UiO : Department of Informatics
University of Oslo

INF 5860 Machine learning for image classification

Lecture 11: Visualization

Anne Solberg

April 4, 2018

Reading material

The lecture is based on papers:

- Deep Dream: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
- https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html
- Zeiler and Fergus 2013
- Springberger et al. (2015)
- http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf
- Visualising explanations from Deep Networks via gradient-based localication
- Simonyan, Veldaldi, Zisserman
- Understanding deep image representations by inverting them
- Multifaceted Feature Visualization
- A neural algorithm of artistic style
- Perceptual losses for real-time style transfer and super-resolution

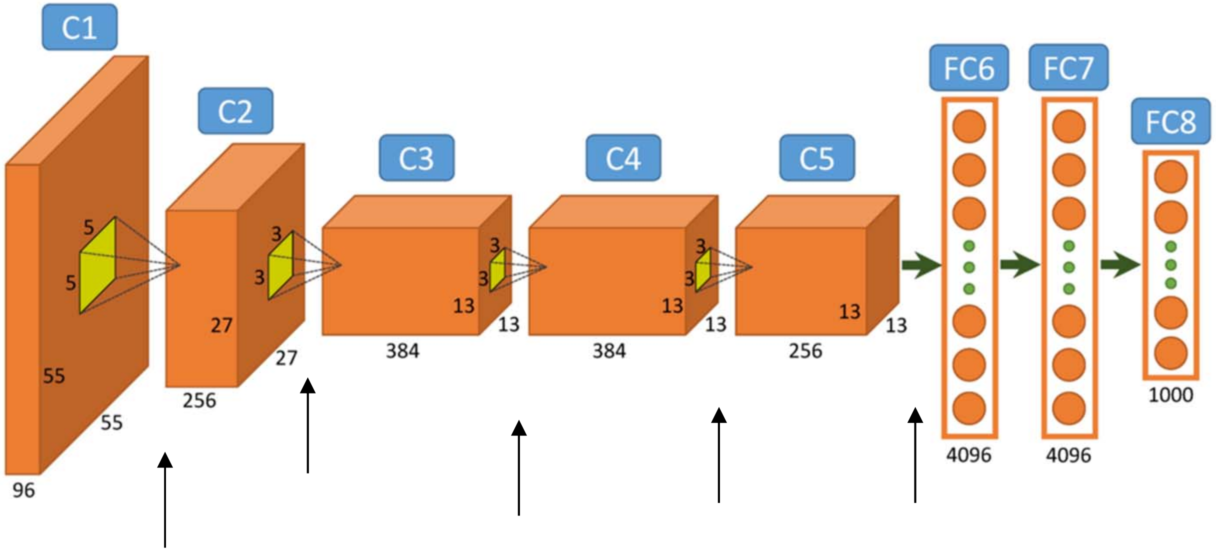
Today – overview of many methods

- Visualization filters
- Visualizing activations:
  - Occlusion experiments
- Visualizing class activation maps
  - Guided backprop
  - Gradcam
- Gradient with respect to the image
  - Saliency maps
- Fooling the network (more in a later lecture)
- Feature inversion
- Neural style transfer

Introductory read

- https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

What do the layers learn?



What does these intermediate features look like?
Can this help us gain confidence in what the network learns?
How can we fool the network?

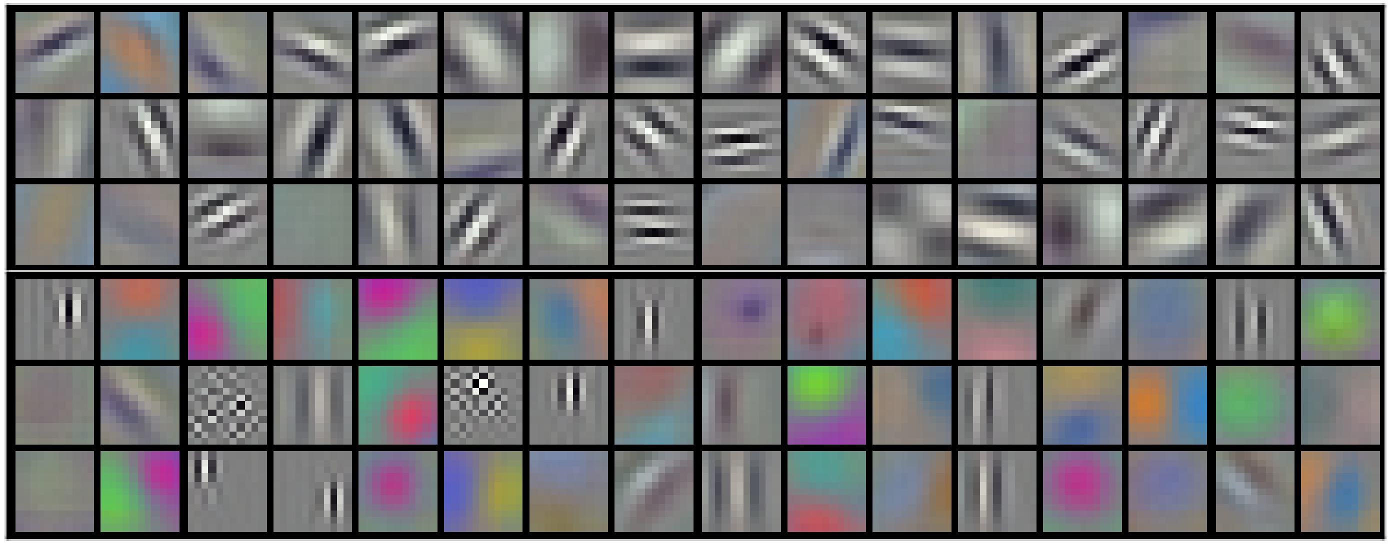
Visualizing filters

Visualizing the filters directly

Can we visualize the filters themselves?

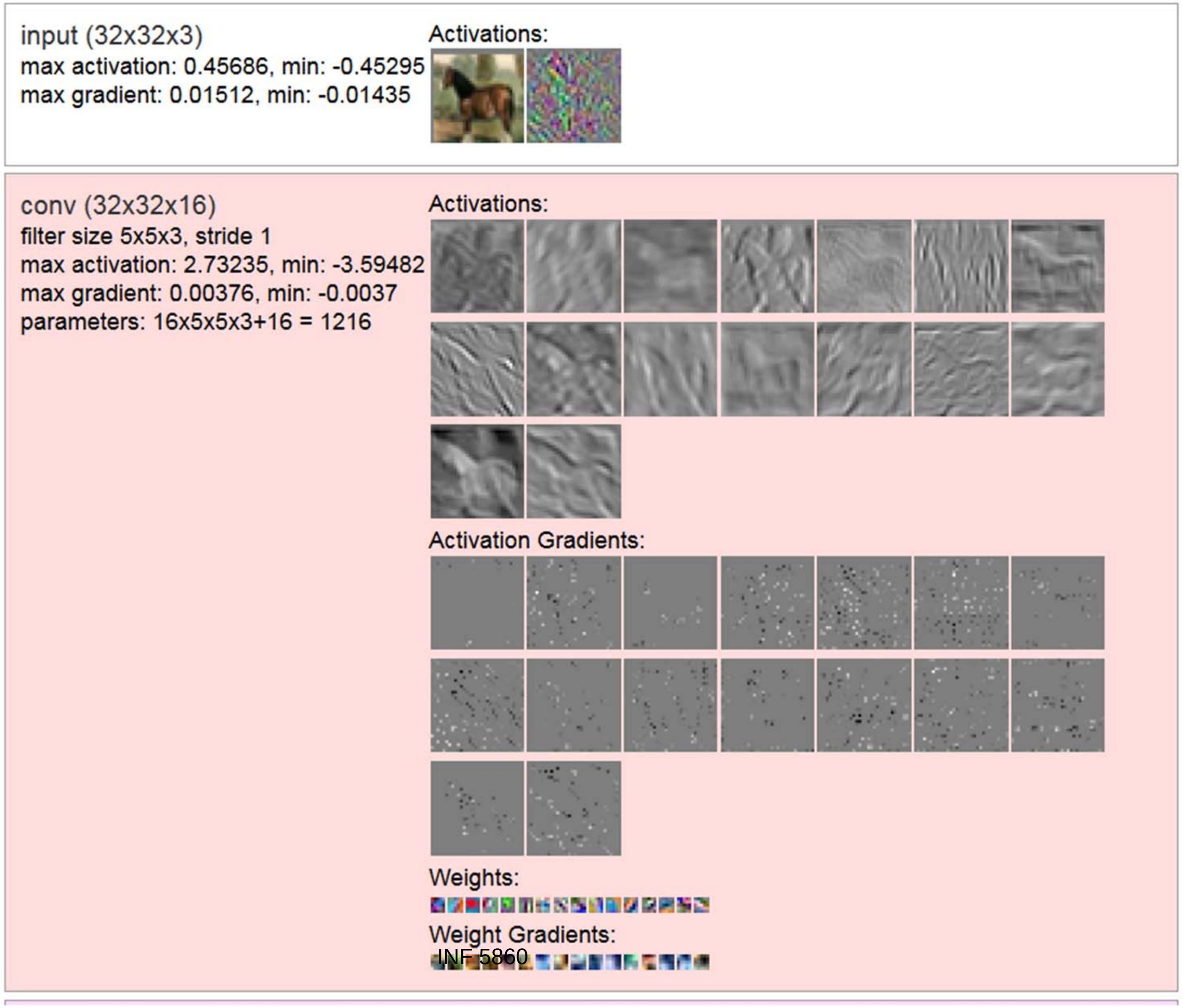
- Useful for the first couple of layers, then difficult

