Interpretability

"Interpretability is the degree to which a human can understand the cause of a decision." Tim Miller, 2017

Diane Lingrand

POLYTECH NICE SOPHIA
UNIVERSITÉ CÔTE D'AZUR

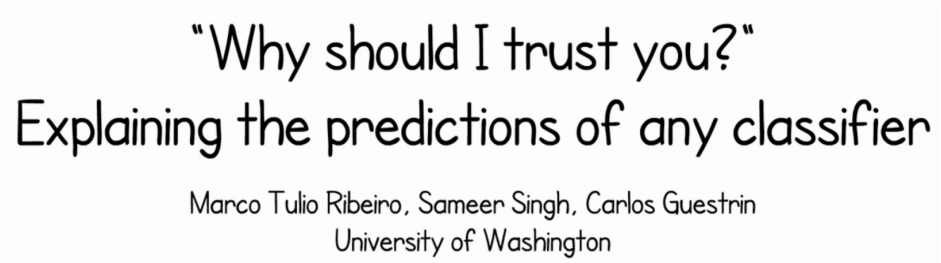
Polytech SI5

2022 - 2023

Outline

Outline

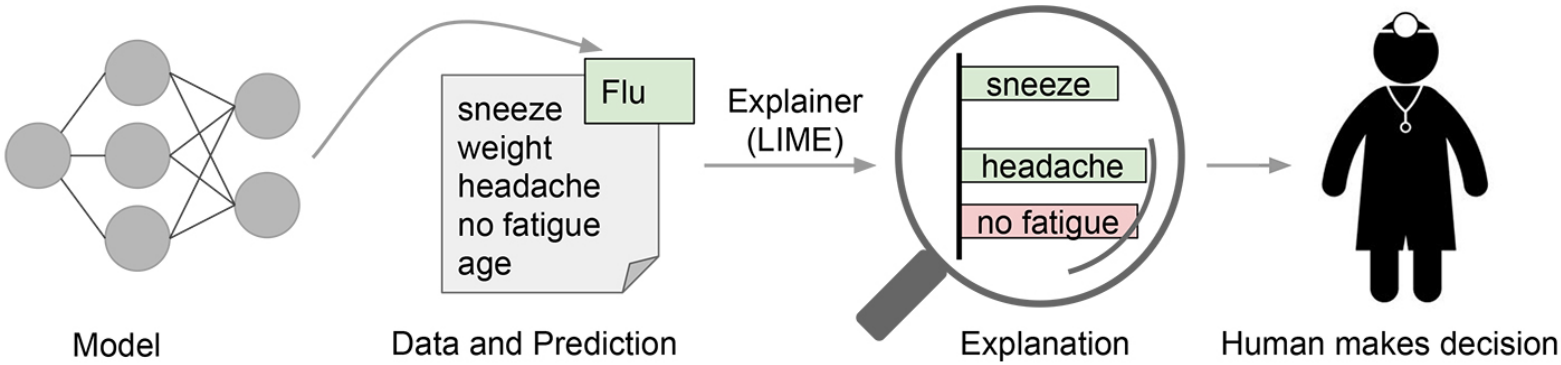
Introduction



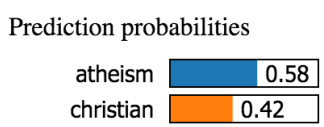
https://www.youtube.com/watch?time_continue=165&v=hUnRCxnydCc&feature=emb_logo

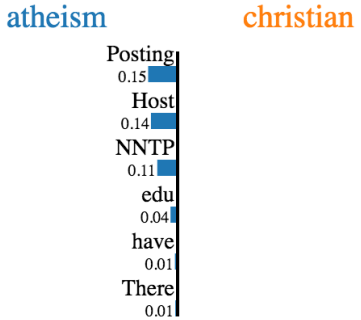
Motivation

- Explaining the decision for decision making



- Trust in the machine learning model
  - Example using the 20newsgroups dataset and highlighting 2 classes : Christianity and Atheism. RF with 500 trees, test acc. 92.4%





Text with highlighted words
From: johnchad@triton.unm.edu (jchadwic)
Subject: Another request for Darwin Fish
Organization: University of New Mexico, Albuquerque
Lines: 11
NNTP-Posting-Host: triton.unm.edu

Hello Gang,

There have been some notes recently asking where to obtain the DARWIN fish.
This is the same question I have and I have not seen an answer on the
net. If anyone has a contact please post on the net or email me.

- *"Why Should I Trust You ?" Explaining the Predictions of Any Classifier* by Ribeiro, Singh and Guestrin, 2016 `https://arxiv.org/pdf/1602.04938v1.pdf`

Outline

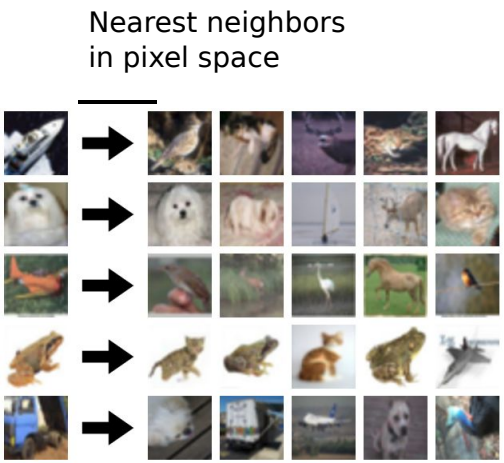
Last Layer: Nearest Neighbors

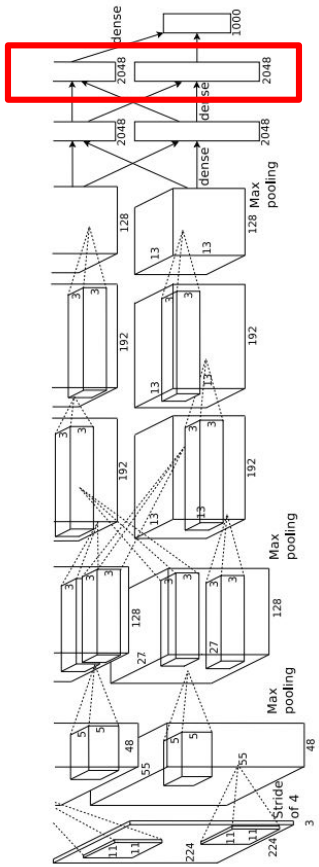
4096-dim vector

Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

Test image L2 Nearest neighbors in feature space

Nearest neighbors in pixel space





look similar

same semantic

Description layer

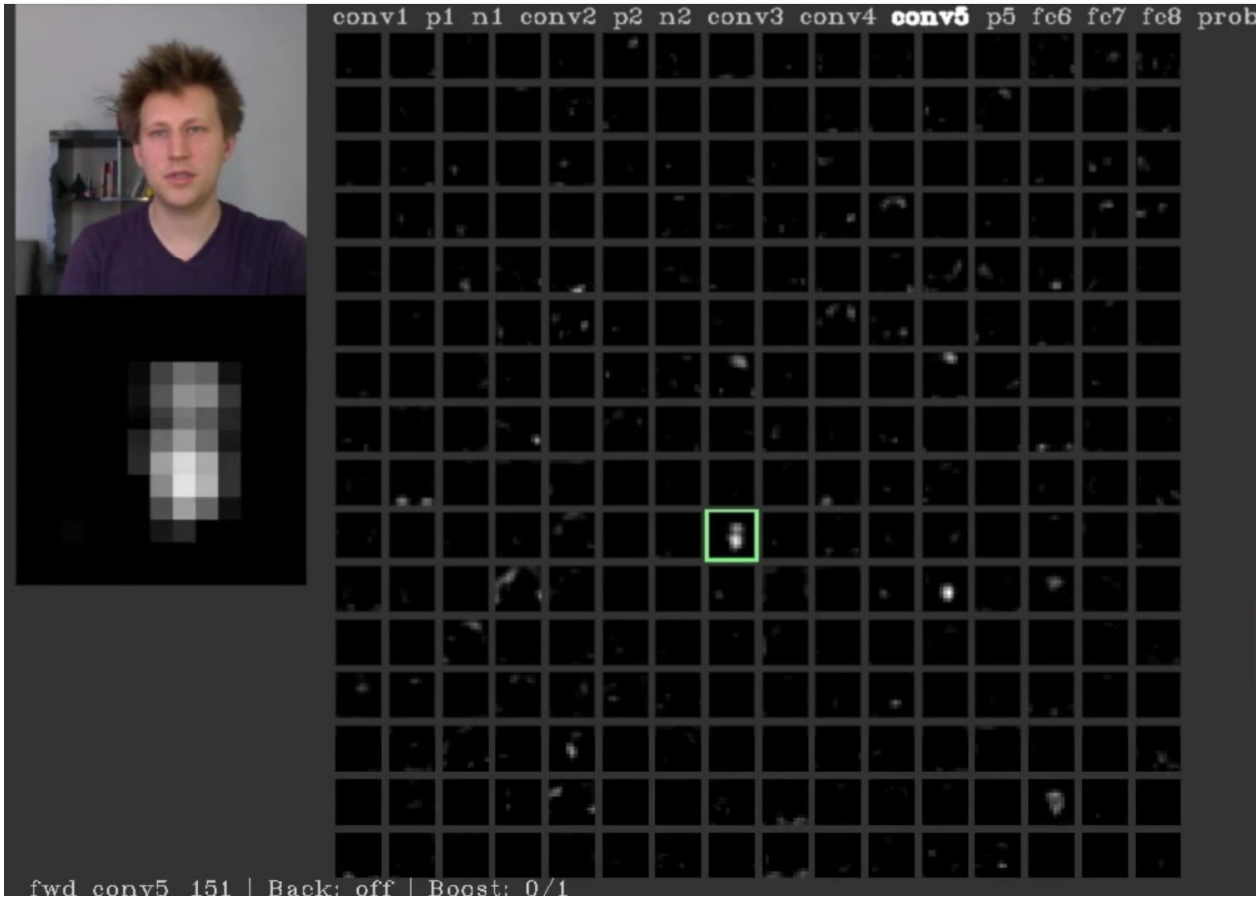
- 4096 dimensional vectors
- tSNE for dimension reduction
- nearest neighbor
- see `https://cs.stanford.edu/people/karpathy/cnnembed/`

