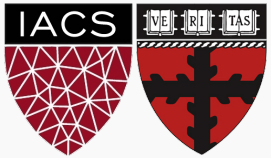
Advanced Section #3: Visualization of convolutional networks and neural style transfer

AC 209B: Data Science

Javier Zazo Pavlos Protopapas



Lecture Outline

Visualizing convolutional networks

Image reconstruction

Texture synthesis

Neural style transfer

DeepDream

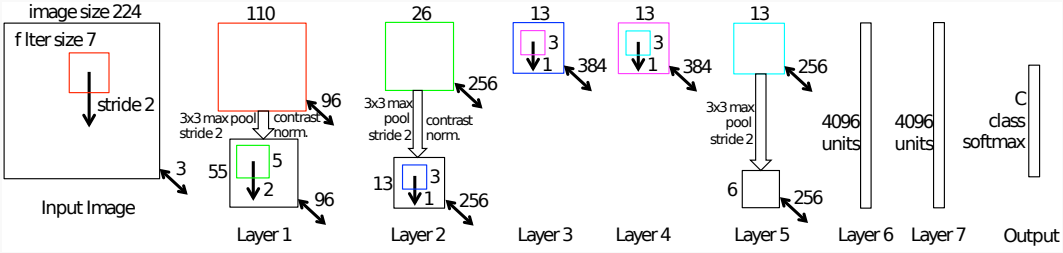
Visualizing convolutional networks

Motivation for visualization

- When studying NN we have little insight about what the network is actually learning and the internal operations.
- Through visualization we may
  - understand how the input stimuli excites the individual feature maps.
  - observe the evolution of features and diagnose potential problems during training.
  - help us make more substantiated designs, rather than simply building models through trial and error.
- All in all, improve general performance if we can address all of these matters.

Architecture

- Architecture similar to AlexNet, i.e., [1]
  - Trained network on the ImageNet 2012 training database for 1000 classes.
  - Input are images of size $256 \times 256 \times 3$.
  - Uses convolutional layers, max-pooling and fully connected layers at the end.



[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

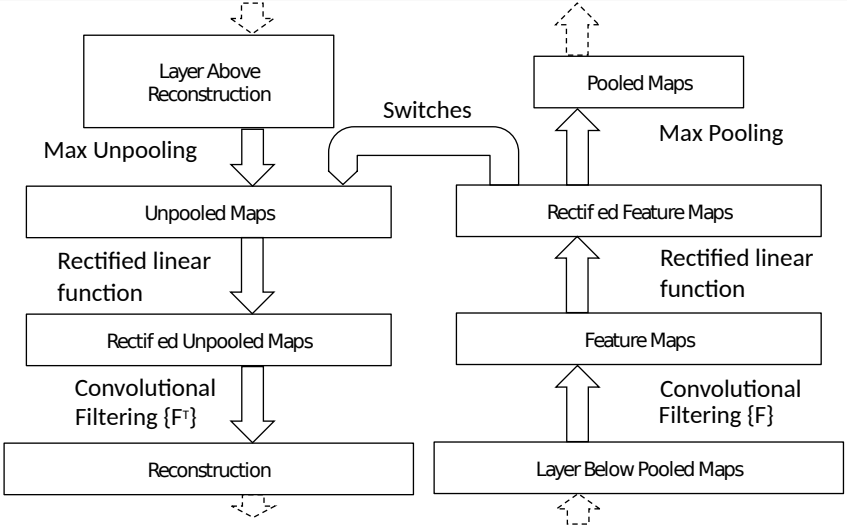
[2] Matthew D. Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision*. 2014, pp. 818–833, Springer.

Deconvolutional network

- For visualization, the authors employ a ***deconvolutional network***.
- The objective is to project the hidden feature maps into the original input space.
  - A common alternative is to visualize the activation functions of a specific filter.
- The name "deconvolutional" network may be unfortunate, since the network does not perform any deconvolutions.