The weights of layer 1 Alex-net



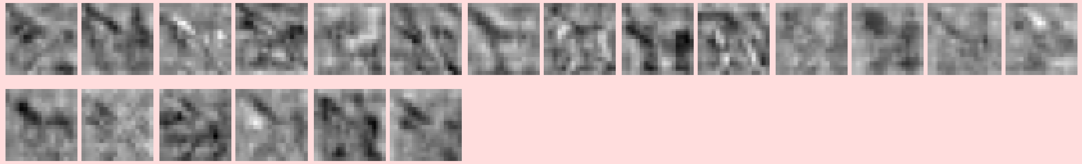
96 filters of size 11x11x3

Check it out at https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

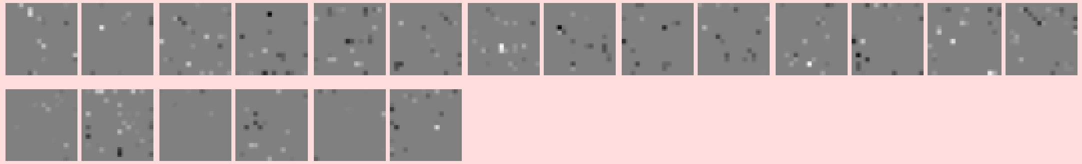


conv (16x16x20)
filter size 5x5x16, stride 1
max activation: 3.9723, min: -7.03267
max gradient: 0.00261, min: -0.00266
parameters: 20x5x5x16+20 = 8020

Activations:



Activation Gradients:



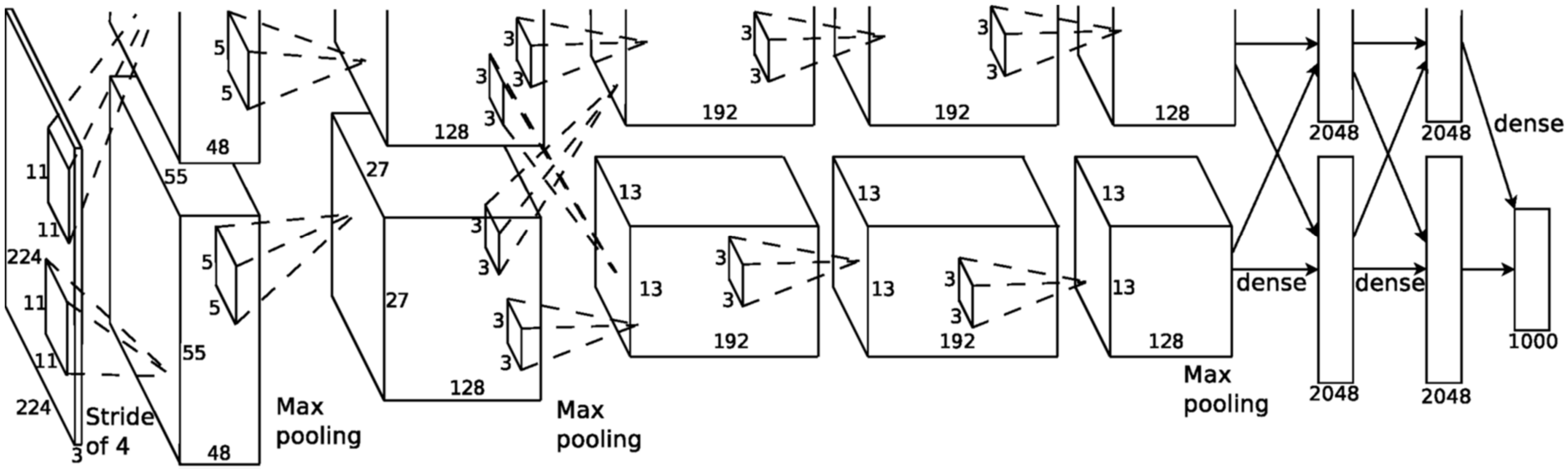
Weights:

Weight Gradients:

Visualizing the final layer

Visualizing the features just before softmax

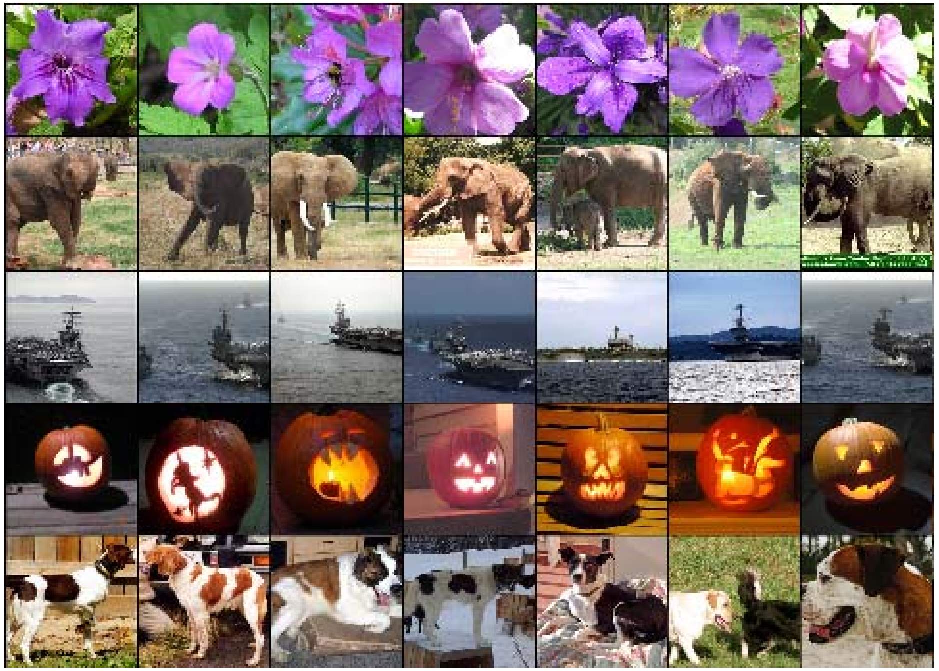
Visualizing the final FC layer in Alex-net



Classification (softmax) is done on 4096 features

Nearest neighbors in feature space FC7

- Propagate a lot of images through the net, and store the features
- Take the 4096 features in FC7 for a given image, find the K-nearest neighbors in feature space FC7 and display those.



Visualizing which pixels are most important for a class

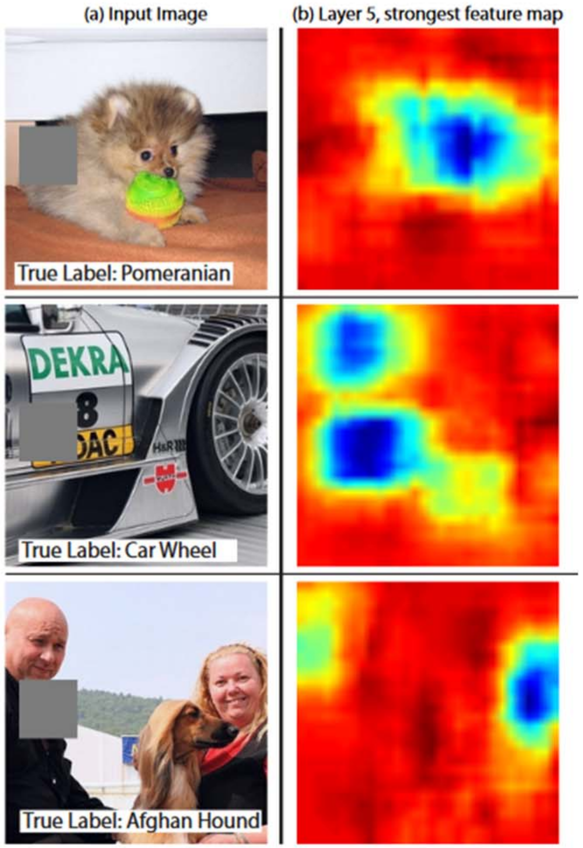
Occlusion experiments

Saliency maps

Occlusion experiments

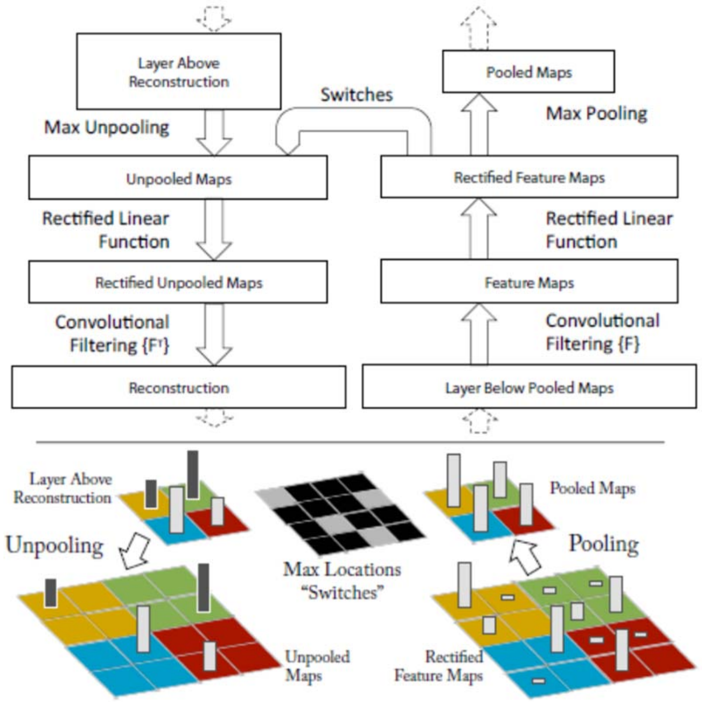
- Create a small patch of zeros.
- Slide this over the image and zero out pixels inside the patch.
- Classify all these images.
- Record how the probability for the given class change over the image as the mask shifts.

