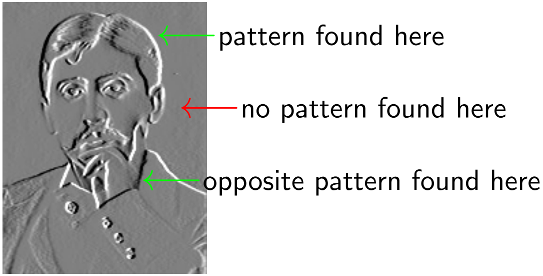
# Gradient ascent on input
# or
# How CNNs see the world

Suggested reading
An exploration of convnet filter with keras

## Patterns

Each neuron in a neural network gets activated by specific patterns in the input image, defined by the weights in it receptive field

$$\text{pattern} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$



## Complex (deep) patterns

The intuition is that neurons at higher layers should recognize increasingly complex patterns, obtained as a combination of previous patterns, over a larger receptive field.

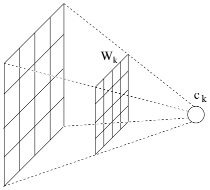
In the highest layers, neurons may start recognizing patterns similar to features of objects in the dataset, such as feathers, eyes, etc.

In the final layers, neurons gets activated by “patterns” identifying objects in the category.

**can we confirm such a claim?**

# Visualization of hidden layers

Goal: find a way to visualize the kind of patterns a specific neuron gets activated by.



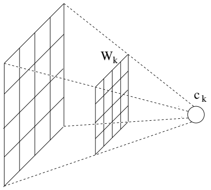
The loss function $\mathcal{L}(\theta, x)$ of a NN depends on the parameters $\theta$ and the input $x$.

During training, we fix $x$ and compute the partial derivative of $\mathcal{L}(\theta, x)$ w.r.t the parameters $\theta$ to adjust them in order to decrease the loss.

In the same way, we can fix $\theta$ and use partial derivatives w.r.t. input pixels in order to syntehsize images minimizing the loss.

## Visualization of hidden layers

Goal: find a way to visualize the kind of patterns a specific neuron gets activated by.



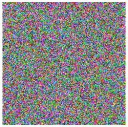
The loss function $\mathcal{L}(\theta, x)$ of a NN depends on the parameters $\theta$ and the input $x$.

During training, we fix $x$ and compute the partial derivative of $\mathcal{L}(\theta, x)$ w.r.t the parameters $\theta$ to adjust them in order to decrease the loss.

In the same way, we can fix $\theta$ and use partial derivatives w.r.t. input pixels in order to syntehsize images minimizing the loss.

## The “gradient ascent” technique

Start with a random image, e.g.

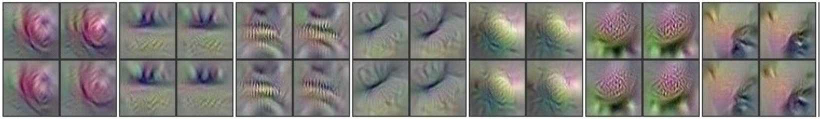


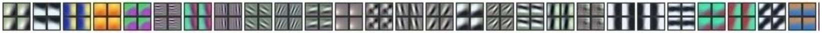
- do a forward pass using this image x as input to the network to compute the activation $a_i(x)$ caused by x at some neuron (or at a whole layer)
- do a backward pass to compute the gradient of $\partial a_i(x)/\partial x$ of $a_i(x)$ with respect to each pixel of the input image
- modify the image adding a small percentage of the gradient $\partial a_i(x)/\partial x$ and repeat the process until we get a sufficiently high activation of the neuron

# First layers

Some neurons from the first two layers of AlexNet

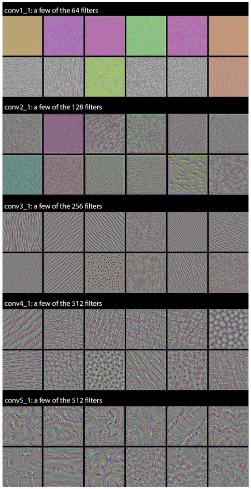
(Understanding Neural Networks Through Deep Visualization by A.Nguyen et al., 2015)





First features (lower picture) are very simple, and get via via more complex at higher levels, as their receptive field get larger due to nested convolutions.