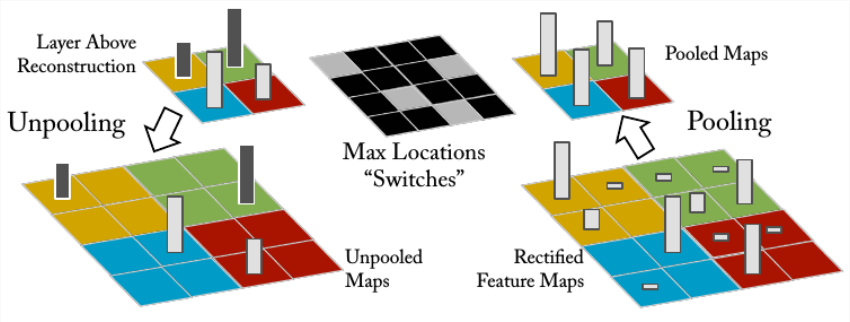
Matthew D Zeiler, Graham W Taylor, and Rob Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2018–2025

Devonvolutional network structure



Devonvolutional network description

- **Unpooling:**
  - The max-pooling operation is non-invertible.
  - Switch variables: record the locations of maxima.
  - It places the reconstructed features into the recorded locations.



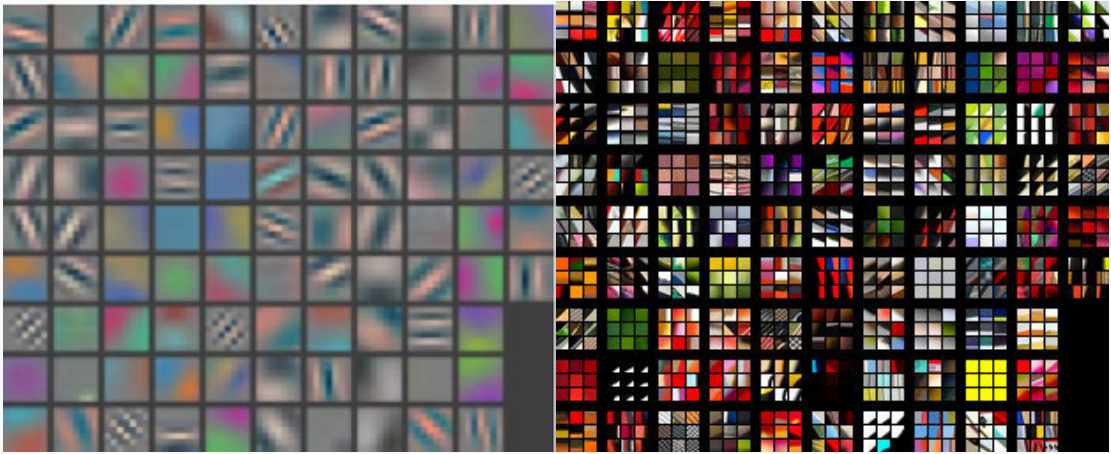
Devonvolutional network description

- **Rectification:** To obtain a valid reconstructed signal at each layer (which should be positive), the signals go through a ReLu operation.
- **Filtering:**
  - The deconvnet uses a transposed convolution of the learned filters from the convnet.
  - In practice, the filters have to be flipped horizontally and vertically, but care has to be taken if padding or stride was used.
  - Neural network frameworks such as tensorflow and others implement the transposed convolution efficiently.
- The purpose of the transposed convolution is to project the feature maps computed by the convnet back to input space.
- The transposed convolution corresponds to the backpropagation gradient computation of convolutional networks (an analogy from MLPs).