Zeiler and Fergus 2013 – occlusion experiments



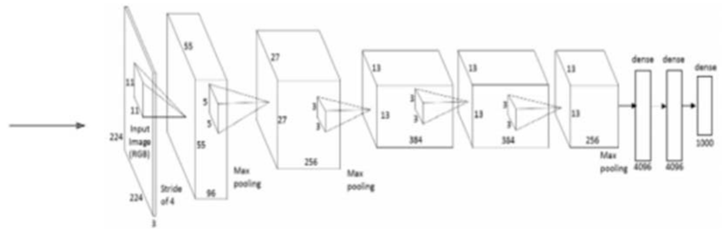
Deconvnet (Zeiler and Fergus 2013)

- Go through the CONVNET operations in reverse order
- Each layer has a deconv-operations.
- To visualize ONE activiation:
  - Set all other activations in this layer to 0
  - Use feature map as input to deconvnet
  - Unpool using switching
  - Rectify using a ReLU
  - Filter with the transpose of the weight matrix
  - Continue to input and display the resulting image.



Visualizing and understanding convolutional networks

DECONVNET: Gradient of a neuron with respect to the image



Treat the image as a variable and the network weights as constants

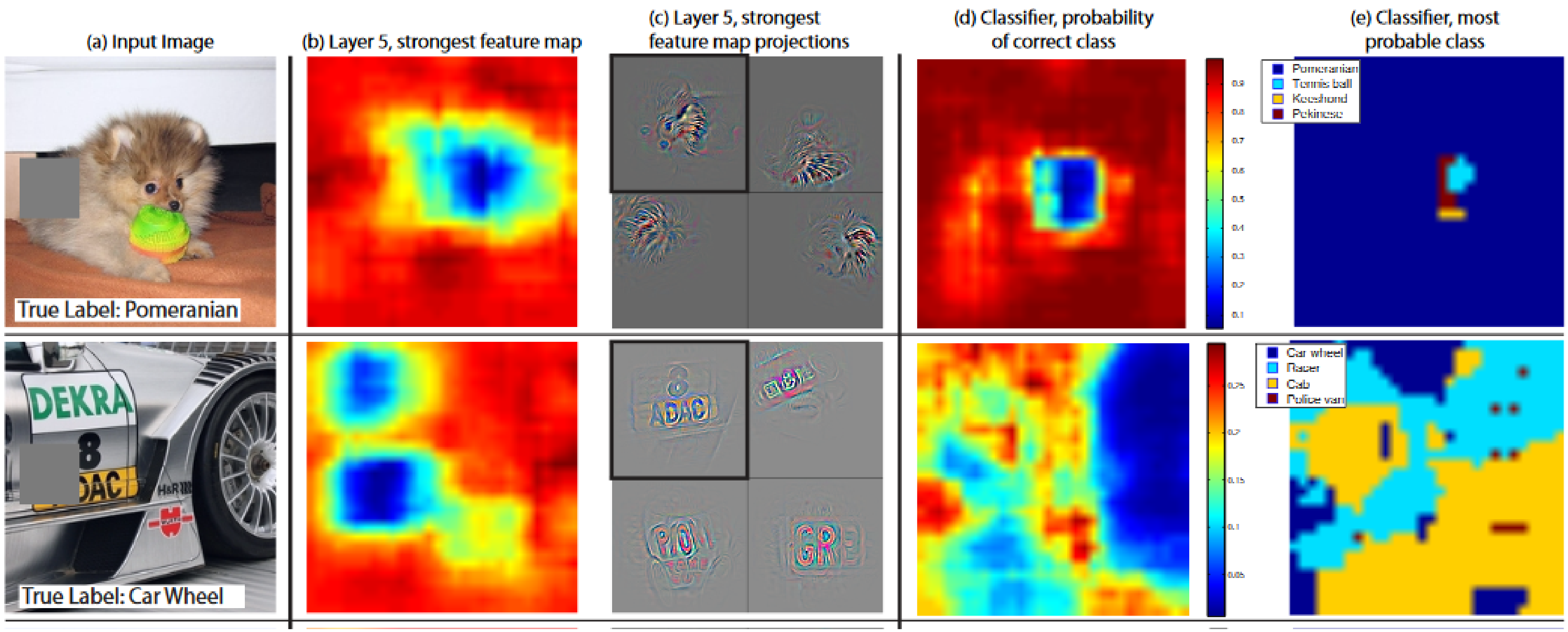
1. Run the image through the network
2. Set the gradients at the layer you want to be zero, except for the neuron of interest
3. Backprop all the way back to the image

Zeiler and Fergus - visualizations

- See details:
- [Visualizing and understanding convolutional networks]

Figure 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activations in a random subset of feature maps across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are *not* samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form.

Zeiler and Fergus results

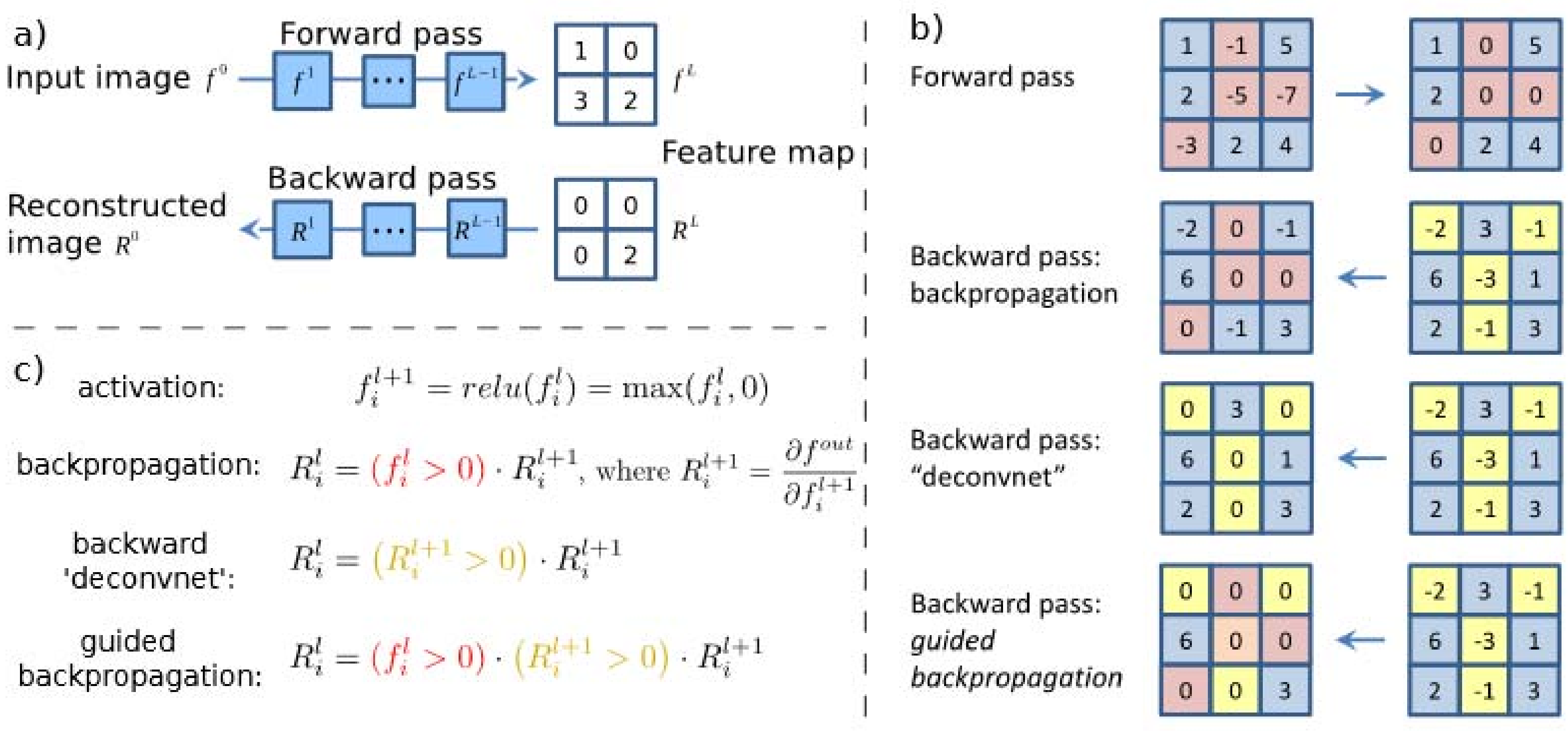


Guided backprop

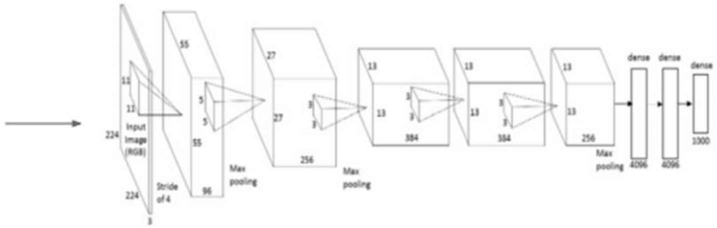
- Springberger et al. (2015) has a couple of interesting points
  - They show that pooling can often be replaced by strided convolution
  - They show that deconv can be improved by only backpropagating positive gradients (called Guided backprop)

