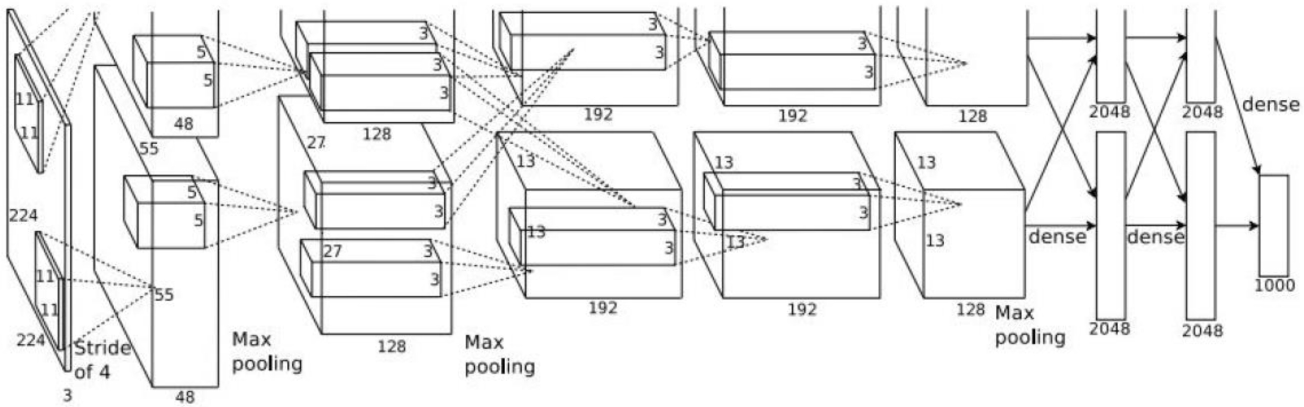
CENG 506 Deep Learning

Lecture 9 – Explainability in Deep Learning

Why Explainability?

- Deep learning is usually more successful than other ML methods like logistic regression, k-NN, decision trees etc. but its explainability level is low.
- It becomes more important to be able to explain ML to various stakeholders (customers, managers etc)
- Regulators/authorities ask more information on how ML models make their decisions in various domains (healthcare, insurance, finance etc)

What is going on inside ConvNets?



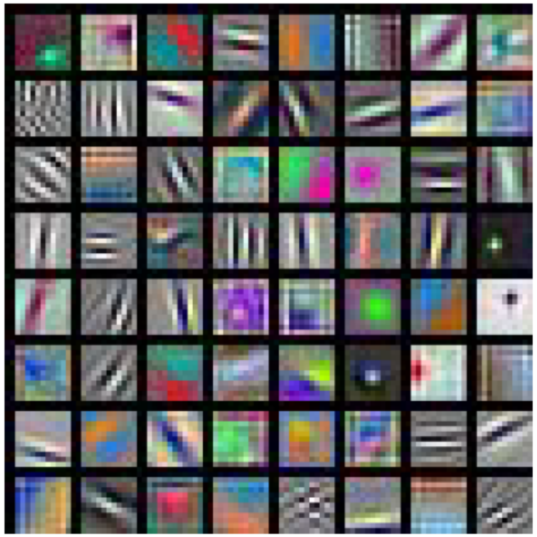
Input Image:
3 x 224 x 224

What are the intermediate features looking for?

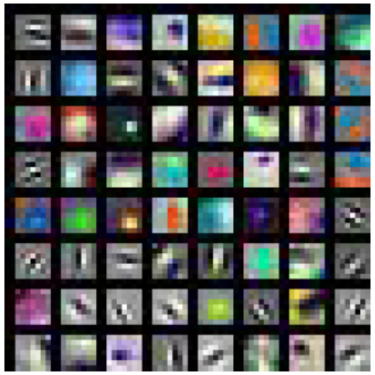
First Layers

We can easily visualize them (as FxFx3 RGB images)

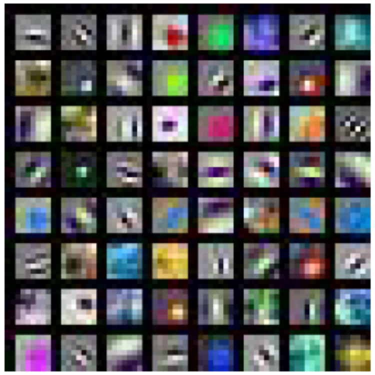
They are sensitive to oriented edges, dots, dominant or opposing colors etc.



