

# CNN Visualizations

Seoul AI Meetup

Martin Kersner, 2018/01/06

# Content

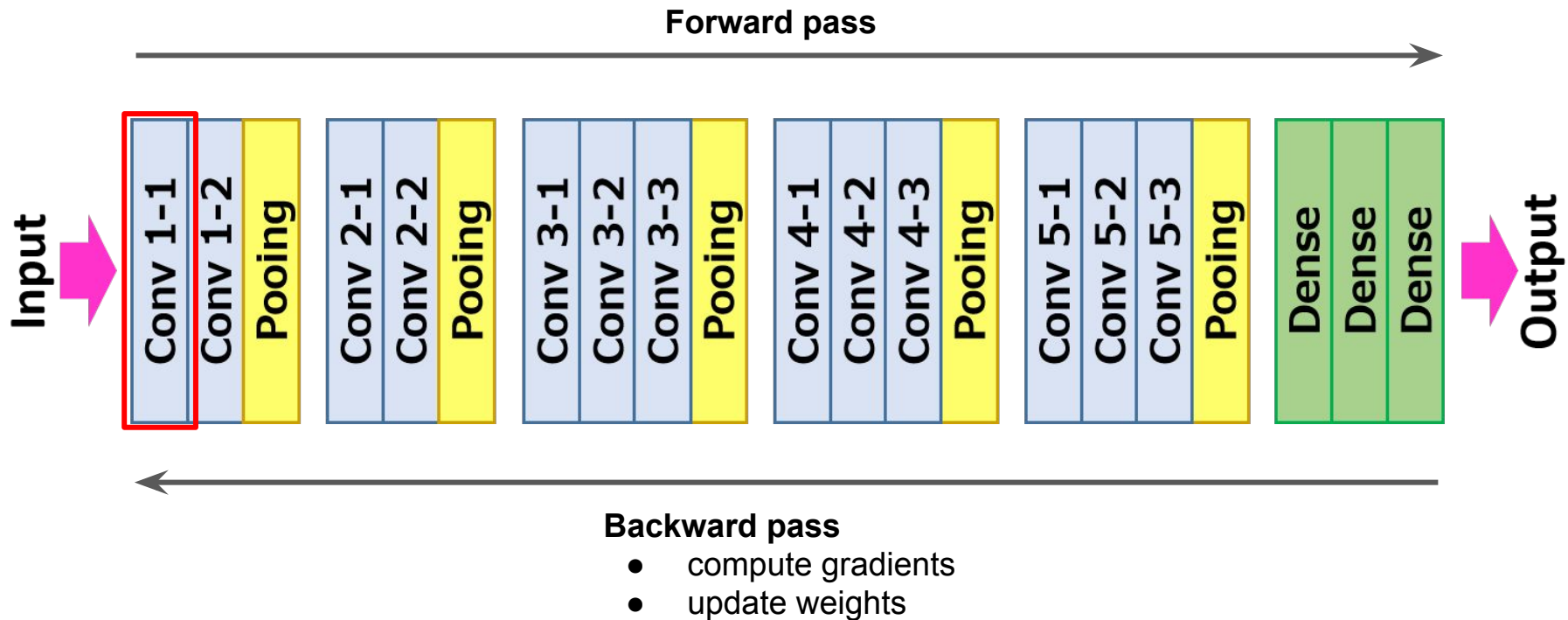
1. Visualization of convolutional weights from the **first layer**
2. Visualization of patterns learned by **higher layers**
3. Weakly Supervised Object Localization

# Motivation

- Understand better dynamics of CNN
- Debugging of network
- Verification of network decisions

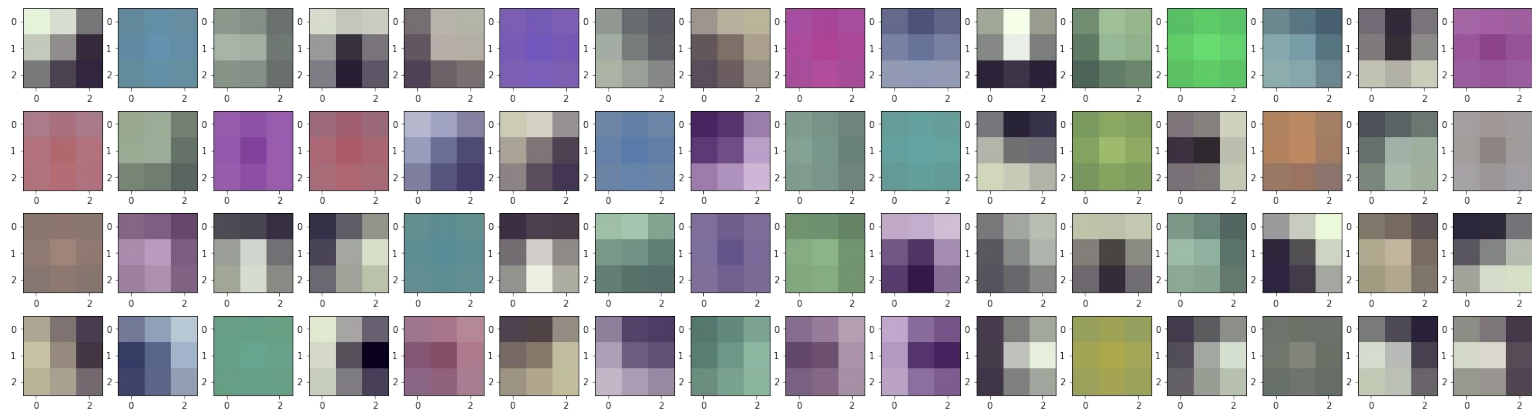
# Visualization of convolutional weights from the first layer