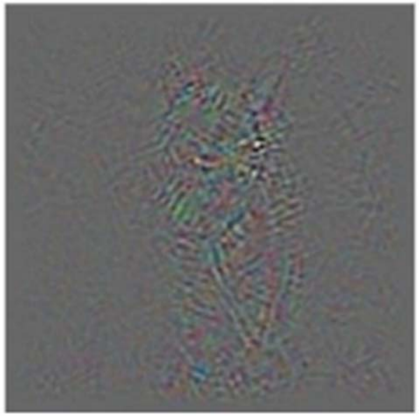
Guided backprop vs. deconv

- From https://arxiv.org/pdf/1412.6806.pdf



Deconvnet vs. Guided backprop





Deconvnet

Guided backprop

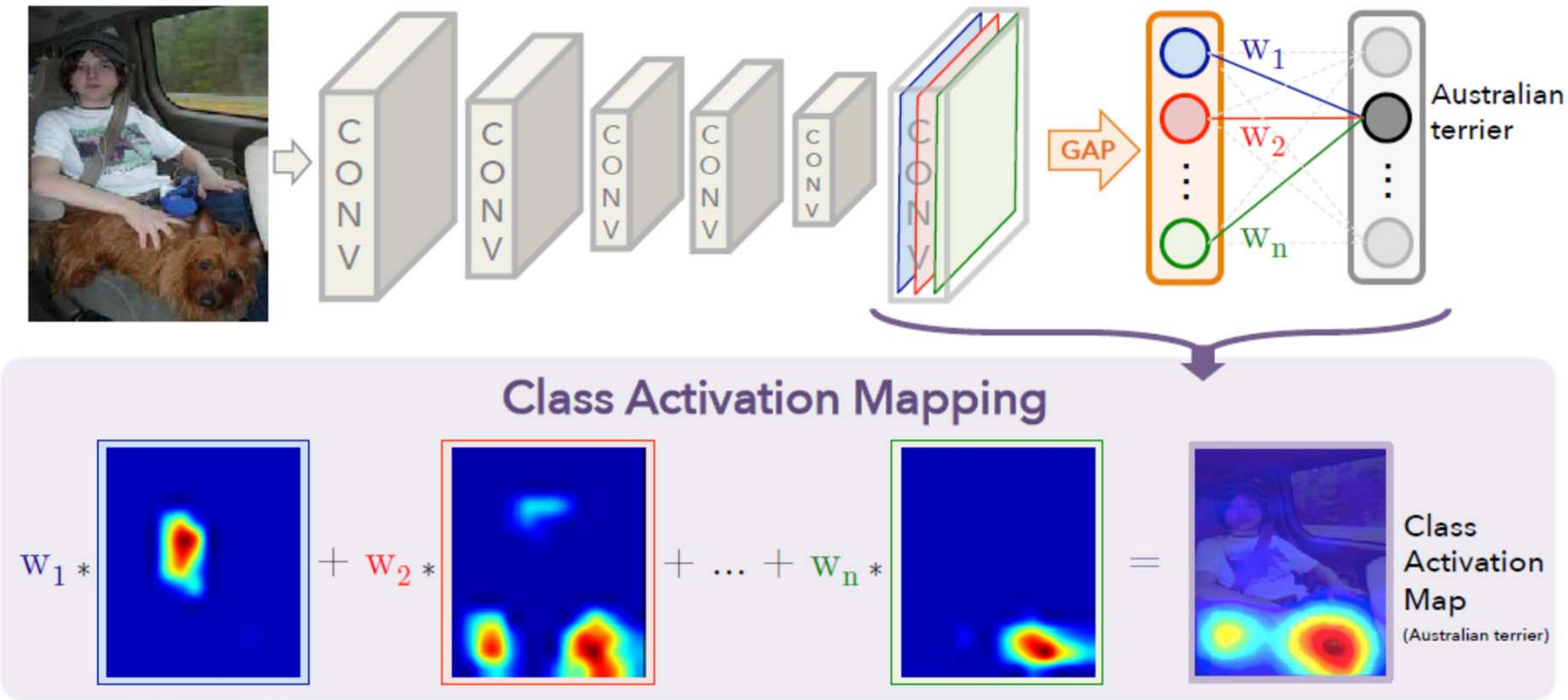
More focused

Results with guided backprop for different channels

Visualizing class activation maps

CAM – Class activation mapping

- http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf
- Main idea: Replace fully connected layers with convolutional layers and global pooling to facilitate visualizing features



CAM – main principles

- Normally: FC→Softmax (no ReLU)
- CAM:
  - f_k: activation of node k in last conv-layer
  - Global average pooling: $Fk=\Sigma_{x,y} f_k (x,y)$
  - Input to softmax for class c: $S_c=\Sigma_k w_k^c F_k = \Sigma_k w_k^c \Sigma_{x,y} f_k (x,y)$
  - Define Class Activation Map, $M_c(x,y)= \Sigma_k w_k^c f_k (x,y)$
  - This gives a pixelwise importance of pixel (x,y) for class c

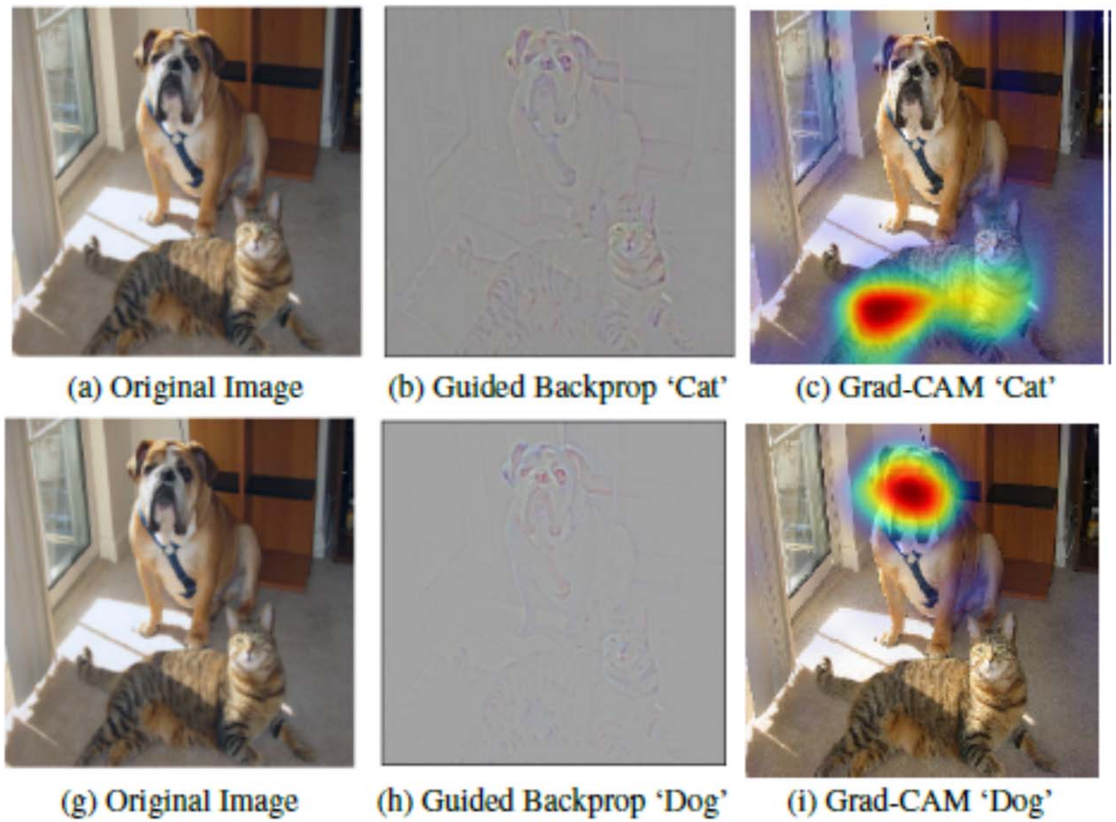
GradCam

- [Visualising explanations from Deep Networks via gradient-based localication]
- Drawback of CAM: only works for pure convolutional architectures with general average pooling before softmax.
- GradCAM: use gradients flowing into the last conv-layer.

