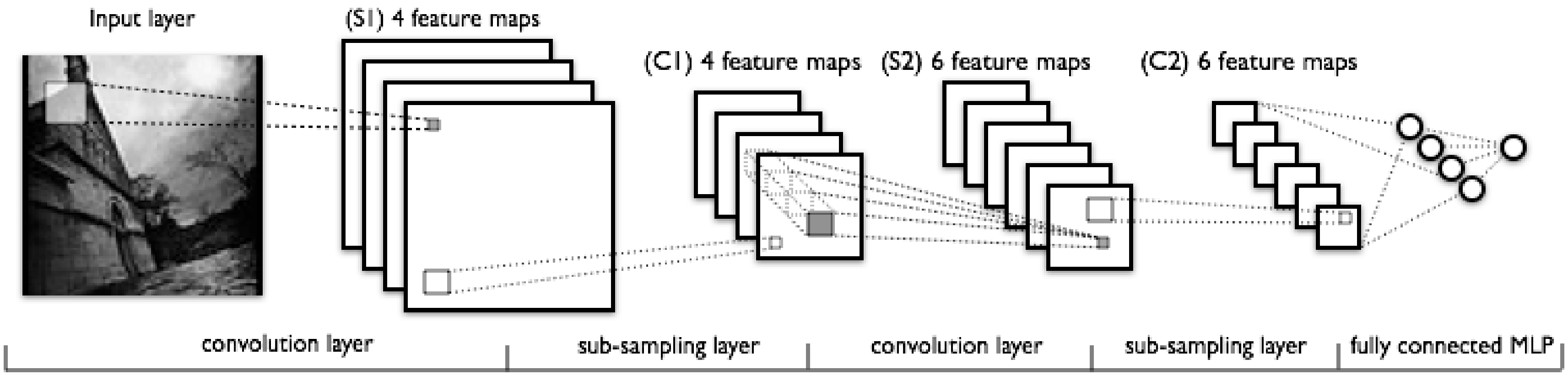
More ConvNets and Transfer Learning

SML310: Research Projects in Data Science, Fall 2019

Michael Guerzhoy

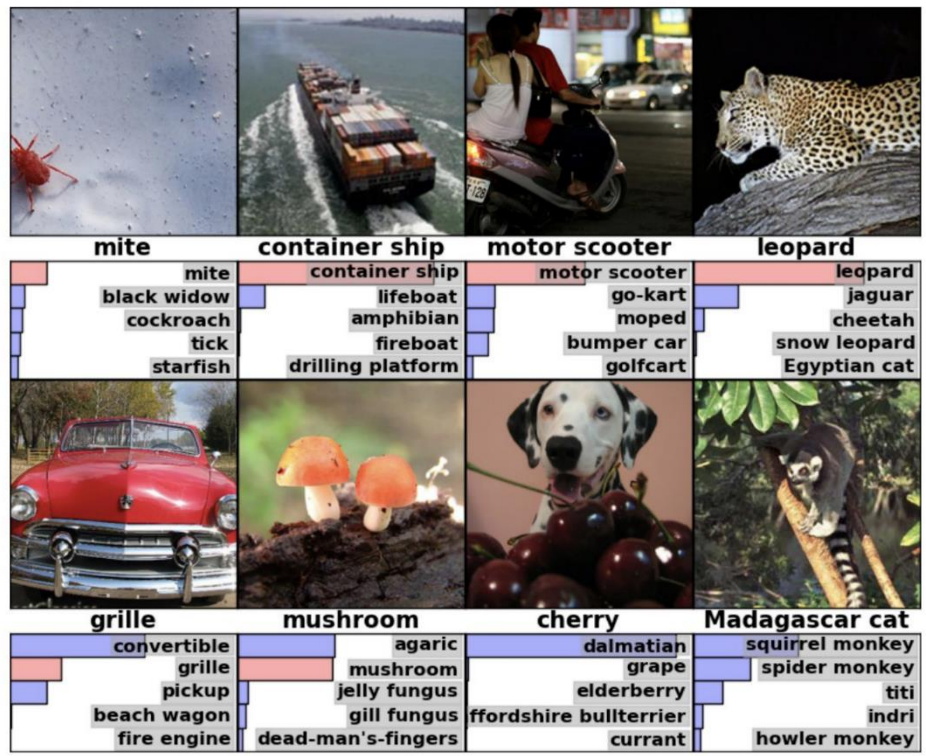
ConvNets


Input layer
(S1) 4 feature maps
(C1) 4 feature maps
(S2) 6 feature maps
(C2) 6 feature maps
convolution layer
sub-sampling layer
convolution layer
sub-sampling layer
fully connected MLP

