# Lecture 6. Explanation of Neural Networks

activation visualisation · gradients · signals

Tatiana Makhalova
MADE, 23.10.2021

# Understanding ConvNets (in computer vision)

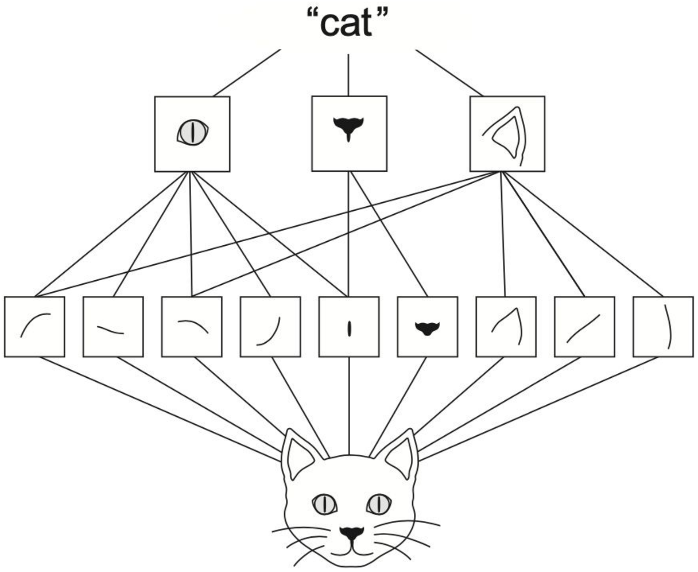
**Main questions** to study

- **How** ConvNets see a dataset?
  - visualization of feature maps / filters
- **What** is a model of a given class?
  - obtaining a “typical image” for a class
- **Why** a given image is classified as an instance of a certain class?
  - identification of image fragments that most affect the ConvNet output