GradCAM principles

- Start with the score for class c before softmax y_c
- Compute the gradient of this with respect to the feature maps A_k of a conv-layer.
- Then apply GAP of these for all locations to get a weight α_k^c
- Then get the GradCAM localization map as ReLU of a linear combination Ak and α_k^c

GradCAM


(a) Original Image
(b) Guided Backprop 'Cat'
(c) Grad-CAM 'Cat'
(g) Original Image
(h) Guided Backprop 'Dog'
(i) Grad-CAM 'Dog'

Guided Grad-CAM

- Grad-CAM fails to show fine details like Guided backprop
- Guided Grad-CAM fuses Guided backprop and Grad-CAM using pointwise multiplication

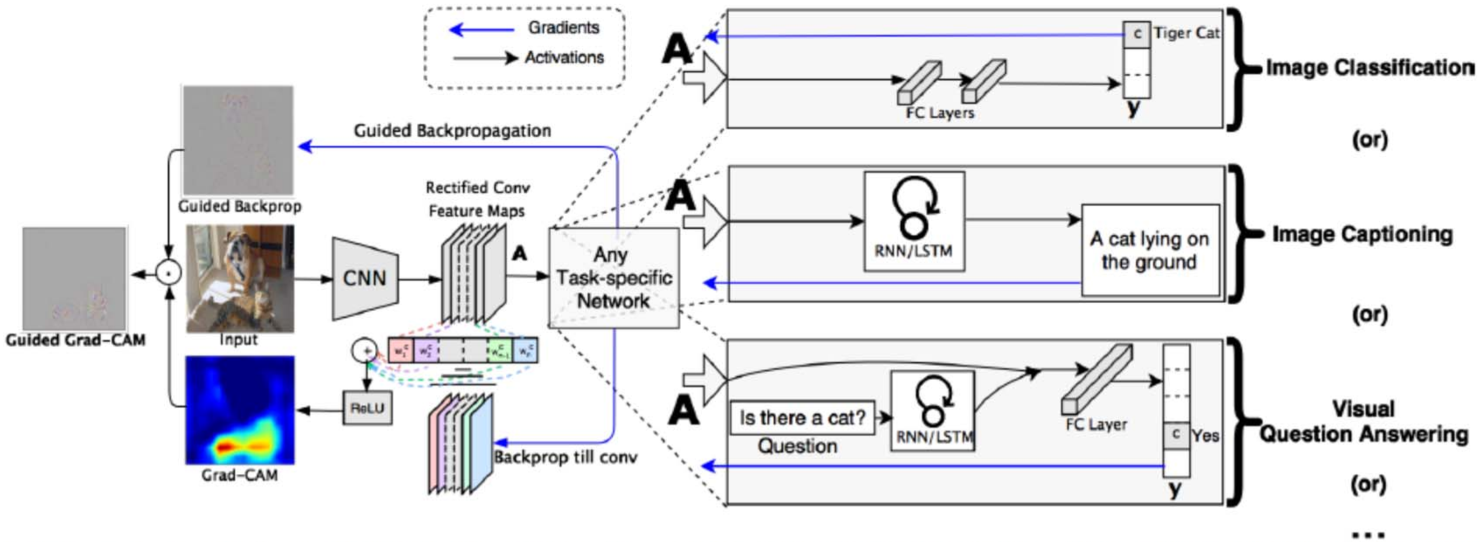
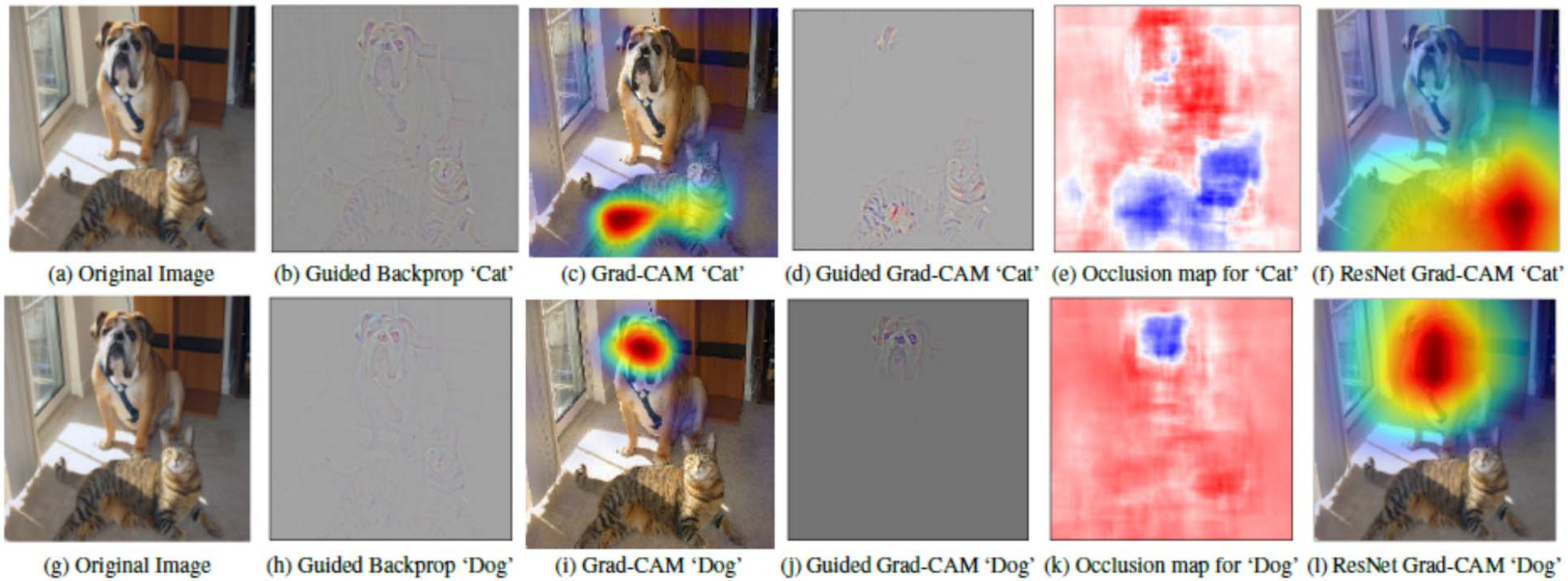


Figure 2: Grad-CAM overview: Given an image and a class of interest (*e.g.*, 'tiger cat' or any other type of differentiable output) as input, we forward propagate the image through the CNN part of the model and then through task-specific computations to obtain a raw score for the category. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then backpropagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization (blue heatmap) which represents where the model has to look to make the particular decision. Finally, we pointwise multiply the heatmap with guided backpropagation to get Guided Grad-CAM visualizations which are both high-resolution and concept-specific.

Guided Grad-CAM



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d) Guided Grad-CAM 'Cat' (e) Occlusion map for 'Cat' (f) ResNet Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j) Guided Grad-CAM 'Dog' (k) Occlusion map for 'Dog' (l) ResNet Grad-CAM 'Dog'

Applications for image gradients

Gradient ascent with respect to the image

Creating saliency maps

- Simonyan, Veldaldi, Zisserman
- Goal: find **the parts of** a L2-regularized image such that the score S_c for class c a maximized.

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2,$$

- Question they rise: **What is the spatial support for class c in a given image?**

Simonya – taking the derivative with respect to image I to create saliency maps

- If we had a vectorized linear layer, the size of w_c for each pixel would tell the importance of the pixels with the largest weights.

$$S_c(I) = w_c^T I + b_c,$$

- Locally, around pixel I_0, this is approximated by a Taylor-expansion around I_0,

$$S_c(I) \approx w^T I + b,$$

- w is the derivative of Sc with respect to image I at the point (or image) I_0 :

$$w = \left. \frac{\partial S_c}{\partial I} \right|_{I_0} .$$

- The derivative of S_c with respect to I tells us which locations require the smallest change to affect the score S_c the most.

Simonya – create saliency maps

- Given an image I_0 and a class c:
- Create a saliency map M as:
  - Compute w by backpropagation
  - Reshape elements in w to match pixel locations
  - Take max magnitude across color channels

Simonyada: Class saliency maps

- Examples from https://arxiv.org/pdf/1312.6034.pdf

Gradient ascent with respect to the image

- Simonyan, Veldaldi, Zisserman
- Goal: find **an image** such that the score S_c for class c a maximized.

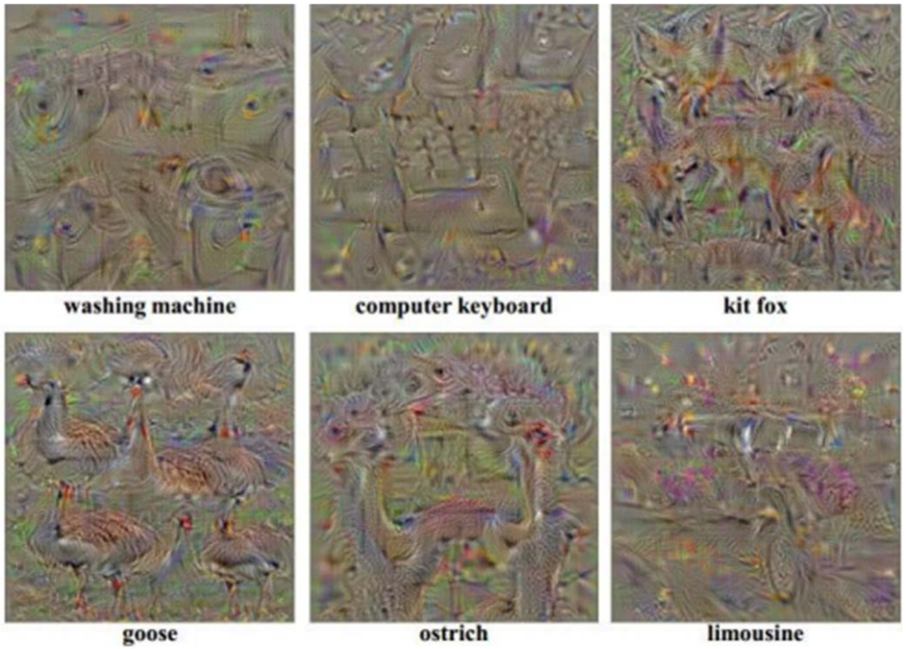
$$\arg\max_I S_c(I) - \lambda \|I\|_2^2,$$