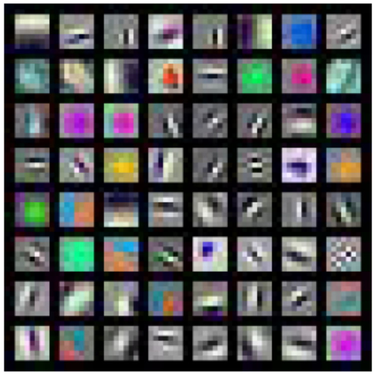
AlexNet:
64 x 3 x 11 x 11



ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7



DenseNet-121:
64 x 3 x 7 x 7

Visualizing Intermediate Layers

Further layers are less interpretable

layer 2 weights

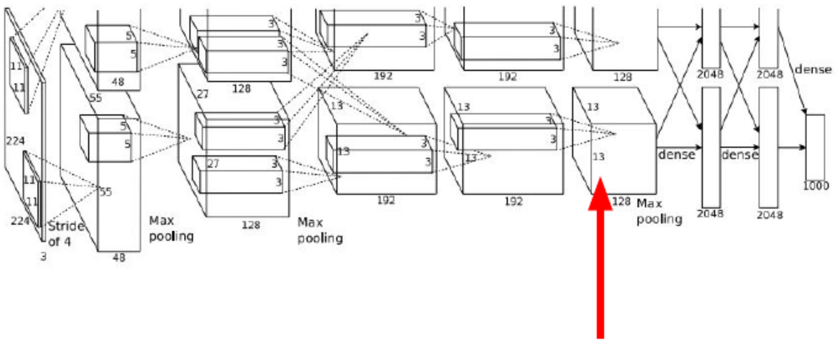
20 x 16 x 7 x 7

layer 3 weights

20 x 20 x 7 x 7

Source: ConvNetJS CIFAR-10 demo

Maximally Activating Patches

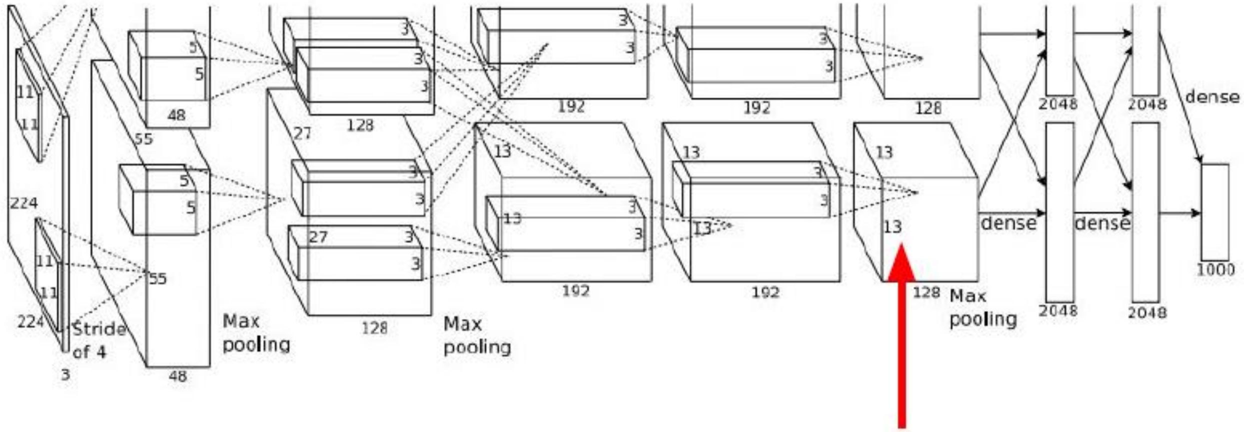


Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick 17th channel

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

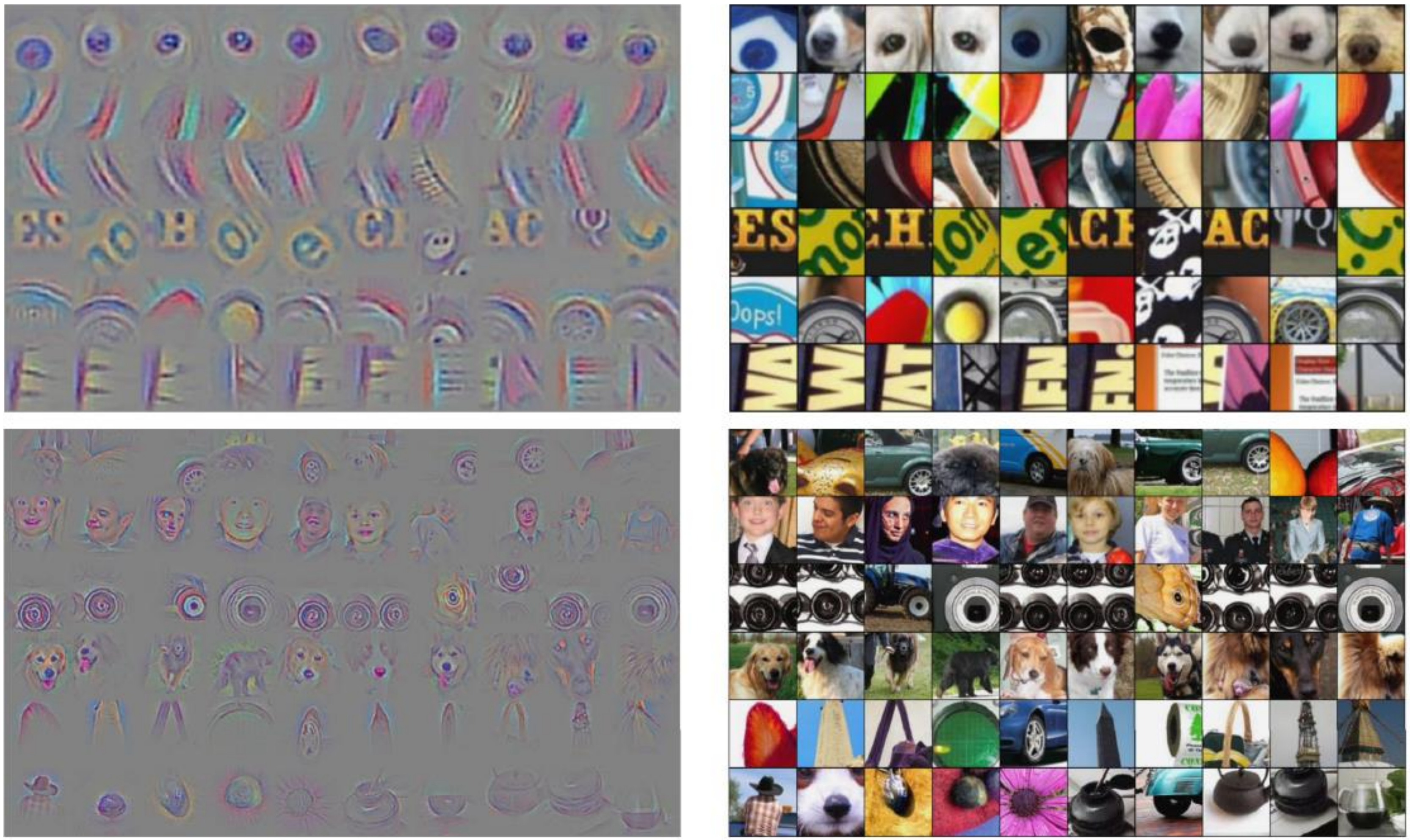
Intermediate features via backprop



Pick a single intermediate neuron,
e.g. one value in 128 x 13 x 13 conv5 feature map.

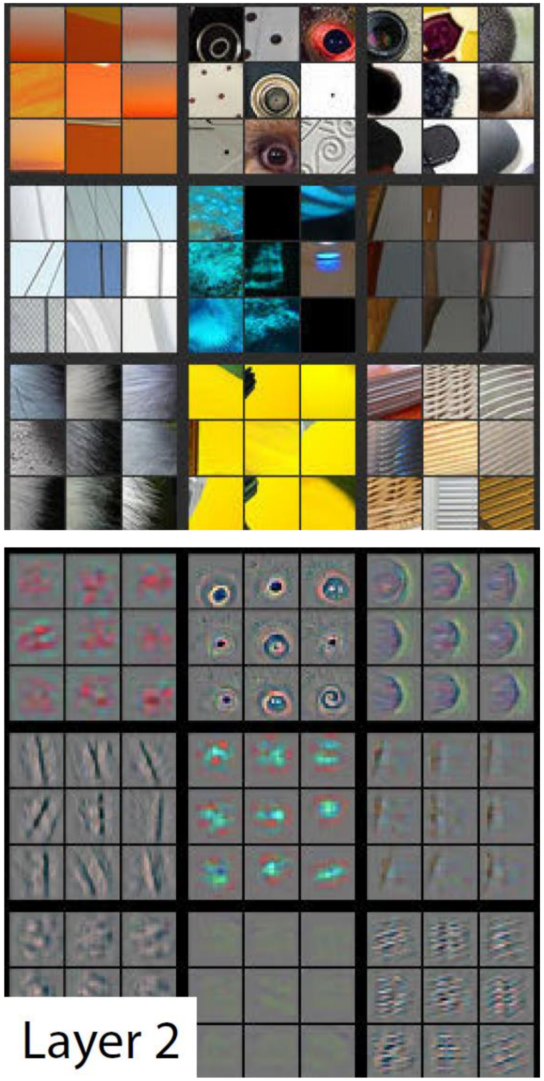
Compute gradient of neuron value with respect to image pixels (only backprop positive gradients - **guided backprop**).