# VGG-16

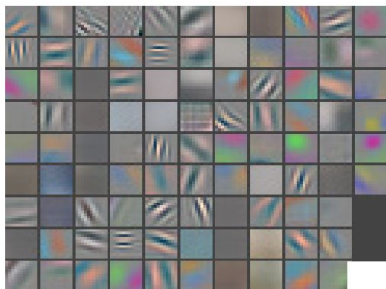


# conv1\_1

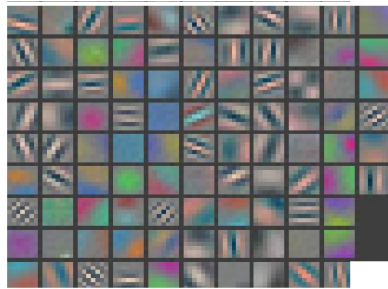
3x3



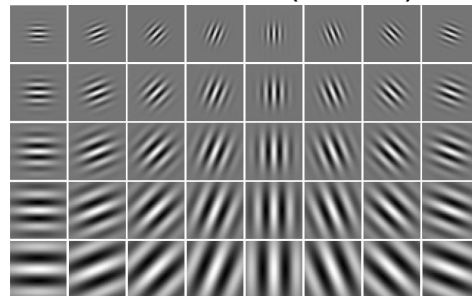
11x11



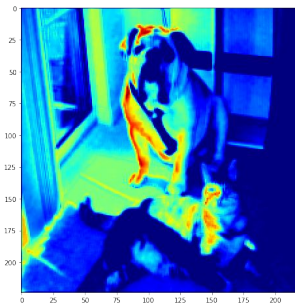
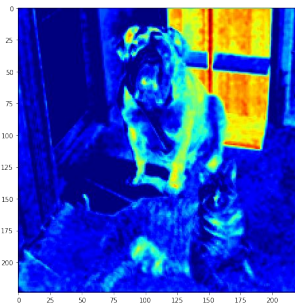
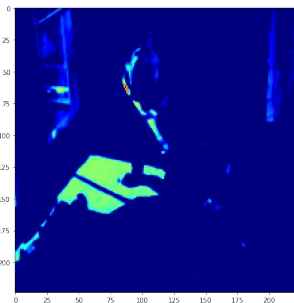
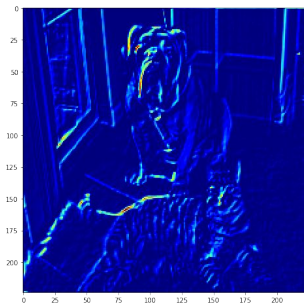
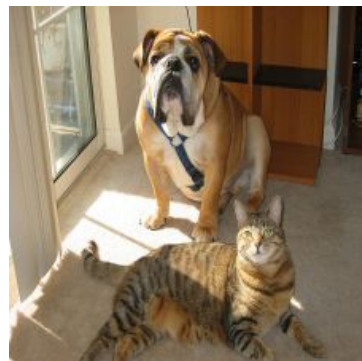
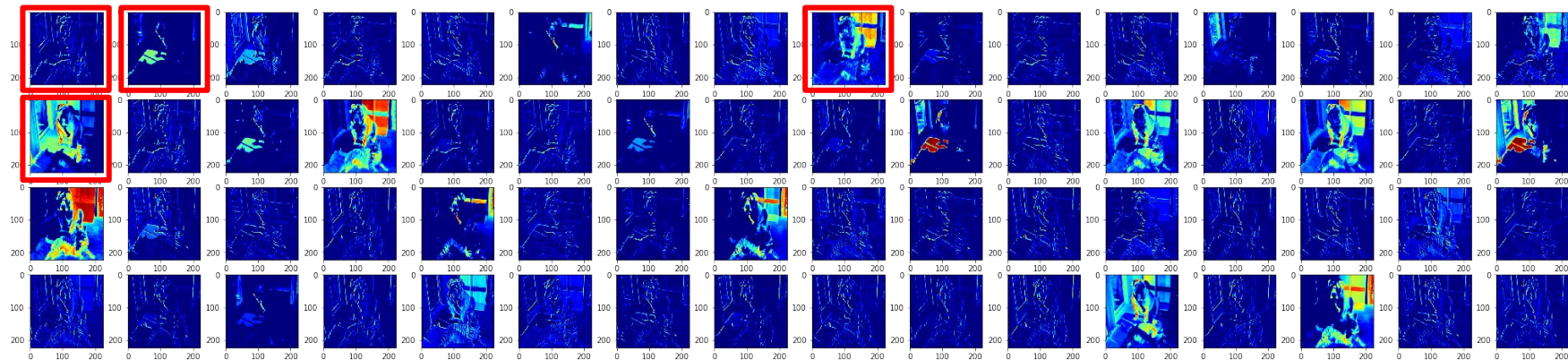
7x7