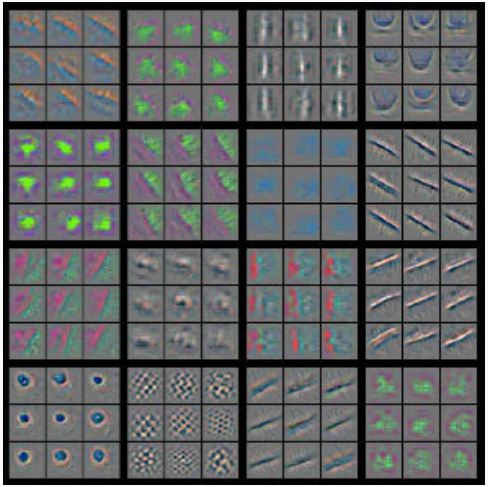
Feature visualization

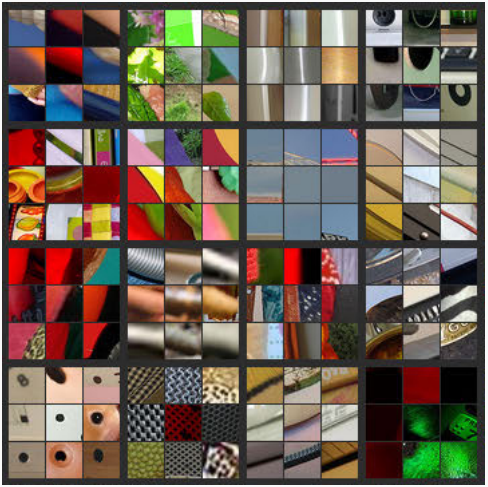
- To visualize the features that activate a specific neuron, the authors evaluate the validation database on the trained network.
- Record the nine highest activation values of each neuron's output.
- Then, project the recorded 9 outputs into input space for every neuron.
  - When projecting, all other activation units in the given layer are set to zero.
  - This operation ensures we only observe the gradient of a single neuron.
  - ***Switch variables*** are used in the unpooling layers