# Part 1. Simple network visualisations

# Activation visualization

**Idea**: to see how the the layers are activated for a given image

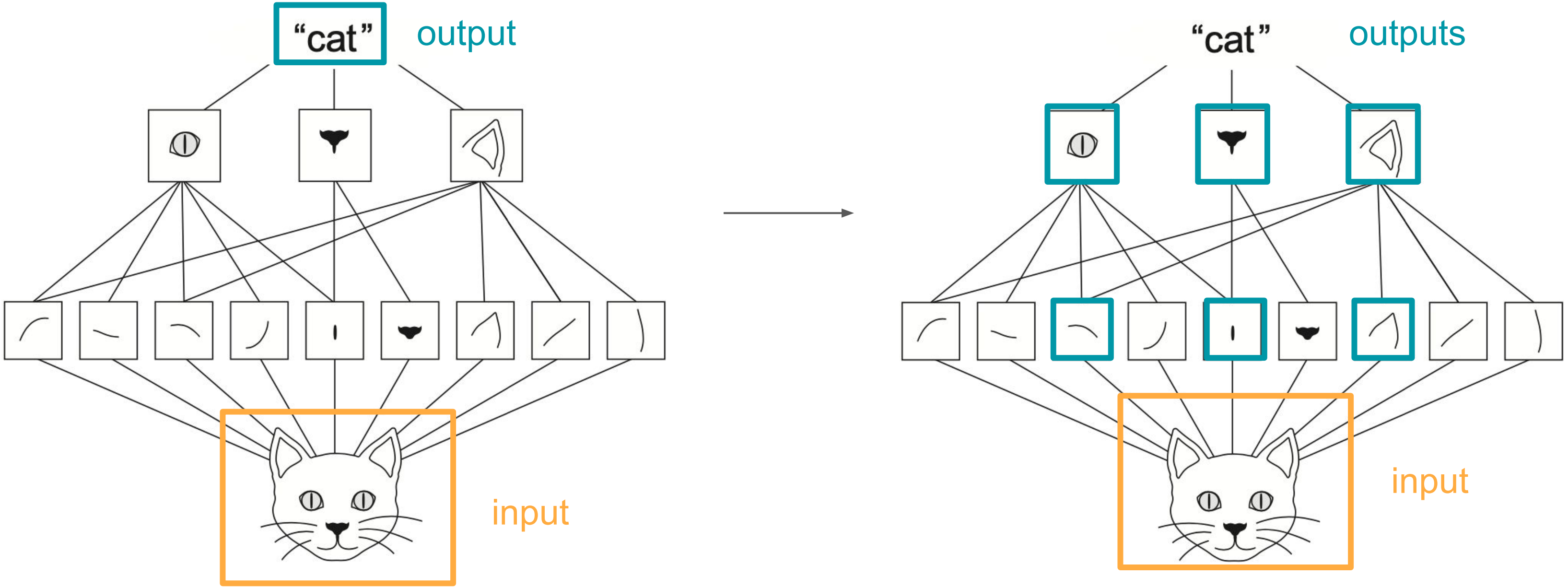



It is known that **lower layers** capture the elementary patterns, e.g., the horizontal, vertical, diagonal lines, textures, etc

**High layers** capture more complicated patterns, e.g., eyes, ears, noses, etc

# How to do

Take the intermediate layers and build a new multi-output model:



# Drawbacks

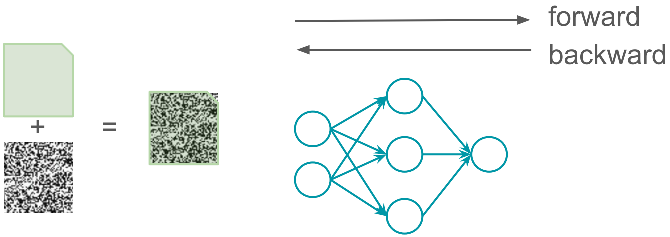
- Gives the explanation for a single image (local explanation)
- Too many filters to inspect

# Part 2. Explaining CNN with the backpropagation-based approaches

# Approaches for attributing importance to the input example

## Perturbation-based approaches

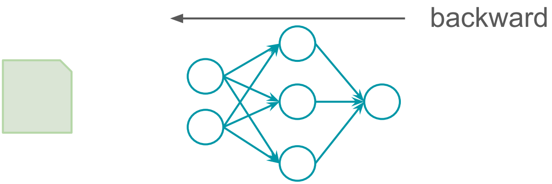
Introducing perturbations to individual inputs or neurons and observe the impact on later neurons or the output




1. Requires 2 passes
2. May underestimate the **importance of features** that have **saturated** their contribution to the output

## Backpropagation-based approaches

Propagating back to the input neurons the output of the neural network



# Sources


**Perturbation-based models:**

Zhou, Jian and Troyanskaya, Olga G. Predicting effects of noncoding variants with deep learning-based sequence model. Nat Methods, 12:931–4, 2015 Oct 2015. ISSN 1548-7105. doi: 10.1038/nmeth.3547.

Zintgraf, Luisa M, Cohen, Taco S, Adel, Tameem, and Welling, Max. Visualizing deep neural network decisions: Prediction difference analysis. ICLR, 2017. URL https://openreview.net/pdf? id=BJ5UeU9xx.

**Backpropagation-based models:**

Gradients: Simonyan, Karen, Vedaldi, Andrea, and Zisserman, An- drew. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.

DeconvNets: Zeiler, Matthew D. and Fergus, Rob. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013. URL http://arxiv.org/ abs/1311.2901.

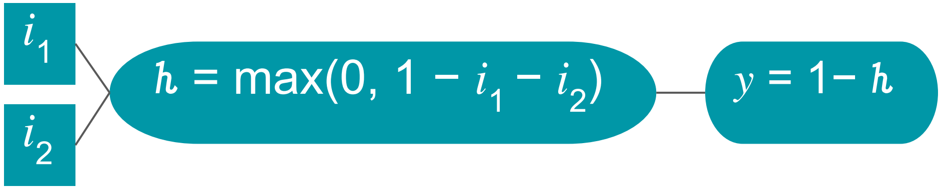
# Saturation problem

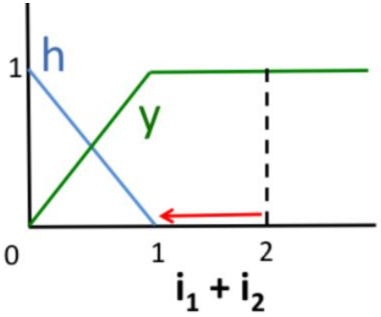
For **perturbation** and **gradient-based** fail to model saturation.