Gabor filters (~1980)



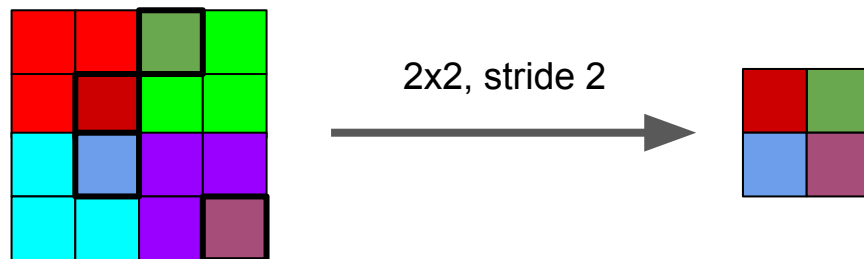
# conv1\_1 output



Visualization of patterns  
learned by higher layers

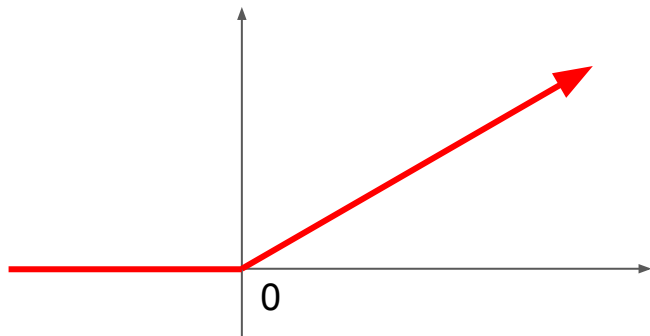


# Max pooling



# Rectified Linear Unit (ReLU)

```
class ReLU(object):  
    def __init__(self):  
        self.input, self.output = None, None  
        self.bottom, self.up = None, None  
        self.grad = None  
  
    def forward(self, input):  
        self.input = input  
        self.output = np.maximum(0.0, input)  
  
    def backward(self):  
        self.grad = self.up.grad.copy()  
        self.grad[self.input <= 0.0] = 0.0
```



# Backpropagation

An alternative way of visualizing the part of an image that most activates a given neuron is to use a simple backward pass of the activation of a single neuron after a forward pass through the network; thus computing the gradient of the activation w.r.t. the image.

