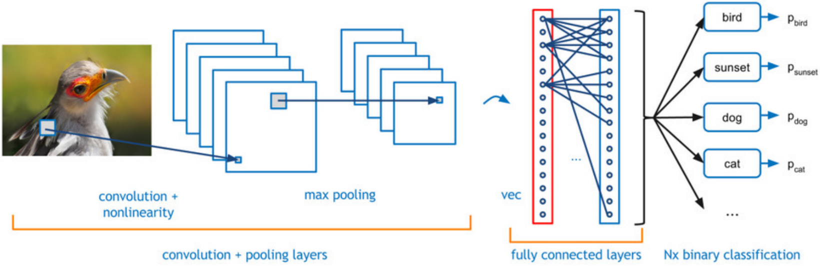
Convolutional network



Convolutional networks – architecture

- Denote
  - X a set of *input* neurons
  - Y a set of *output* neurons
  - Z a set of *all* neurons ($X, Y \subseteq Z$)
- individual neurons denoted by indices i, j etc.
  - ξ_j is the inner potential of the neuron j *after the computation stops*
  - y_j is the output of the neuron j *after the computation stops*

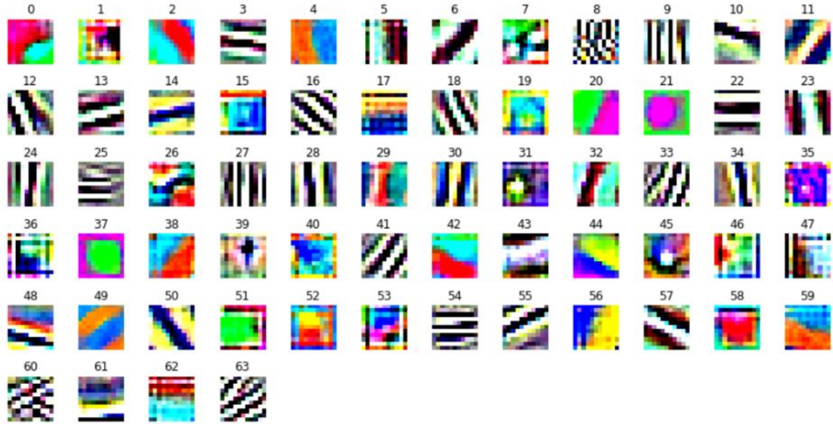
  (define $y_0 = 1$ is the value of the formal unit input)
- w_{ji} is the weight of the connection **from** i **to** j

  (in particular, w_{j0} is the weight of the connection from the formal unit input, i.e. $w_{j0} = -b_j$ where b_j is the bias of the neuron j)
- $j_{\leftarrow}$ is a set of all i such that j is adjacent from i
  (i.e. there is an arc **to** j from i)
- $j^{\rightarrow}$ is a set of all i such that j is adjacent to i
  (i.e. there is an arc **from** j to i)
- $[ji]$ is a set of all connections (i.e. pairs of neurons) sharing the weight w_{ji}.

Visualzation methods

- Visualize weights
- Visualize most "important" inputs for a given class
- Visualize effect of input perturbations on the output
- Construct a local "interpretable" model

Alex-net - filters of the first convolutional layer



64 filters of depth 3 (RGB).

Combined each filter RGB channels into one RGB image of size 11x11x3.

Maximizing input

Assume a trained model giving a score for each class given an input image.

- Denote by $y_i(I)$ the value of the output neuron i on an input image I.
- Maximize

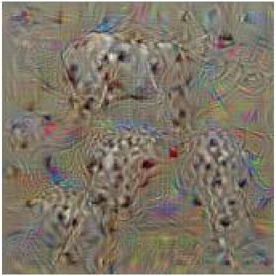
  $$y_i(I) - \lambda \|I\|_2^2$$

  over all images I.
- A maximum image computed using gradient descent.
- Gives the most "representative" image of the class c.

Maximizing input - example

dumbbell

cup



dalmatian

Image specific saliency maps

- Let us fix an output neuron i and an image I_0.
- Rank pixels in I_0 based on their influence on the value $y_i(I_0)$.

