MAC 0460 / 5832
Introduction to Machine Learning

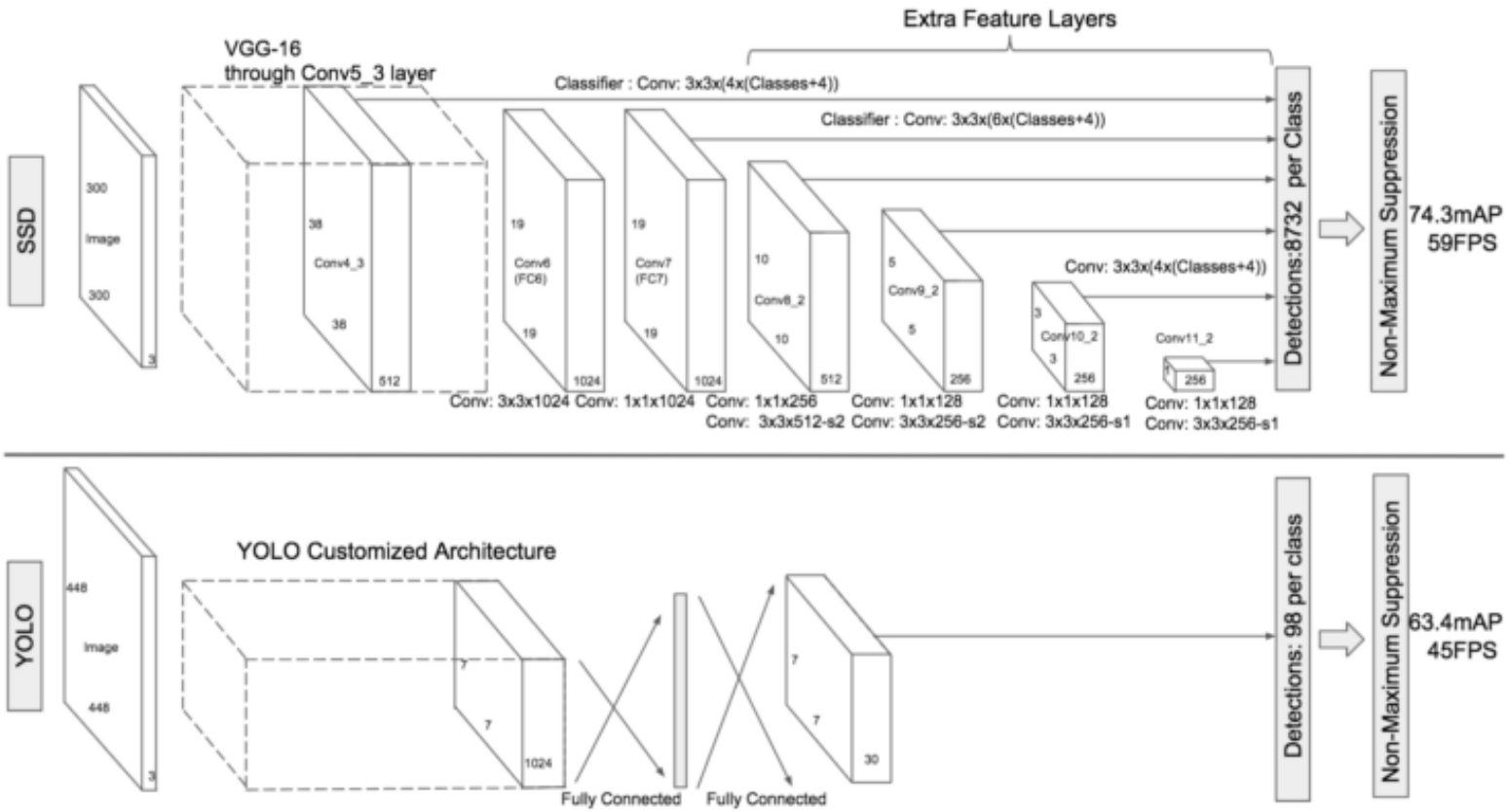
24 – Visualization and interpretability of deep neural nets

IME/USP (12/07/2021)

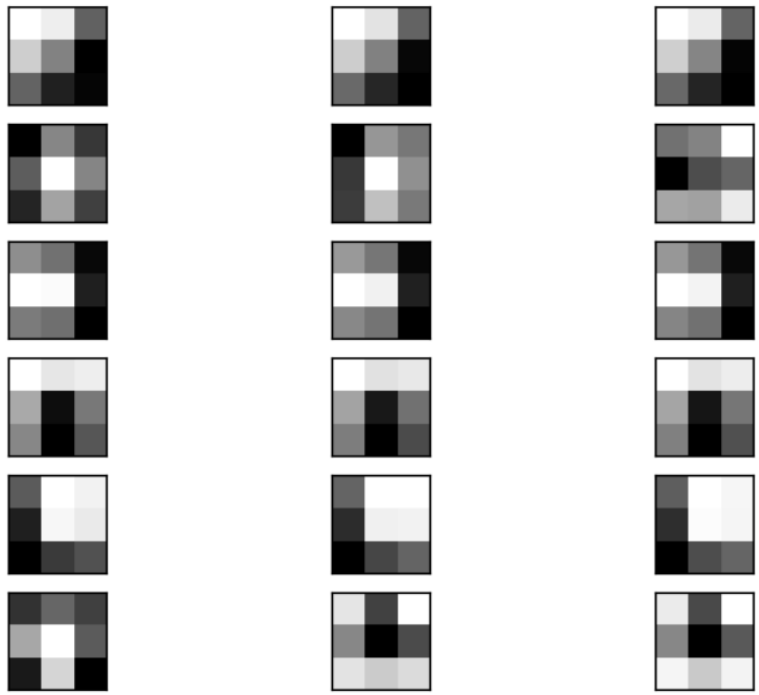
Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 1000)              0
_________________________________________________________________
embedding_1 (Embedding)      (None, 1000, 100)         17410600
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 996, 128)          64128