## ReLU

Forward pass

1	-1	5
2	-5	-7
-3	2	4

→

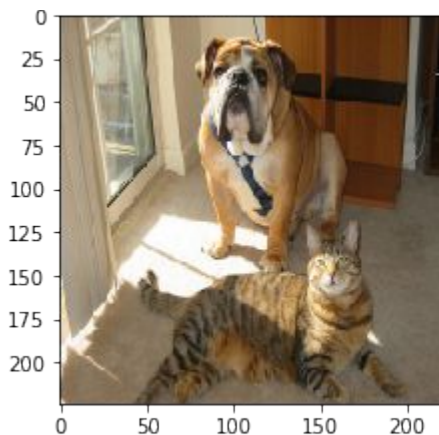
1	0	5
2	0	0
0	2	4

Backward pass:  
backpropagation

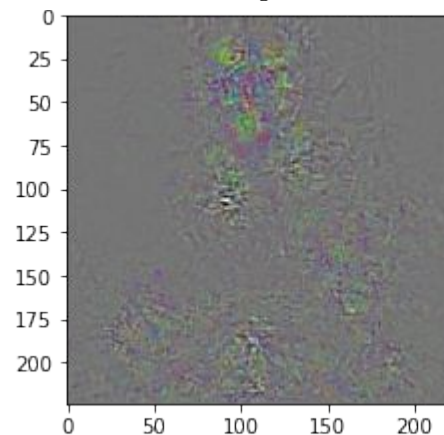
-2	0	-1
6	0	0
0	-1	3

←

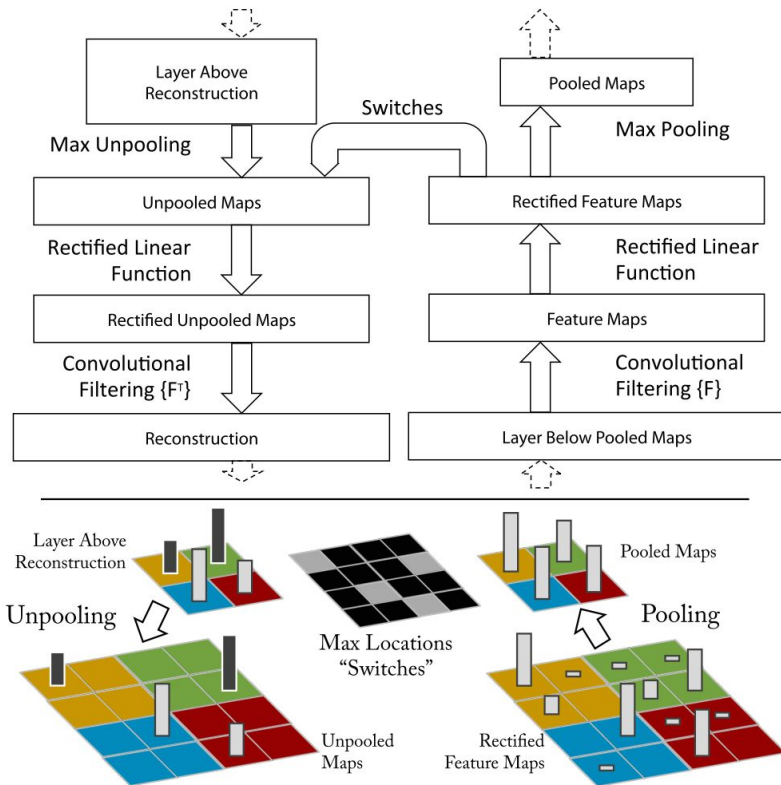
-2	3	-1
6	-3	1
2	-1	3



$$\frac{\partial \text{cost}}{\partial \text{image}}$$



# Deconvnet



A deconvnet can be thought of as a convnet model that uses the same components (filtering, pooling) but in reverse, so **instead of mapping pixels to features does the opposite.**

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.

# Deconvnet

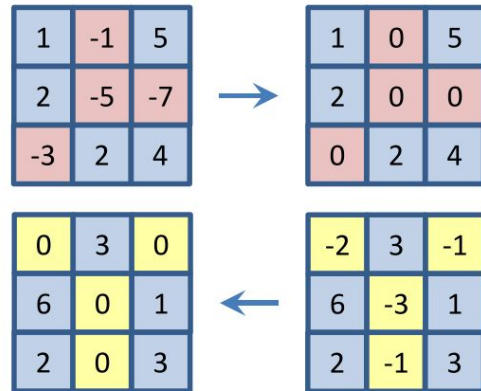
The “**deconvolution**” is equivalent to a **backward pass** through the network, except that when propagating through a nonlinearity, its **gradient is solely computed based on the top gradient signal, ignoring the bottom input**.

In case of the ReLU nonlinearity this amounts to setting to zero certain entries based on the top gradient.

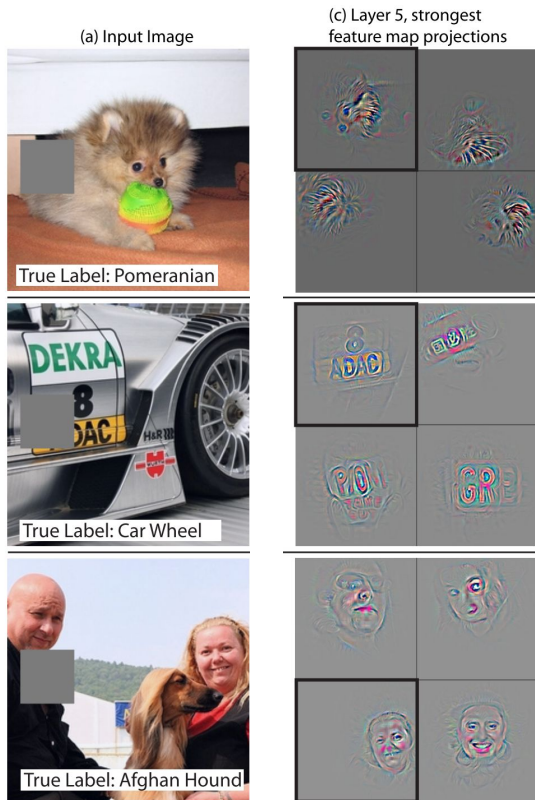
## ReLU

```
def backward(self):  
    self.grad = self.up.grad.copy()  
    self.grad[self.up.grad <= 0.0] = 0.0
```

Forward pass



# Deconvnet



(c) a visualization of this feature map projected down into the input image (black square), along with visualizations of this map from other images.

# Guided Backpropagation

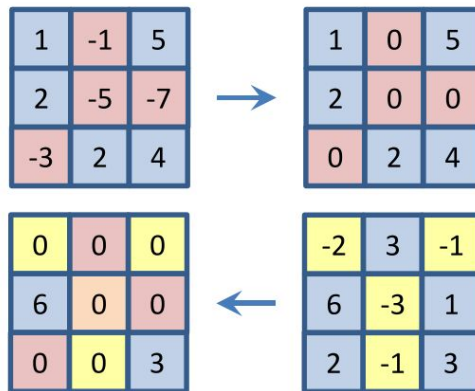
Combination of Deconvolution and Backpropagation approach.

Rather than masking out values corresponding to **negative entries** of the top gradient ('deconvnet') or bottom data (backpropagation), we mask out the values for which **at least one of these values is negative**.

## ReLU

```
def backward(self):
    self.grad = self.up.grad.copy()
    self.grad[np.logical_or(self.grad <= 0.0, \
                             self.input <= 0.0)] = 0.0
```

Forward pass



# Guided Backpropagation

**Prevents backward flow of negative gradients**, corresponding to the neurons which decrease the activation of the higher layer unit we aim to visualize.

Works well **without switches**.

The **bottom-up signal** in form of the pattern of bottom ReLU activations **substitutes the switches**.