Optimizing image - Gradient ascent on image

When you have the image gradient you can iterate with gradient ascent, to get very "catty" pictures…

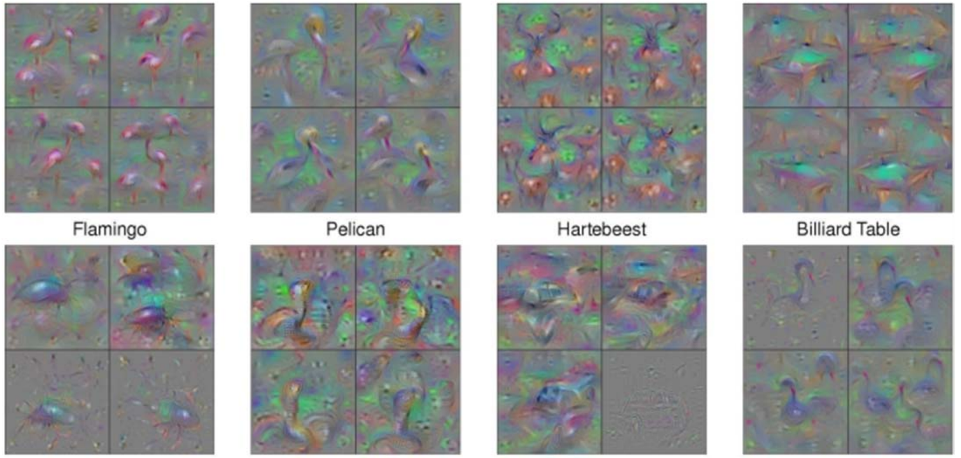
They often look weird:

- Most images are not natural images
- Many cats are more cat than one cat :)



Optimizing image - Gradient descent on image

- Adding **blur** between each iteration
- Natural images usually don’t have very high frequent information
- Setting pixels close to zero to **zero** removes some of the “overlapping” effects



Accounting for variations

- A tie neuron/output can respond to many different situations
- A neuron is often reused for many different classes
- This can contribute to noise in the visualizations

Multifaceted Feature Visualization

Accounting for variations

A possible approach:

1. Run images through the network
2. Save activations at a given layer
3. Do dimensionality reduction
4. Then clustering
    a. You find images with similar representation on that level
5. Find average image of each cluster
6. Start optimization with this average image

Accounting for variations

- You find neurons that represent ties or theaters
- You find different variations that the neuron is invariant to.

Feature inversion

Feature inversion
What is stored in a set of neurons

- Find an image that gives same response/similar features
- How much is stored in the weights and output layer?

- Important for privacy
- Important to see what the network includes and excludes

Understanding deep image representations by inverting them

Feature inversion

Understanding deep image representations by inverting them

- From the features inside the net – can we reconstruct the image?
- Given a feature representation Θ_0, reconstruct **x** as

$$x = argmin_x l(\Theta(x), \Theta_0) + \lambda R(\lambda)$$

- l is the Euclidean loss:

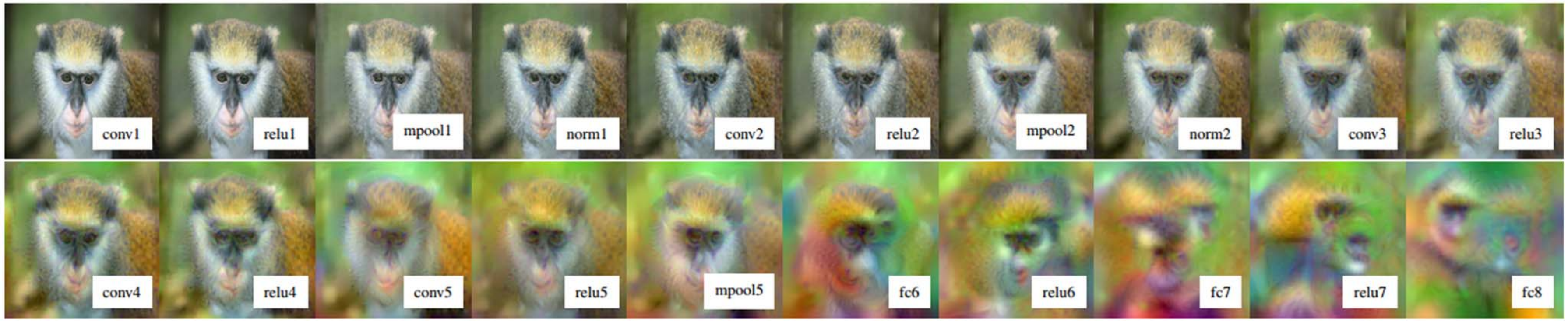
$$l(\Theta(x), \Theta_0) = \frac{||\Theta(x) - \Theta_0||^2}{||\Theta_0||^2}$$

- With TV (total variation)-regularization:

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

- Solve using gradient descent

Feature inversion: Reconstruction from different layers



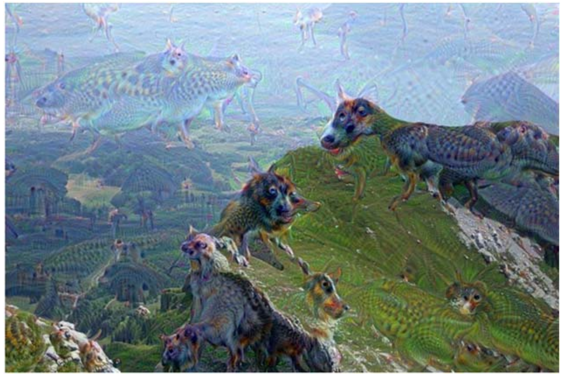
UiO : Department of Informatics
University of Oslo

Applications for image gradients

Visualizing activation/Dreaming?

Optimizing on a pre-existing image

- If you optimize on a pre-existing network it can have strange effects
  - Optimize for a activation channel
  - Optimize for whole final output
- Optimizing for whole final output
  - Smallest change to the image that affect the output most
  - The traits a network first “discover” will be enhanced

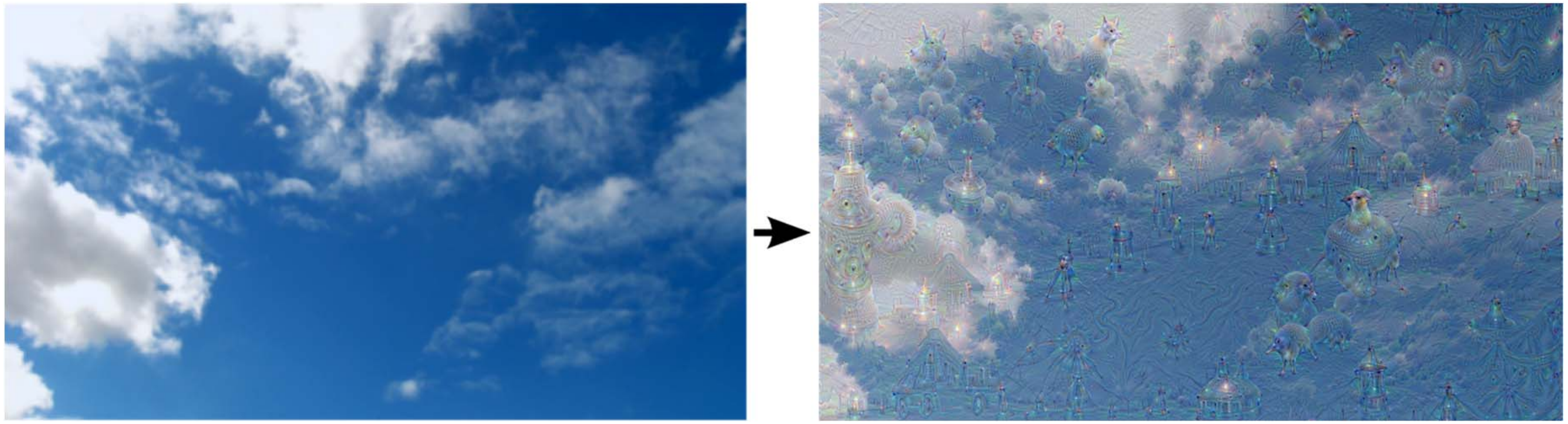


Full code in tensorflow notebook:
https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb

DeepDream

- https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html
- Start with either a noise image or a natural image.
- Forward propagate the image to a given layer.
- Modify the gradient of this layer to equal its activation : see more of what the layer sees
- Add some tricks
- Backward propagate and update image