## First layers


conv1_1: a few of the 64 filters
conv2_1: a few of the 128 filters
conv3_1: a few of the 256 filters
conv4_1: a few of the 512 filters
conv5_1: a few of the 512 filters

For a visualization of the first layers of VGG see:

An exploration of convnet filter with keras

## A different approach

Try to understand the inner representation at some layer by generating an image indistinguishable from the original one.

"DeConv nets look at how certain network outputs are obtained, whereas we look for what information is preserved by the network output – Mahendran, Vevaldi 2014"

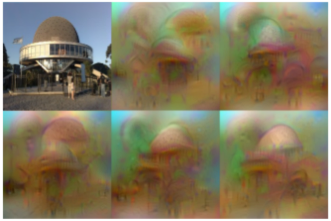


Figure 1. **What is encoded by a CNN?** The figure shows five possible reconstructions of the reference image obtained from the 1,000-dimensional code extracted at the penultimate layer of a reference CNN[13] (before the softmax is applied) trained on the ImageNet data. From the viewpoint of the model, all these images are practically equivalent. This image is best viewed in color/screen.

**Understanding Deep Image Representations by Inverting Them**

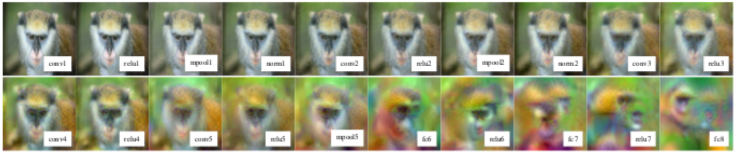
A. Mahendran, A. Vedaldi (2014)

## the technique

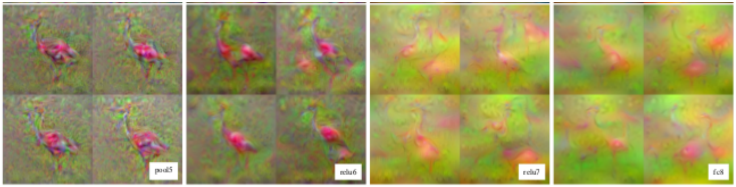
- **Goal**: given an input image $x_0$ with an internal representation $\Theta_0 = \Theta(x_0)$, generate a different image $x$ such that $\Theta(x) = \Theta_0$,
- **Approach**: via gradient ascent starting form a noise image. Instead of optimizing towards a given category or the activation of a neuron, minimize the distance from $\Theta_0$:

$$\operatorname*{argmin}_x \underbrace{\ell(\Theta(x), \Theta_0)}_{\textit{loss}} + \underbrace{\lambda \mathcal{R}(x)}_{\textit{regularizer}}$$

## Results



First layers are an invertible code, progressively becoming fuzzier.



Last layers invert back to multiple copies of object's parts at different positions and scales.

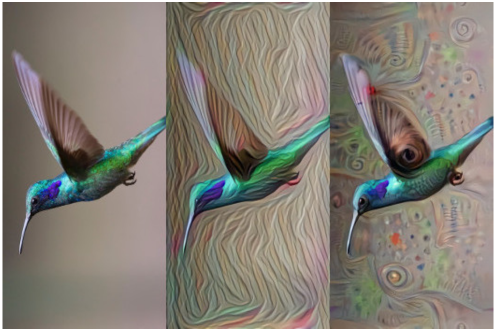
These details must be relevant for discriminative classification.

