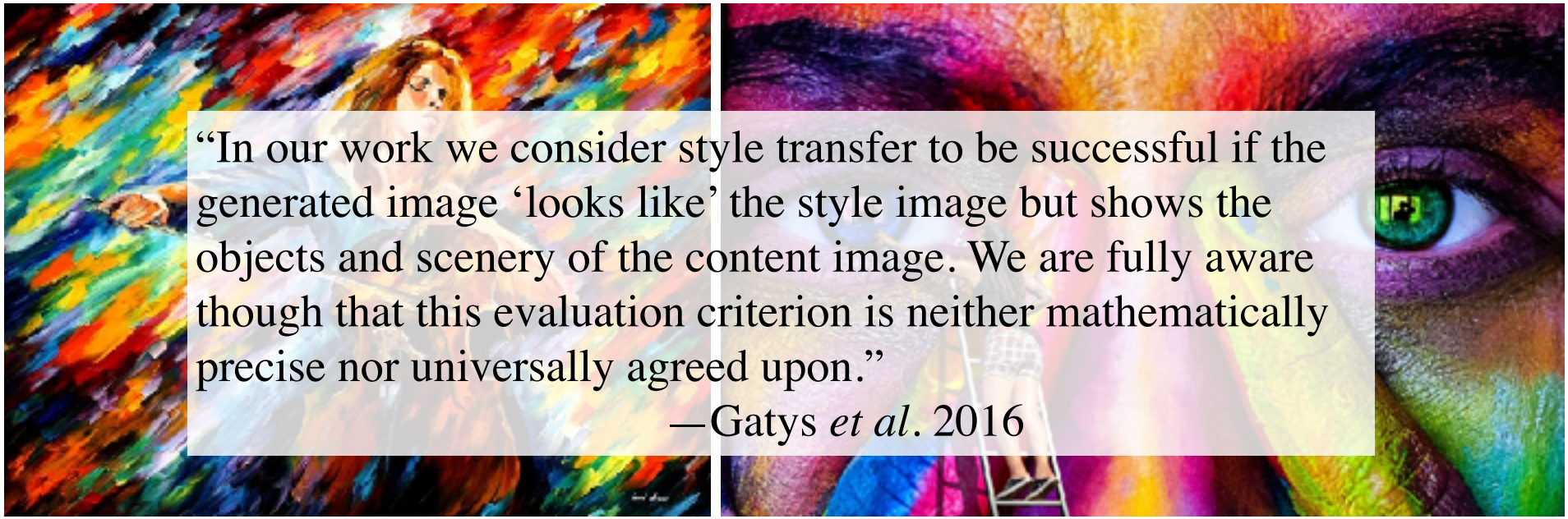
- Logistics
  - Next Assignment: Style Transfer
  - Next Lecture: “Fast Style Transfer” Student Presentation
- Agenda
  - A History of Style Transfer (today)
  - Image Optimization Algorithms (today)
  - Model Optimization Algorithms
  - One Shot Algorithms
  - Evaluating Style Transfer Performance
  - Extensions in Other Domains

Style Transfer: A History



The Premise

- "Style" can be transferred from one domain to another
- While preserving the "content" of an image
- Style and content are in the eye of the beholder
- Can you define style versus content?
- Do you know it when you see it?