# Guided Backpropagation Tensorflow

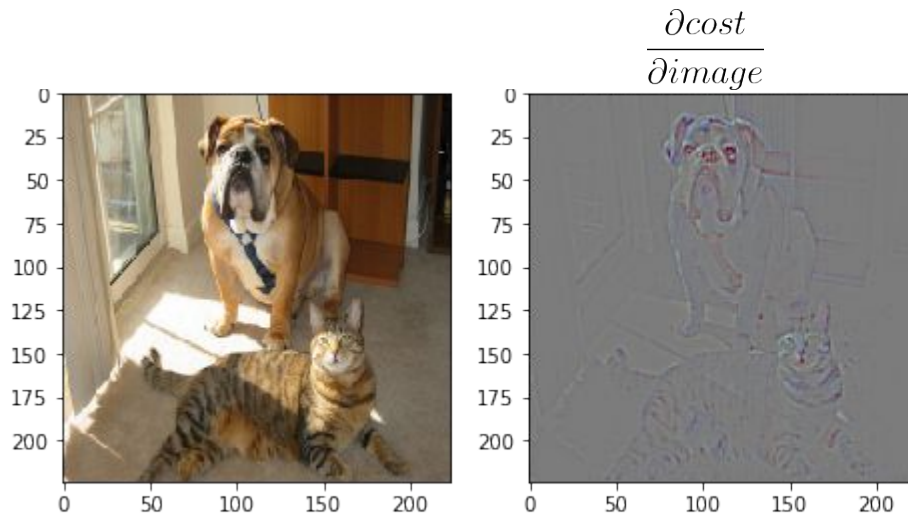
```
import tensorflow as tf

from tensorflow.python.framework import ops
from tensorflow.python.ops import gen_nn_ops

@ops.RegisterGradient("GuidedRelu")
def _GuidedReluGrad(op, grad):
    return tf.where(grad > 0.0,
                    gen_nn_ops._relu_grad(grad, op.outputs[0]),
                    tf.zeros(grad.get_shape()))

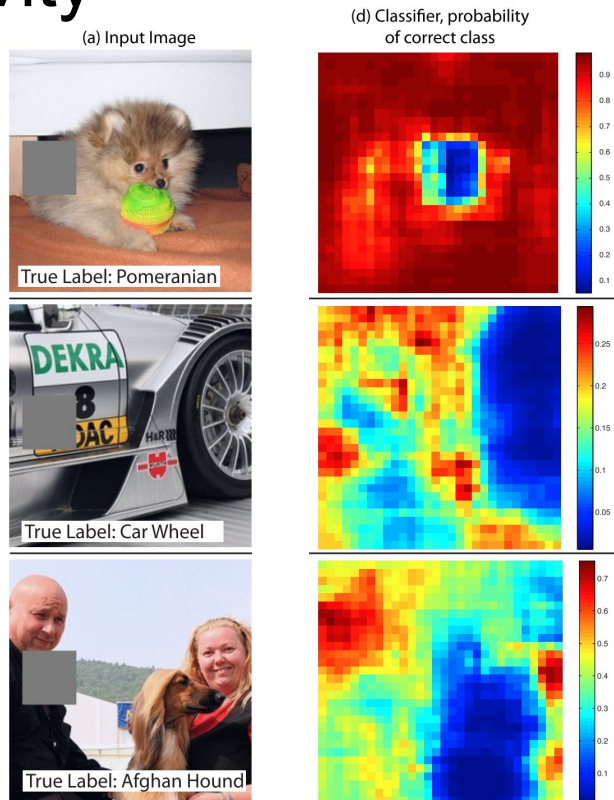
g = tf.Graph()
with g.as_default():
    with g.gradient_override_map({"Relu": "GuidedRelu"}):
        M = model() # example
```