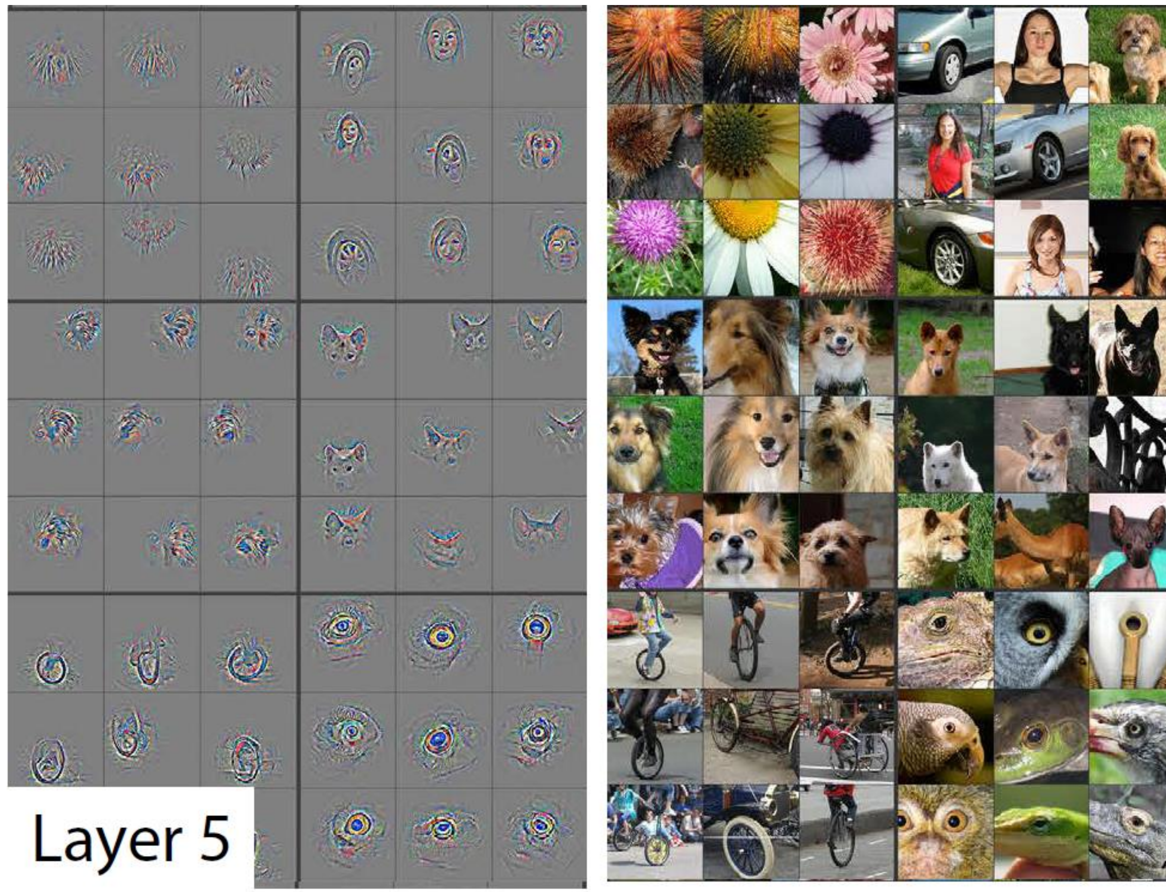
Guided backprop



Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014
Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015

Backprop at different level features





Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Fooling Images / Adversarial Examples