First layer of Alexnet

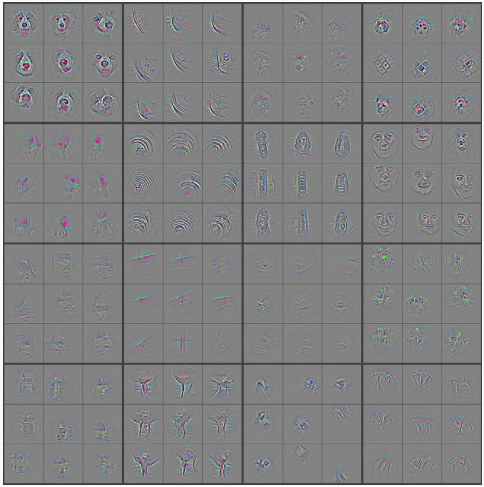


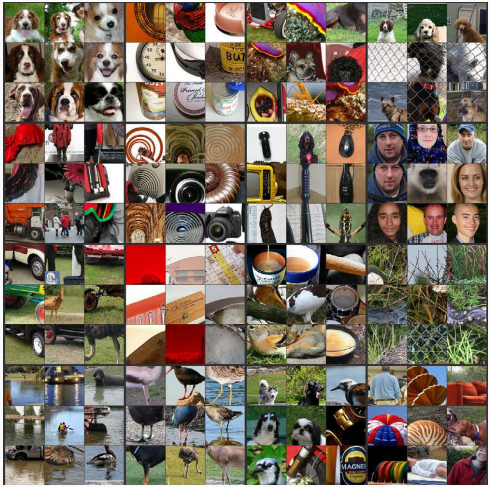
Second layer of Alexnet



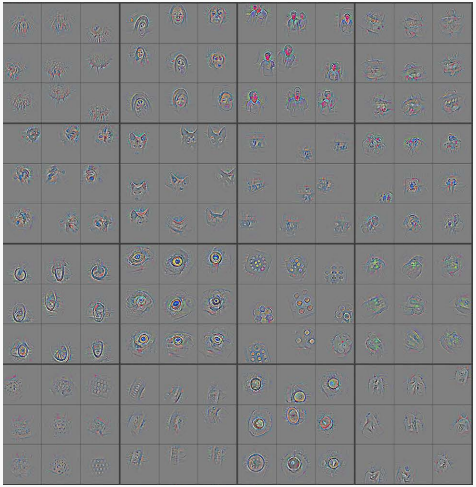


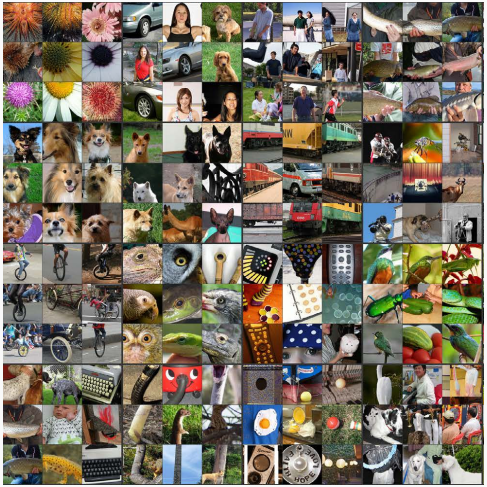
Fourth layer of Alexnet





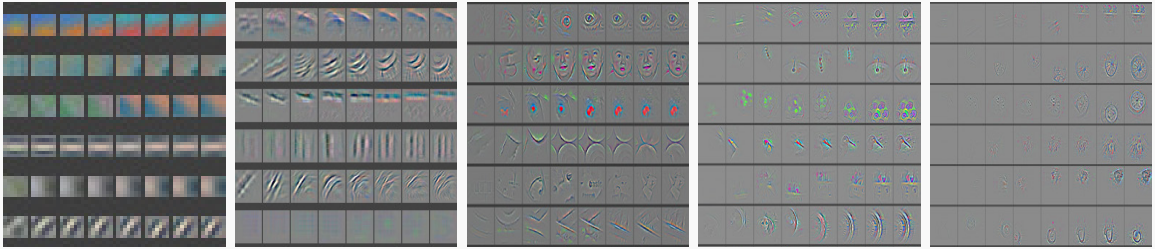
Fifth layer of Alexnet





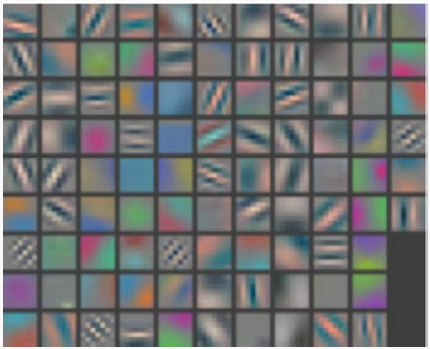
Feature evolution during training

- Evolution of features for 1, 2, 5, 10, 20, 30, 40 and 64 epochs.
- Strongest activation response for some random neurons at all 5 layers.
- Low layers converge soon after a few single passes.
- Fifth layer does not converge until a very large number of epochs.
- Lower layers may change their feature correspondence after converge.



Architecture comparison

- Check if different architectures respond similarly or more strongly to the same inputs.
- Left picture used filters 7×7 instead of 11×11, and reduced the stride from 4 to 2.
- Evidence that there are less dead units on the modified network.
- More defined features, whereas Alexnet has more aliasing effects.



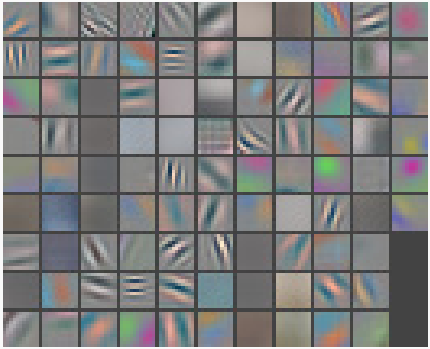


Image reconstruction

Image reconstruction

- Reconstruction of an image with latent features, or encoding.
- Layers in the network retain an accurate photographical representation about the image, retaining geometric and photometric invariance.
- Assume $a^{[l]}$ corresponds to the latent representation of layer l for some input image $\mathbf{x}$, for some mapping $\Phi^{[l]}(\mathbf{x}) = a^{[l](C)}$.
- Solve the optimization problem:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{y}} J^{[l](C)}(\mathbf{x}, \mathbf{y}) + \lambda R(\mathbf{y}),$$

where

$$J_C^{[l]}(\mathbf{x}, \mathbf{y}) = \left\| \Phi^{[l]}(\mathbf{y}) - \Phi^{[l]}(\mathbf{x}) \right\|_{\mathcal{F}}^2 = \left\| a^{[l](G)} - a^{[l](C)} \right\|_{\mathcal{F}}^2.$$

Aravindh Mahendran and Andrea Vedaldi, "Understanding deep image representations by inverting them," Nov. 2014

Regularization and optimization

- Regularization
  - Use a combination of α-norm regularizer,

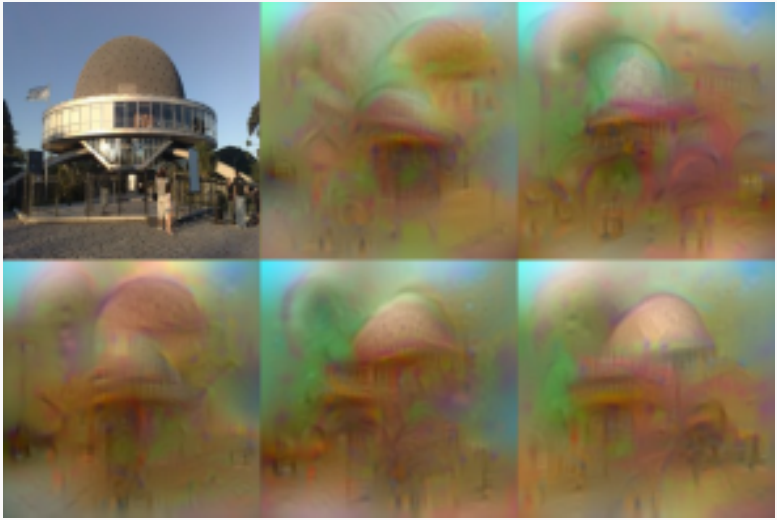
$$R_\alpha(\mathbf{y}) = \lambda_\alpha \|\mathbf{y}\|_\alpha^\alpha$$