Activations

- Look at activations (e.g. conv5)
- Maximally activated patches
- Occlusions for saliency detection
- Saliency maps
- DeconvNet
- CAM
- Grad-CAM

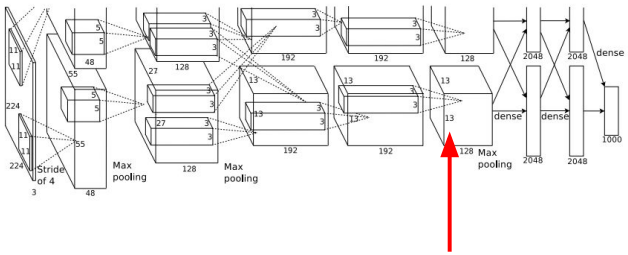
Visualising activations

- conv5 feature map (128x13x13)
- as 128 gray images (13x13)

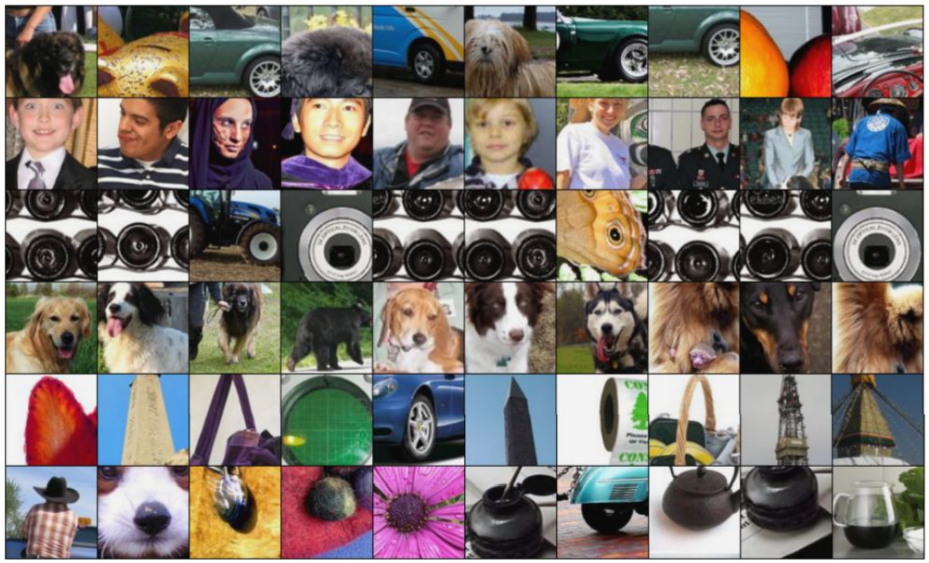


Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Maximally activating patches



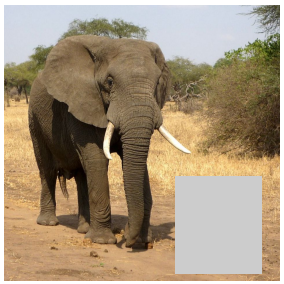
- pick a layer and a channel
  - e.g. conv5 (128x13x13) and channel 17
- run many images through the network
  - record values of chosen channel
- visualize image patches corresponding to max. activations

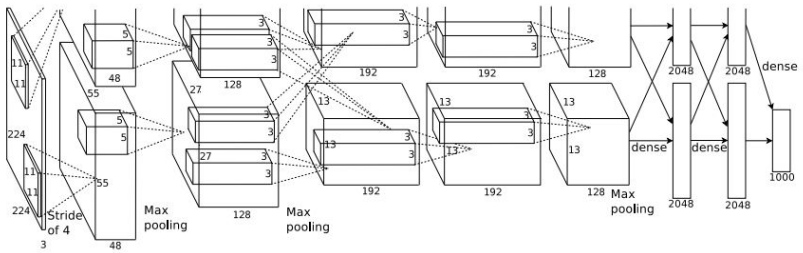


Springenberg et al, "Striving for Simplicity : The All Convolutional Net", ICLR

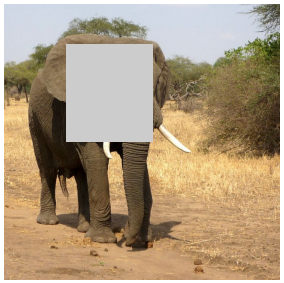
Which are the "important" pixels ?

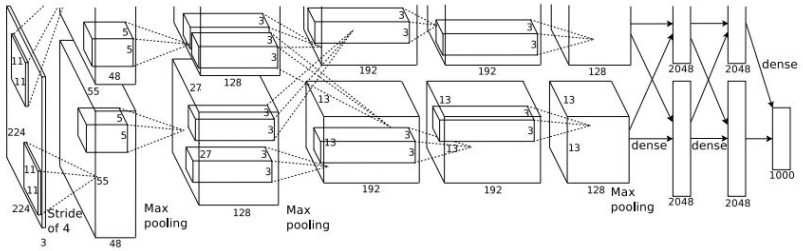
- apply a mask to the image before feed forward to CNN
- check how much predicted probabilities change





P(elephant) = 0.95



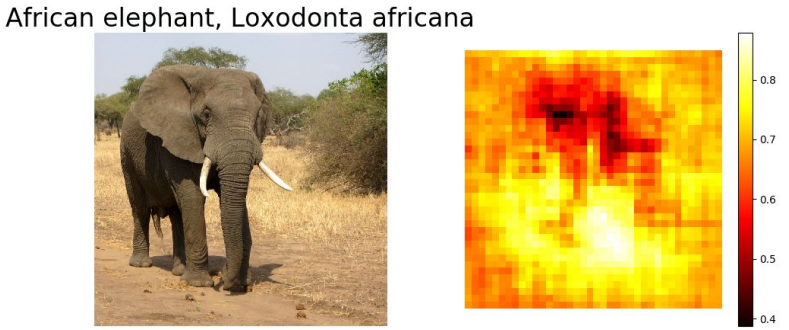


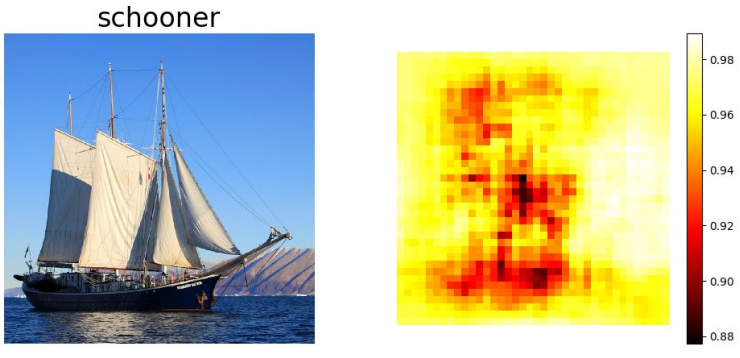
P(elephant) = 0.75

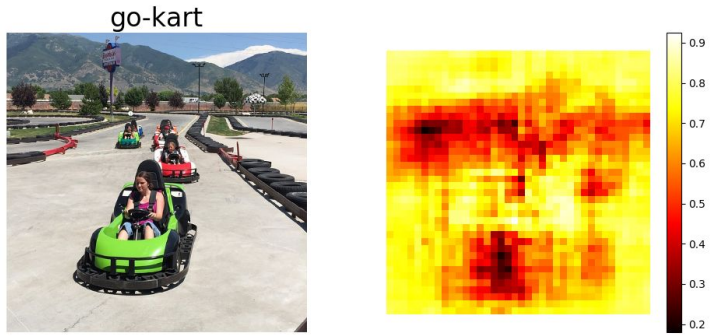
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks" ECCV 2014

`https://arxiv.org/pdf/1311.2901.pdf`

Occlusion map





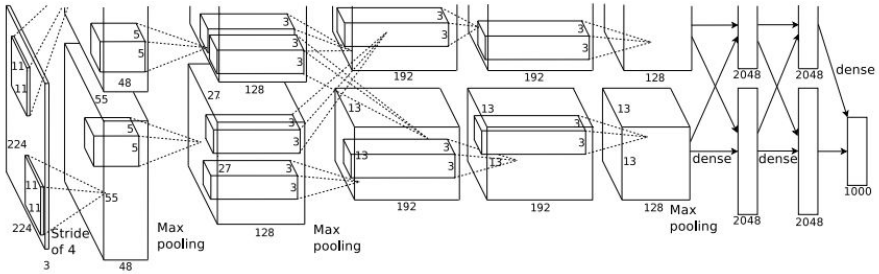


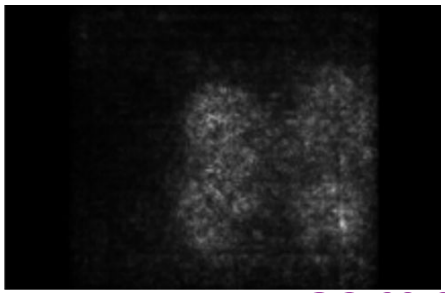
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks" ECCV 2014