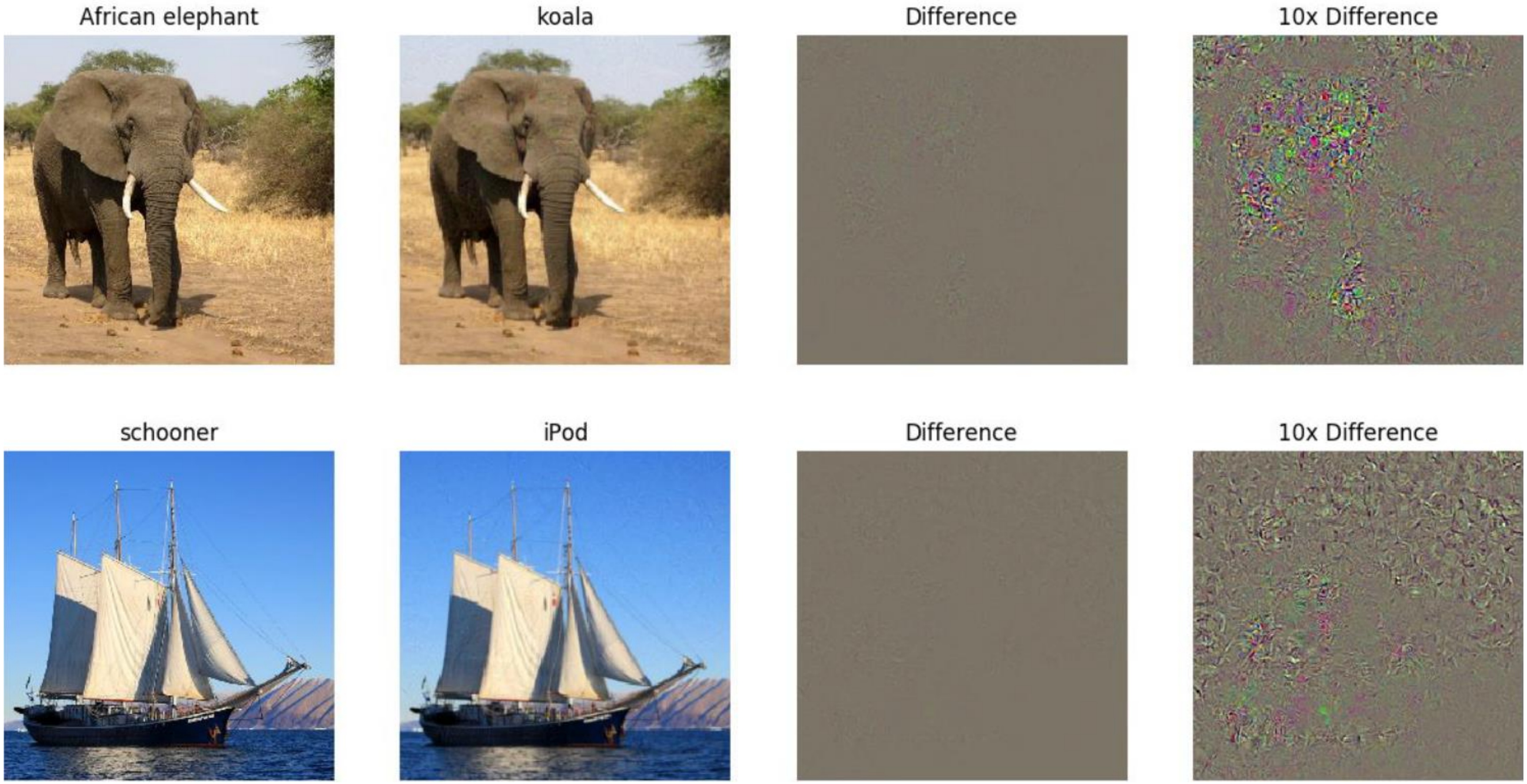
(1) Start from an arbitrary image

(2) Pick an arbitrary class

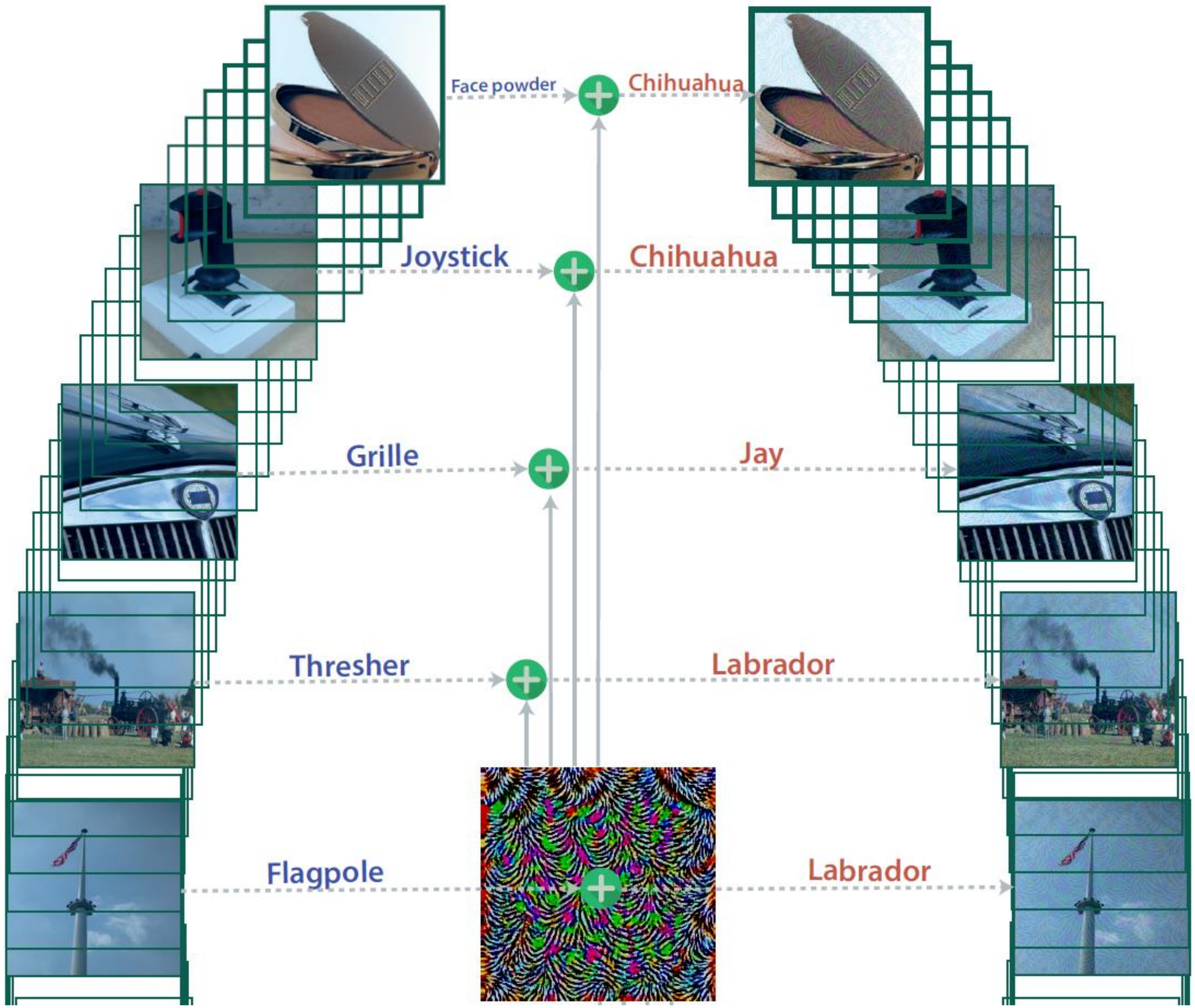
(3) Modify the image to maximize the class

(4) Repeat until network is fooled

Fooling Images / Adversarial Examples



Training Perturbations to Fool Networks

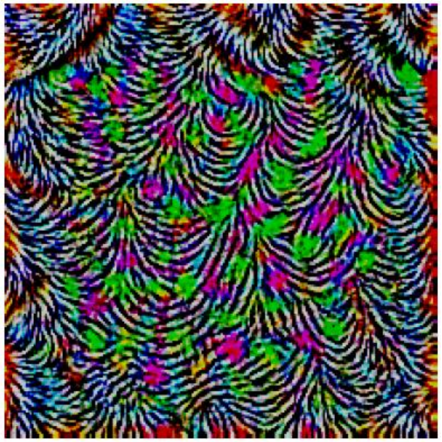


Moosavi-Dezfooli, S.M. Fawzi, A., Fawzi, O., Frossard, P. “Universal adversarial perturbations”, CVPR’17