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 199, 128)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 195, 128)          82048
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 39, 128)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 35, 128)           82048
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 1, 128)            0
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 20)                2580
=================================================================
Total params: 17,657,916
Trainable params: 247,316
Non-trainable params: 17,410,600
_________________________________________________________________
None
```

Architecture



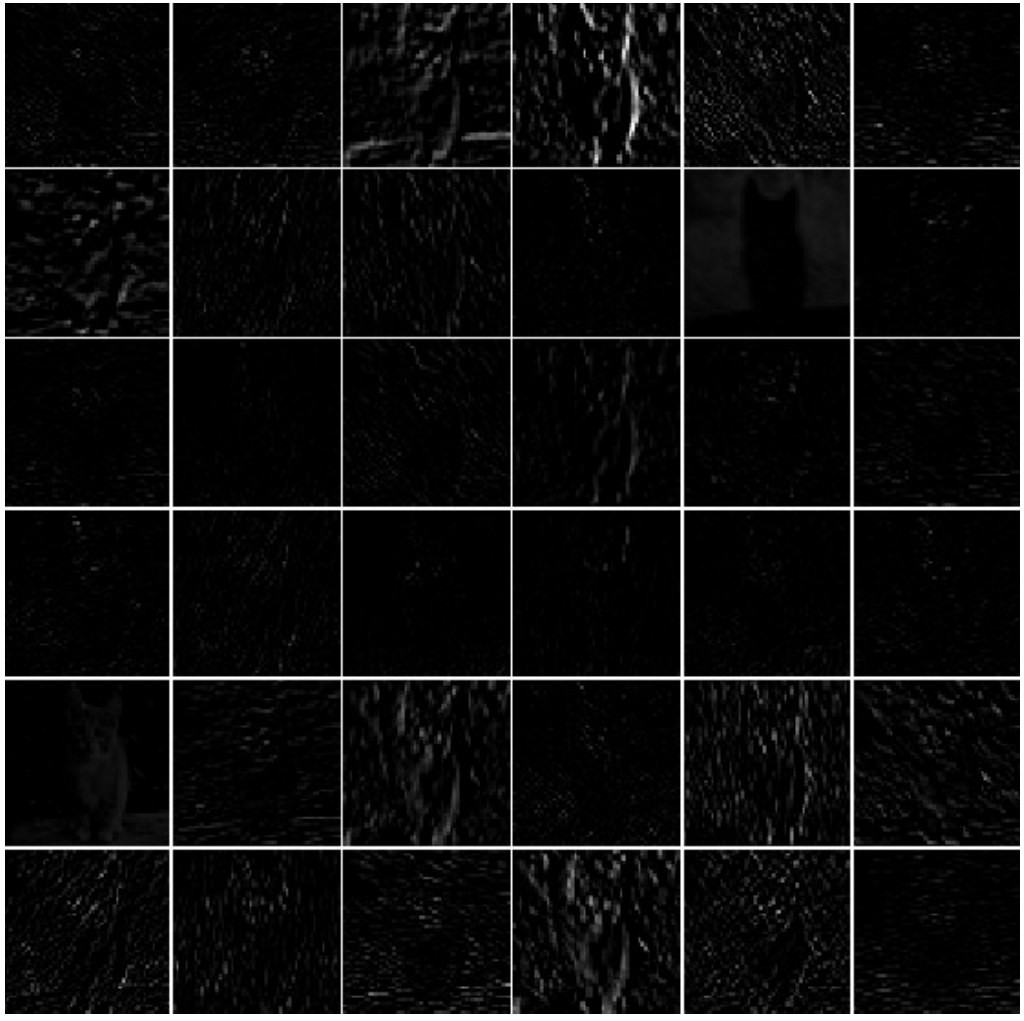
Filters (kernels)



https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

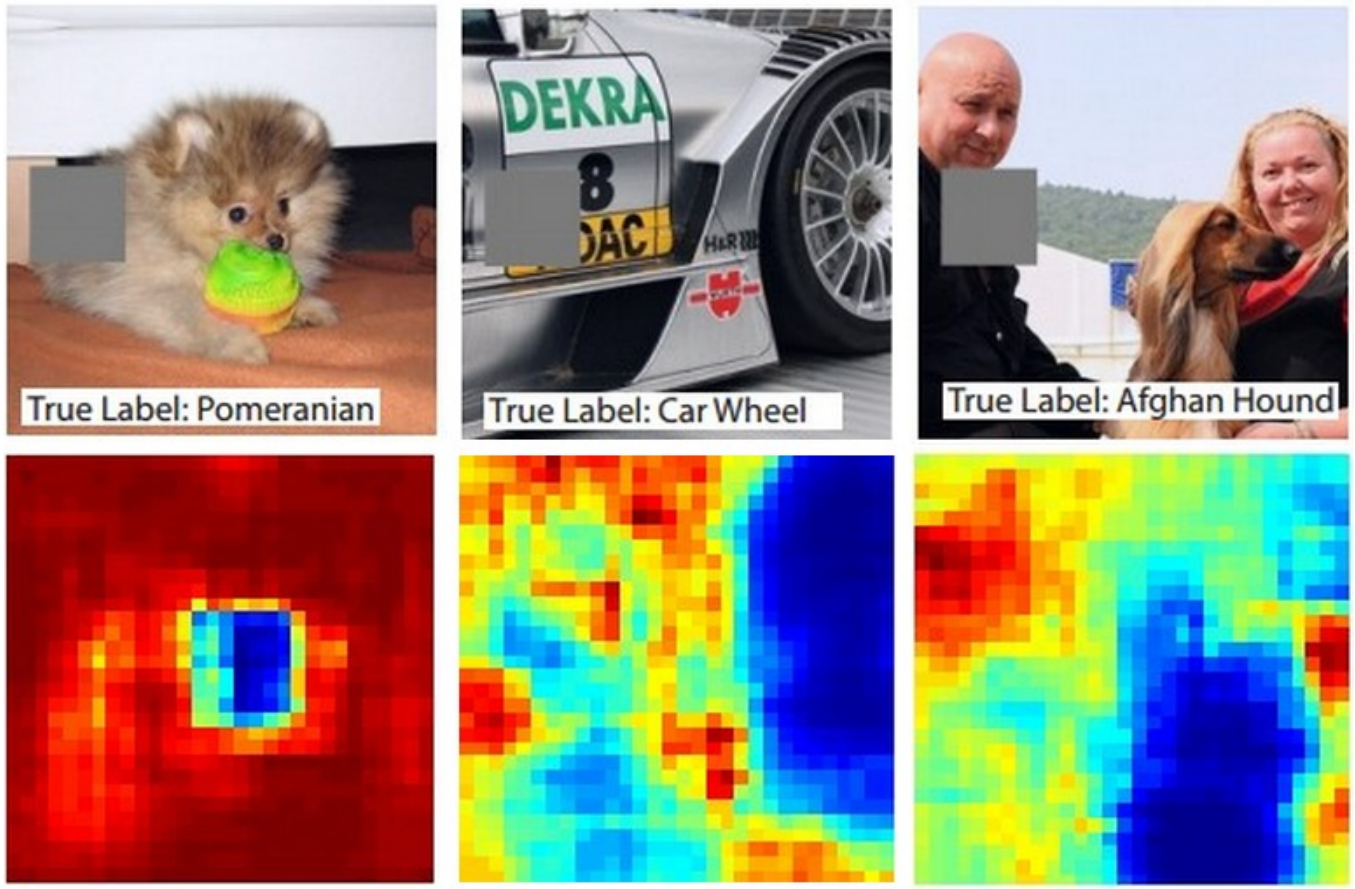
Each row corresponds to a 3-channel filter, from the first layer of VGG

Feature map



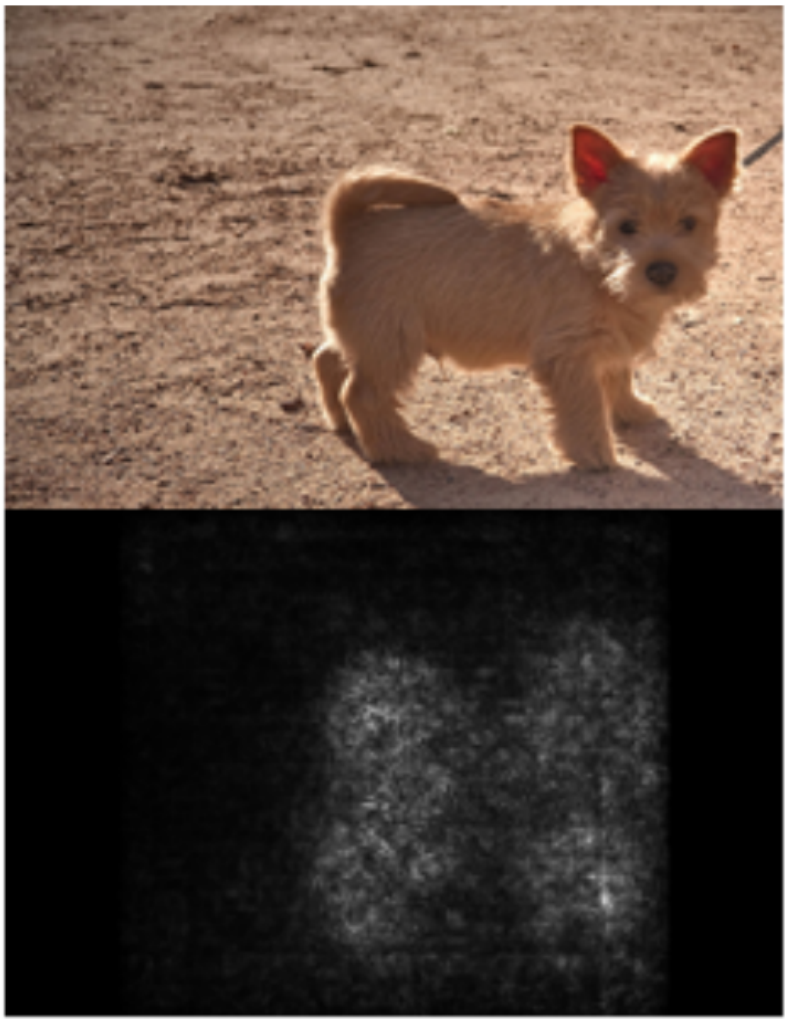
Occlusion sensitivity (arXiv:1311.2901, Zeiler et al., 2013)