ImageNet

ImageNet Challenge

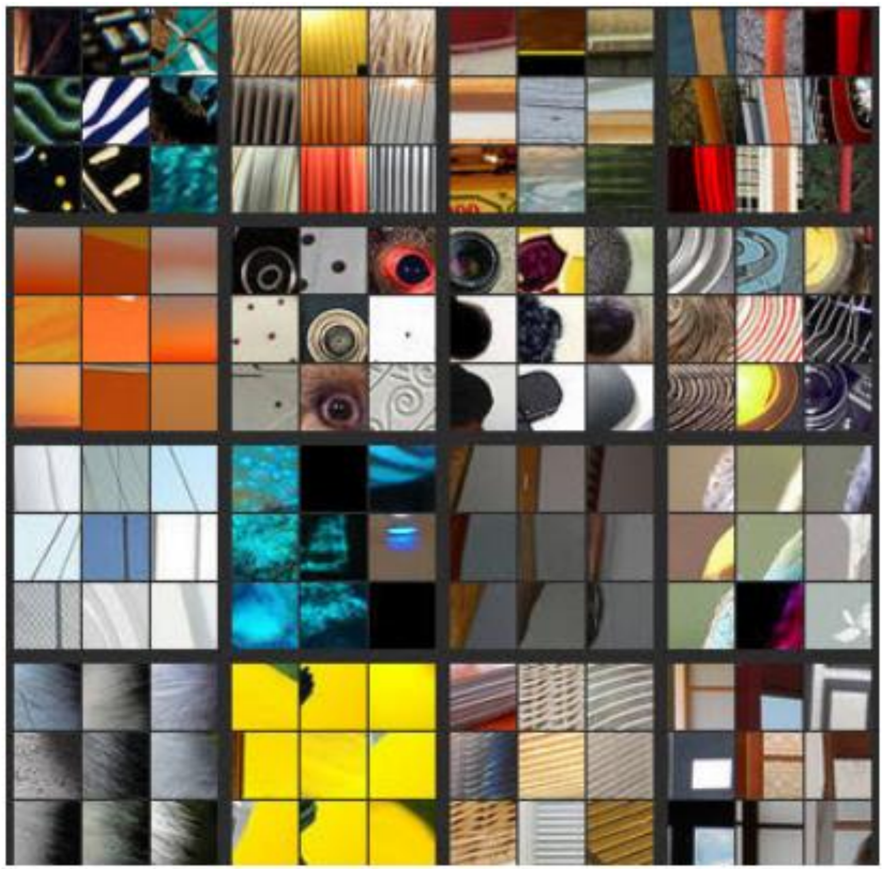
IMAGENET



- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Neurons of a ConvNet trained on ImageNet

Units in Layer 3



Matthew Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks” (ECCV 2014)