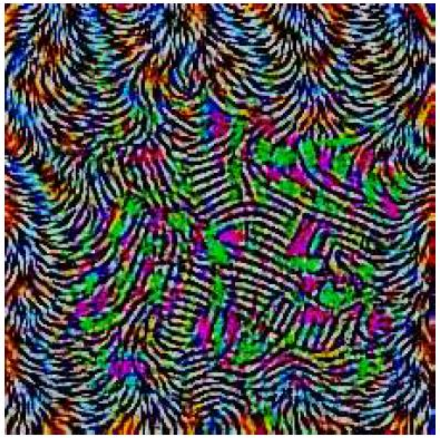
Training Perturbations to Fool Networks

(a) CaffeNet

(b) VGG-F

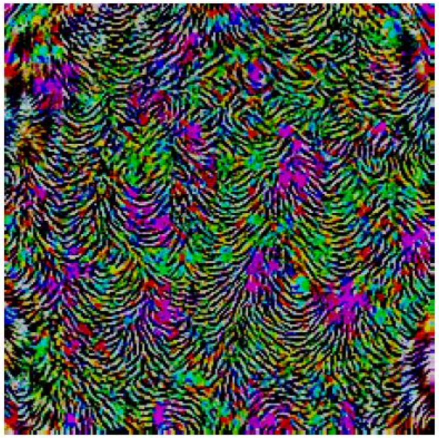


(c) VGG-16



(d) VGG-19

(e) GoogLeNet



(f) ResNet-152

Moosavi-Dezfooli, S.M. Fawzi, A., Fawzi, O., Frossard, P. “Universal adversarial perturbations”, CVPR’17

Training Perturbations to Fool Networks

	VGG-F	CaffeNet	GoogLeNet	VGG-16	VGG-19	ResNet-152
VGG-F	**93.7%**	71.8%	48.4%	42.1%	42.1%	47.4 %
CaffeNet	74.0%	**93.3%**	47.7%	39.9%	39.9%	48.0%
GoogLeNet	46.2%	43.8%	**78.9%**	39.2%	39.8%	45.5%
VGG-16	63.4%	55.8%	56.5%	**78.3%**	73.1%	63.4%
VGG-19	64.0%	57.2%	53.6%	73.5%	**77.8%**	58.0%
ResNet-152	46.3%	46.3%	50.5%	47.0%	45.5%	**84.0%**

Generalizability of the universal perturbations across different networks. **The percentages indicate the fooling rates (for ImageNet data).** The rows indicate the architecture for which the universal perturbations is computed, and the columns indicate the architecture for which the fooling rate is reported. Not surprisingly, networks are best fooled if perturbation is computed on itself.

Moosavi-Dezfooli, S.M. Fawzi, A., Fawzi, O., Frossard, P. “Universal adversarial perturbations”, CVPR’17

Why Test Accuracy is not Enough to trust on ConvNets?

Test accuracy may not capture critical issues

- Bad data
- Biases
- Poor performance in critical cases

Why Test Accuracy is not Enough?