# Guided Backpropagation



# Weakly Supervised Object Localization

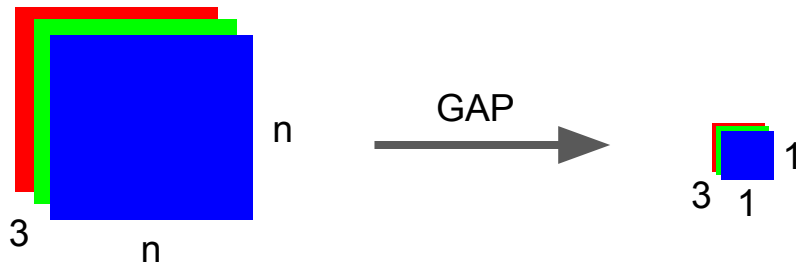
# Occlusion sensitivity



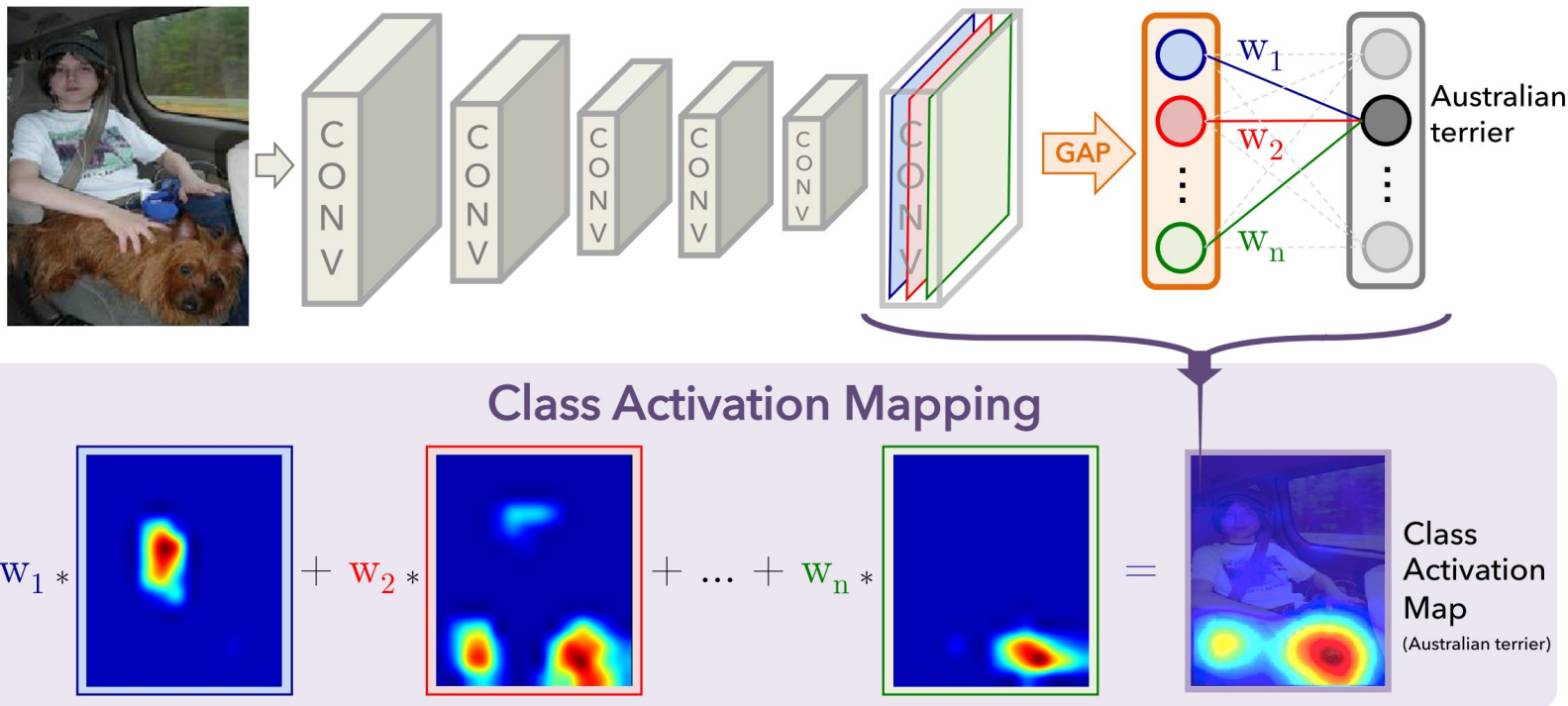
# Class Activation Mapping (CAM)

- “A class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category.”
- Top-5 error for object localization on ILSVRC 2014
  - CAM 37.1%
  - VGG 25.3 %
  - Cldi-KAIST 46.8%

## Global Average Pooling (GAP) layer



# Class Activation Mapping (CAM)



# Class Activation Mapping (CAM)

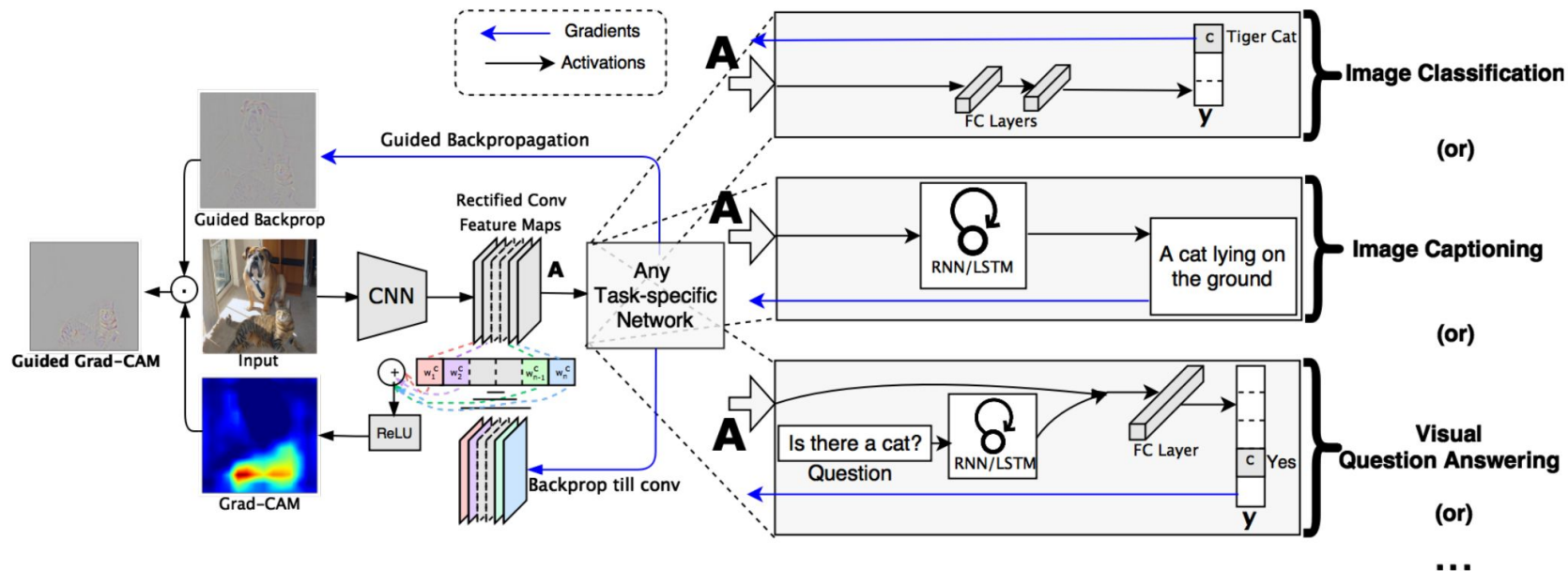
- Advantages
  - Simple way of obtaining localization per class
- Disadvantages
  - Need to be retrained if network is lacking GAP layer.
  - (if we want to apply to VGG-16 we have to remove 2 fully-connected layers)
- Details
  - **Does not average, just sum, could be a problem if directly connected to softmax**  
(<http://seoulai.com/2017/12/23/knowledge-distillation.html>)