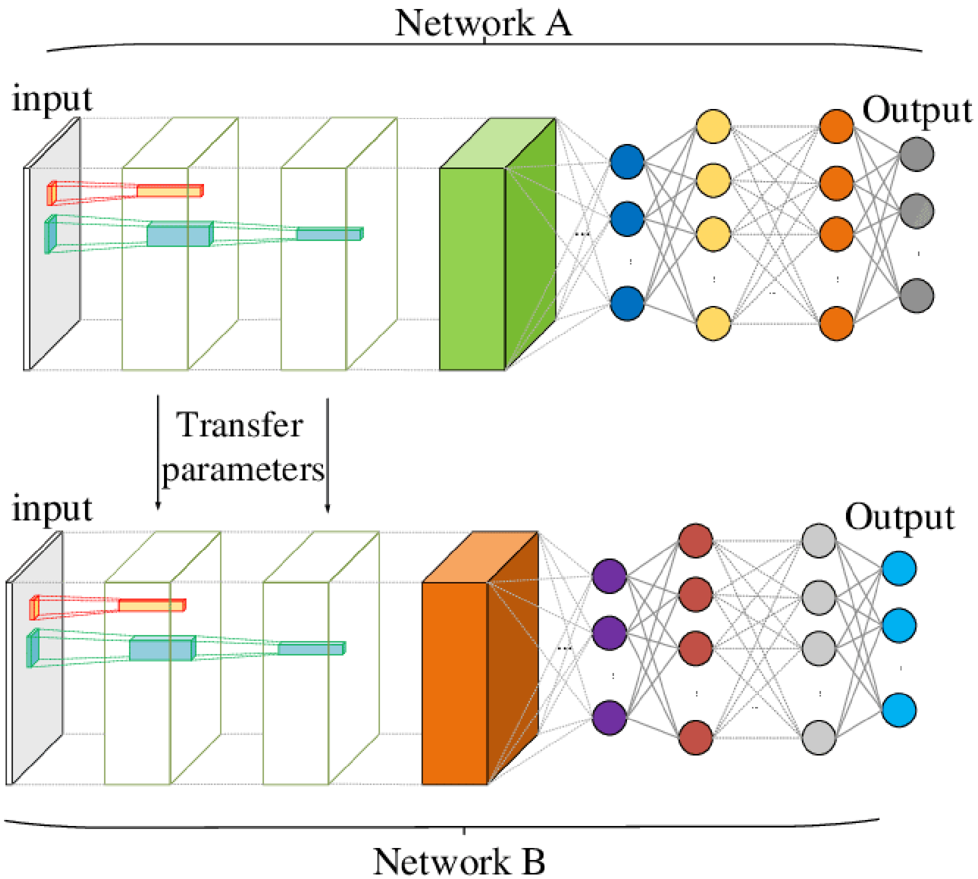
Units in Layer 4

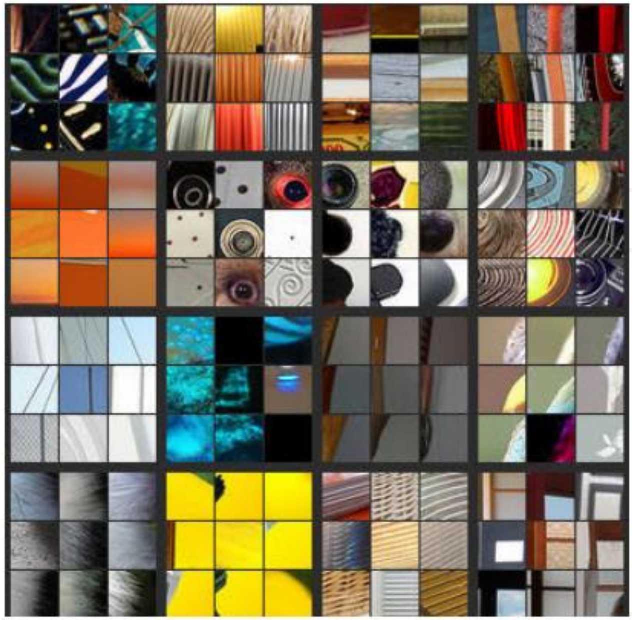
Matthew Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks” (ECCV 2014)

Units in Layer 5

Matthew Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks” (ECCV 2014)