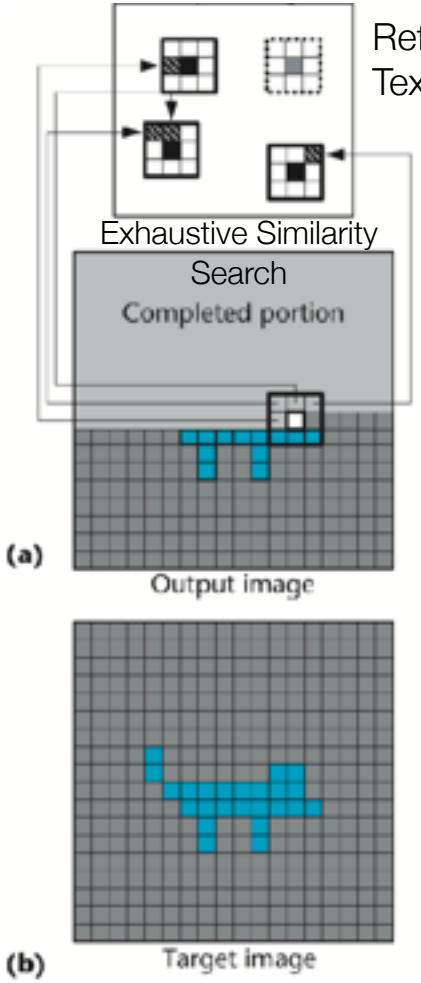


"In our work we consider style transfer to be successful if the generated image 'looks like' the style image but shows the objects and scenery of the content image. We are fully aware though that this evaluation criterion is neither mathematically precise nor universally agreed upon."

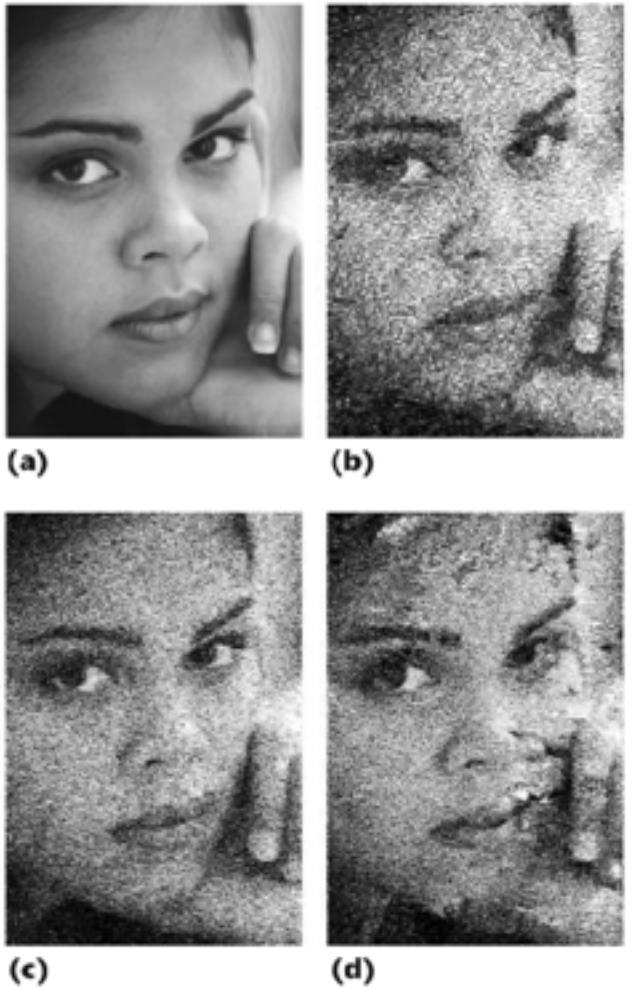
—Gatys *et al*. 2016

Before Neural Networks

- Ashekhmin, Fast Texture Transfer, 2003



1 (a) Each pixel in this L-shaped neighborhood generates a shifted candidate pixel (black) according to its original position in the input image (hatched). A single random candidate (light blue with dashed lines) is added with probability *p*. The candidate whose neighborhood best matches the one in the output image according to an application-specific similarity metric is chosen as the next pixel value. In the original algorithm, the complete neighborhood for matching is composed from two L-shaped halves, with top half coming from the already synthesized part of the output image and (b) the bottom half from the target image.