**Example**: $y(i_1,i_2) = 1 - \max(0, 1 - i_1 - i_2)$

The function is equivalent to $y(i_1,i_2) = i_1 + i_2$, if $(i_1 + i_2) < 1$ and

= 0, otherwise




$i_1 = 1$ and $i_2 = 1$ then $y(1,1) = 1$

$i_1 = 0$ and $i_2 = 1$ then $y(0,1) = 1$

$i_1 = 1$ and $i_2 = 0$ then $y(1,0) = 1$

Moreover, for $i_1 + i_2 > 1$ $\partial y/\partial i_1 = \partial y/\partial i_2 = 0$

Credit: Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences." International Conference on Machine Learning. PMLR, 2017

# Roadmap: backpropagation-based methods

Gradients

SmoothGrad

**Functions**

DeconvNet

Guided Backpropagation

PatternNet

**Signals**

propagating back an “assumed signal” toward the signal direction of each neuron

Input * Gradient

LRP

DeepLIFT

Integrated gradients

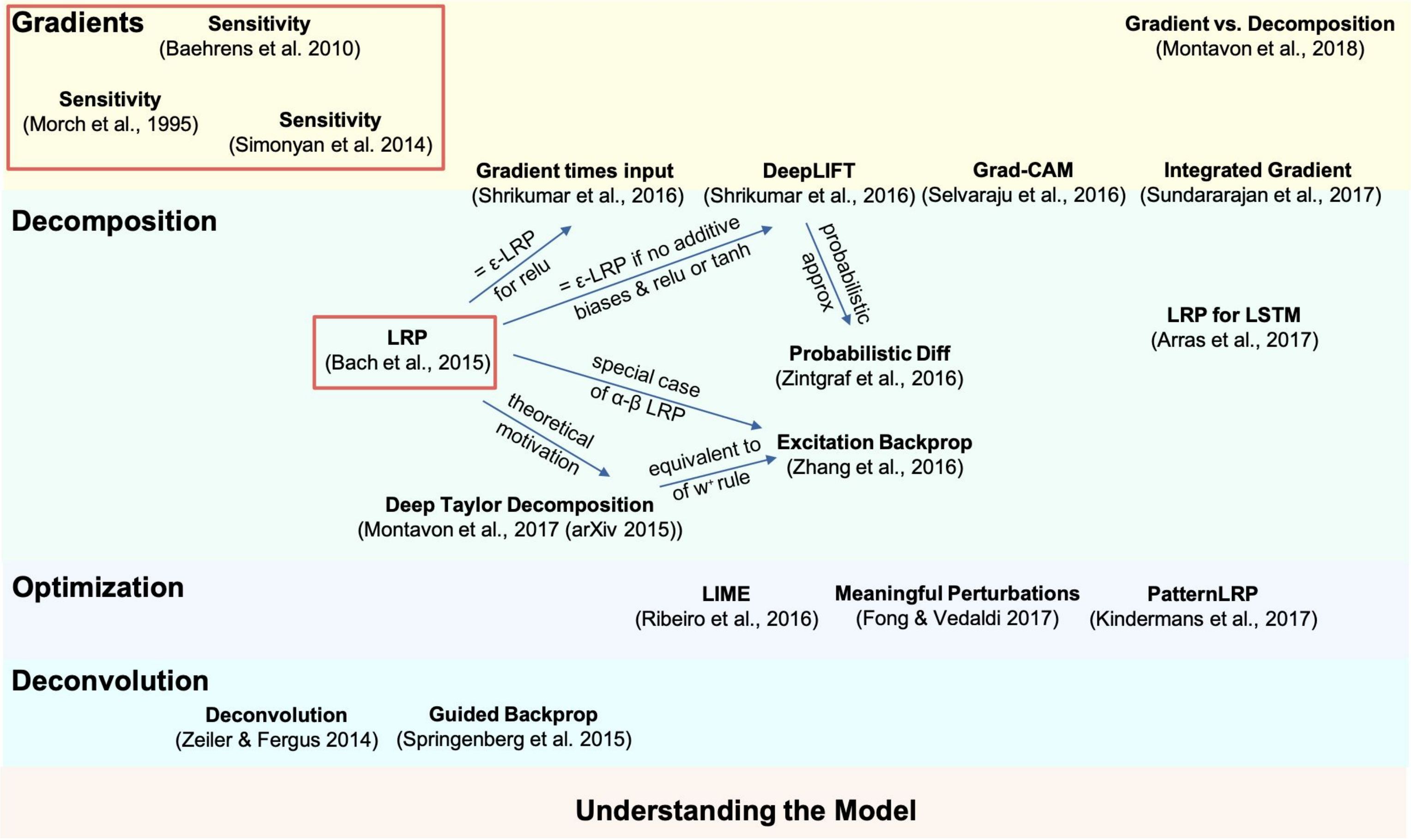
Deep Taylor Decomposition

Pattern Attribution

**Attribution**

assessing how much the signal dimensions contribute the the output through the layers