Transfer Learning





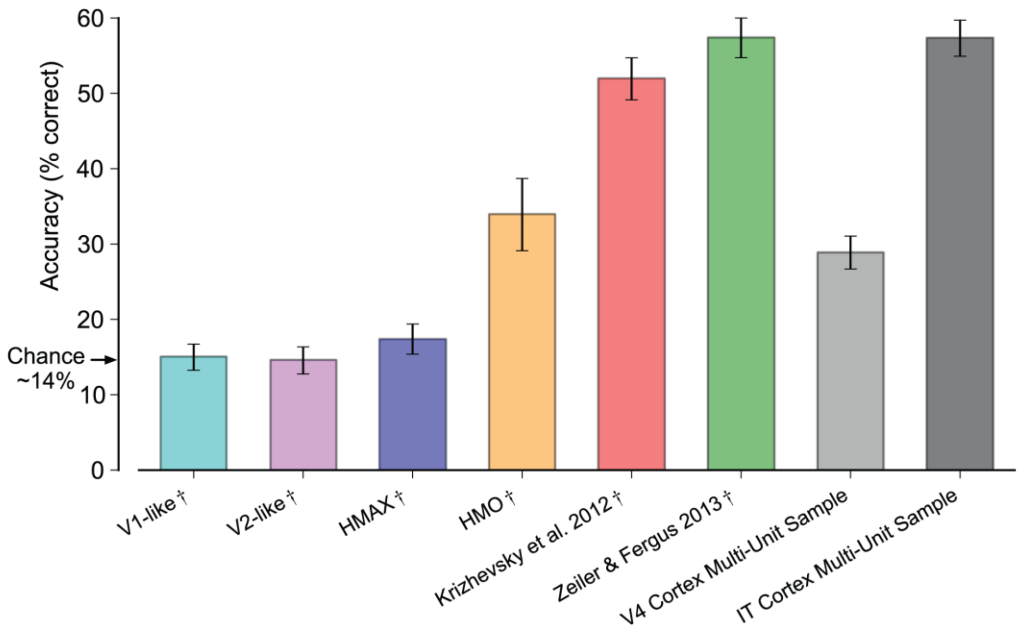
Images that activate neurons in layer 3

Idea: can use the weights that detect basic image features for a *different* task

Transfer Learning

- Useful when the training set we have is small
  - Leverage the size e.g. ImageNet (or a large linguistic corpus)
  - If we trained the network from scratch, it would overfit on a small training set

Transfer Learning from Monkey Brains



Cadieu et al, **Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition**

Idea: show monkeys images and record the neural activity in their inferior temporal cortex. Classify images based on the neural activity in the IT cortex
Worked about as well as the state of the art in transfer learning in 2014. (Not as well as the SOTA in 2019.)

Optimizing the Input

- So far, we trained the network by modifying the weights so that we get the outputs y that we want when inputting x
- Could vary the *input* in order to get the output we want