E.g. Train a neural network to predict wolf vs. husky

Husky

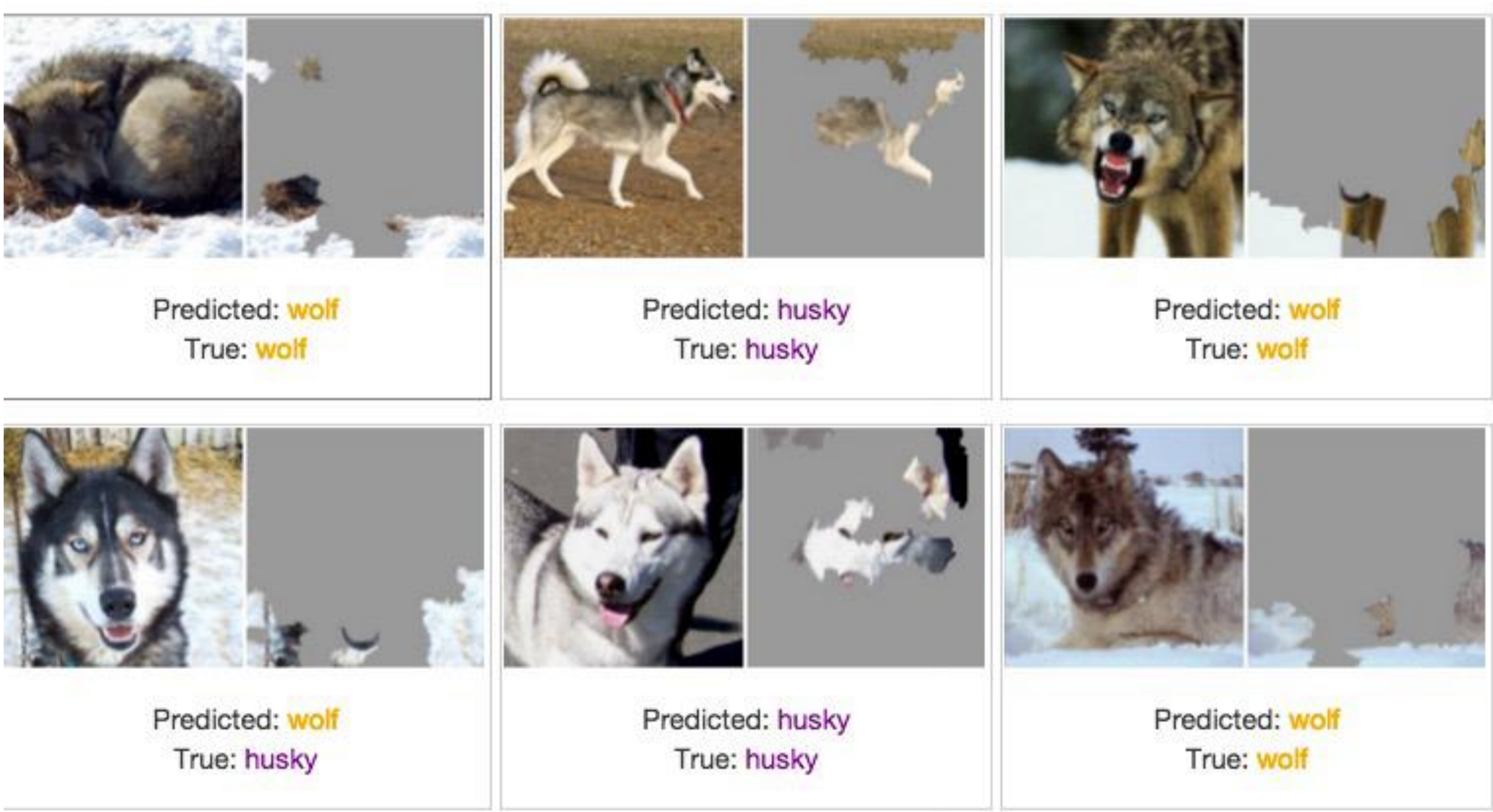
Wolf

Some results:

Source:
Why should I trust you?: Explaining the Predictions of Any Classifier. Ribeiro,Singh&Guestrin KDD'16

Why Test Accuracy is not Enough?

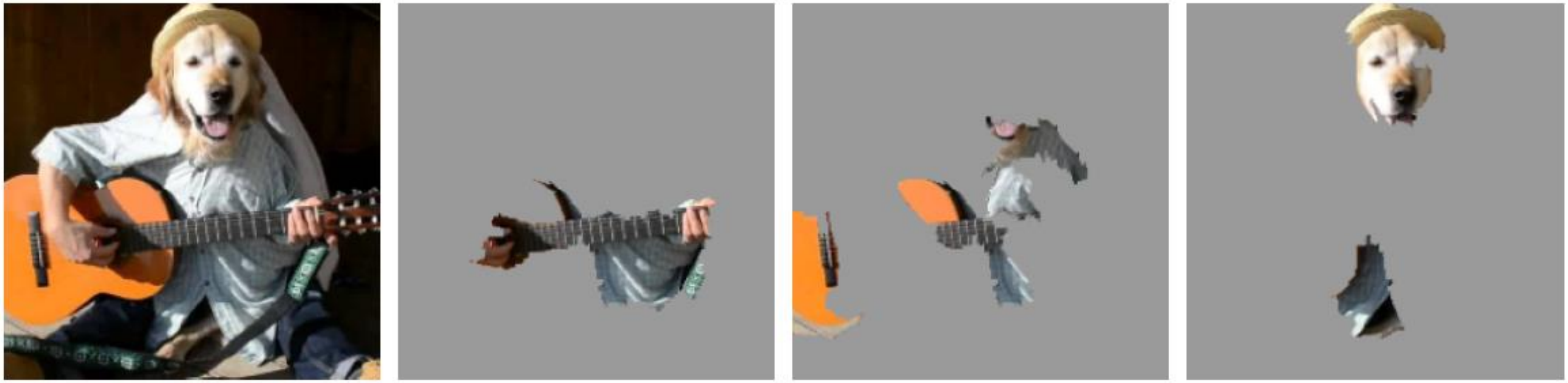
Explanations for neural network prediction



Source:
Why should I trust you?: Explaining the Predictions of Any Classifier. Ribeiro,Singh&Guestrin KDD'16

Why Test Accuracy is not Enough?

Explanations for neural network prediction



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

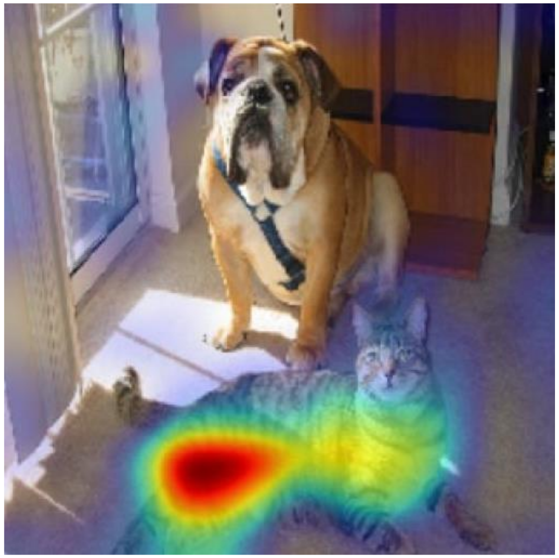
Figure 4: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$), "Acoustic guitar" ($p = 0.24$) and "Labrador" ($p = 0.21$)

Source:
Why should I trust you?: Explaining the Predictions of Any Classifier. Ribeiro,Singh&Guestrin KDD'16

Gradient Class Activation Mapping (Grad-CAM*)

Grad-CAM is a technique to visualize where a CNN is looking. It is class-specific, meaning it can produce a separate visualization for every class.