# Historical remarks on Explaining Predictors



# “Gradents”: class saliency map

**Goal**: generate an image which is representative of the class in the terms of the ConvNet class scoring model

**More formally,** the objective is to find image $I$: $\arg\max_I S_c(I) - \lambda||I||_2^2$

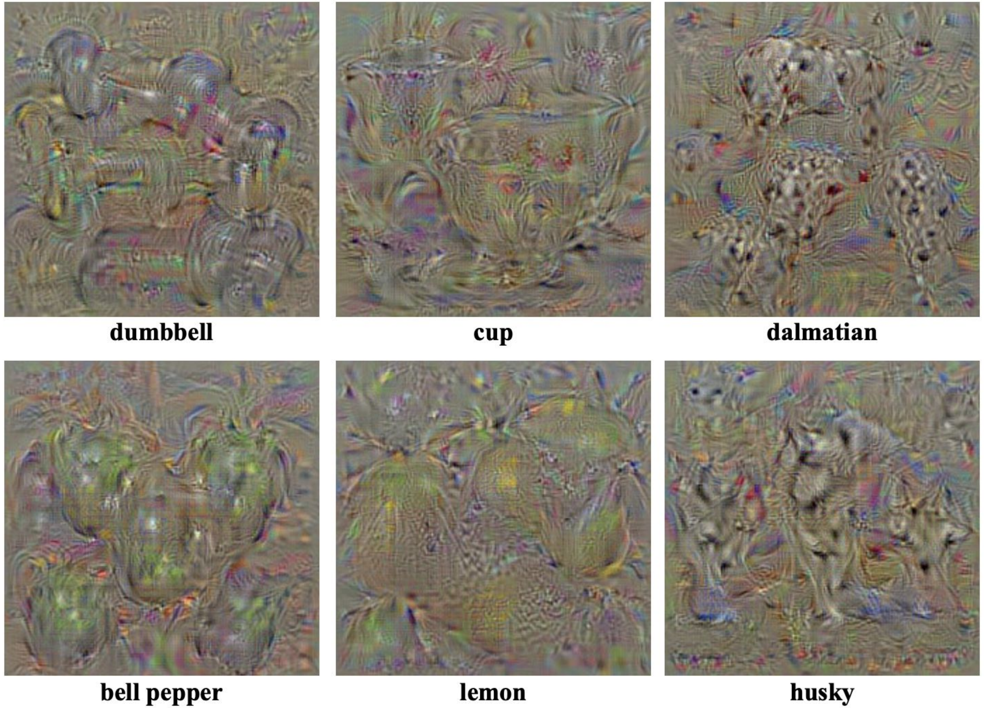
where $S_C(I)$ be the score of the class $c$, computed by the classification layer of the ConvNet for an image $I$

**How**: similar to the ConvNet training except for optimization is performed not w.r.t. weights but w.r.t. the input image $I$

**Details:** instead of considering the class posteriors returned by the softmax layer, i.e., $P_c = \frac{\exp S_c(I)}{\sum_c \exp S_c(I)}$, here one minimizes only the class scores $S_C(I)$ , since maximization of the class posterior can be achieved by minimizing the scores of other classes, thus does not allow to concentrate on the class of interest. In experiments, maximization of the class posteriors did not ensure good results*.

*Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013

# Example of application




the saliency maps for centered and averaged images by classes

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013

# “Gradients”: image-specific saliency map

**Goal**: identify the spatial support of a particular class in a given image

**Motivating example:** In case of the liniar score model for the class $c$: $S_C(I) = w_C^T I + b_C$, thus the magnitude of the elements of $w_C$ defines the importance of the corresponding pixels of $I$ for the class $c$

**Interpretation**: the magnitude of the derivative indicates which pixels need to be changed the least to affect the class score the most. One can expect that such pixels correspond to the object location in the image.

**Reality**: In case of a highly non-linear function $S_C(I)$, given an image $I_0$, we can approximate $S_C(I)$ with a linear function in the neighbourhood of $I_0$ by computing the first-order Taylor expansion $S_C(I) \approx w^T I + b$, where $w = \left.\frac{\partial S_c}{\partial I}\right|_{(I_0)}$

The class saliency map $M \in \mathcal{R}^{m \times n}$ is computed as follows $M_{ij} = |w_{h(i,j)}|$, where $h(i,j)$ for gray-scale images and $M_{ij} = \max_c |w_{h(i,j,c)}|$ for RGB images.

Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013

# Gradients (or backward pass for ConvNets) in detail

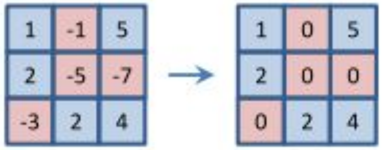
| FORWARD PASS (image classification) | BACKWARD PASS (explanation) |
|---|---|
| - convolutional layer: $$X_{n+1} = X_n * K_n$$ | $$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} * \hat{K_n}$$ |
| - ReLU layer $$X_{n+1} = \max(X_n, 0)$$ | $$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1(X_n > 0)$$ |
| - Max pooling $$X_{n+1}(p) = \max_{q \in \Omega(p)} X_n(q)$$ | $$\frac{\partial f}{\partial X_n(s)} = \frac{\partial f}{\partial X_{n+1}(p)} 1(s = \arg\max_{q \in \Omega(p)} X_n(q))$$ |

$f$ is visualized neuron activity, $K_n$ and $\hat{K_n}$ are the convolutional kernel and its flipped version

# Difference in backpropagating through ReLU

## FORWARD PASS

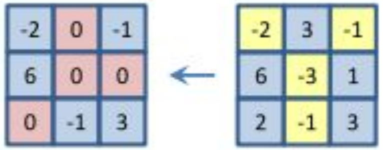
$$X_{n+1} = \max(X_n, 0)$$




## BACKWARD PASS

$$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1(X_n > 0)$$

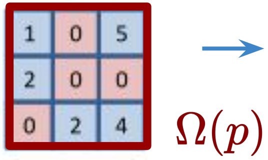
$$\frac{\partial f}{\partial X_{n+1}}$$



# Backpropagation through MaxPooling

FORWARD PASS

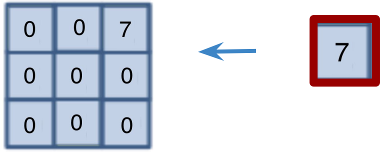
$$X_{n+1}(p) = \max_{q \in \Omega(p)} X_n(q)$$