## Next argument

Inceptionism

## Deep dreams and inceptionism

Initially intended to visualize what a deep neural network is seeing when it is looking in a given image.



It evolved into a new form of psychedelic and abstract art.
See Google AI blog and deepdream generator

## Deep dreams

Usual approach:

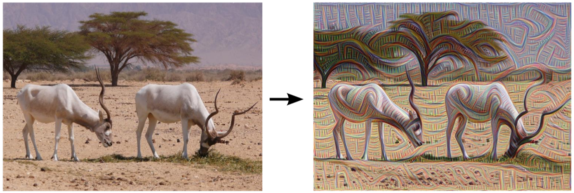
- train a network for image classification
- revert the network to slightly adjust (via backpropagation) the original image to improve activation of a specific neuron
  - after enough reiterations, even imagery initially devoid of the sought features will be incepted by them, creating psychedelic and surreal effects;
  - the generated images take advantage by **strong regularizers** privileging inputs that have **statistics similar to natural images**, like e.g. correlations between neighboring pixels (texture).

Many videos on youtube: for instance

What Google's Hallucinating Computers See

## Enhancing content

Instead of prescribing which feature we want to amplify, we can also fix a layer and enhance whatever it detected.

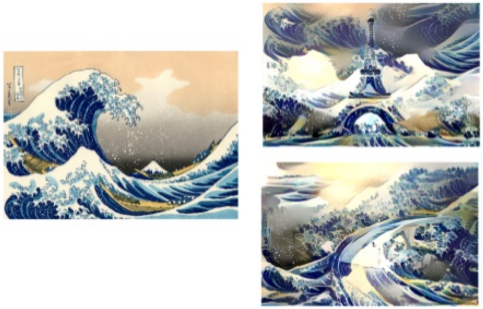


Each layer of the network deals with features at a different level of abstraction.

Lower layers will produce strokes or simple ornament-like patterns, because those layers are sensitive to basic features such as edges and their orientations.

## Next argument

# Style transfer

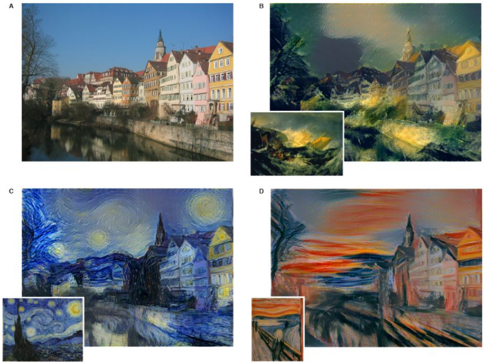


Source: A Neural Algorithm of Artistic Style. Leon A. Gatys et. al.

## Mimicking artistic style

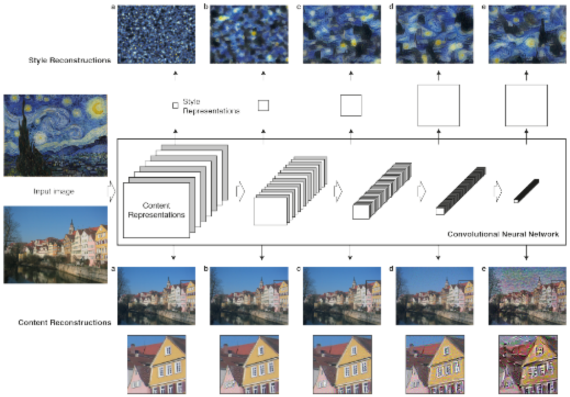
The gradient ascent technique can also be adapted to superimpose a specific style to a given content:

A neural algorithm of artistic style L.A.Gatys, A.S.Ecker, M.Bethge



## Capturing style

Add a feature space on top of the original CNN representations computing correlations between the different features maps (channels) at each given layer.
A technique already used to compute image textures.



## Capturing style

We know that at layer $\ell$ an image is encoded with $D^\ell$ distinct feature maps $F^\ell_d$, each of size $M^\ell$ (width times height).