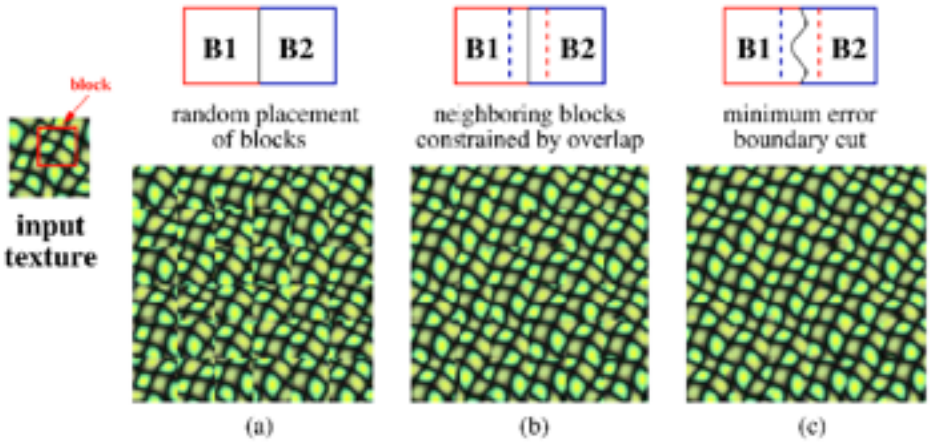
Before Neural Networks

- Efros and Freeman, 2001

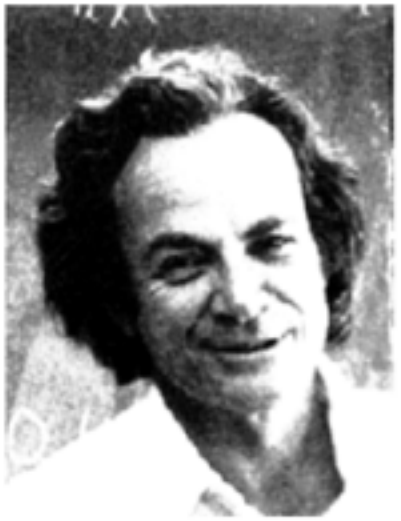
2.2 The Image Quilting Algorithm

The complete quilting algorithm is as follows:

- Go through the image to be synthesized in raster scan order in steps of one block (minus the overlap).
- For every location, search the input texture for a set of blocks that satisfy the overlap constraints (above and left) within some error tolerance. Randomly pick one such block.
- Compute the error surface between the newly chosen block and the old blocks at the overlap region. Find the minimum cost path along this surface and make that the boundary of the new block. Paste the block onto the texture. Repeat.



source texture



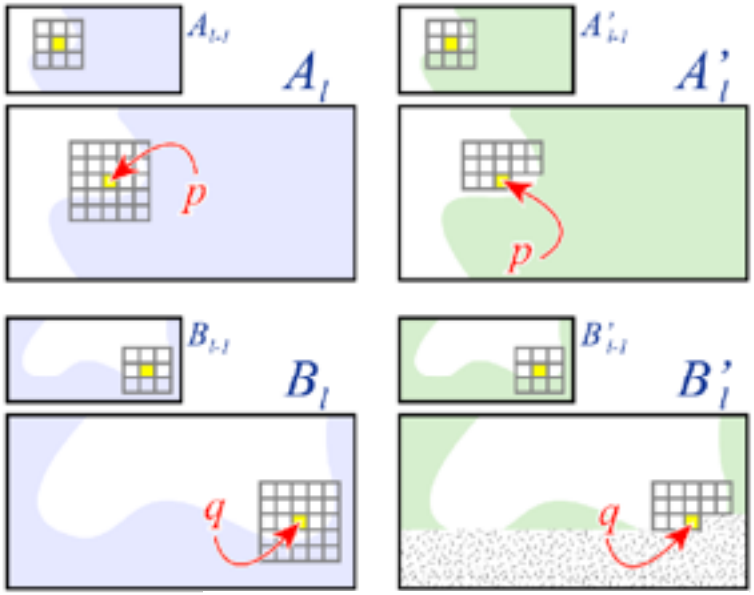
target image

correspondence maps

texture transfer result

Before Neural Networks

- Hertzman *et al.*, 2001
- Image analogies, *A is to A' as B is to "?"*



Input Analogy

Output

Early methods: so many downsides

- Exhaustive patch wise image searches
  - not suitable for any real time processing,
  - took tens of minutes and hours in early 2000's
- Analogy required existing style transfer examples
  - Typically brittle to new types of images
  - …and images without structures in the original analogy
- Research field was dormant for about a decade
- Until, 2016! Convolutional Neural Networks are found to have "Information Distillation Pipeline"

Neural Methods

- 2016 early: Gatys, Ecker, Bethge, Original Paper, Use CNNs instead of patch wise searches to separate style and content **Usually Best Results**
- 2016 mid: Johnson, Alahi, and Fei-Fei. Define loss through neural network as “perceptual” **Usually Fastest Results**
- 2016 late: Li and Wand: GANs for translating style, slightly better results than perceptual loss
- 2017: Lots of small improvement papers based on loss function and normalization tricks
- 2017 late: Li, Fang, Yang, Wang, Lu, and Yang,
  One shot style transfer: no training methods for transferring infinite styles
  Best Quality/Time Tradeoff,
  Generally Impressive that it Even Works!
- 2018 mid: Li, Liu, Li, Yang, Kautz: Photo realistic one shot transfer