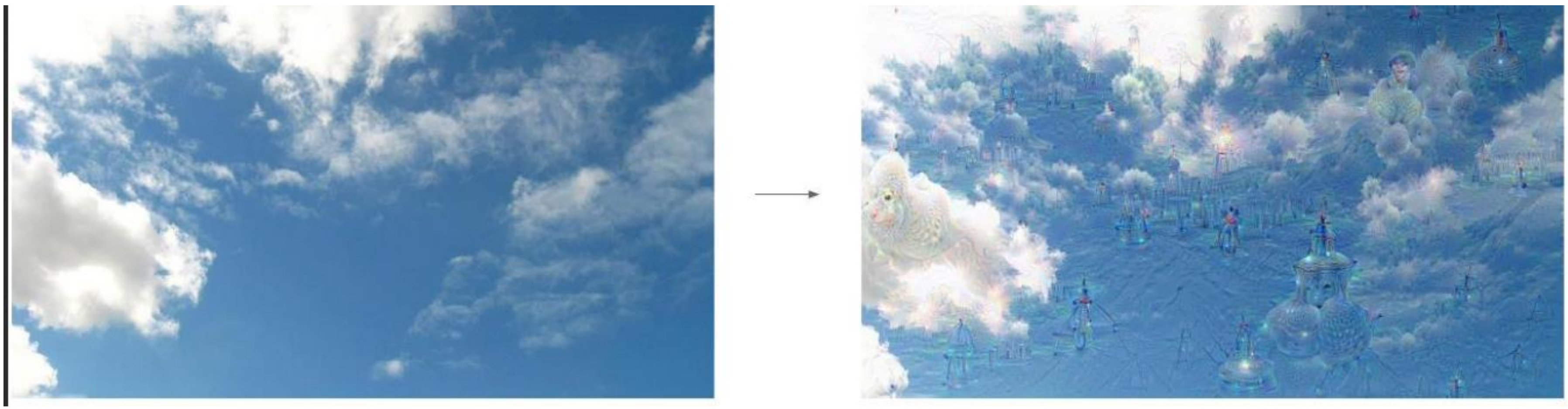
Deep Dream

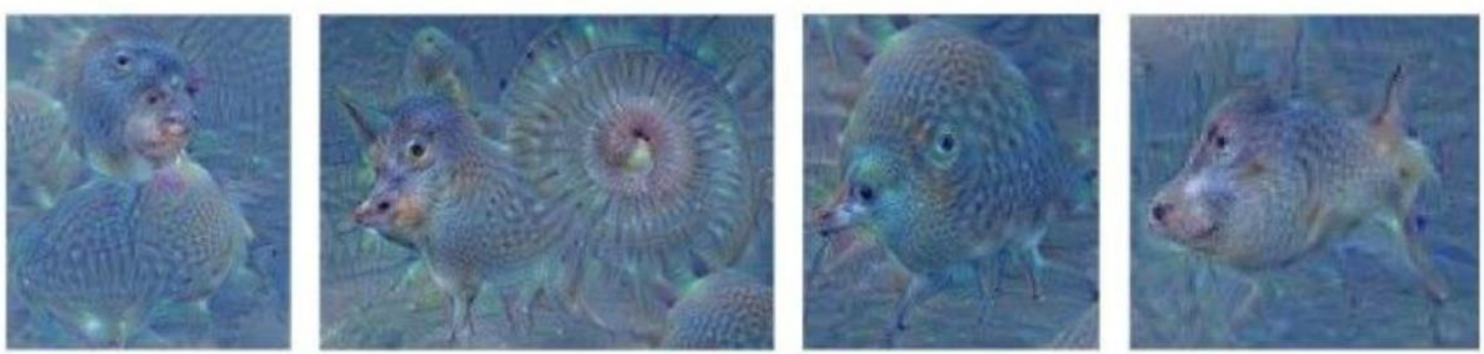
Deep Dream

- Start with a random input x, and apply a ConvNet trained on ImageNet to it
- We'll get something like

Poodle	Car	Hay	Squirrel	…	…	…
0.01	0.015	0.0001	0.02	…	…	…

- Now, adjust the input x so that it looks more like the object that it already looks the most like (i.e., try changing x a little bit so that P(squirrel) increases)
- Repeat
- Can do this with intermediate layers: modify the input x so that the units that are already highly activated are activated even more