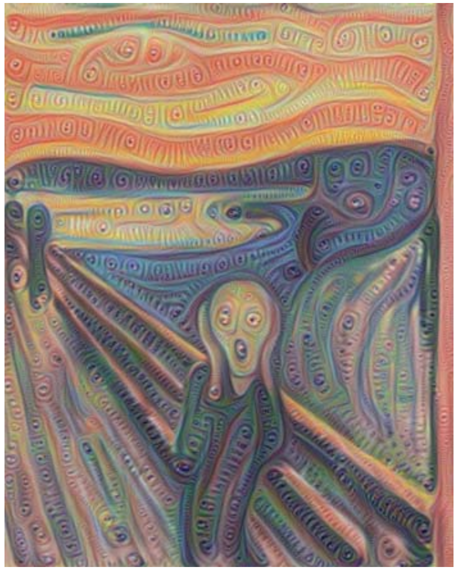
Just for fun



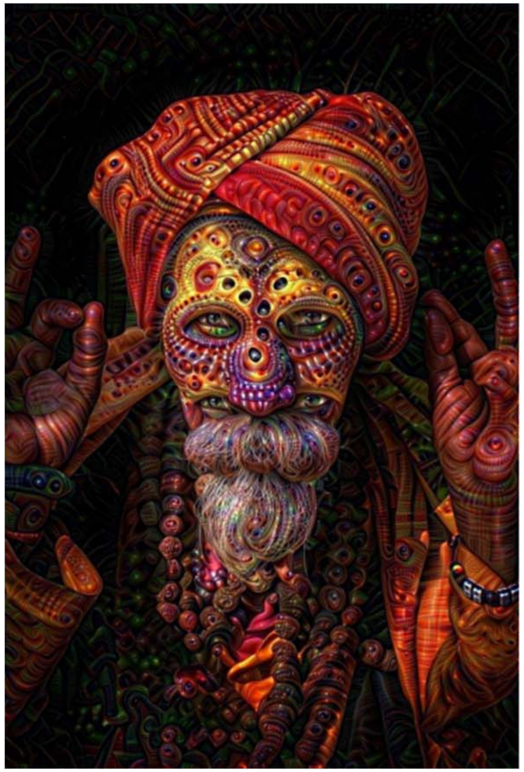
"Admiral Dog!" "The Pig-Snail" "The Camel-Bird" "The Dog-Fish"

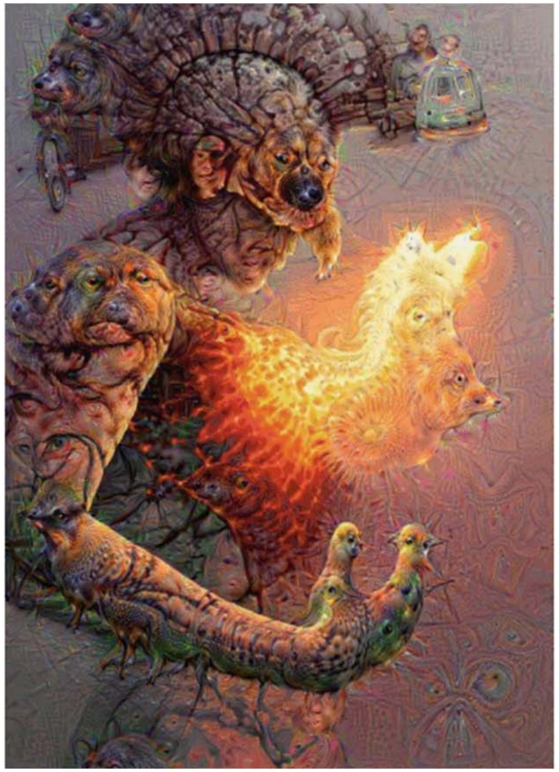
search.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Have a look at the Inception Gallery

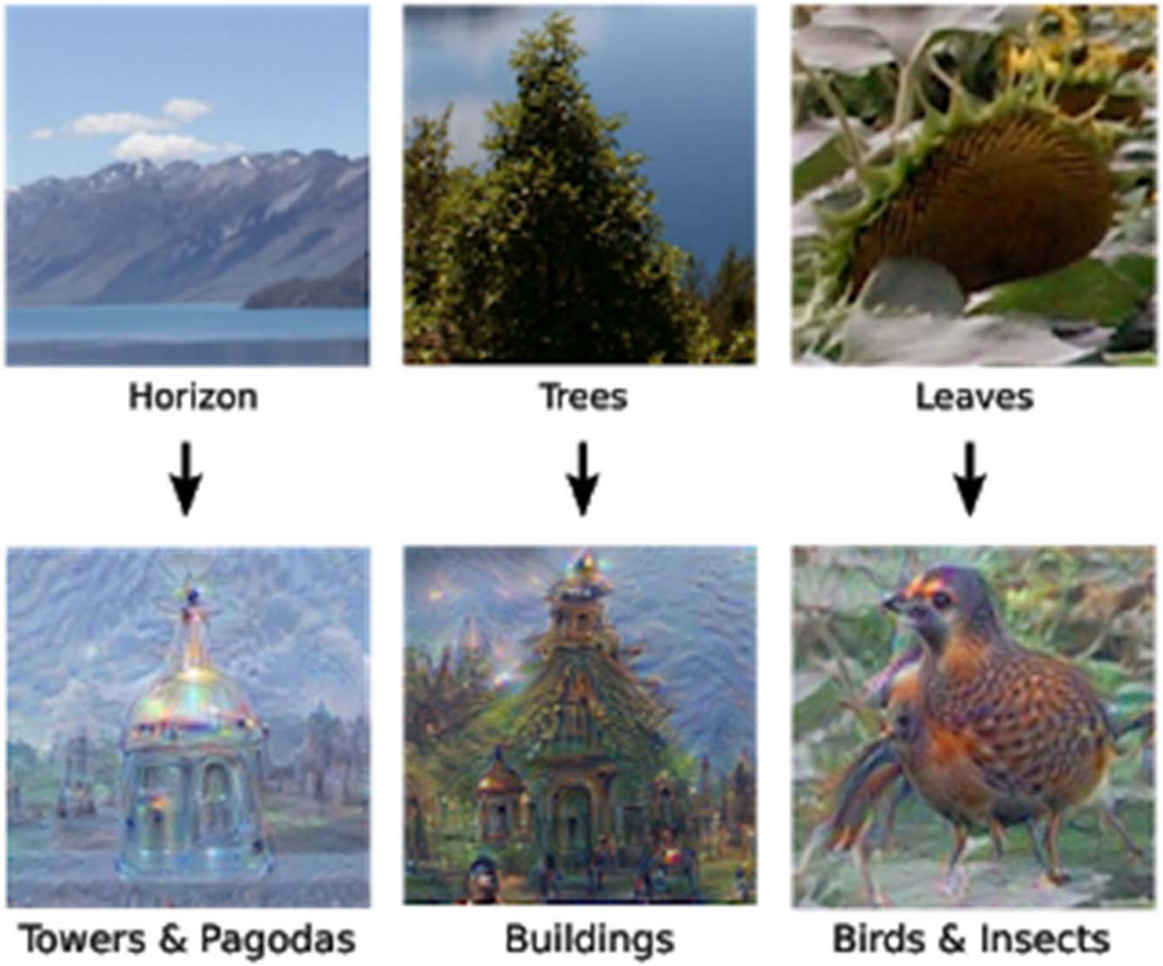


Some cool examples





More dreaming….

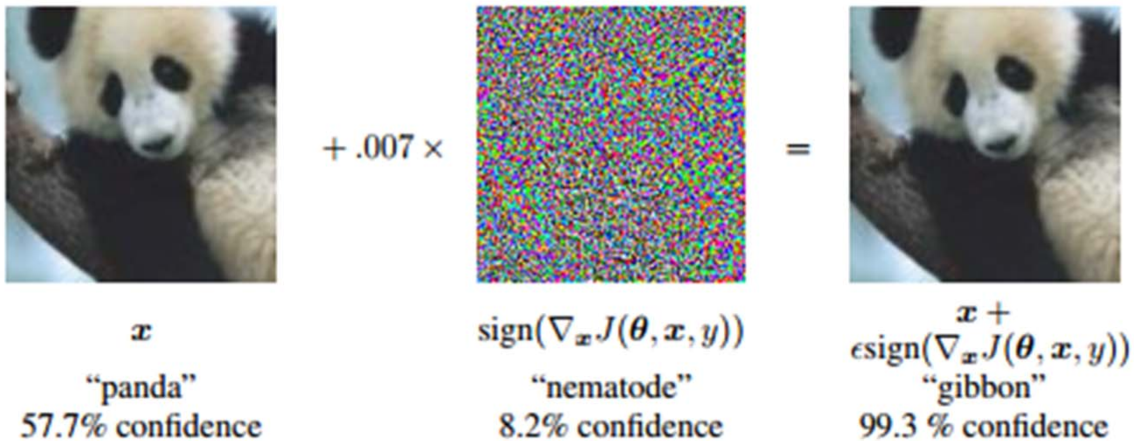


Applications for image gradients

Fooling a network

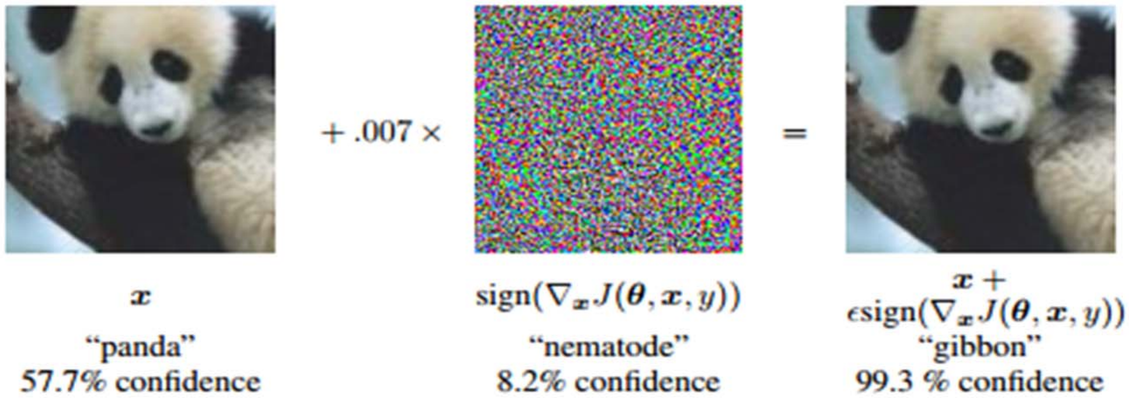
Fooling a neural network

- Adding small values to every pixel gives large change in euclidian distance
- Network representations are different than human representation
- This is not inherent to deep learning or neural networks