Image specific saliency maps

- Let us fix an output neuron i and an image I_0.
- Rank pixels in I_0 based on their influence on the value $y_i(I_0)$.
- Note that we can approximate y_i locally around I_0 with the linear part of the Taylor series:

$$y_i(I) \approx y_i(I_0) + w^T(I - I_0) = w^T I + (y_i(I_0) - w^T I_0)$$

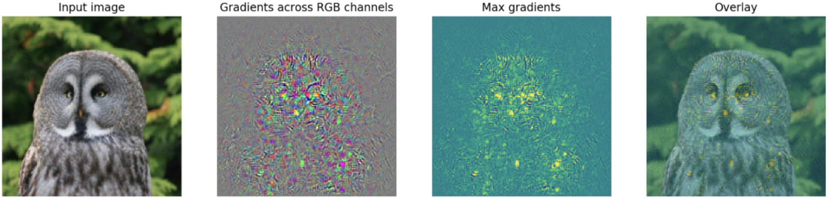
  where

$$w = \frac{\delta y_i}{\delta I}(I_0)$$

- **Heuristics:** The magnitude of the derivative indicates which pixels need to be changed the least to affect the score most.

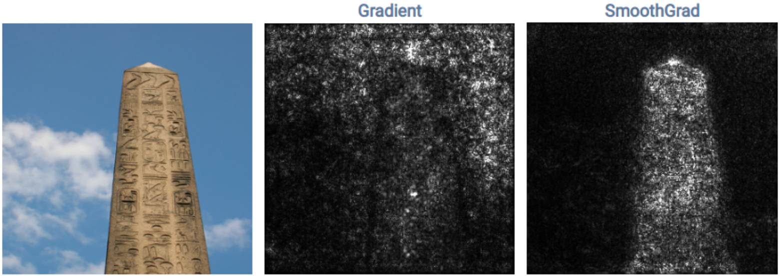
Saliency maps - example

Saliency maps - example



Quite noisy, the signal is spread and does not tell much about the perception of the owl.

SmoothGrad

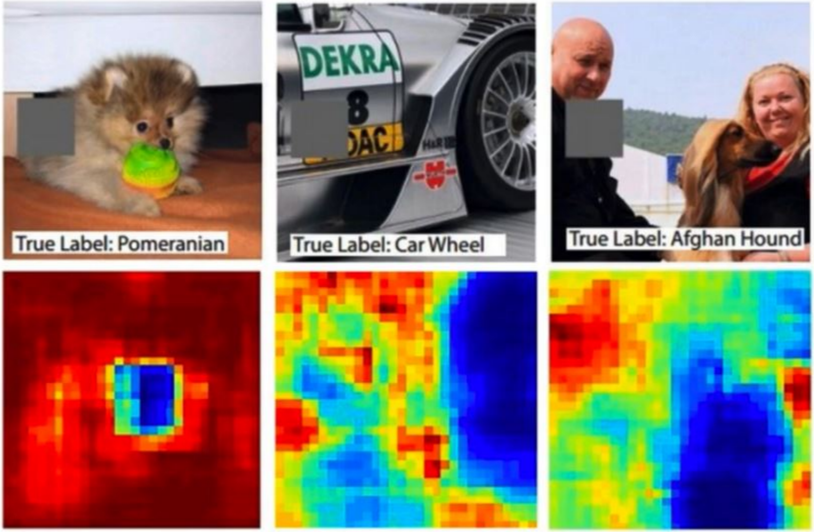


Average several saliency maps of noisy copies of the input.

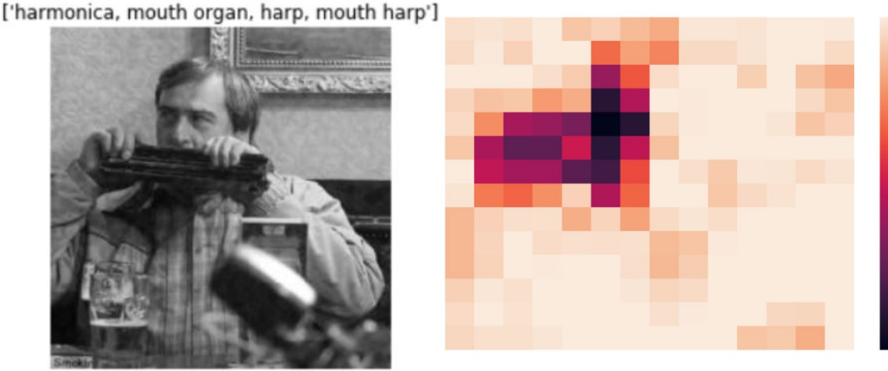
Occlusion

- Systematically cover parts of the input image.
- Observe the effect on the output value.
- Find regions with the largest effect.