Saliency by back-propagation

Forward pass:
Compute probabilities

Dog





Dog

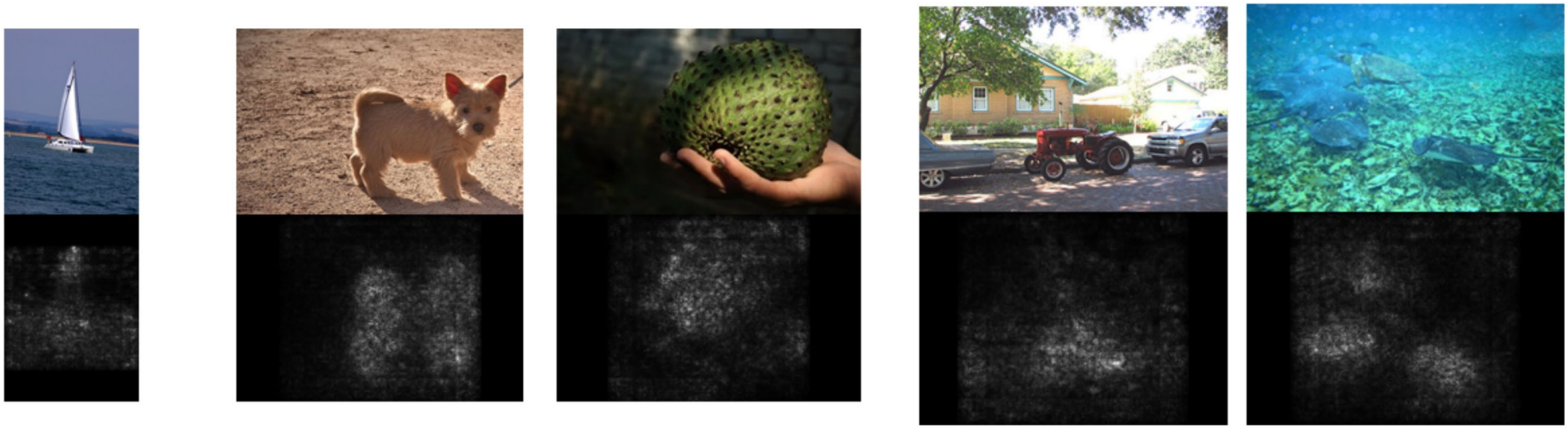
Back-propagation :
compute gradient of class score w.r.t. pixels
take abs. value and max over RGB channels

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks : Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Computation of gradients

In tensorflow, use `GradientTape`:
`https://www.tensorflow.org/guide/autodiff`

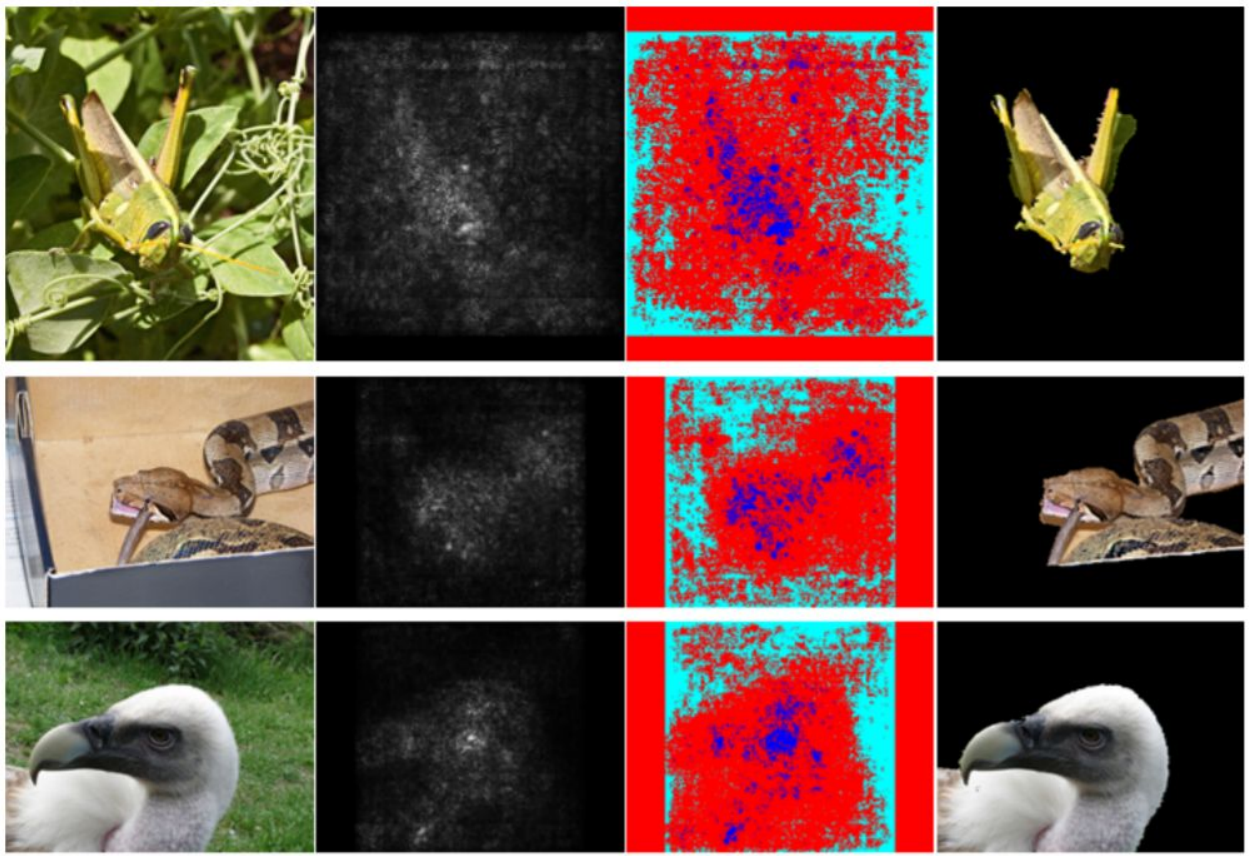
Saliency : examples



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks : Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Segmentation using saliency maps

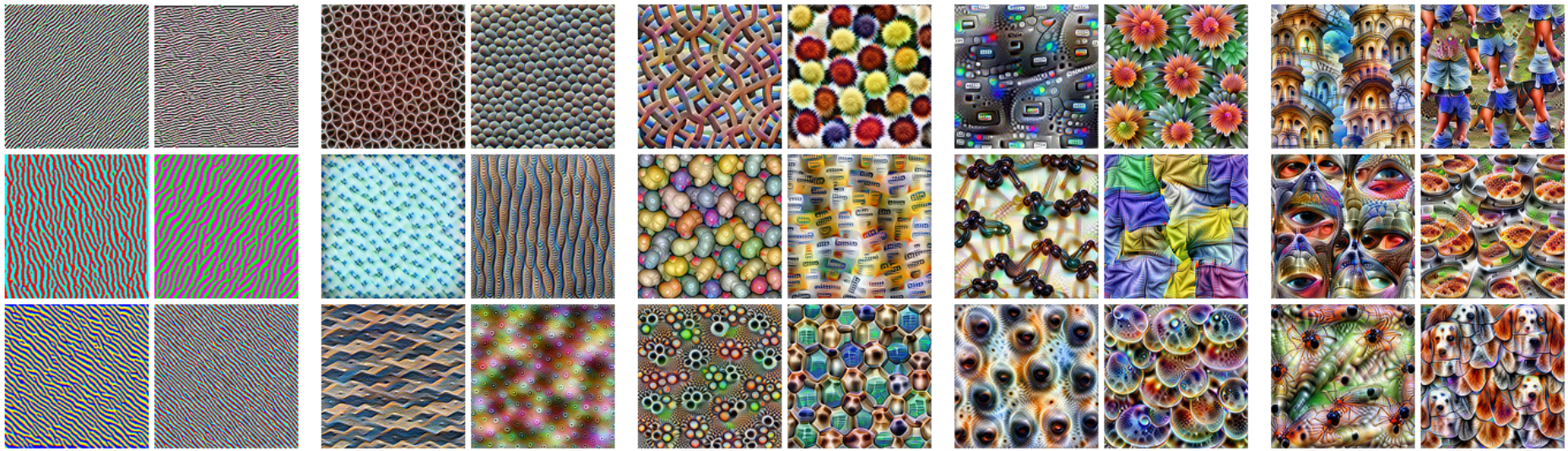
- Using GrabCut (2004)
  - GMM for background / foreground
  - graph connecting pixels with Source node and Sink node
  - graph cut to minimize a cost function



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks : Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014

Feature Visualization, Google, 2017

from `https://distill.pub/2017/feature-visualization/`



Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

CAM : Class Activation Maps (2015)

Discriminative region used by CNN to identify the class.

Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
Computer Science and Artificial Intelligence Laboratory, MIT

`https://arxiv.org/pdf/1512.04150.pdf`

inside CAM

FOR CAM THE LAST DENSE LAYER LEARNING IS MENDATORY.
We have to multiply the weights with feature maps

- CNN network for image classification without FC layer(e.g. Googlenet Inception v1)
- use a GAP (Global Average Pooling) layer before the softmax layer