Image Optimization Based Style Transfer

In the Beginning, there was Gatys…

- How to define content and style with CNNs?
- Use Pre-trained network like VGG.
- Content:

$$\mathscr{L}_c(I_c, I_{new}) = \sum_{l \in L_c} \lambda_l \cdot \|A_c^{(l)} - A_{new}^{(l)}\|^2$$

Content Layers

Vectorized Activations
(Conv Layer Outputs)

- Style:

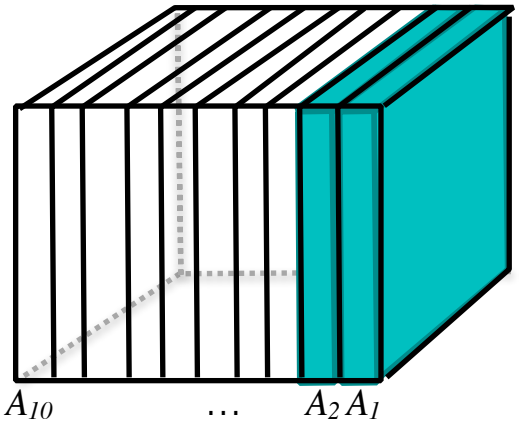
$$\mathscr{L}_s(I_s, I_{new}) = \sum_{l \in L_s} \beta_l \cdot \|G_s^{(l)} - G_{new}^{(l)}\|^2$$

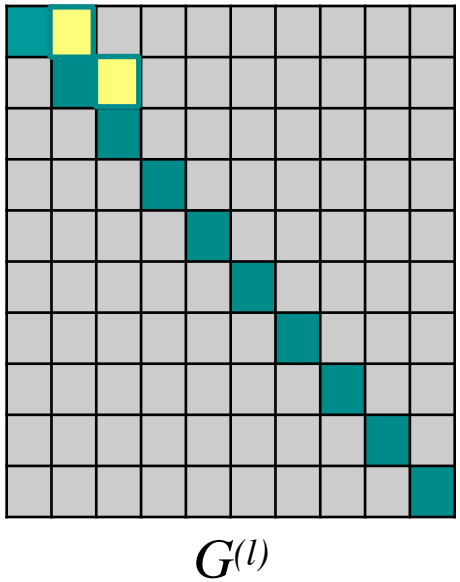
Grammian of Each
$(l)^{th}$ Activation Tensor

$$G_{i,j}^{(l)} = \sum A_i^{(l)} \cdot A_j^{(l)}$$

Into the Grammian… Inner Product

- Grammian is a square matrix, defining covariance among filter activations:





So this is the non-normalized "covariance" among the different filters, where spatial information is aggregated away.

We reduce "style" to the correlated responses of filters in a specific layer.