$F^\ell_{d,x}$ is thus the activation of the filter $d$ at position $x$ at layer $l$.

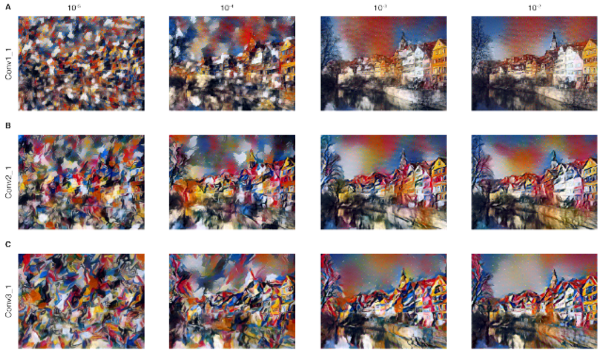
Feature correlations for the given image are given by the Gram matrix $G^\ell \in \mathcal{R}^{D^\ell \times D^\ell}$, where $G^\ell_{d_1,d_2}$ is the dot product between the feature maps $F^\ell_{d_1}$ and $F^\ell_{d_2}$ at layer $\ell$:

$$G^\ell_{d_1,d_2} = F^\ell_{d_1} \cdot F^\ell_{d_2} = \sum_k F^\ell_{d_1,k} \cdot F^\ell_{d_2,k}$$

This is just yet another internal representation of the image, useful to encode style.

## Combine style and content

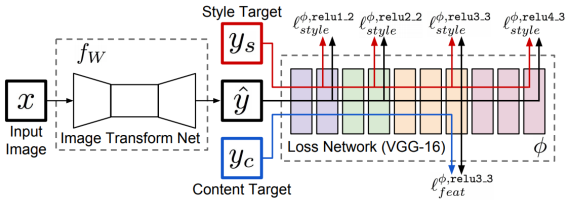
Content and style are separable:



Different combinations varying the reconstruction layer (rows) and the relevance ratio between style and content (columns).

## Variants and improvements

Perceptual Losses for Real-Time Style Transfer and Super-Resolution



Style Target
$y_s$
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
$f_W$
$x$
Input Image
Image Transform Net
$\hat{y}$
$y_c$
Content Target
Loss Network (VGG-16)
$\phi$
$\ell_{feat}^{\phi,relu3\_3}$

An image transformation network is trained to transform input images into output images. A pretrained loss network for image classification is used to define perceptual loss functions that measure perceptual differences in content and style between images. The loss network remains fixed during the training process.

In the case of style transfer, $y_c$ is $x$.

## Some additional links

- Rethinking and Improving the Robustness of Image Style Transfer
- Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization
- Adversarially Robust Neural Style Transfer
- Universal Style Transfer via Feature Transforms

Somehow superseded by diffusion techniques ...

## Recap: possible applications of gradient ascent

- if the loss corresponds to the activation of a specific neuron (or a specific layer) we may try generate images that cause a strong activation of it, hence explaining the role of the neuron inside the network (what neurons see of the world)
- if the loss is the distance, in the latent space, from the internal representation of a given image, we may try to generate other images with the same internaal representation (hence explaining what features have been captured in the internal representation)
- if the loss is the similarity to a given texture, we may try to inject stylistic information in the input image
- what if the loss is the **distance from a target category** in a classification network?
  Can we hope to automatically synthesize images belonging to that category? (or at least having distintictive features of that category?)

## Recap: possible applications of gradient ascent

- if the loss corresponds to the activation of a specific neuron (or a specific layer) we may try generate images that cause a strong activation of it, hence explaining the role of the neuron inside the network (what neurons see of the world)
- if the loss is the distance, in the latent space, from the internal representation of a given image, we may try to generate other images with the same internaal representation (hence explaining what features have been captured in the internal representation)
- if the loss is the similarity to a given texture, we may try to inject stylistic information in the input image
- what if the loss is the **distance from a target category** in a classification network?
  Can we hope to automatically synthesize images belonging to that category? (or at least having distintictive features of that category?)