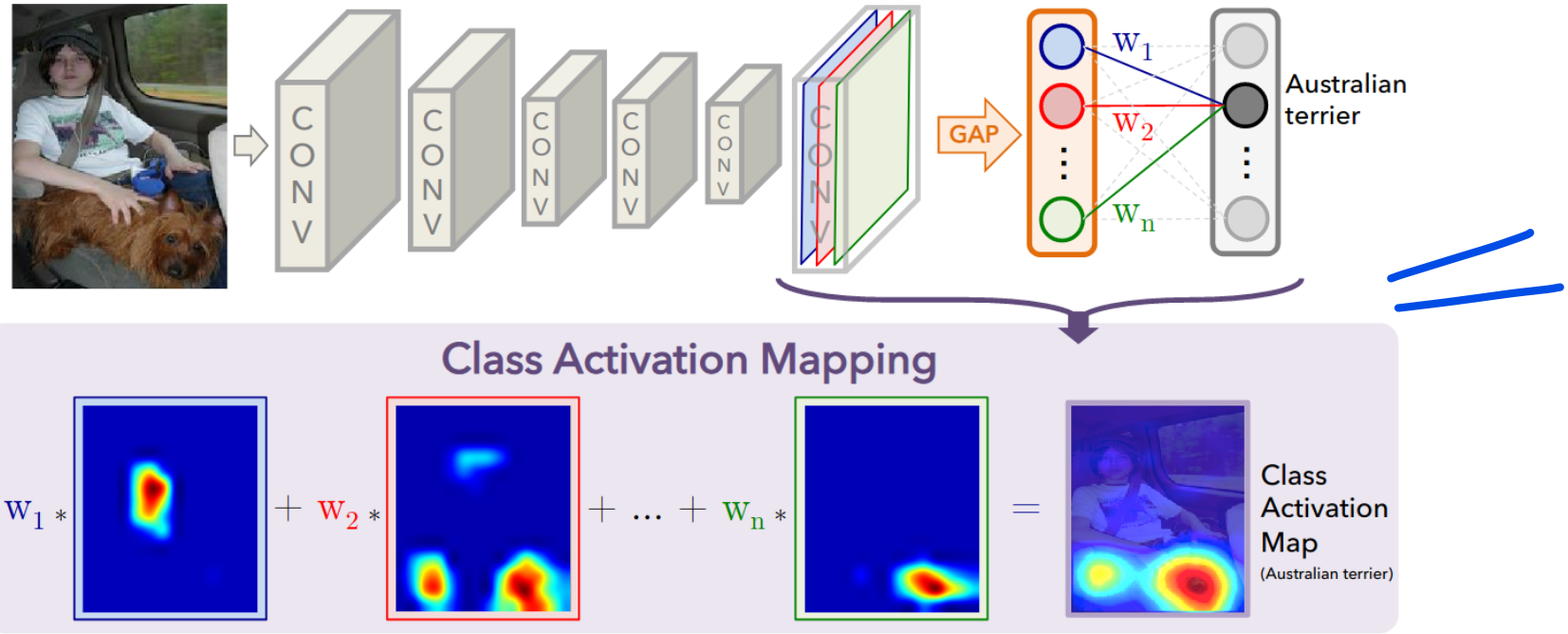


illustration from `https://arxiv.org/pdf/1512.04150.pdf`

- upsampling

CAM and EfficientNet

From the EfficientNet paper
`https://arxiv.org/pdf/1905.11946v5.pdf`.

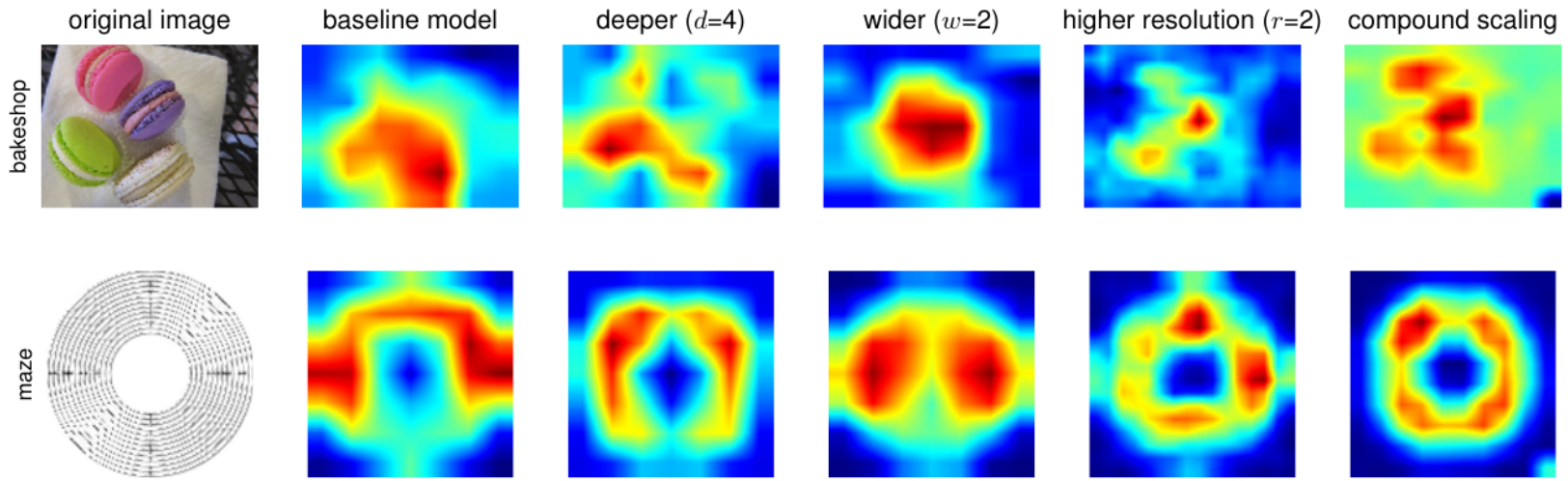


Figure 7. **Class Activation Map (CAM) (Zhou et al., 2016) for Models with different scaling methods-** Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

Implementation of CAM

- needs a learned CNN
  - remove the top (fc layers)
  - add a global average pooling (GAP)
  - add a dense layer for classification with a softmax activation
  - learn this last layer
- Identify :
  - the detected class

```
predClasses = model.predict(xTest)
index = 800 # one of the images
cl = np.argmax(predClasses[index])
```

  - the weights w_i, $1 \leq i \leq n$

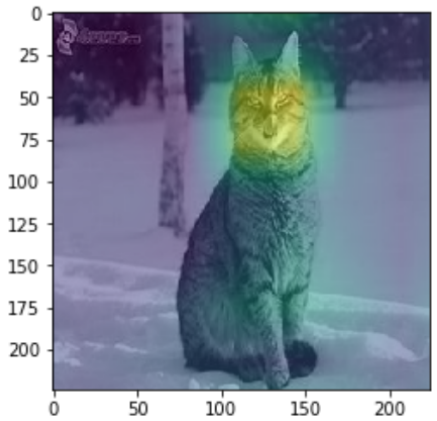
```
layer = model.get_layer('dense')
we = layer.get_weights()[0]
w = we[:,cl]
```

  - the activation maps M_i, $1 \leq i \leq n$

```
allFeatures = VGGmodel.predict(xTest)
maps = allFeatures[index]
```

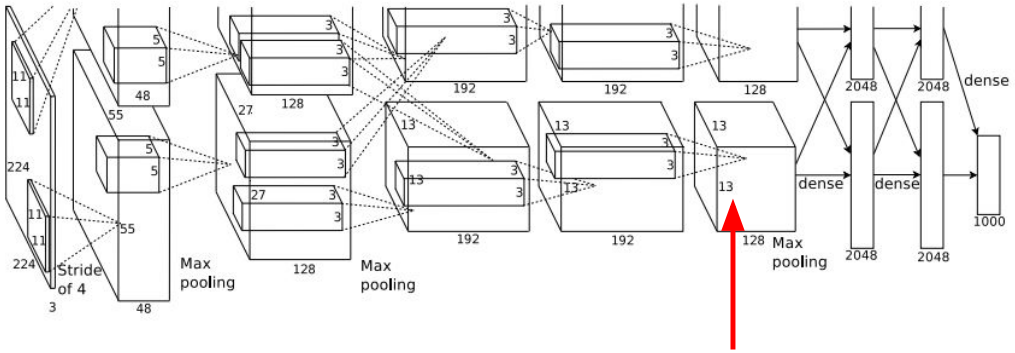
- Compute :
  - the weighted sum : $\sum_{i=1}^{n} w_i M_i$: `cam = np.inner(maps,w)`
  - upscaling

```
from skimage.transform import resize
plt.imshow(resize(camimg, (224,224)))
plt.imshow(image[:,:,1],cmap=plt.cm.gray, alpha=0.5)
plt.show()
```



Guided Back-propagation

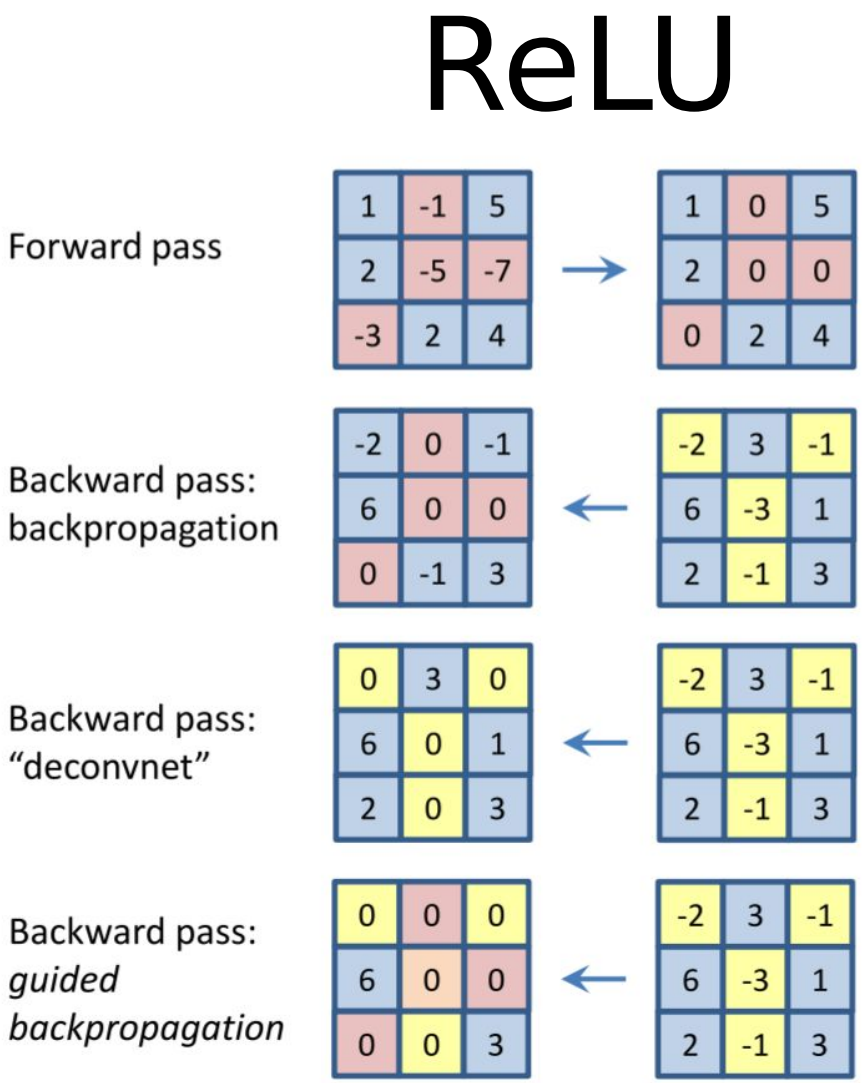
Back track the



- Remember the back-progation :
  - Pick a single intermediate neuron
  - e.g. one from the conv5 feature map
  - gradient of neuron value w.r.t. image pixels
- Guided Back-propagation
  - goal : detect which pixels are responsible of neurons activation
  - avoid non activation information $\Rightarrow$ neg gradients set to 0
  - propagation of only positive gradients