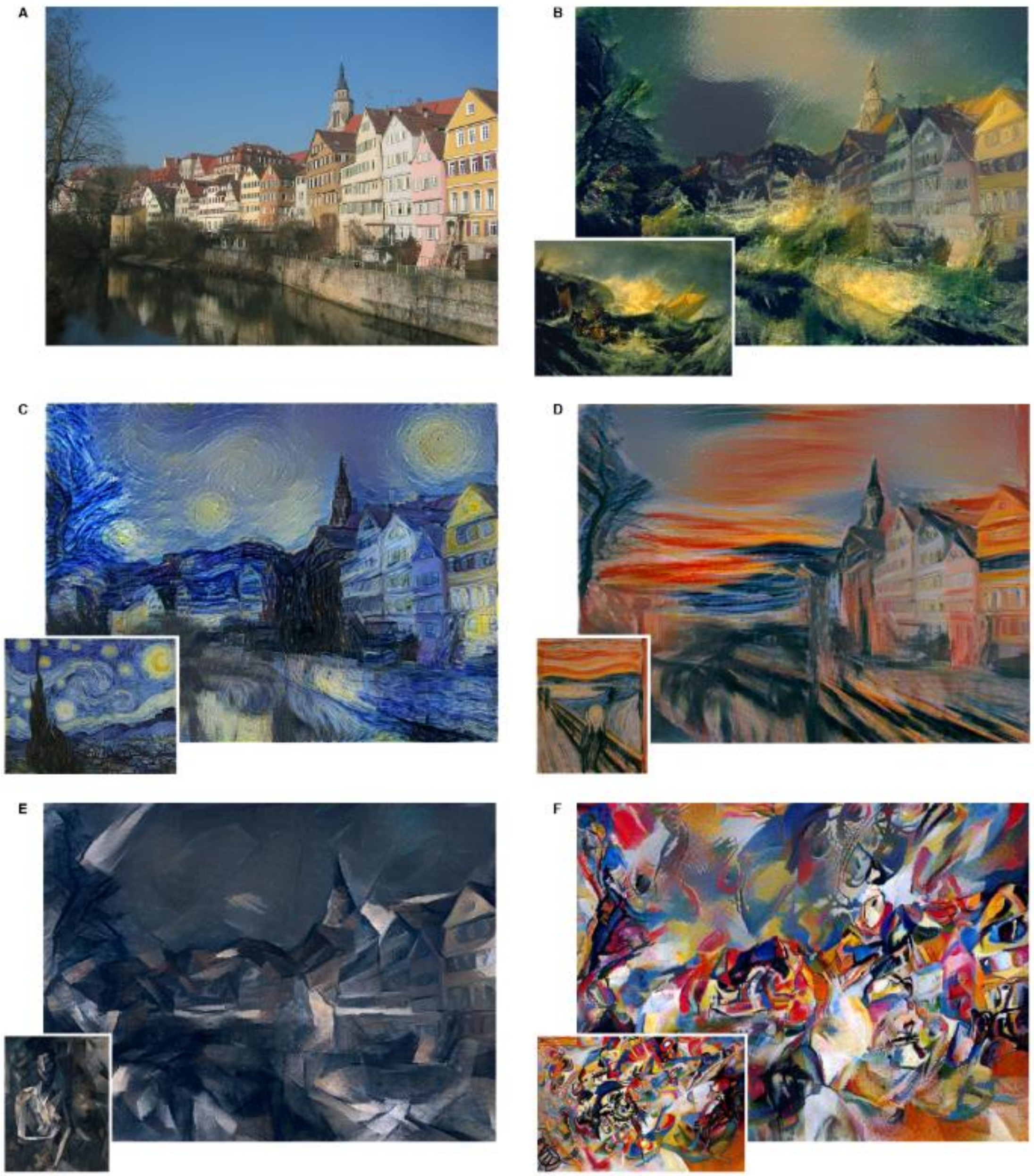


"Admiral Dog!" "The Pig-Snail" "The Camel-Bird" "The Dog-Fish"

Deep Dream

- A result of using gradient descent to alter x (rather than w) to increase the outputs that are already high
- The network weights are fixed, and we are adjusting the input

Style Transfer


A
B
C
D
E
F

- Want an image that matches the photo in *content* and the painting in *style*
- Start with a random image, and change it with gradient descent to simultaneously match the photo in content and the painting in style

Cost function: content

- To keep the content close to the photo I, make sure that $|x - I|$ is small
  - OK, but the Euclidean distance is not the best at being small when the content is similar (because the image could shift, or be brighter, etc.)
- Better: make the activations in all the different layers for the original image and for x stay the same: minimize

$$L_{content}(x, I) = \sum \ (F(x)^l_{i,j} - F(I)^l_{i,j})^2$$

  - $F(y)^l_{(i,j)}$ is the activation at layer I at location j in feature map I, for input y

Cost function: Style

- Define the Gram matrix at layer *l* as

$$G_{i,j}^{l}(y) = \sum_k F(y)_{ik}^{l} F(y)_{jk}^{l}$$

- Discovery: images that have similar Gram matrices tend to look like they have similar style
- Difference in Gram matrices at layer *l* between image x and image P*:*

Square layer size

  - $E_l(x, P) = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{i,j}^{l}(x) - G_{i,j}^{l}(P) \right)^2$
  - $L_{style}(x, P) = \sum_l w_l E_l(x, P)$

Cost function: overall

- $cost(x, I, P) = \alpha L_{style}(x, P) + \beta L_{content}(x, I)$
- Adjust x to minimize the cost: try to keep both $L_{style}(x, P)$ and $L_{content}(x, I)$ small

The intermediate layers of deep neural networks contain lots of info!

- The semantic content of images
- The semantic content of sentences
  - Even language-independent semantic content