  - and a ***total variation*** regularizer:

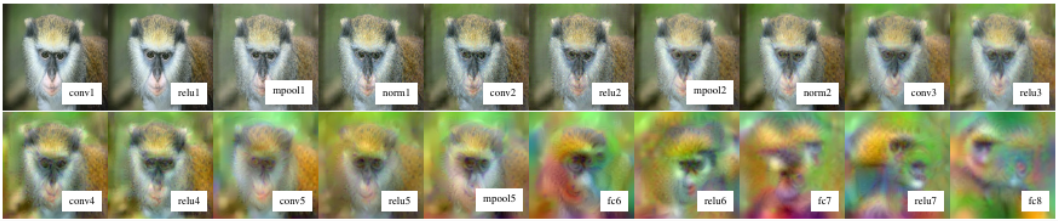
$$R_{V_\beta}(\mathbf{y}) = \lambda_{V_\beta} \sum_{i,j,k} \left(\left(a_{i,j+1,k}^{[l](G)} - a_{i,j,k}^{[l](C)} \right)^2 + \left(a_{i+1,j,k}^{[l](G)} - a_{i,j,k}^{[l](C)} \right)^2 \right)^{\beta/2}.$$

- Image reconstruction:
  1. Initialize $\mathbf{y}$ with random noise.
  2. Compute gradients of the cost and backpropagate to input space.
  3. Update generated image G with a gradient step.

Example of image reconstruction



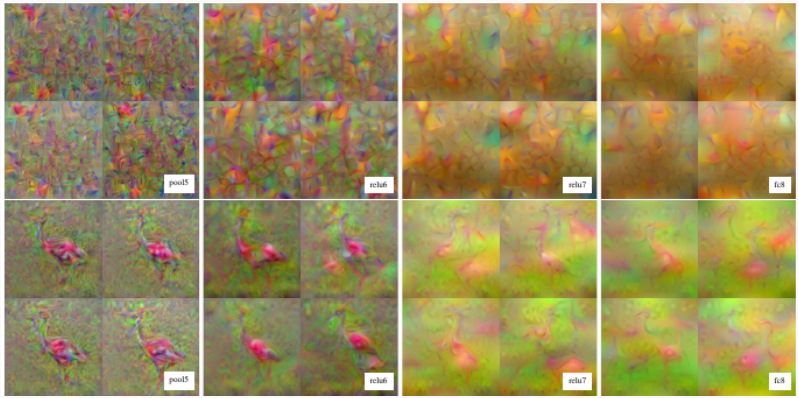
Example of image reconstruction



λ_{V^β}	1 conv1	2 relu1	3 pool1	4 norm1	5 conv2	6 relu2	7 pool2	8 norm2	9 conv3	10 relu3	11 conv4	12 relu4	13 conv5	14 relu5	15 pool5	16 fc6	17 relu6	18 fc7	19 relu7	20 fc8
λ_1	**10.0** ±5.0	**11.3** ±5.5	**21.9** ±9.2	**20.3** ±5.0	**12.4** ±3.1	**12.9** ±5.3	15.5 ±4.7	15.9 ±4.6	14.5 ±4.7	16.5 ±5.3	14.9 ±3.8	13.8 ±3.8	12.6 ±2.8	15.6 ±5.1	16.6 ±4.6	12.4 ±3.5	15.8 ±4.5	12.8 ±6.4	10.5 ±1.9	5.3 ±1.1
λ_2	20.2 ±9.3	22.4 ±10.3	30.3 ±13.6	28.2 ±7.6	20.0 ±4.9	17.4 ±5.0	**18.2** ±5.5	**18.4** ±5.0	**14.4** ±3.6	**15.1** ±3.3	**13.3** ±2.6	**14.0** ±2.8	15.4 ±2.7	13.9 ±3.2	15.5 ±3.5	14.2 ±3.7	13.7 ±3.1	15.4 ±10.3	10.8 ±1.6	5.9 ±0.9
λ_3	40.8 ±17.0	45.2 ±18.7	54.1 ±22.7	48.1 ±11.8	39.7 ±9.1	32.8 ±7.7	32.7 ±8.0	32.4 ±7.0	25.6 ±5.6	26.9 ±5.2	23.3 ±4.1	23.9 ±4.6	**25.7** ±4.3	**20.1** ±4.3	**19.0** ±4.3	**18.6** ±4.9	**18.7** ±3.8	**17.1** ±3.4	**15.5** ±2.1	**8.5** ±1.3