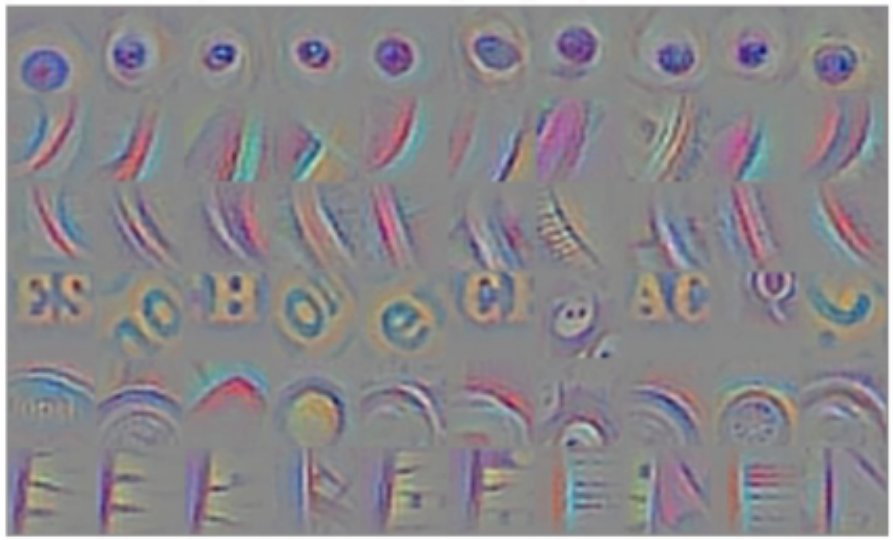
Guided Back-propagation

Images come out nicer if you only backpropagate positive gradients through each ReLU.



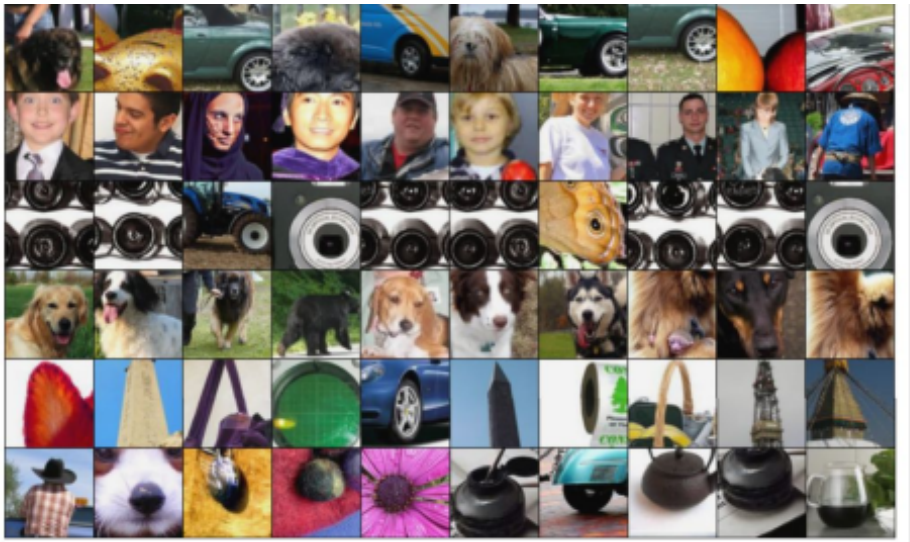
Intermediate features using Guided Back-propagation

Maximally activating patches
(Each row is a different neuron)



Guided Backprop

Intermediate features using Guided Back-propagation



Maximally activating patches
(Each row is a different neuron)

Guided Backprop

Remember what "deconvolution" means

Gaussian smoothing filter

- Example : gaussian filter, dim 3x3, stride 1, paddind VALID (discard borders) applied one a 4x4 grey levels image :

120	110	80	84
105	80	60	70
110	90	54	62
112	94	62	44

$*$

1	2	1
2	4	2
1	2	1

$/16 \Rightarrow$

88.375	72.125
90	64.5

  - From

88.375	72.125
90	64.5

and the filter, how to recover the 4x4 image ?

Transposed convolution

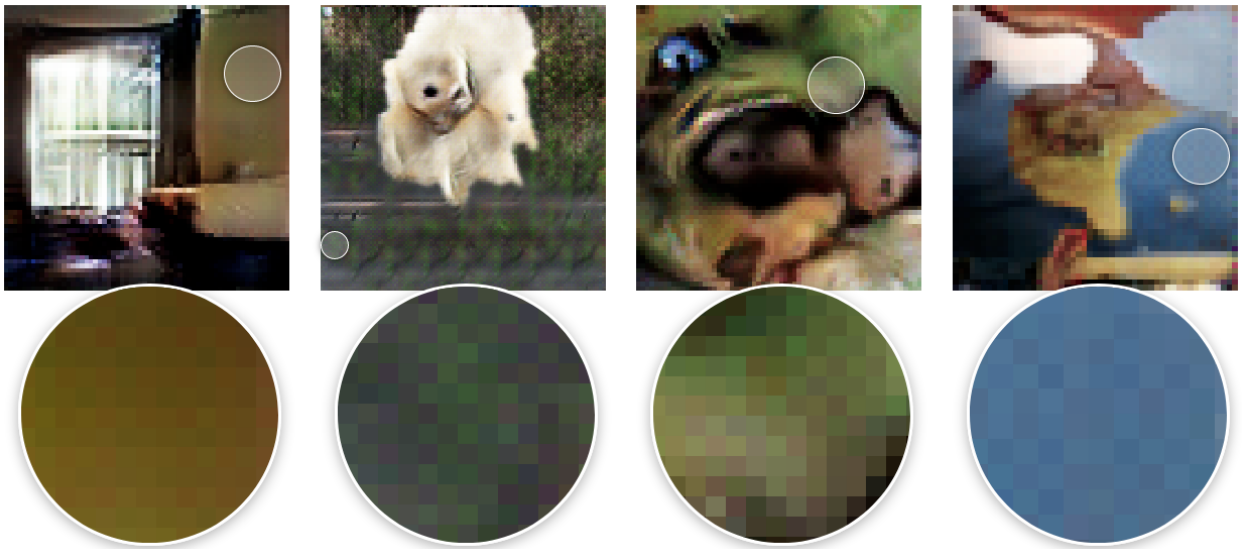
- also called *backwards convolution* or *deconvolution*
- rewrite the convolution operation :

$$\underbrace{\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 2 & 4 & 2 & 0 & 1 & 2 & 1 \end{bmatrix}}_{\mathbf{C}} \cdot \underbrace{\begin{bmatrix} 120 \\ 110 \\ 80 \\ 84 \\ 105 \\ 80 \\ 60 \\ 70 \\ 110 \\ 90 \\ 54 \\ 62 \\ 112 \\ 94 \\ 62 \\ 44 \end{bmatrix}}_{\mathbf{I_1}} = \underbrace{\begin{bmatrix} 88.375 \\ 72.125 \\ 90 \\ 64.5 \end{bmatrix}}_{\mathbf{I_2}}$$

- transpose the matrix $\mathbf{C}$: $\mathbf{I_1} = \mathbf{C^T I_2}$
  - $\mathbf{C}$: not the same coefficients but learnable
  - transposed convolution : upsampling

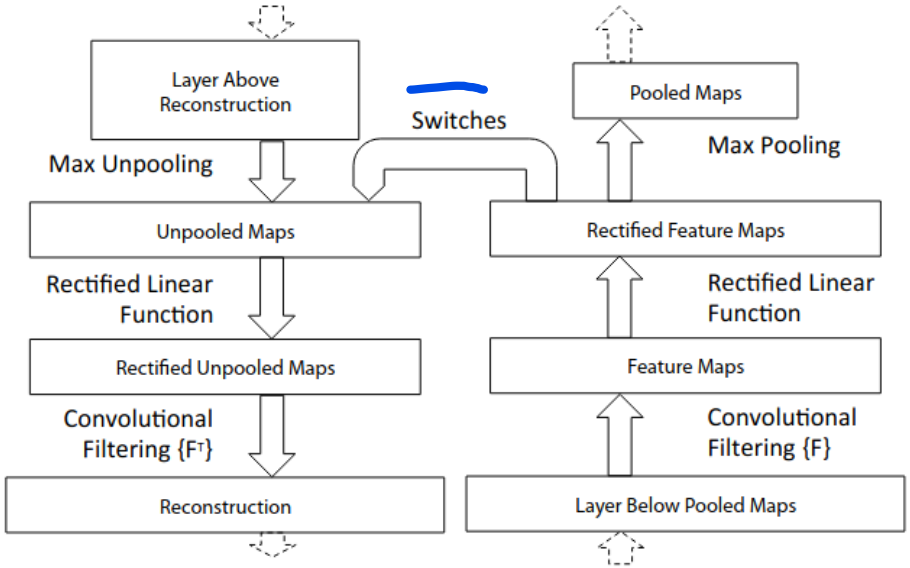
Upsampling or deconvolution ?