## Next arguments

Distance from a target category

or also

How to fool a Neural Networks

## Next arguments

Distance from a target category

or also

How to fool a Neural Networks

## Next arguments

Distance from a target category

or also

How to fool a Neural Networks

## Reducing distance from a target category

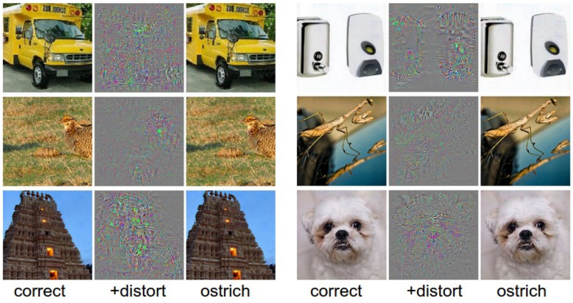
In an image calssification framework, we can use the gradient ascent technique to increase, starting from noise or any given picture, the score of whatever class we want.

Demo!

We shall try to transform an elephant into a tigershark.

## Fooling Neural Networks

Since we have many pixels, a tiny (imperceptible to humans!), consistent perturbation of all of them is able to fool the classifier.



Intriguing Properties of Neural Networks, C. Szegedy et al., 2013

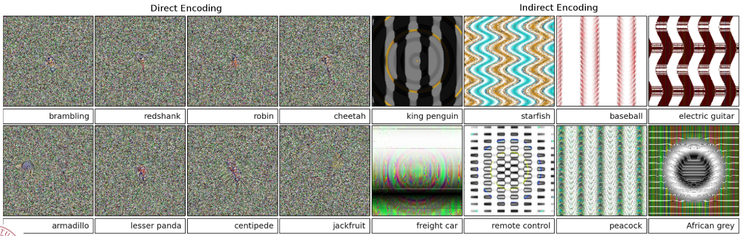
# A different technique

The previous technique, being based on gradient ascent, requires the knowledge of the neural net in order to fool it.

We can do something similar using the network as a **black box**, for instance by means of evolutionary techniques

- Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled. In Computer Vision and Pattern Recognition (CVPR '15), IEEE, 2015.

They were able to produce not only "noisy" adversarial images, but also geometrical examples with high regularities (meaningful for humans).



# Evolutionary approach

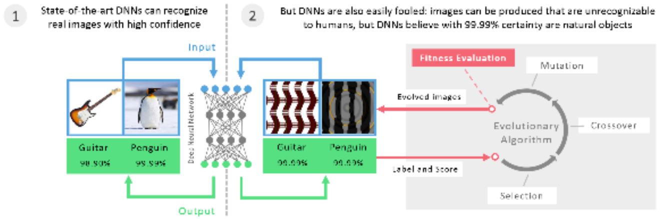

1
State-of-the-art DNNs can recognize real images with high confidence
2
But DNNs are also easily fooled: images can be produced that are unrecognizable to humans, but DNNs believe with 99.99% certainty are natural objects
Input
Output
Deep Neural Network
Guitar 98.90%
Penguin 99.99%
Guitar 99.99%
Penguin 99.99%
Fitness Evaluation
Mutation
Evolved images
Evolutionary Algorithm
Crossover
Label and Score
Selection

Figure 2. Although state-of-the-art deep neural networks can increasingly recognize natural images (*left panel*), they also are easily fooled into declaring with near-certainty that unrecognizable images are familiar objects (*center*). Images that fool DNNs are produced by evolutionary algorithms (*right panel*) that optimize images to generate high-confidence DNN predictions for each class in the dataset the DNN is trained on (here, ImageNet).

- start with a random population of images
- alternately apply selection (keep best) and mutation (random perturbation/crossover)