Occlude small regions of input image and check the target class output score



Blue (most sensible; small score); red (less sensible; high score)

Saliency map (arXiv:1312.6034, Simonyan et al., 2014)



$S_c(I)$ score function of class c

Saliency map of input image I_0:

$$M = \left| \frac{\partial S_c}{\partial I}\Big|_{I_0} \right|$$

($S_c(I)$ is non-linear, but we can approximate it linearly around I_0)

Vanilla saliency

$$M_c(x) = \frac{\partial S_c(x)}{\partial x}$$

SmoothGrad (2017)

$$\hat{M}_c(x) = \frac{1}{n} \sum^{n} M_c(x + \mathcal{N}(0, \sigma^2))$$

SmoothGrad (2017)

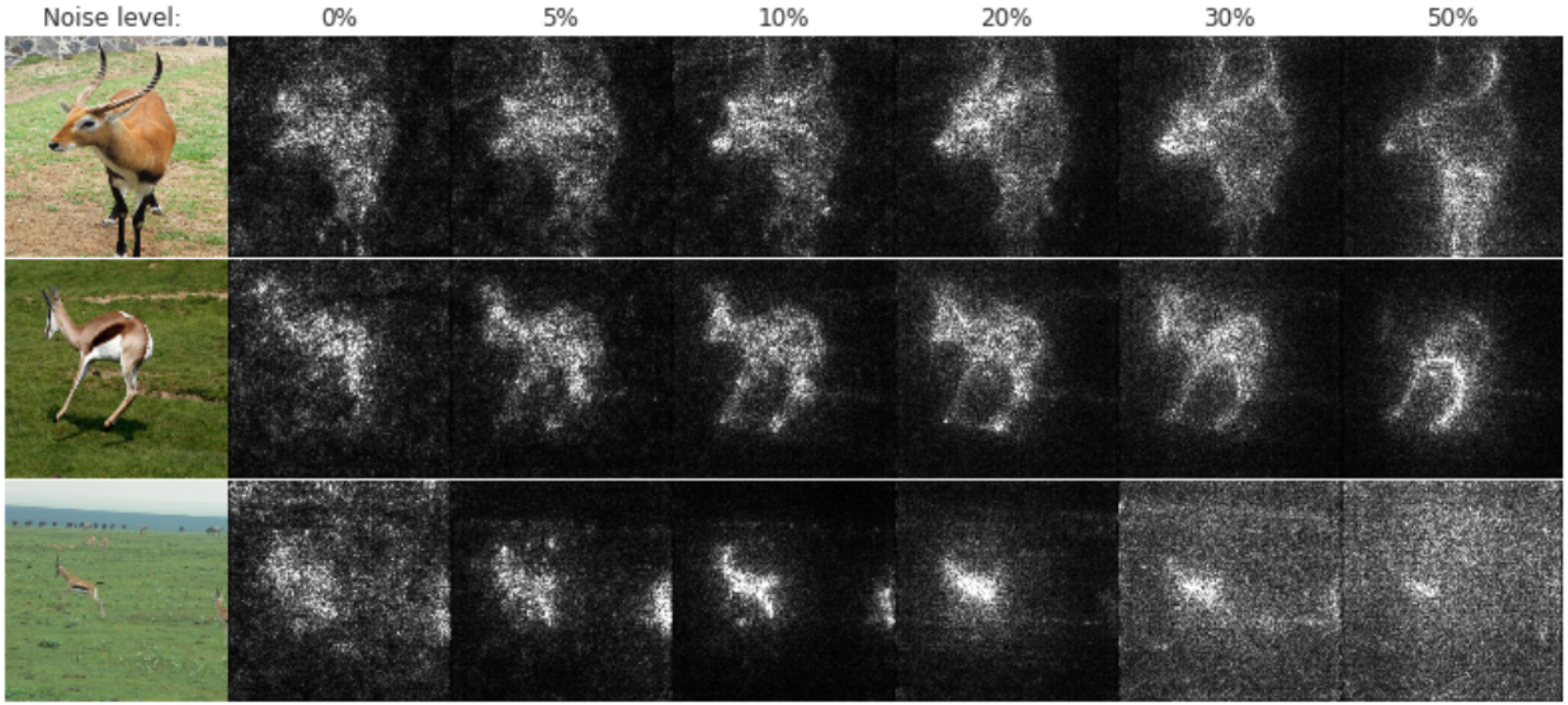
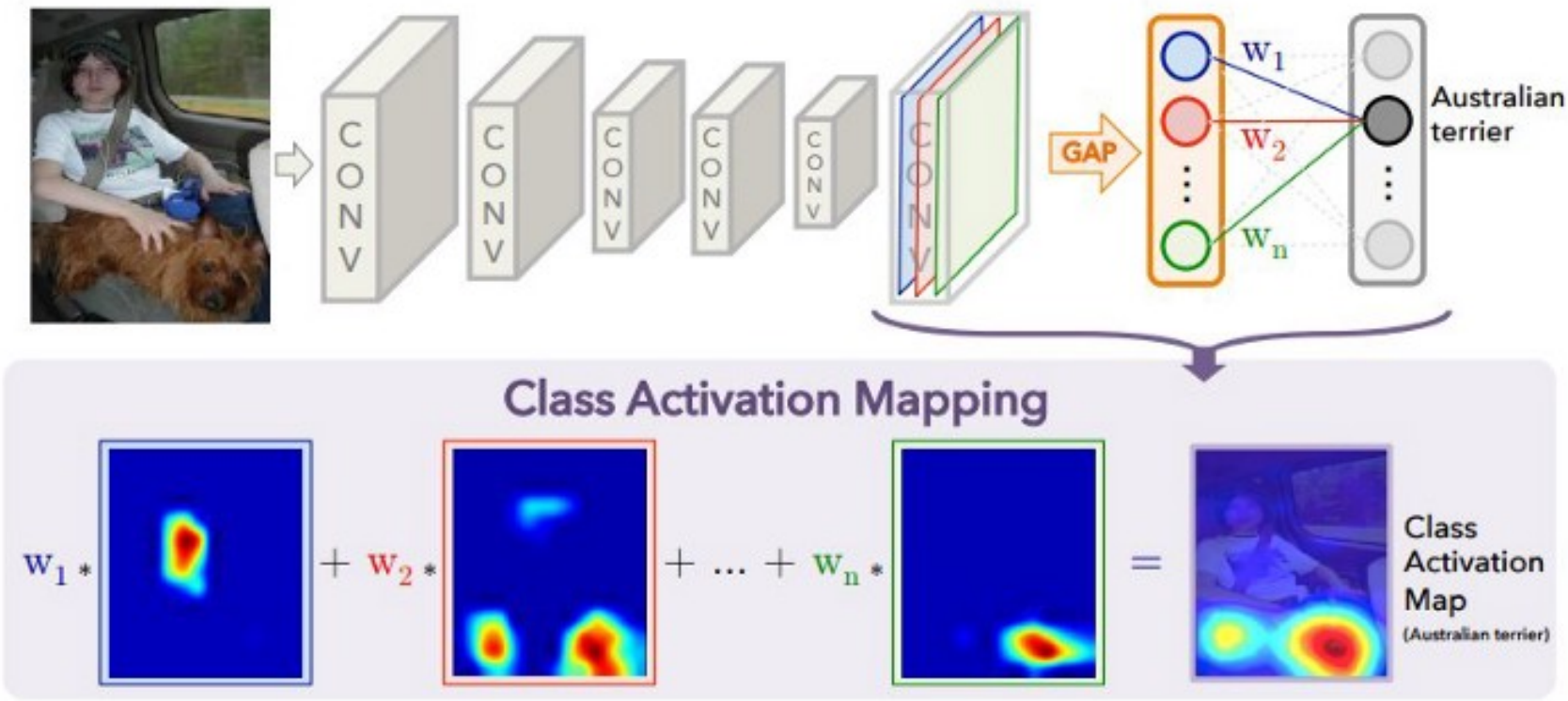


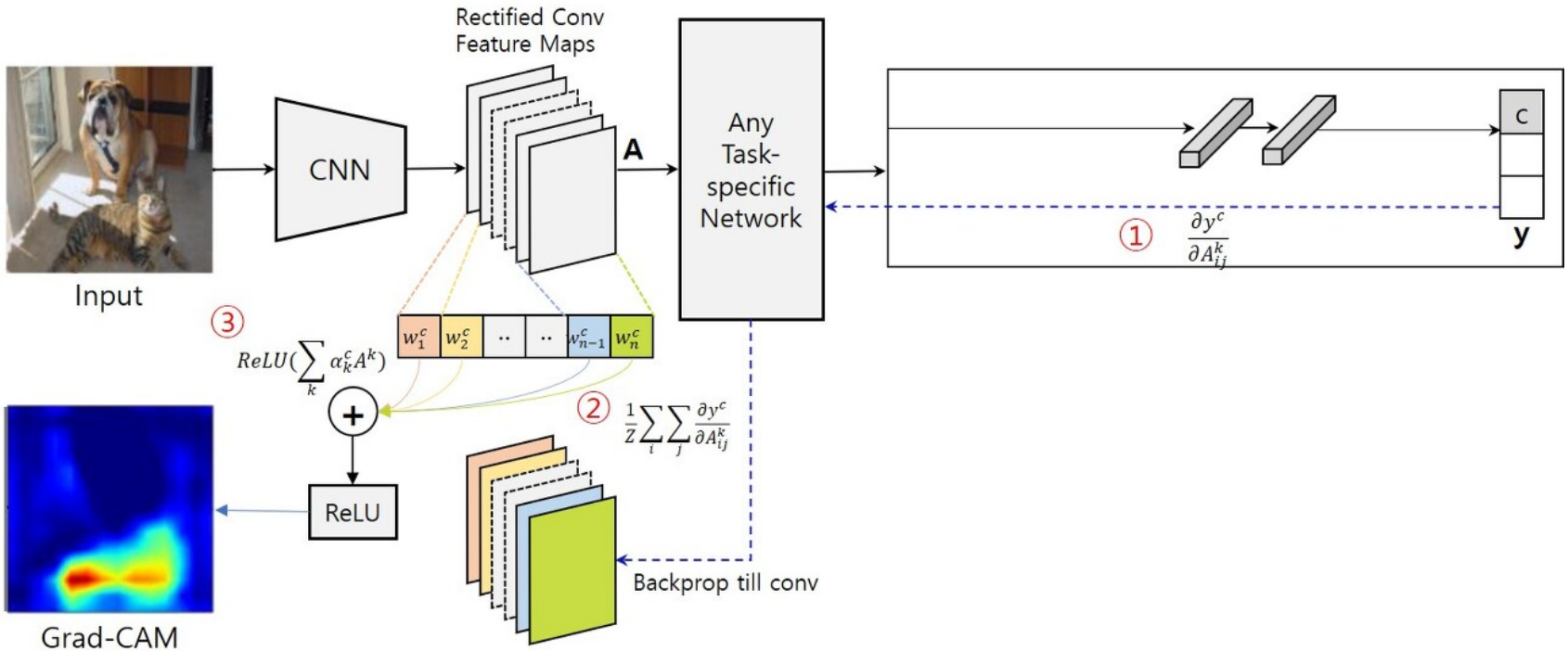
Figure 3. Effect of noise level (columns) on our method for 5 images of the gazelle class in ImageNet (rows). Each sensitivity map is obtained by applying Gaussian noise $\mathcal{N}(0,\, \sigma^2)$ to the input pixels for 50 samples, and averaging them. The noise level corresponds to $\sigma/(x_{max} - x_{min})$.