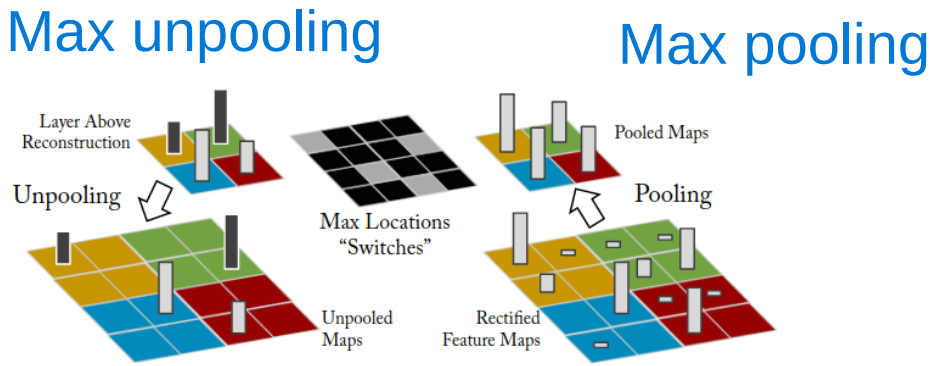
- transposed convolution matrix :
  - bigger than the original kernel
  - lots of zeros
  - in fact, could be rewritten as a convolution with different kernel, strides and padding
- checkerboard artifacts



  - nice illustrations and demos on `https://distill.pub/2016/deconv-checkerboard/`
  - this is why it's better to separate upsampling and deconvolution
- See `https://arxiv.org/pdf/1603.07285v1.pdf` for a complete discussion about dimensions.

Deconvolution

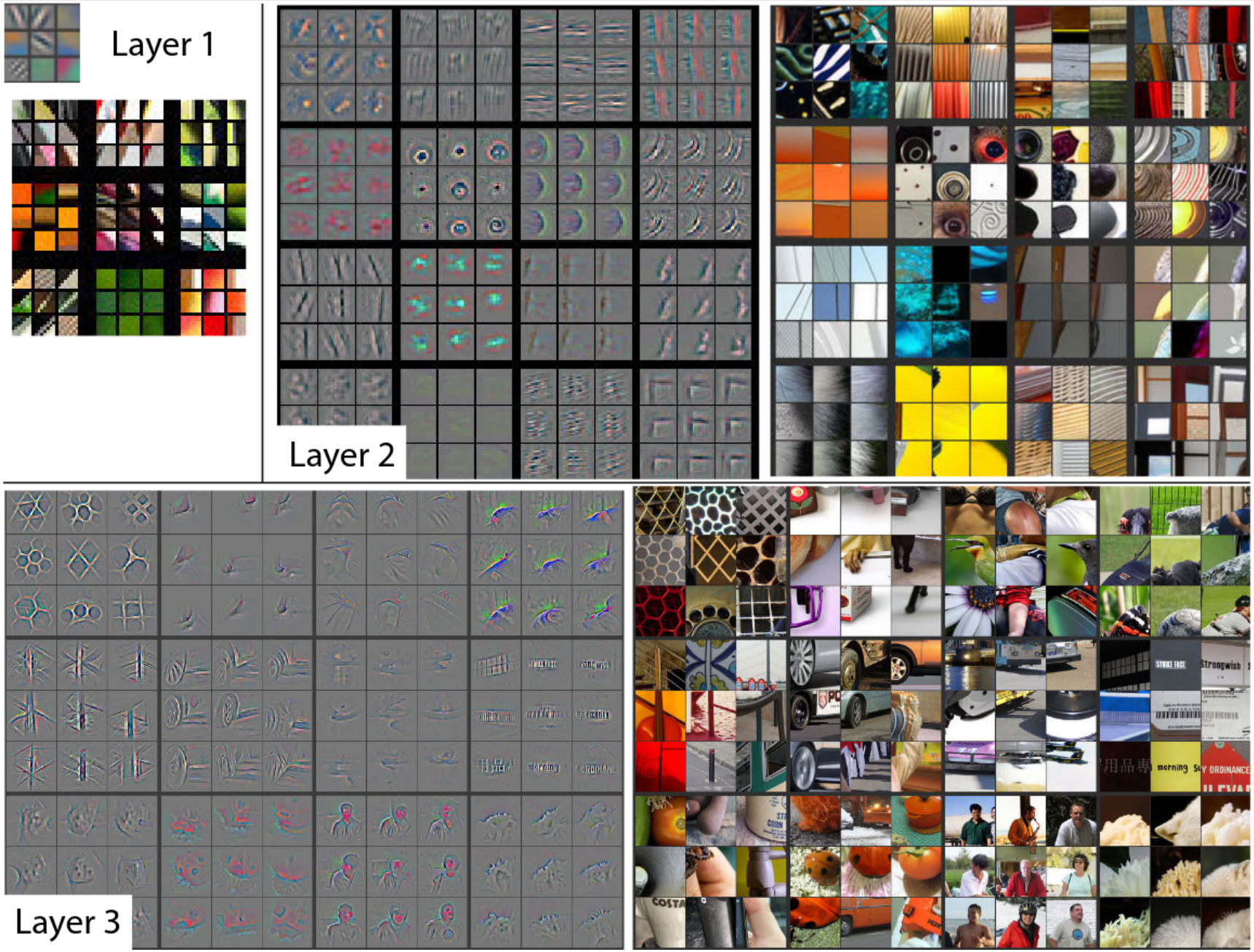




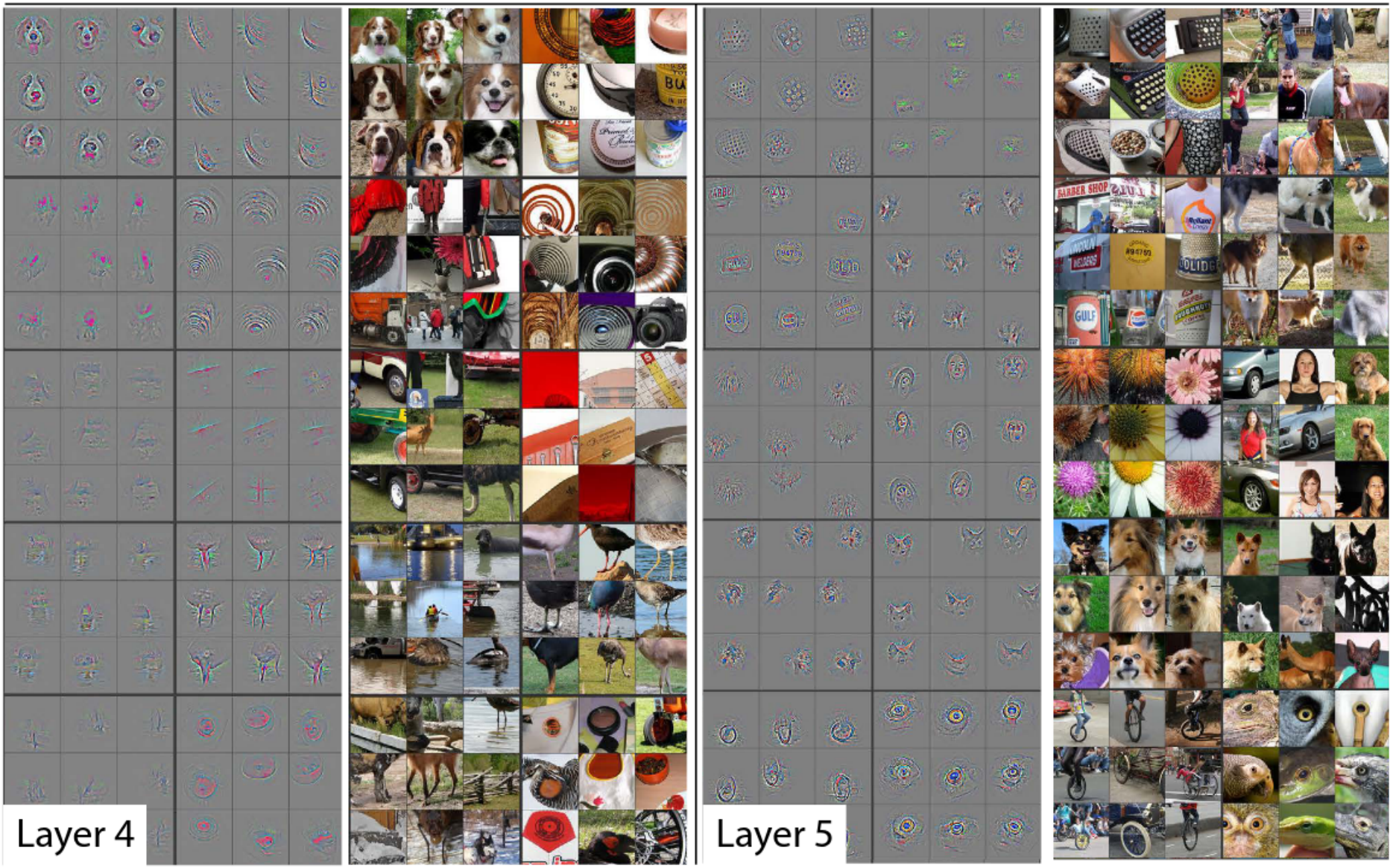
A deconvnet layer attached to a convnet layer. The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath.

from `https://arxiv.org/pdf/1311.2901.pdf`

Deconvnet (illustration from paper)



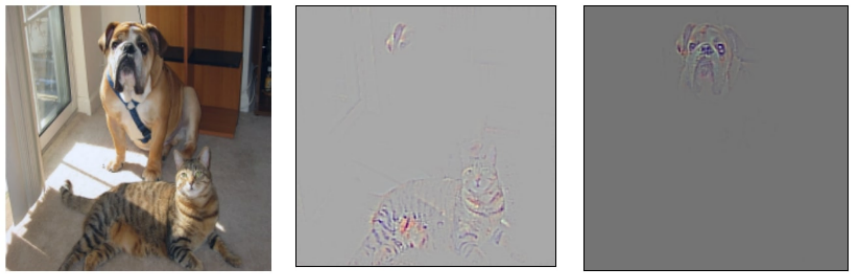
Deconvnet (illustration from paper)