$$G_{i,j}^{(l)} = \sum A_i^{(l)} \cdot A_j^{(l)}$$

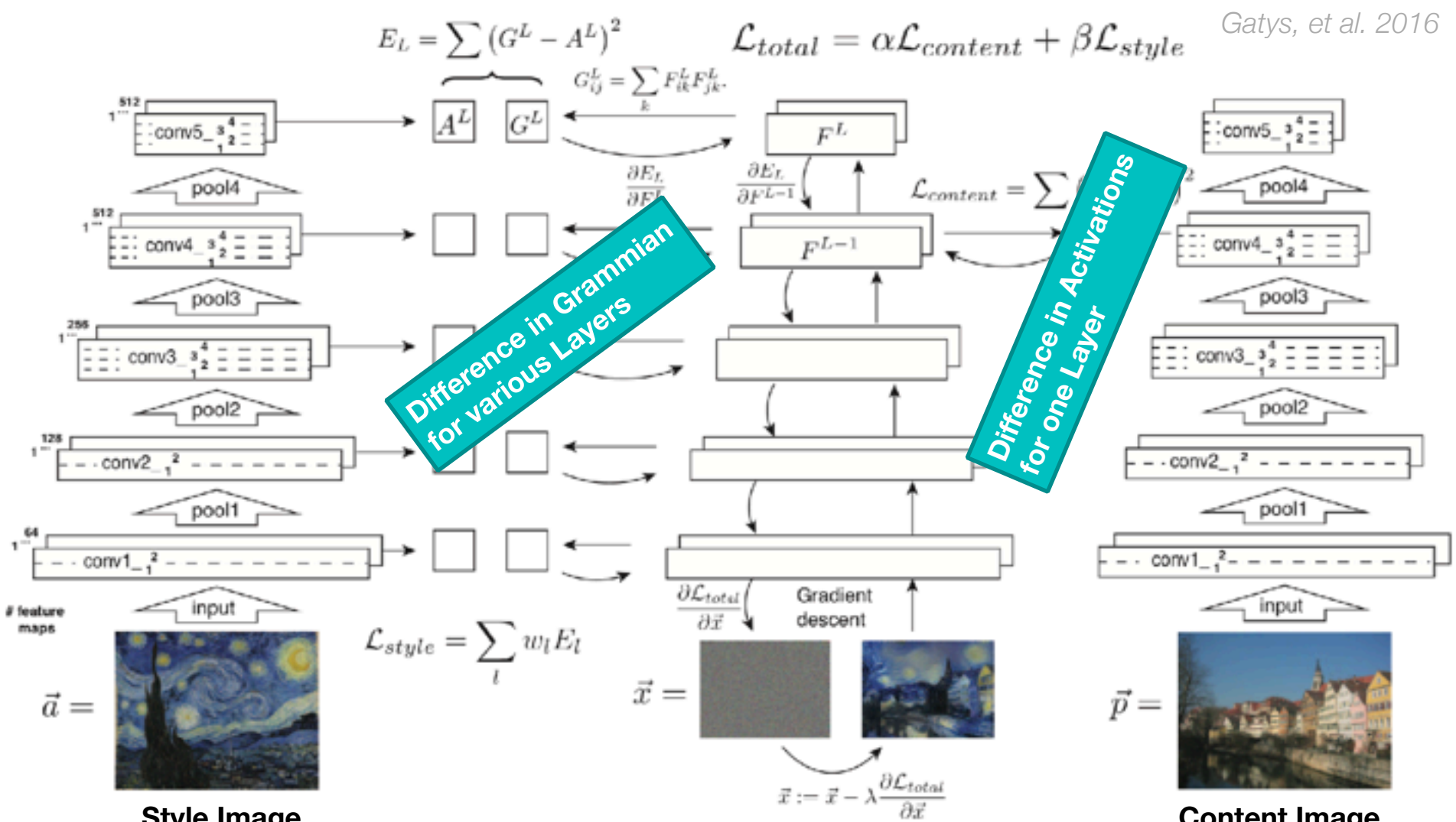
```
# Easy to Implement
Av =  A.reshape( (channels, rows*cols) )
G  =  Av @ Av.T
```


Is that really style?

- No.
- But, the vision system and brain are complex.
- The vision system does classify texture through correlated responses of cortical cells… So we are approximating correlation between neurons in the vision system…
  - …at best, this is an argument about inspiration, not explanatory
  - but it is independent of the content, or at least not completely dependent on the content…
- Or, in the words of Gatys when presenting the paper:
  "The Gram matrix encodes second order statistics of a set of filters. It sort of mushes up all the features at a given layer, tossing out spatial information in favor of a measure of how correlated the different features [are]."

Gatys's Procedure

Gatys, et al. 2016



Style Image into VGG

Iteratively Optimized Image

Content Image into VGG

The Loss Functions

$$\mathscr{L}_c(I_c, I_{new}) = \sum_{l \in L_c} \lambda_l \cdot \|A_c^{(l)} - A_{new}^{(l)}\|^2$$

Content Loss

$$\mathscr{L}_s(I_s, I_{new}) = \sum_{l \in L_s} \beta_l \cdot \|G_s^{(l)} - G_{new}^{(l)}\|^2$$

Style Loss

$$\mathscr{L}_{tv}(I_{new}) = \sum_{i,j} \|I_{i,j} - I_{i,j+1}\|^2 + \|I_{i,j} - I_{i+1,j}\|^2$$