Class activation map (CAM)



CAM requires a global pooling layer followed by the output layer

GradCAM: for each feature map A^k in the last convolutional layer, compute $\sum_{i,j} \frac{\partial S_c(x)}{\partial A^k_{i,j}}$ and use this as weight w_k



Gradient ascent

Given a trained network, keep its weights fixed, and iteratively update the input image through backpropagation so as to maximize

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

The regularization term is there to keep some smoothness.

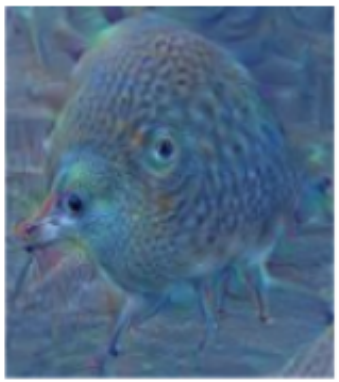
goose

Deep dream

Use gradient ascent to maximize activation of a filter at some specific layer

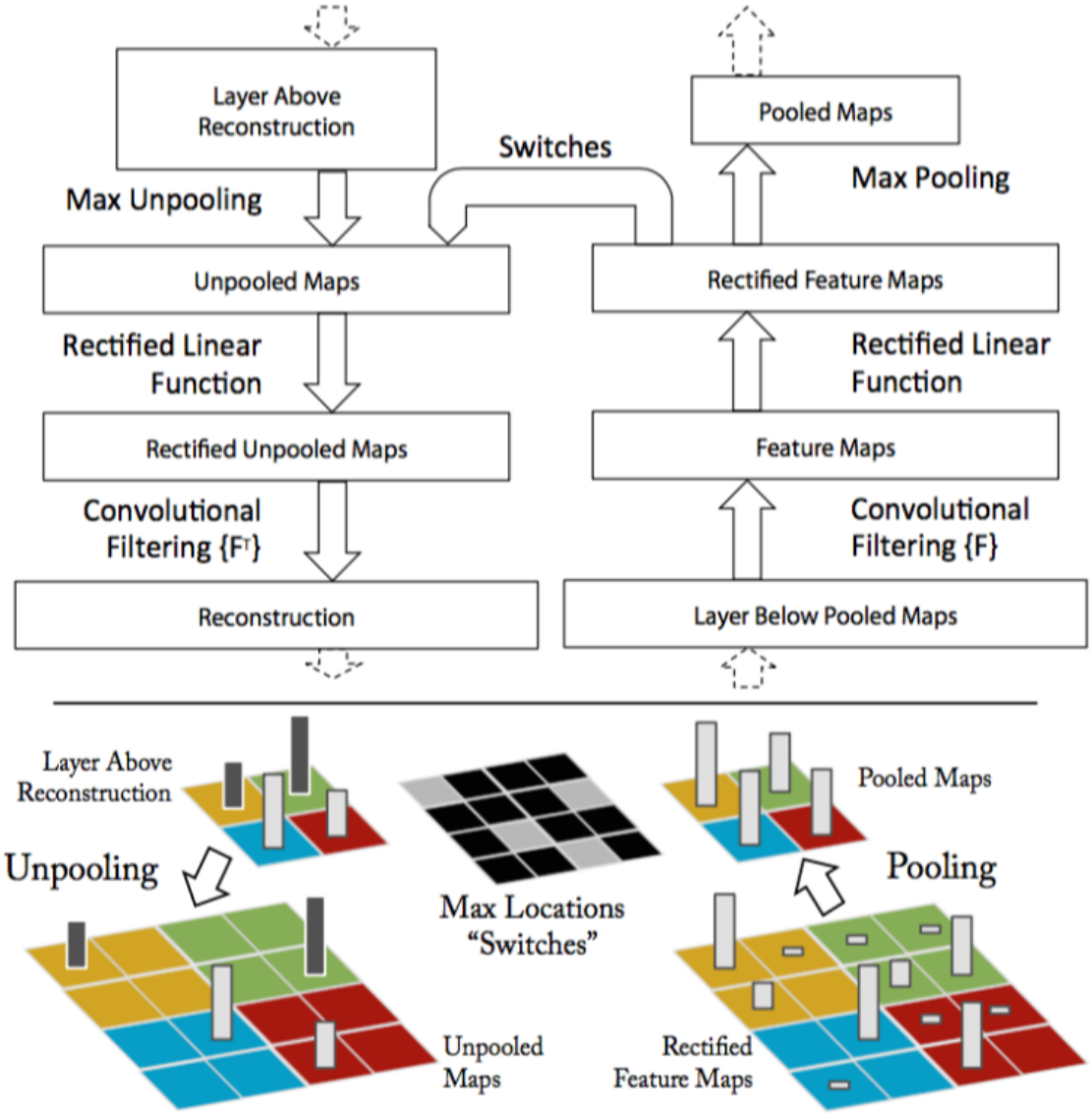

"Admiral Dog!"