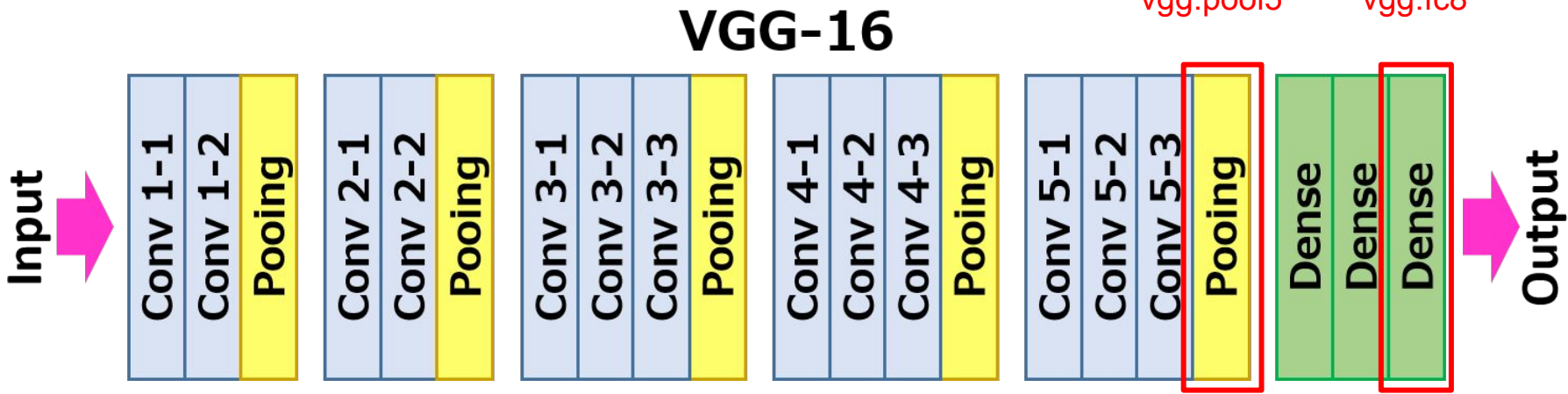
# Grad-CAM

- Another “visual explanation” method
- Advantages
  - No need for architectural changes or re-training
  - Applicable to many types on networks
    - CNNs with fully-connected layers (e.g. VGG)
    - CNNs used for structured outputs (e.g. captioning)
    - CNNs used in tasks with multi-modal inputs (e.g. VQA) or reinforcement learning
- Disadvantages
  - Need to compute gradients up to feature map of interest