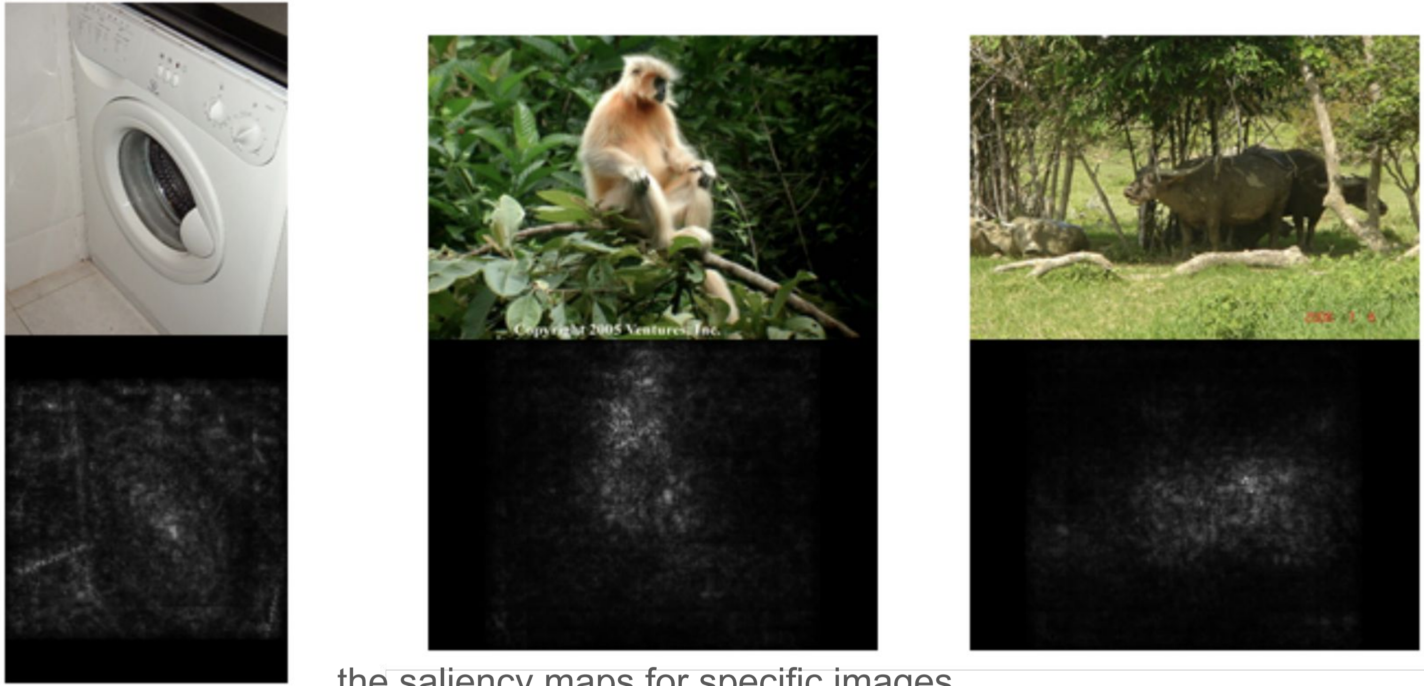

BACKWARD PASS

$$\frac{\partial f}{\partial X_n(s)} = \frac{\partial f}{\partial X_{n+1}(p)} 1(s = \arg\max_{q \in \Omega(p)} X_n(q))$$

$$\frac{\partial f}{\partial X_{n+1}(p)}$$



# Example of application




the saliency maps for specific images

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013

# Input * Gradient

**Idea**: to show the importance for a feature. For example, in a linear system, i.e., $y = w^{\mathsf{T}}x$, it makes sense to consider $w_i x_i$ instead of $w_i$ as the contribution of $x_i$ to the final score $y$

**Side effects**: pixels with values of 0 will never show up on the sensitivity map

# DeconvNet (additional)

**Idea**: to find in unsupervised manner a mid and high-level image representation to visualize the stimuli leading to a certain output