"The Pig-Snail"


"The Camel-Bird"


"The Dog-Fish"

Reconstruction of the input from a neuron (Zeiler et al., 2013)



Forward pass:

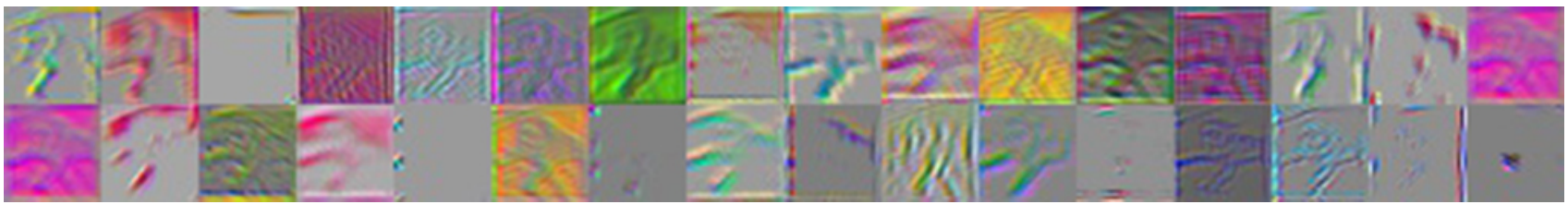
```
Conv ---> ReLU ---> Pooling
```

To reconstruct, we just need to perform the opposite sequence

```
unPool ---> unReLU ---> deConv
```

See Stanford CS230: Deep Learning — Autumn 2018 — Lecture 7 - Interpretability of Neural Network
https://youtu.be/gCJCgQW_LKc

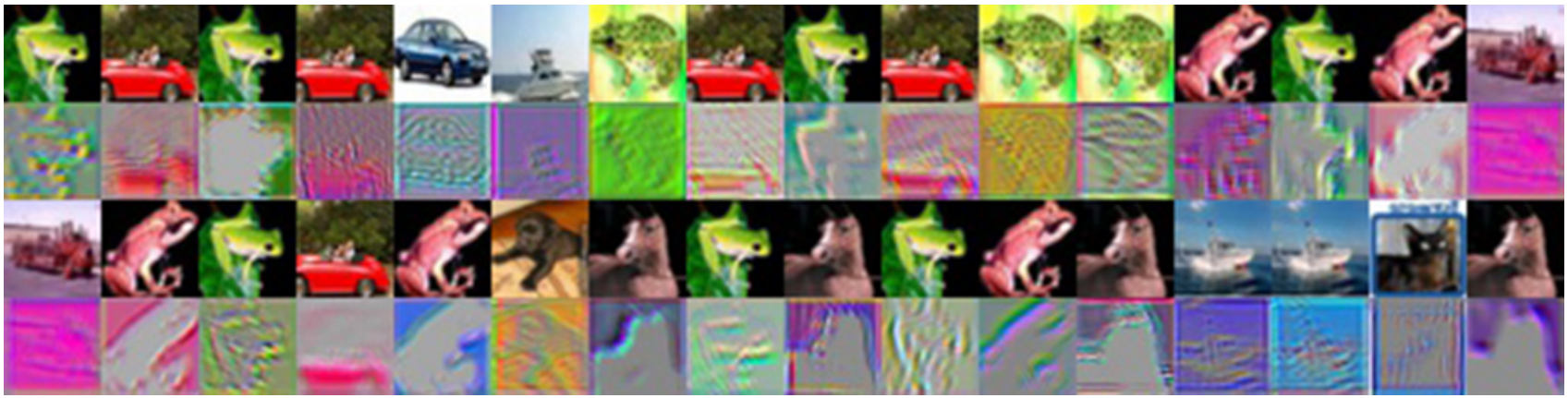
Images reconstructed from neurons 1 to 32 of layer 1, for one input image



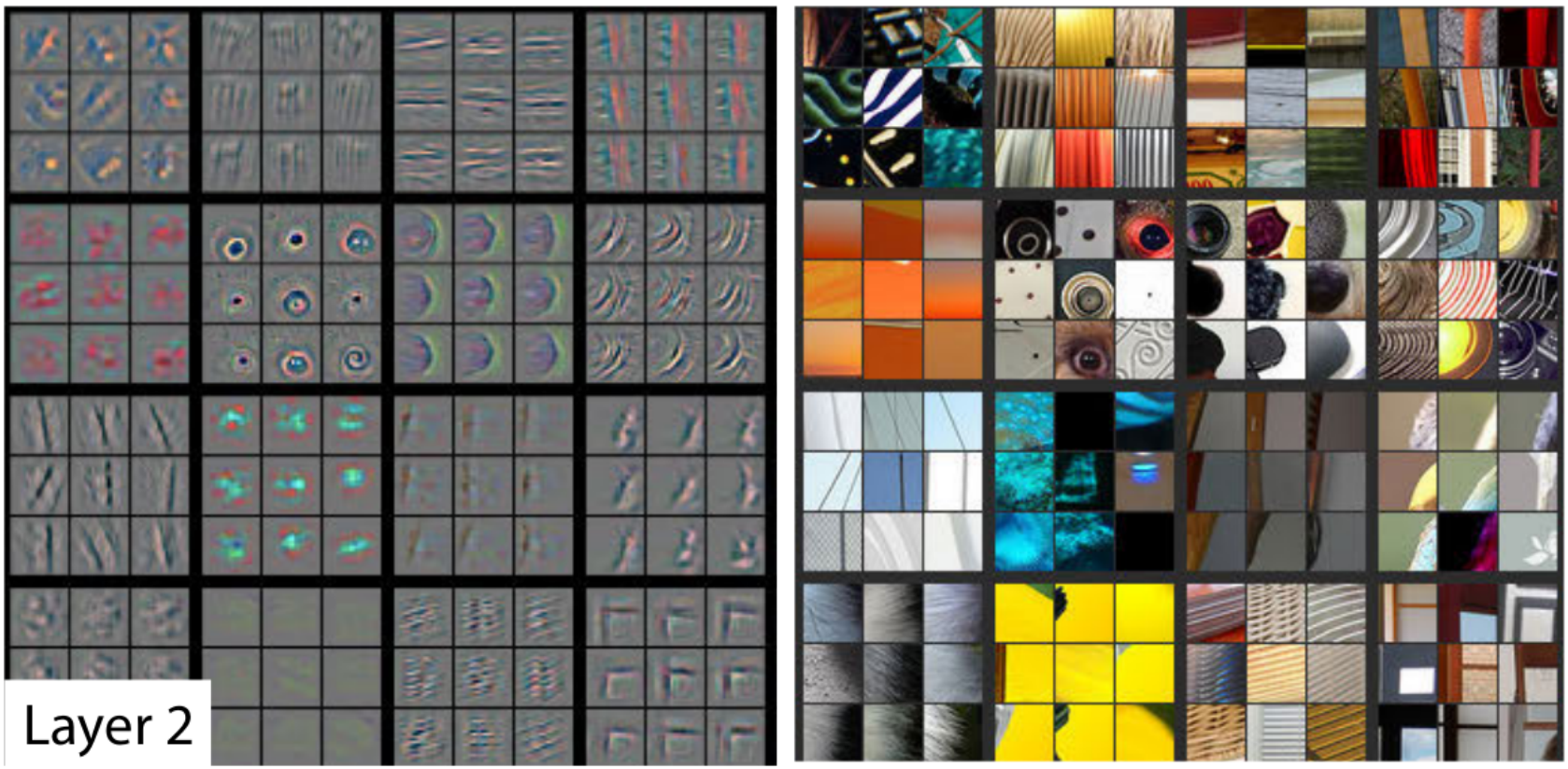
(http://kvfrans.com/visualizing-features-from-a-convolutional-neural-network/)

Images reconstructed from neuron 7 of layer 1, for multiple input images