Total Variation

Total Loss $$\mathscr{L}(I_c, I_s, I_{new}) = \alpha \cdot \mathscr{L}_c(I_c, I_{new}) + \beta \cdot \mathscr{L}_s(I_s, I_{new}) + \mathscr{L}_{tv}(I_{new})$$

- Hyperparameters:
  - alpha/beta ratio
  - I_{in} initialization method: I_c, I_s, *White Noise*
  - Layers to use in VGG

$$I_{new} \leftarrow I_{new} + \eta \nabla \mathscr{L}(I_c, I_s, I_{new})$$

Update Equation

Alpha Beta Ratio

10^{-4}

10^{-3}

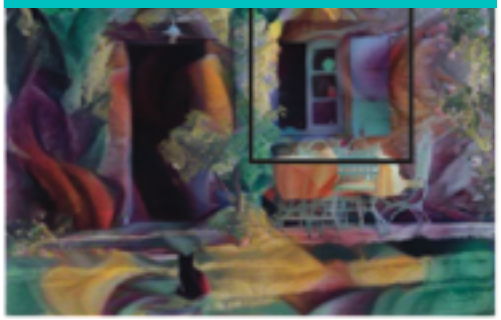
10^{-2}

10^{-1}

Gatys, et al. 2016

Layer Selection and Initialization

Content Loss: Convolutional Layer 2