Grad-CAM

Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

Ramprasaath R. Selvaraju · Michael Cogswell · Abhishek Das · Ramakrishna Vedantam · Devi Parikh · Dhruv Batra

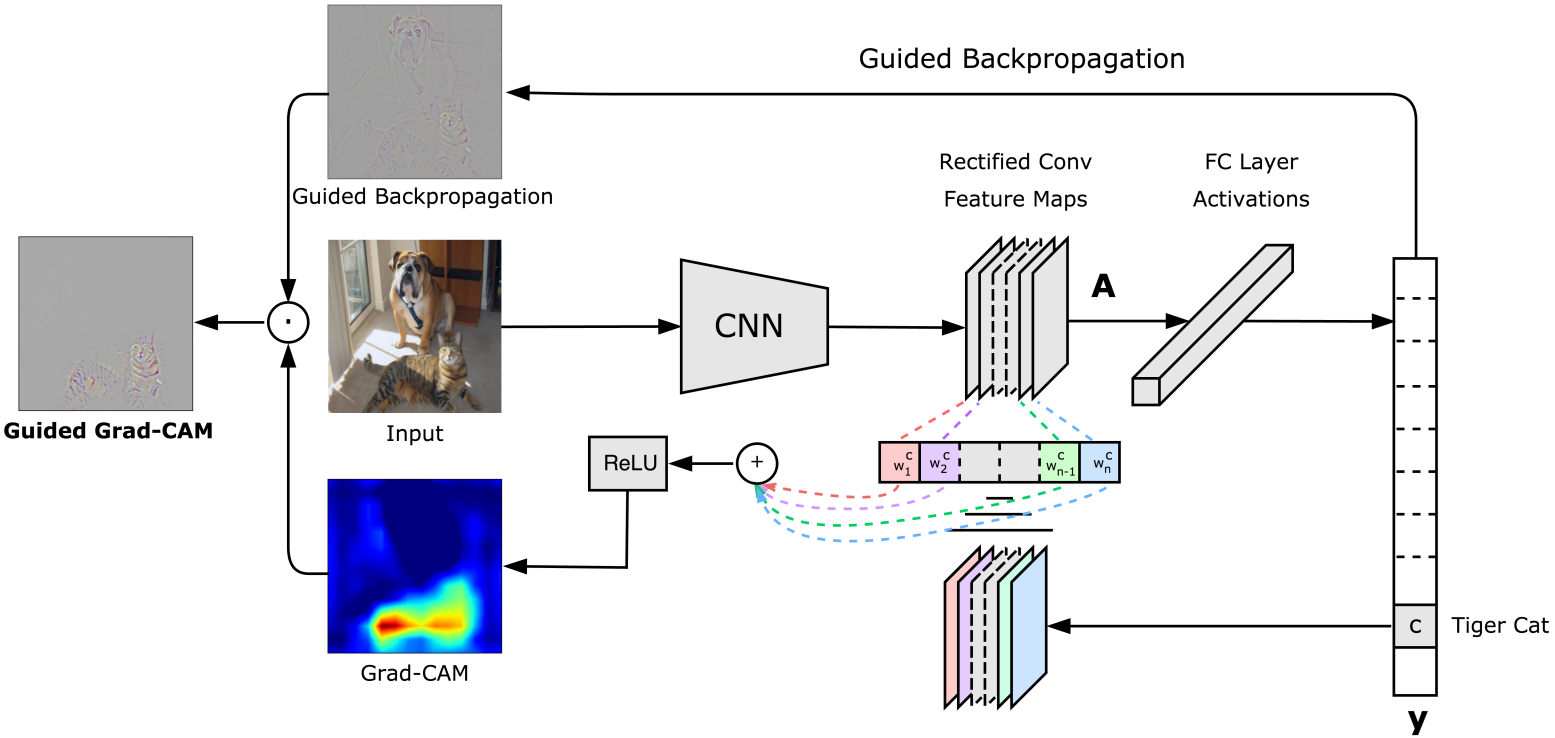


Original Image | Guided Grad-CAM 'Cat' | Guided Grad-CAM 'Dog'

`https://arxiv.org/pdf/1610.02391.pdf`

- Improvement of CAM
  - CAM needs to add GAP between feature map and softmax layer : usually performs worse that other CNNs
- Grad-CAM
  - don't modify the CNN
  - may be use with any CNN (including Image Captioning and VQA)
  - both **class discriminative** (localisation of the target class pixels) and **high-resolution** (e.g. stripes on the tiger cat)

Grad-CAM network



- computation of the score gradient w.r.t. conv. layers
- Grad-CAM is the ReLU activation of the weighted sum of feature maps :

$$L^c = \text{ReLU}(\sum_k \alpha_c^k A^k) \text{ with } \alpha_c^k = \overbrace{\frac{1}{N}\sum_{u,v}}^{GAP} \overbrace{\frac{\partial y^c}{\partial A_{uv}^k}}^{\text{gradient via backprop}}$$

Grad-CAM example (from the paper)

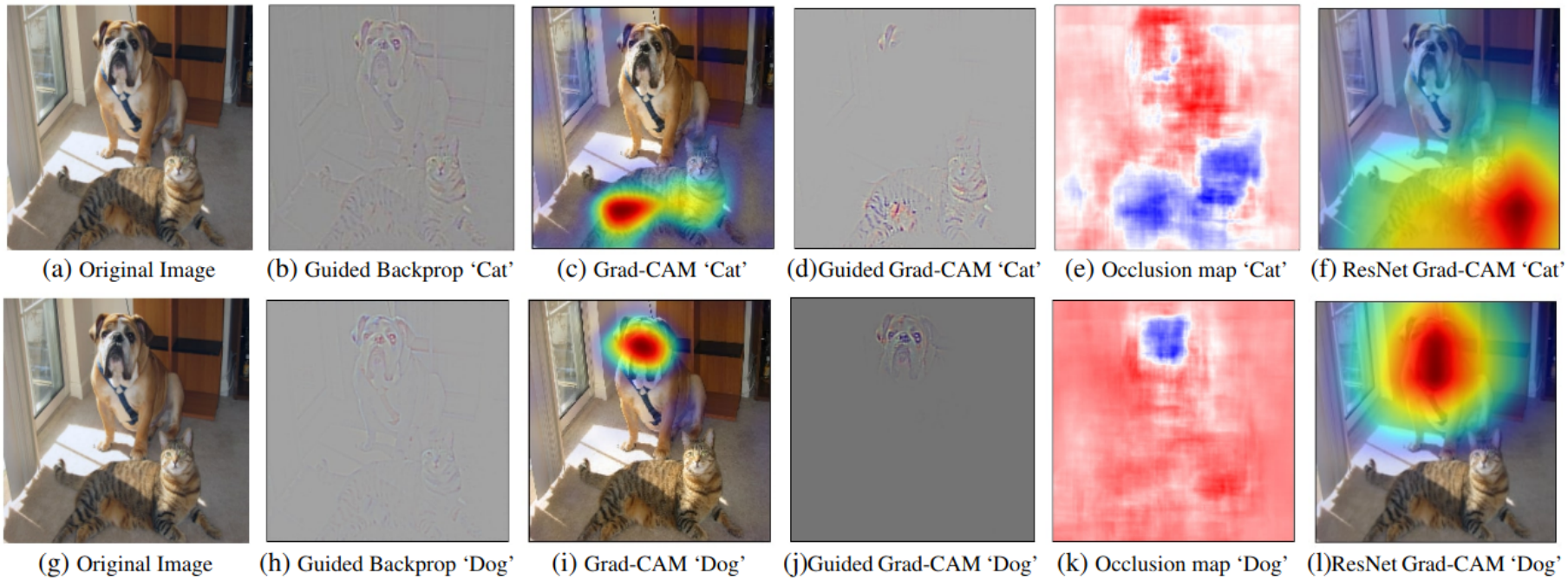


Fig. 1: (a) Original image with a cat and a dog. (b-f) Support for the cat category according to various visualizations for VGG-16 and ResNet. (b) Guided Backpropagation [53]: highlights all contributing features. (c, f) Grad-CAM (Ours): localizes class-discriminative regions, (d) Combining (b) and (c) gives Guided Grad-CAM, which gives high-resolution class-discriminative visualizations. Interestingly, the localizations achieved by our Grad-CAM technique, (c) are very similar to results from occlusion sensitivity (e), while being orders of magnitude cheaper to compute. (f, l) are Grad-CAM visualizations for ResNet-18 layer. Note that in (c, f, i, l), red regions corresponds to high score for class, while in (e, k), blue corresponds to evidence for the class. Figure best viewed in color.