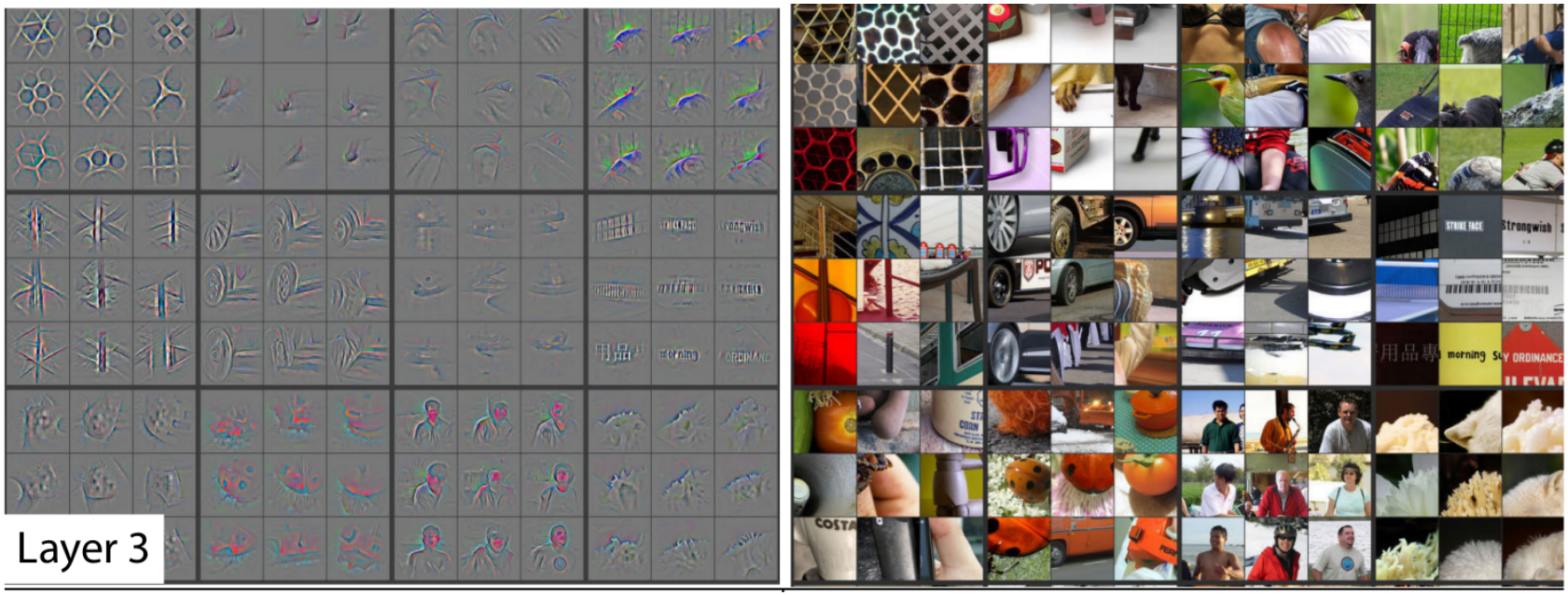
For each of the 32 neurons, reconstruction of the images that most activated the neuron



(Zeiler et al.) 16 neurons in layer 2; for each neuron, reconstruction of the 9 images that correspond to strongest activation of the node



(Zeiler et al.) 12 neurons in layer 3; for each neuron, reconstruction of the 9 images that correspond to strongest activation of the node



~~deconvolution~~

Transposed convolution

In reconstruction, weights are fixed

It can be, however, used in other tasks (segmentation, super-resolution, ...)
In these cases, weights are learned

There is `tf-keras-vis` that implements saliency maps and activation maps

XAI

- Feature importance
- SHAP
- LIME
- Layer-wise relevance propagation (LRP)