Deep layer invariances

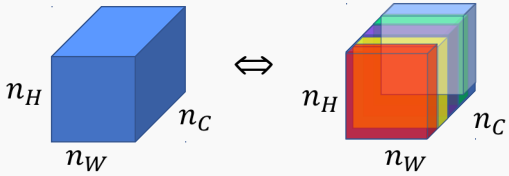
- At deeper layers, feature maps capture more object deformations, at different scales and positions.
- This corresponds to a more abstract and less precise representation of the image. The CNN captures a sketch of the original image.



Texture synthesis

Texture synthesis using convnets

- Generate high perceptual quality images that imitate a given texture.
- Uses a trained convolutional network (such as VGG) for object classification.
- Employs the correlation of features among layers as a generative process to obtain new textures.
- Output of a layer:



Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "Texture synthesis using convolutional neural networks".

Cross-correlation of feature maps: Gram matrices

- Denote the output of a given filter k at layer l with $a^{[l]}_{ijk}$.
- Indexes i and j refer to the spatial latent features, and k to channel.
- The cross-correlation between this output and a different channel k':

$$G^{[l]}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l]}_{ijk} a^{[l]}_{ijk'}.$$

- The Gram matrix:

$$G^{[l]} = A^{[l]}(A^{[l]})^T$$

where $(A^{[l]})^T = (a^{[l]}_{::1}, \ldots, a^{[l]}_{::n_C^{[l]}})$.

Generating new textures

- To create a new texture, we synthesize an image that has similar correlation as the one we want to reproduce.
- $G^{[l](S)}$ refers to the Gram matrix of the ***style*** image, and $G^{[l](G)}$ to the newly ***generated*** image.

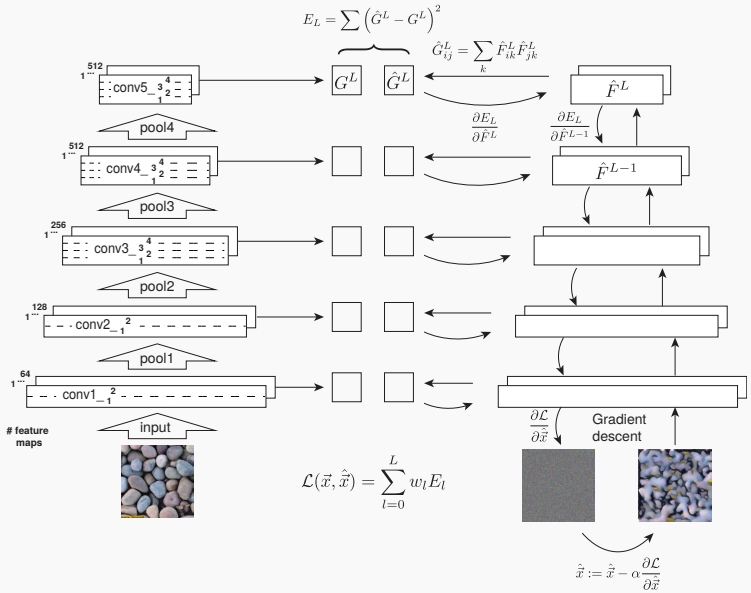
$$J_S^{[l]}(G^{[l](S)}, G^{[l](G)}) = \frac{1}{4(n_W^{[l]} n_H^{[l]})^2} \left\| G^{[l](S)} - G^{[l](G)} \right\|_{\mathcal{F}}^2,$$

  where $\|G\|_{\mathcal{F}} = \sqrt{\sum_{ij}(g_{ij})^2}$ corresponds to the Frobenius norm.
- We combine all of the layer losses into a global cost function:

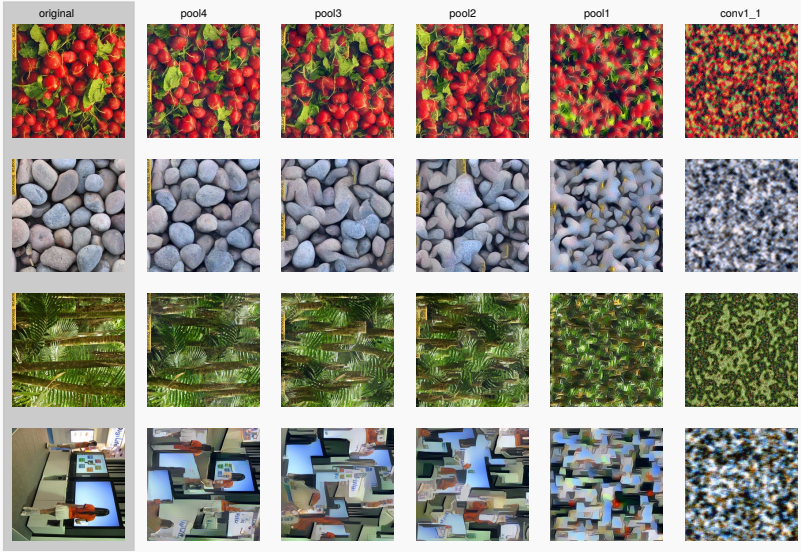
$$J_S(\mathbf{x}, \mathbf{y}) = \sum_{l=0}^{L} \lambda_l J_S^{[l]}(G^{[l](S)}, G^{[l](G)}),$$

  for given weights $\lambda_1, \ldots, \lambda_L$:

Process description



Texture examples



Neural style transfer

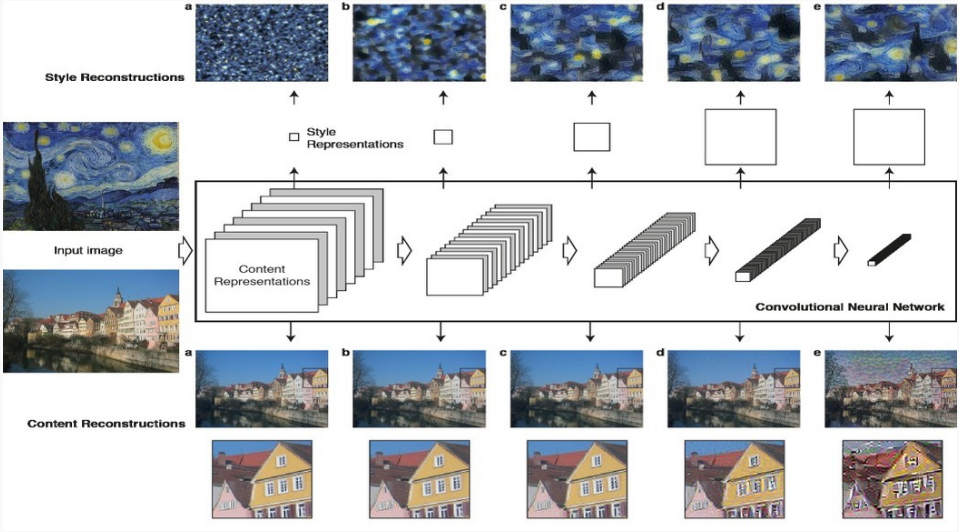
Neural style transfer

- It is the artistic generation of high perceptual quality images that combine the style or texture of some input image, and the elements or content from a different one.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "A neural algorithm of artistic style," Aug. 2015.

Other examples

Methodology



Objective function

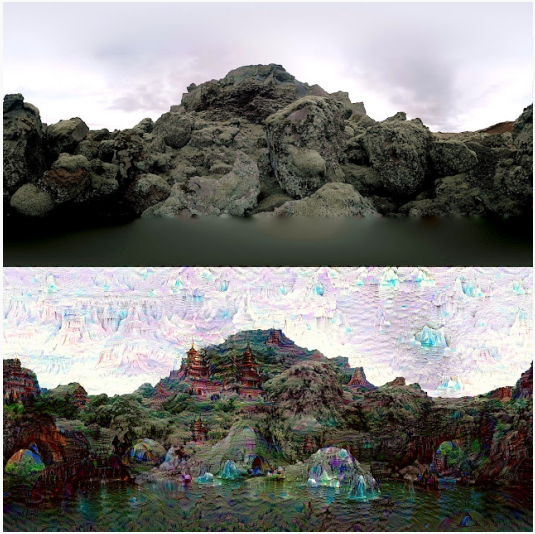
- Neural style transfer combines content reconstruction and style resemblance.