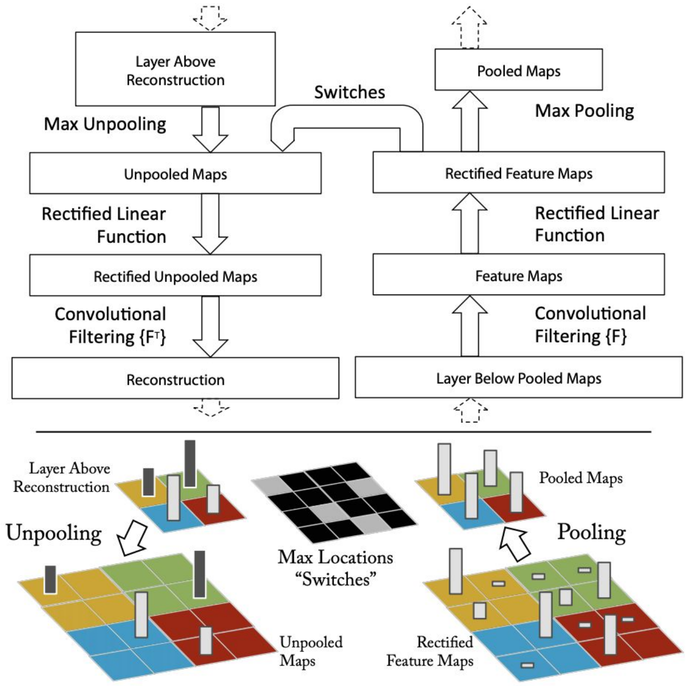



*Figure 1.* Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local max in each pooling region (colored zones) during pooling in the convnet.


Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014. arXiv: 1311.2901

# DeconvNet in detail

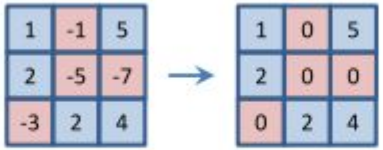
| FORWARD PASS (image classification) | BACKWARD PASS (explanation) |
|---|---|
| - convolutional layer: $$X_{n+1} = X_n * K_n$$ | $$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} * \hat{K_n}$$ |
| - ReLU layer $$X_{n+1} = \max(X_n, 0)$$ | $$\cancel{\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1(X_n > 0)}$$ $$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1\left(\frac{\partial f}{\partial X_{n+1}} > 0\right)$$ |
| - Max pooling $$X_{n+1}(p) = \max_{q \in \Omega(p)} X_n(q)$$ | $$\frac{\partial f}{\partial X_n(s)} = \frac{\partial f}{\partial X_{n+1}(p)} 1(s = \arg\max_{q \in \Omega(p)} X_n(q))$$ |

f is visualized neuron activity, K_n and K_n^hat are the convolutional kernel and its flipped version

# Difference in backpropagating through ReLU

## FORWARD PASS

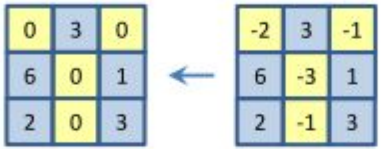
$$X_{n+1} = \max(X_n, 0)$$




## BACKWARD PASS

$$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1\left(\frac{\partial f}{\partial X_{n+1}} > 0\right)$$

$$\frac{\partial f}{\partial X_{n+1}}$$




Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." arXiv preprint arXiv:1412.6806 (2014).

# Guided Backpropagation

**Motivation**: improving Gradients and DeconvNet to get sharper results

**Interesting remark from the paper**: it was shown that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks

Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." arXiv preprint arXiv:1412.6806 (2014)

# Guided Backpropagation

## FORWARD PASS (image classification)

- convolutional layer:

$$X_{n+1} = X_n * K_n$$

- ReLU layer

$$X_{n+1} = \max(X_n, 0)$$

- Max pooling

$$X_{n+1}(p) = \max_{q \in \Omega(p)} X_n(q)$$

## BACKWARD PASS (explanation)

$$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} * \hat{K_n}$$

$$\cancel{\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1(X_n > 0) \quad \frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1\left(\frac{\partial f}{\partial X_{n+1}} > 0\right)}$$

$$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1(X_n > 0) 1\left(\frac{\partial f}{\partial X_{n+1}} > 0\right)$$

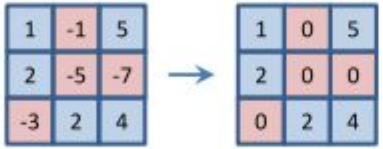
$$\frac{\partial f}{\partial X_n(s)} = \frac{\partial f}{\partial X_{n+1}(p)} 1(s = \arg\max_{q \in \Omega(p)} X_n(q))$$

f is visualized neuron activity, K_n and K_n^hat are the convolutional kernel and its flipped version

# Difference in backpropagating through ReLU

## FORWARD PASS

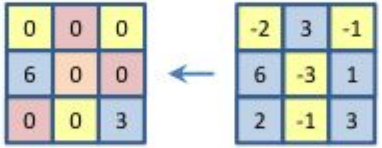
$$X_{n+1} = \max(X_n, 0)$$




## BACKWARD PASS

$$\frac{\partial f}{\partial X_n} = \frac{\partial f}{\partial X_{n+1}} 1(X_n > 0) 1\left(\frac{\partial f}{\partial X_{n+1}} > 0\right)$$

$$\frac{\partial f}{\partial X_{n+1}}$$




Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." arXiv preprint arXiv:1412.6806 (2014).