Occlusion - example



Occlusion - example



LIME - for images

Let us fix an image I_0 to be explained.

Outline:

- Consider superpixels of I_0 as interpretable components.
- Construct a linear model approximating the network aroung the image I_0 with weights corresponding to the superpixels.
- Select the superpixels with weights of large magnitude as the important ones.

Original Image

Interpretable Components

Superpixels as explainable components

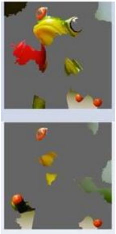
Original Image

Interpretable Components

Denote by $P_1, \ldots, P_\ell$ all superpixels of I_0.

Consider binary vectors $\vec{x} = (x_1, \ldots, x_\ell) \in \{0, 1\}^\ell$.

Each such vector $\vec{x}$ determines a "subimage" $I[\vec{x}]$ of I_0 obtained by removing all P_k with $x_k = 0$.



LIME

- Let us fix an output neuron i, we dnote by $y_i(I)$ the value of i for a given input image I.
- Given an image I_0 to be interpreted, consider the following training set:

$$\mathcal{T} = \{(\vec{x}_1, y_i(I_0[\vec{x}_1])), \ldots, (\vec{x}_p, y_i(I_0[\vec{x}_p])\}$$

  Here $\vec{x}_h = (x_{h1}, \ldots, x_{h\ell})$ are (some) binary vectors of $\{0,1\}^\ell$. E.g. randomly selected.

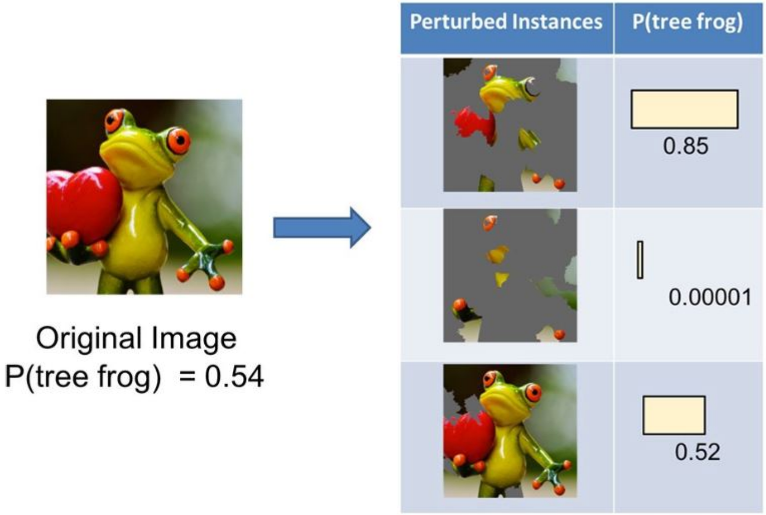
LIME

- Let us fix an output neuron i, we dnote by $y_i(I)$ the value of i for a given input image I.
- Given an image I_0 to be interpreted, consider the following training set:

$$\mathcal{T} = \{(\vec{x}_1, y_i(I_0[\vec{x}_1])), \ldots, (\vec{x}_p, y_i(I_0[\vec{x}_p])\}$$

  Here $\vec{x}_h = (x_{h1}, \ldots, x_{h\ell})$ are (some) binary vectors of $\{0,1\}^\ell$. E.g. randomly selected.
- Train a linear model (ADALINE) with weights $w_1, \ldots, w_\ell$ on $\mathcal{T}$ minimizing the mean-squared error
  (+ a regularization term making the number of non-zero weights as small as possible).
  Intuitively, the linear model approximates the networks on the "subimages" of I obtained by removing unimportant superpixels.
- Inspect the weights (magnitude and sign).

LIME - example



LIME - example

Original Image
P(tree frog) = 0.54

Perturbed Instances	P(tree frog)
	0.85
	0.00001
	0.52

Explanation

LIME - example

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

LIME - example

(a) Husky classified as wolf

(b) Explanation