$$J_{\text{total}}(\mathbf{x}, \mathbf{y}) = \alpha J_C^{[l]}(\mathbf{x}, \mathbf{y}) + \beta J_S(\mathbf{x}, \mathbf{y})$$

- Need to choose a layer to represent content.
  - middle layers are recommended (not too shallow, not too deep) for best results.
- A set of layers to represent style.
- Combined cost is minimized using gradient descent or any other method typical of neural networks combined with backpropagation.
- The input $\mathbf{y}$ is initialized with random noise.
- Replacing the max-pooling layers with average pooling improves the gradient flow, and this produces more appealing pictures.

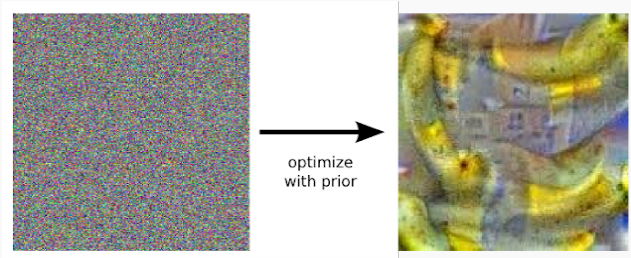
DeepDream

Art from visualization techniques



Inceptionism: Going Deeper into Neural Networks

- Discriminative trained network for classification.
  - First layer maybe looks for edges or corners.
  - Intermediate layers interpret the basic features to look for overall shapes or components, like a door or a leaf.
  - Final layers assemble those into complete interpretations: trees, buildings, etc.
- Turn NN upside down: what sort of image would result in Banana.
  - need to add texture information (prior).

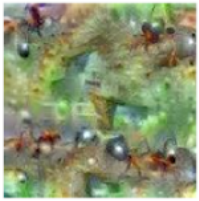


https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Class generation


Hartebeest


Measuring Cup


Ant


Starfish


Anemone Fish


Banana


Parachute


Screw

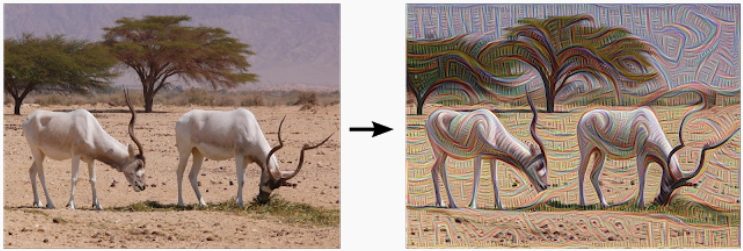
Visualizing mistakes

- Generating dumbbells always pictures them with an arm:

- The network failed to completely distill the essence of a dumbbell.
- Visualization can help us correct these kinds of training mishaps.

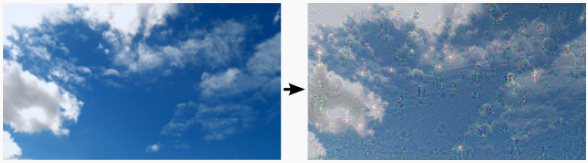
Enhancing feature maps

- Instead of prescribing which feature we want the network to amplify, we can also let the network make that decision.
  - feed the network an image.
  - then pick a layer and ask the network to enhance whatever it detected.
- Lower layers tend to produce strokes or simple ornament-like patterns:

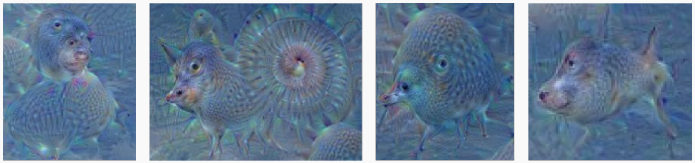


Enhancing feature maps: higher layers

- With higher level layers complex features or even whole objects tend to emerge.
  - these identify more sophisticated features in images...
- The process creates a feedback loop: if a cloud looks a little bit like a bird, the network will make it look more like a bird.



- If we train on pictures of animals:



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

Enhancing features: bias

- Results vary quite a bit with the kind of image, because the features that are entered bias the network towards certain interpretations.

We must go deeper: Iterations

- Apply the algorithm iteratively on its own outputs and apply some zooming after each iteration.
- We get an endless stream of new impressions.
- We can even start this process from a random-noise image.