# Way to compute gradients

1. Gradients (vanilla backpropagation) - the true gradients...

   Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. 2013

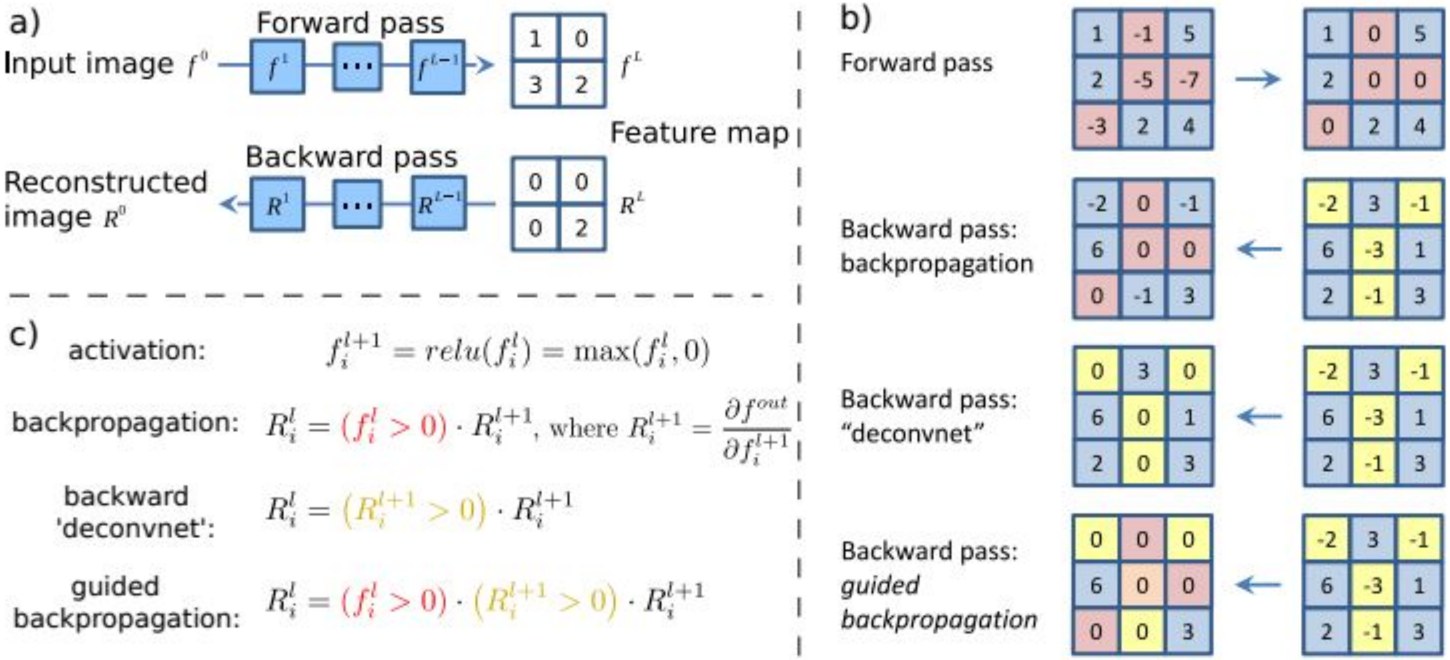
2. DeconvNet-based approach

   Zeiler, Matthew D., and Rob Fergus. Visualizing and understanding convolutional networks. 2014

3. Guided backpropagation

   Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M. Striving for simplicity: The all convolutional net. 2014

# Difference in backpropagating through ReLU




Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." arXiv preprint arXiv:1412.6806 (2014)