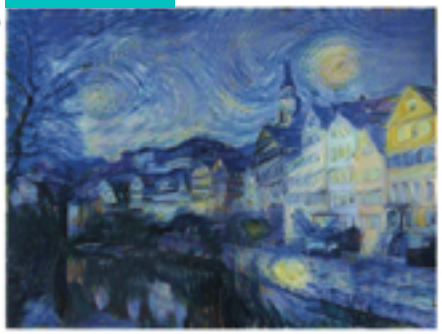


Content Loss: Convolutional Layer 4

Gatys, et al. 2016

Init Content

A

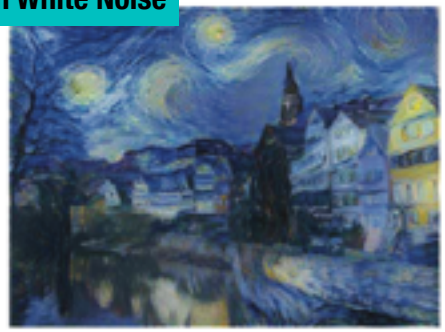


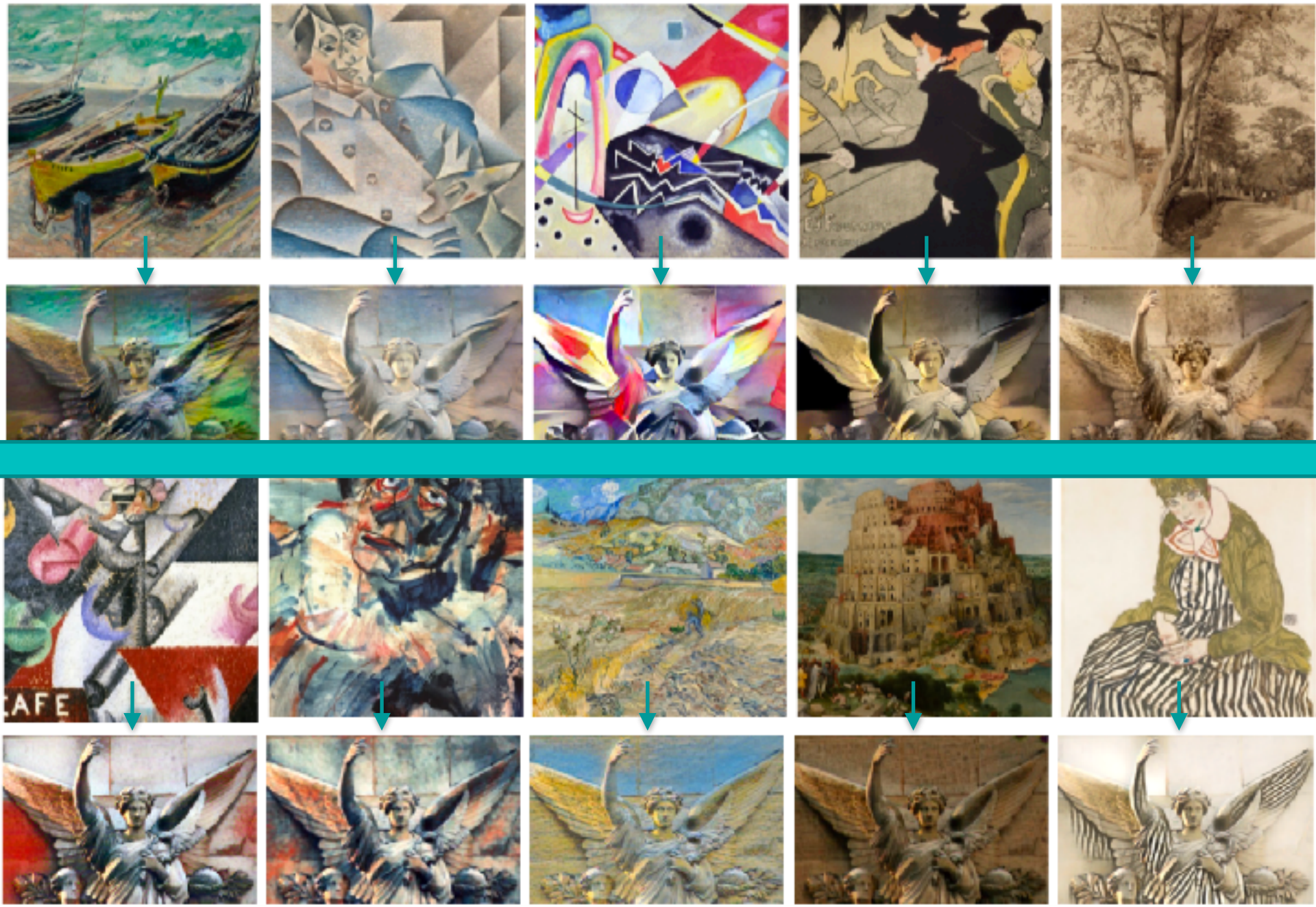
Init Style

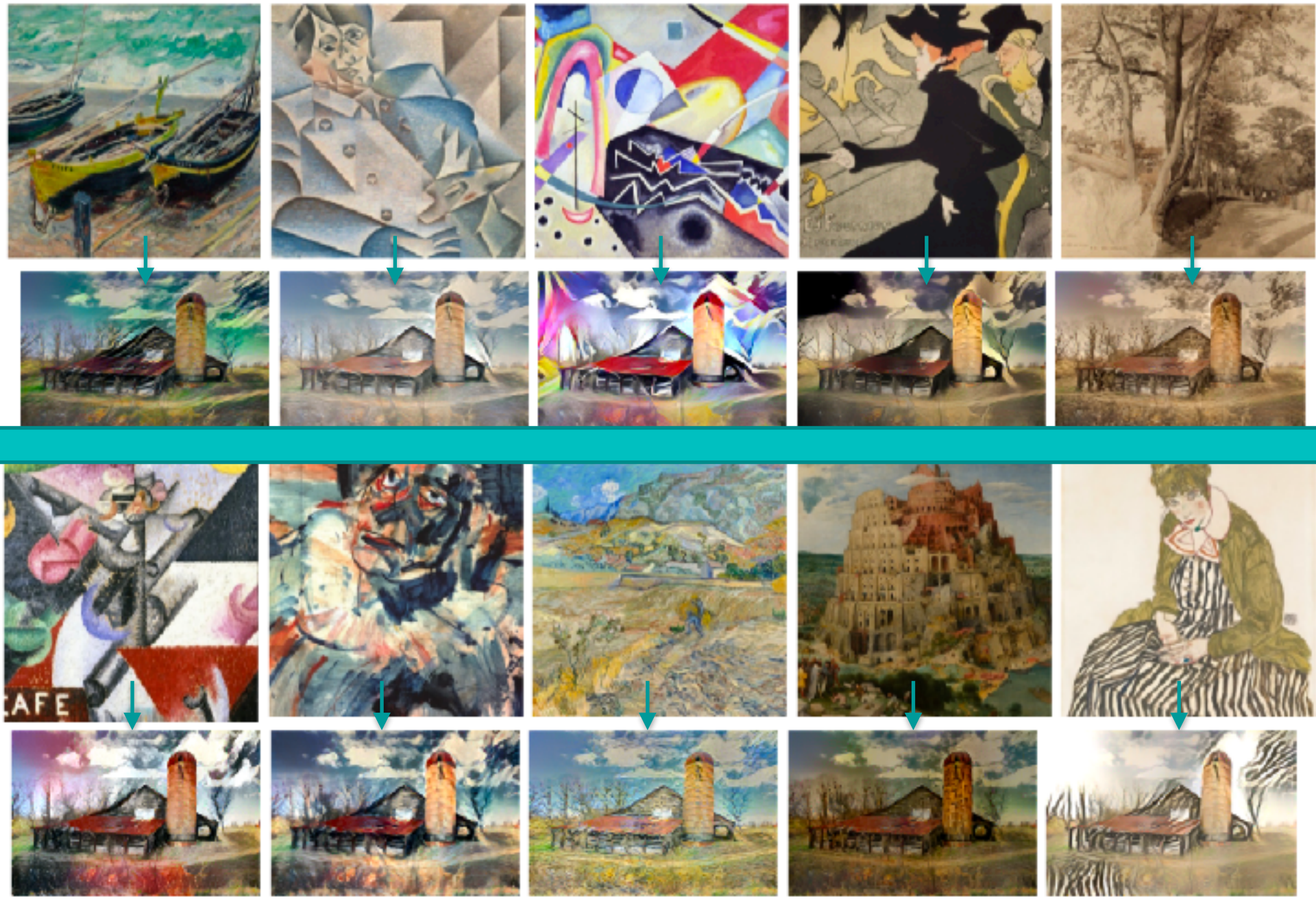
B

C

Init Random White Noise




AFE



Specifics of Implementation

- Uses basic tensorflow operations
  - GradientTape
  - Einsum
  - Loading from VGG
- Normalizes Style loss
  - sum / size^2 channels^2

Demo by Google

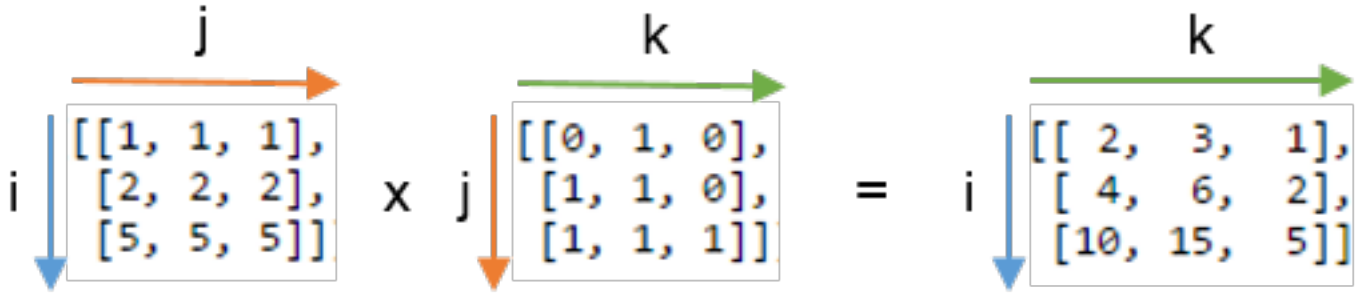
$$G_{i,j}^{(l)} = \sum A_i^{(l)} \cdot A_j^{(l)}$$

size squared

$$\sum_{l \in L_s} \beta_l \cdot \| G_s^{(l)} - G_{new}^{(l)} \|^2$$

channels squared

```
np.einsum('ij,jk->ik', A, B)
```



https://ajcr.net/Basic-guide-to-einsum/

Image Optimization Based Style Transfer

Gatys, et. al

Our master class repository:
05a GatysStyleTransfer.ipynb

Alternative Implementation:

Follow Along: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/8.3-neural-style-transfer.ipynb

Demo by Francois Chollet

Lecture Notes for

Neural Networks and Machine Learning

Style Transfer: Image Opt.

Next Time:
Model Opt. and One Shot

Reading: None