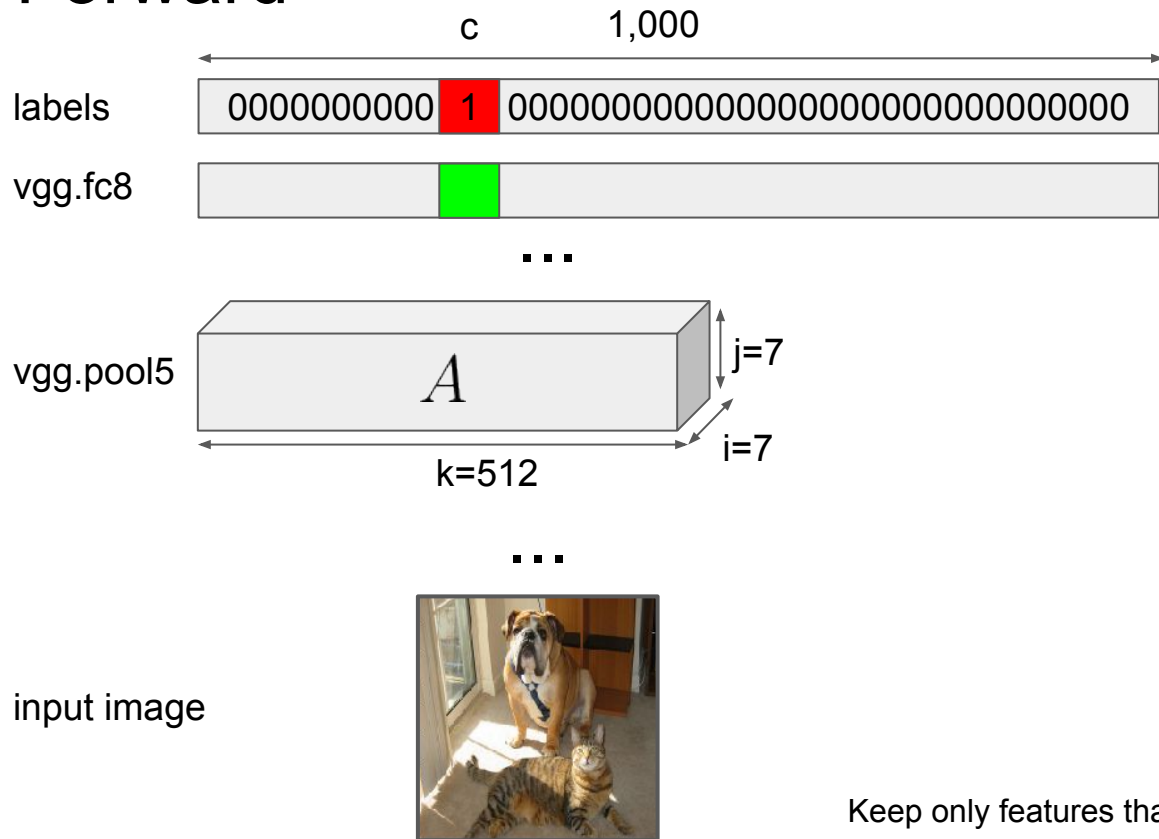


# Grad-CAM





# Forward



# Backward

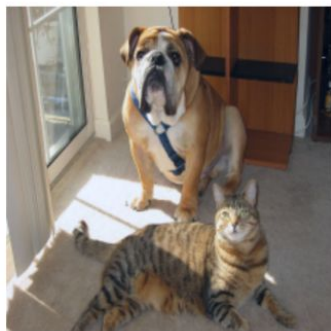
$$y^c = \text{red } 1 * \text{green box}$$

$$\alpha_k^c = \overbrace{\frac{1}{Z} \sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left( \underbrace{\sum_k \alpha_k^c A^k}_{\text{linear combination}} \right)$$

Keep only features that have positive influence.

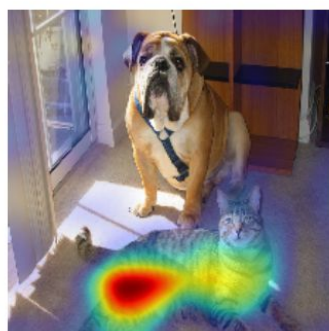
# Guided Grad-CAM



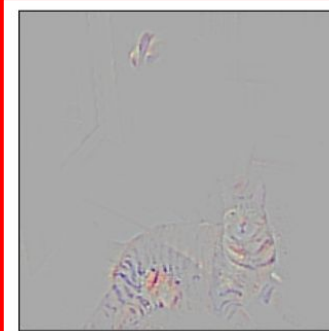
(a) Original Image



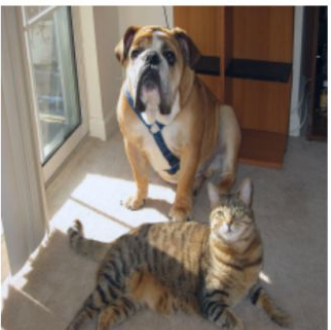
(b) Guided Backprop 'Cat'



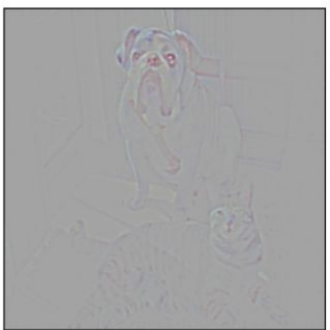
(c) Grad-CAM 'Cat'



(d) Guided Grad-CAM 'Cat'



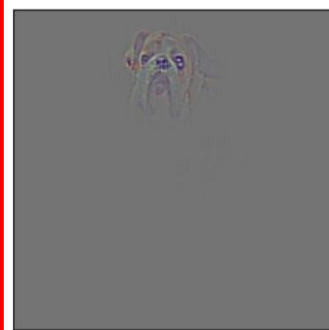
(g) Original Image



(h) Guided Backprop 'Dog'



(i) Grad-CAM 'Dog'



(j) Guided Grad-CAM 'Dog'

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

# References

## **Deconvnet**

Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler and Rob Fergus, 2013

## **Guided Backpropagation**

Striving For Simplicity: The All Convolutional Net, Jost Tobias Springenberg et al., 2015

## **CAM**

Learning Deep Features for Discriminative Localization, Bolei Zhou et al., 2015

**Grad-CAM:** Visual Explanations from Deep Networks via Gradient-based Localization, Ramprasaath R. Selvaraju et al., 2017