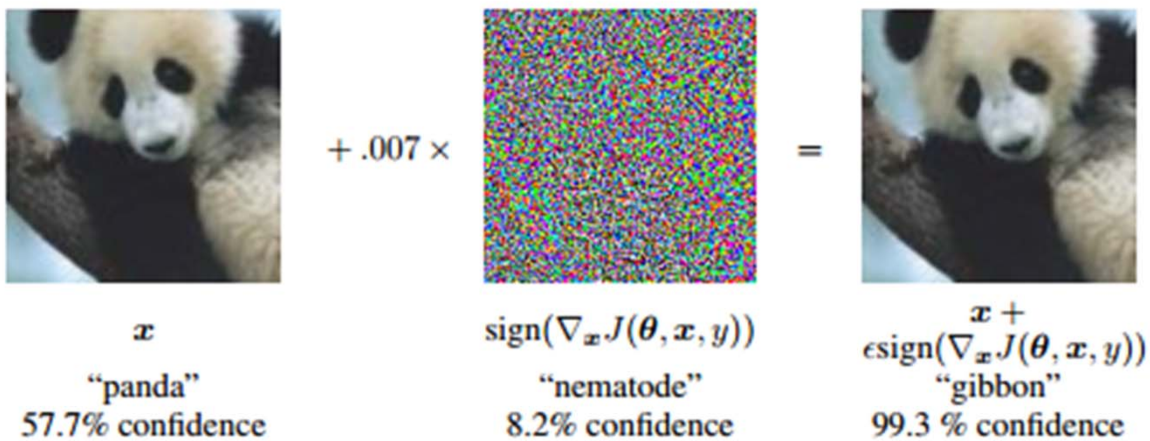
Is there a fix?

- Forcing perturbed images to give similar representations at different levels
- Training on adversarial examples
- Adding noise to training
- Adding noise and smoothing on input images



A final solution is hard

- If you have access to the gradient, there will alway be some “small” direction that can fool the network
- This is not inherent for deep learning, but also exists in other machine learning models



UiO : Department of Informatics
University of Oslo

Visualization

Neural style transfer

Background: texture synthesis

- Constructing an image from a small texture sample
- What is texture (From INF 4300)?

What is texture?

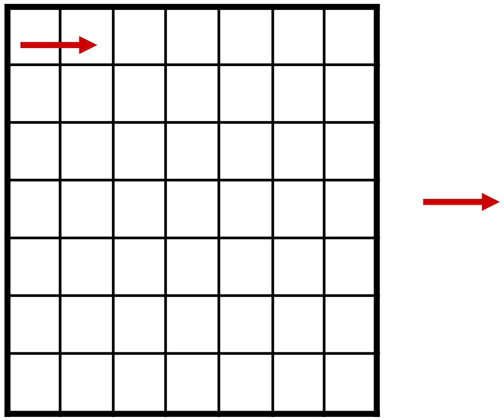
- Intuitively obvious,
  but no precise definition exists
  - "fine, coarse, grained, smooth" etc
- Texture consists of texture primitives, texels,
  - a contiguous set of pixels with some tonal and/or regional property
- Texture can be characterized by
  - intensity (color) properties of texels
  - Structure & spatial relationships of texels
- A texel is the characteristic object
  that the texture consists of (the "brick in the wall")
- Textures are highly scale dependent.

"Spatially extended patterns of more or less accurate repetitions of some basic texture element, called texels."

F2 31.08.17 INF 4300 5

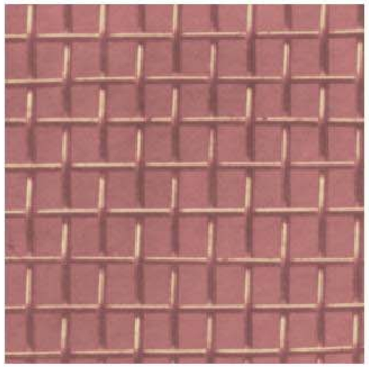
Classical texture descriptors: Grey-Level Co-occurrence Matrix

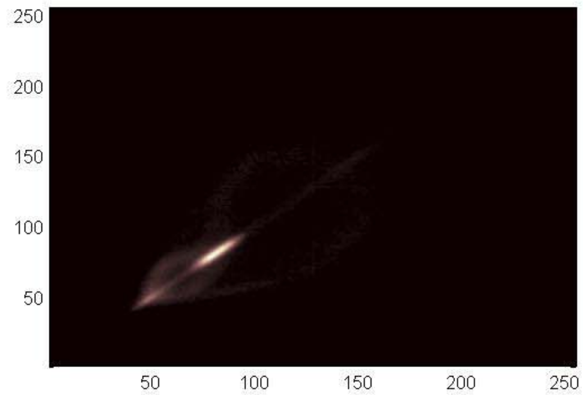
- Create a matrix of the co-occurence of a change in graylevel from *i* to *j* when moving distance *d* in direction θ.
- Dimension of co-occurrence matrix is *GxG* (*G* = gray-levels in image)
- Choose a distance *d* and a direction θ

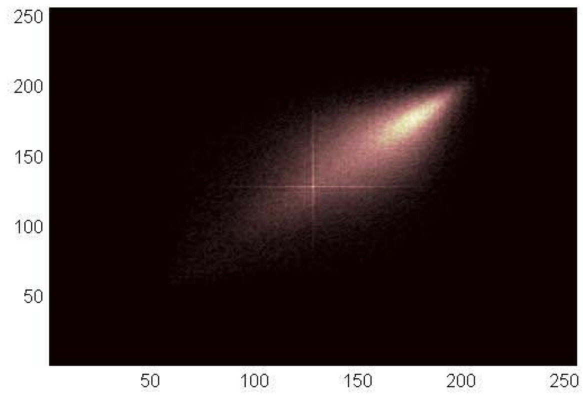


In this example,
d=1 and θ=0

GLCM example (from INF 4300)

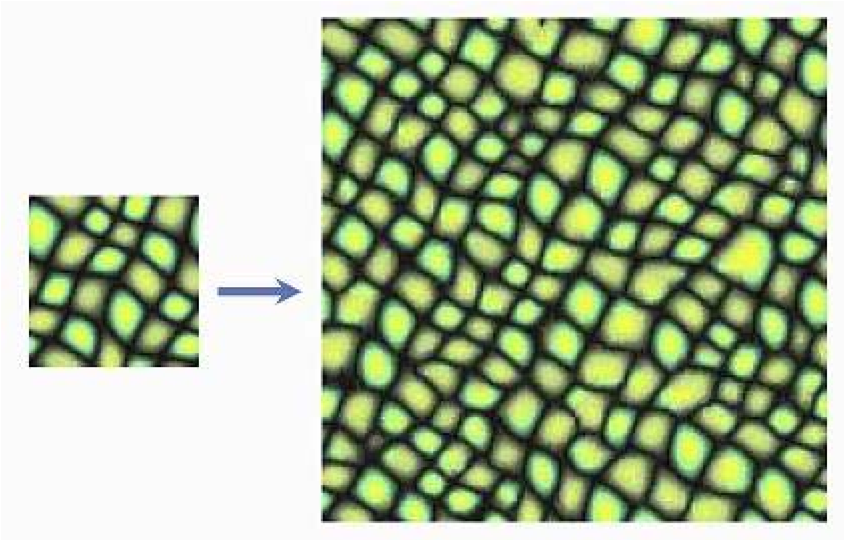






Texture synthesis

- Task: given a texture sample, generate images with similar texture



An introduction

- Traditional methods need texture models
  - Markov models, pyramids, wavelets……

Texture synthesis by deep learning

- Deep nets are well suited for creating texture images.
- One very interesting application is to generate artificial images combining an image and a painting style (a texture).
- This is called Neural Style Transfer

Neural Style Transfer – Content-style loss

- A neural algorithm of artistic style
- Given a texture image with feature F_{ij}^{l} at level l
- Do gradient descent on a noise image to create features P_{ij}^{l}
- Measure the content loss between the feature representation
- $L_{content} = 1/2 \sum_{ij}(F_{ij}^{l} - P_{ij}^{l})^2$

Gram matrices of the filter responses

- Build a style representation on top.
- For each layer, efficiently compute a measure of similarity between filters using Gram matrices:
- $G_{ij}^{l} = \sum_k F_{lj}^{k} F_{jk}^{l}$
- Generate another image that models the style representation of an artistic image by minimizing the MSE of the two Gram-matrices G and A:
- $E_l = const \sum_{ij} \left(G_{ij}^{l}. A_{ij}^{l}\right)^2$
- Set the style loss:
- $L_{style} = \sum_l w_l E_l$

Combine the two images

- Minimise the feature distance between a noise image from the content image and the style representation of a painting
- $L_{total} = \alpha L_{content} + \beta L_{style}$

Examples from A neural algorithm of artistic style

A faster algorithm

- Perceptual losses for real-time style transfer and super-resolution
- Read details yourself.

Learning goals today

- Have an overview of major techniques for visualization
- Know the limitations of visualizing filters directly
- Know that networks can be fooled
- You are not expected to know details of the presented methods
- Have some fun exploring them!