Grad-CAM practice

- An useful tool : `tf.GradientTape`

Grad-CAM practice

- An useful tool : `tf.GradientTape`
- Guided Back Prop part
- Grad CAM part
  - based on VGG16, features maps after `block5_conv3`
  - model :

```
VGG = VGG16(weights='imagenet', include_top=True)
model = Model(inputs=VGG.input,
              outputs=[VGG.get_layer('block5_conv3').output,VGG.output])
```

Activation maps

Probability values of classification

Grad-CAM practice

- An useful tool : `tf.GradientTape`
- Guided Back Prop part
- Grad CAM part
  - based on VGG16, features maps after `block5_conv3`
  - model :

```
VGG = VGG16(weights='imagenet', include_top=True)
model = Model(inputs=VGG.input,
              outputs=[VGG.get_layer('block5_conv3').output,VGG.output])
```

  - gradients :

```
with tf.GradientTape() as tape:
    output, predictions = model(np.array([img]))
    cl = np.argmax(predictions)
    score = predictions[:, cl]
featureMaps = output[0]
gradients = tape.gradient(score, output)[0]
```

Outline

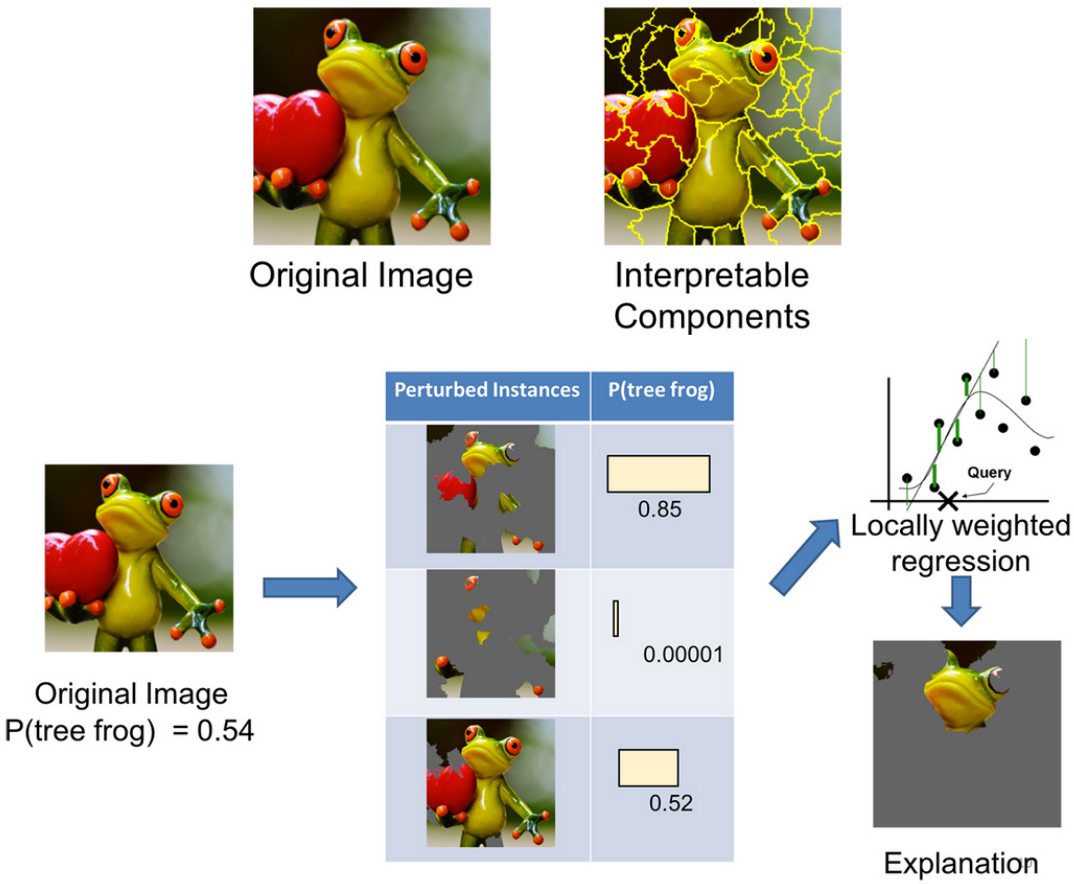
Local Interpretable Model-Agnostic Explanations(LIME)

If we dont know the algo

- maching learning algorithm as a black box
- perturb the input and see how the predictions change
  - benefit in terms of interpretability : changing components that make sense to humans
- local approximation of the model by a linear model

LIME example for image classification

- interpretable components are contiguous superpixels
- generate pertubed instances by turning some of the interpretable components “off” (grey)
- for each pertubed instance : compute the probability of frog class
- local linear regression model



Practice with LIME

- installation : `python3 -pip install lime`
- tutorial `http://marcotcr.github.io/lime/tutorials/Lime%20-%20basic%20usage%2C%20two%20class%20case.html`

LIME is a great tool but ...

- linear models are restrictive
- superpixels do not explain all the problems
  - from the paper : *For example, a model that predicts sepia-toned images to be retro cannot be explained by presence or absence of super pixels.*

SHAP (SHapleyAdditive exPlanations), NIPS 2017

A Unified Approach to Interpreting Model Predictions

Scott M. Lundberg
Paul G. Allen School of Computer Science
University of Washington
Seattle, WA 98105
slund1@cs.washington.edu

Su-In Lee
Paul G. Allen School of Computer Science
Department of Genome Sciences
University of Washington
Seattle, WA 98105
suinlee@cs.washington.edu

http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Use of game theory in order to highlight the contribution of each feature to the result.

Shapley value (1953)

- coalition game (observation, model output) with N players (features)
- S is a subset of N
- $v(S)$ is the total value of the S players
- if player i joins the group S, its marginal contribution is $v(S \cup \{i\}) - v(S)$.
- contribution of player i : average contribution over all possible permutations

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|S| - N - 1)!}{N!}(v(S \cup \{i\}) - v(S))$$

Shapley set four axioms in order to achieve a fair contribution :

- Efficiency. The sum of the Shapley values of all players equals the value of the total coalition.
- Symmetry. All players have a fair chance to join the game.
- Dummy. If player i contributes nothing to any coalition S, then the contribution of player i is zero, i.e., $\phi(v) = 0$.
- Additivity. For any pair of games v, w : $\phi(v + w) = \phi(v) + \phi(w)$,

Shapley value - an example

taken from https://towardsdatascience.com/interpreting-your-deep-learning-model-by-shap-e69be2b47893

- a project of 100 lines of codes with 3 programmers : L, M and N

V(X)	Line of codes
L	10
M	30
N	5
L, M	50
L, N	40
M, N	35
L, M, N	100

- consider different orders :

Order	L Contribution	M Contribution	N Contribution
L, M, N	V(L) = 10	V(L,M) – V(L) = 50 – 10 = 40	V(L,M,N) – V(L,M) = 100 – 50 = 50
L, N, M	V(L) = 10	V(L,M,N) – V(L, N) = 100 – 40 = 60	V(L,N) – V(L) = 40 – 10 = 30
M, L, N	V(L,M) – V(M) = 50 – 30 = 20	V(M) = 30	V(L,M,N) – V(L,M) = 100 – 50 = 50
M, N, L	V(L,M,N) – V(M,N) = 100 – 35 = 65	V(M) = 30	V(M,N) – V(M) = 35 – 30 = 5
N, L, M	V(L,N) – V(L) = 40 – 5 = 35	V(L,M,N) – V(L,N) = 100 – 40 = 60	V(N) = 5
N, M, L	V(L,M,N) – V(M,N) = 100 – 35 = 65	V(M,N) – V(N) = 35 – 5 = 30	V(N) = 5

- compute Shapley values :

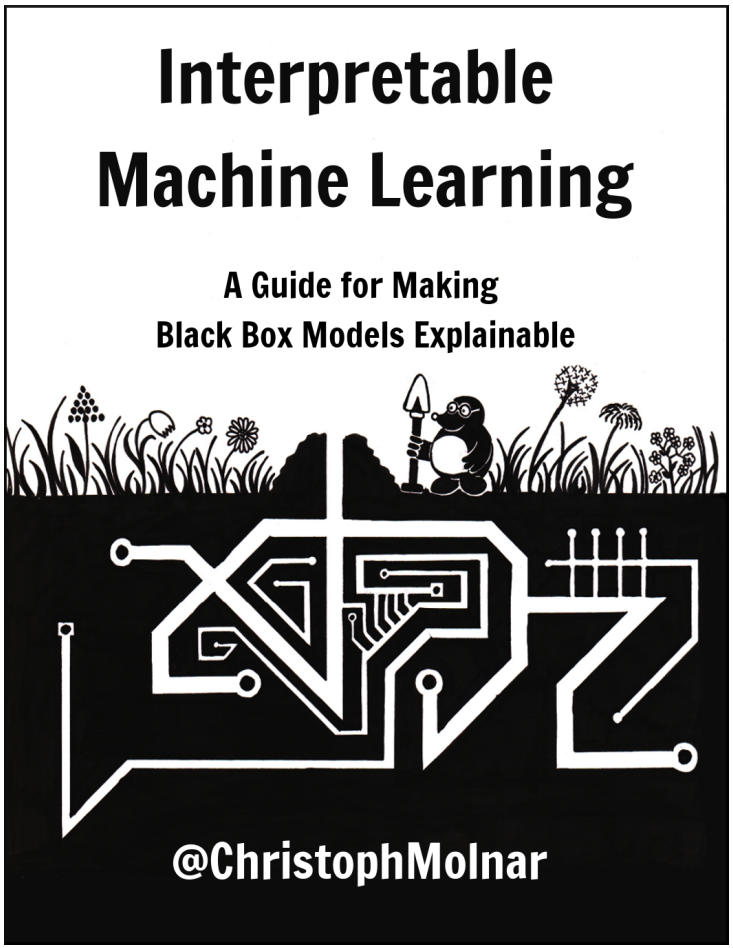
Contributor	Shapley Calculation	Shapley Value
L	1/6(10+10+20+65+35+65)	34.17
M	1/6(40+60+30+30+60+30)	41.7
N	1/6(50+30+50+5+5+5)	24.17

Shapley value : simplification

Number of coalitions in case of n features : $2^n \Rightarrow$ simplification

- for all m = 1,...,M :
  - random z from X
  - random permutation o of the feature values
  - order instance x : $x_o = (x(1), \ldots, x(j), \ldots, x(p))$
  - order instance z : $z_o = (z(1), \ldots, z(j), \ldots, z(p))$
  - two new instances :
    - with feature j : $x_{+j} = (x(1), \ldots, x(j-1), x(j), z(j+1), \ldots, z(p))$
    - without feature j : $x_{-j} = (x(1), \ldots, x(j-1), z(j), z(j+1), \ldots, z(p))$
  - compute marginal contribution : $\phi_j^m = \hat{f}(x_{+j}) - \hat{f}(x_{-j})$
- compute Shapley value as the average : $\phi_j(x) = \frac{1}{M} \sum_{m=1}^{M} \phi_j^m$

Reading



https://christophm.github.io/interpretable-ml-book/