Original Image



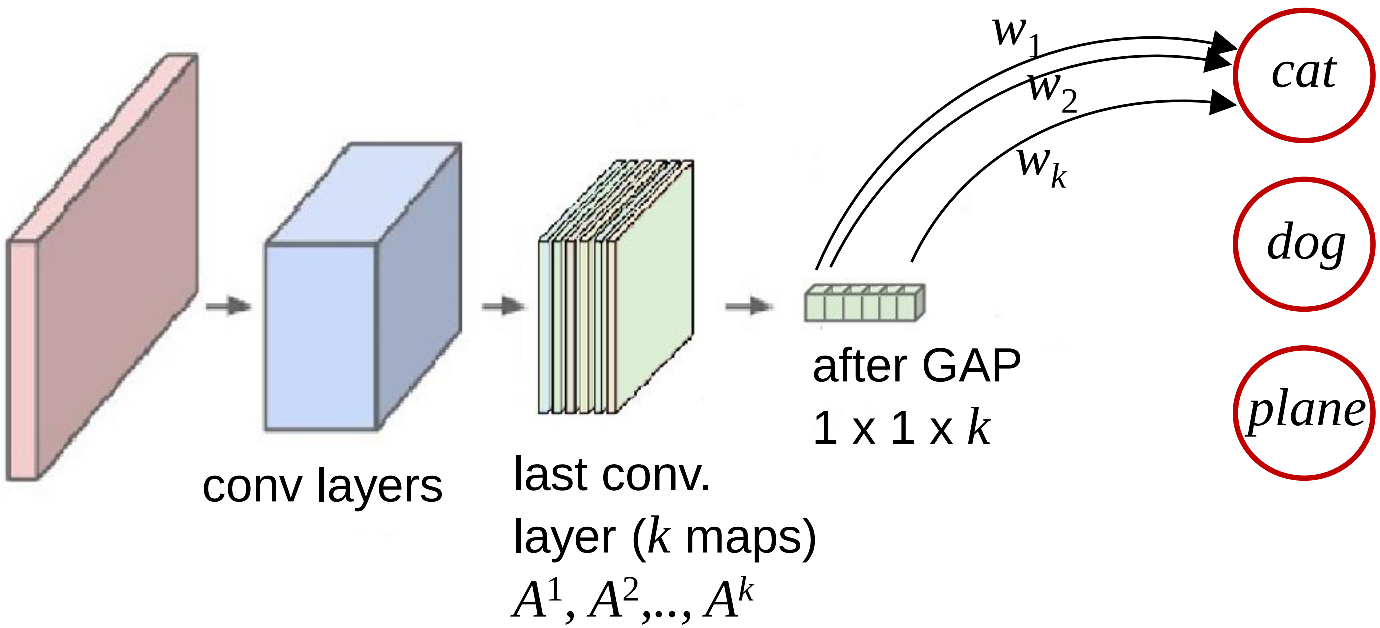
Grad-CAM ‘Cat’

Grad-CAM ‘Dog’

* Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization.” International Journal of Computer Vision 2019.

Class Activation Mapping (CAM*)

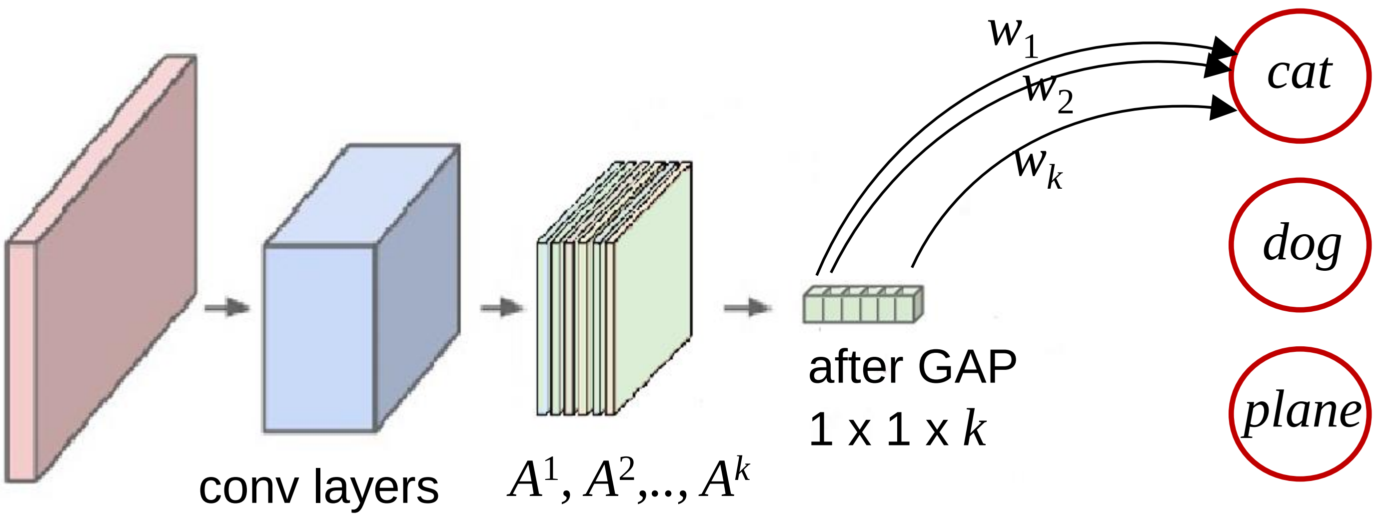
CAM requires applying global average pooling (GAP) to the final convolutional feature maps, followed by a single fully connected layer that produces the predictions:



* https://towardsdatascience.com/grad-cam-2f0c6f3807fe

Class Activation Mapping (CAM*)

For class "cat", the prediction depends on k weights ($w_1, w_2, .., w_k$). To make a CAM heatmap for "cat", we perform a weighted sum of the feature maps, using the "cat" weights of the final FC layer:

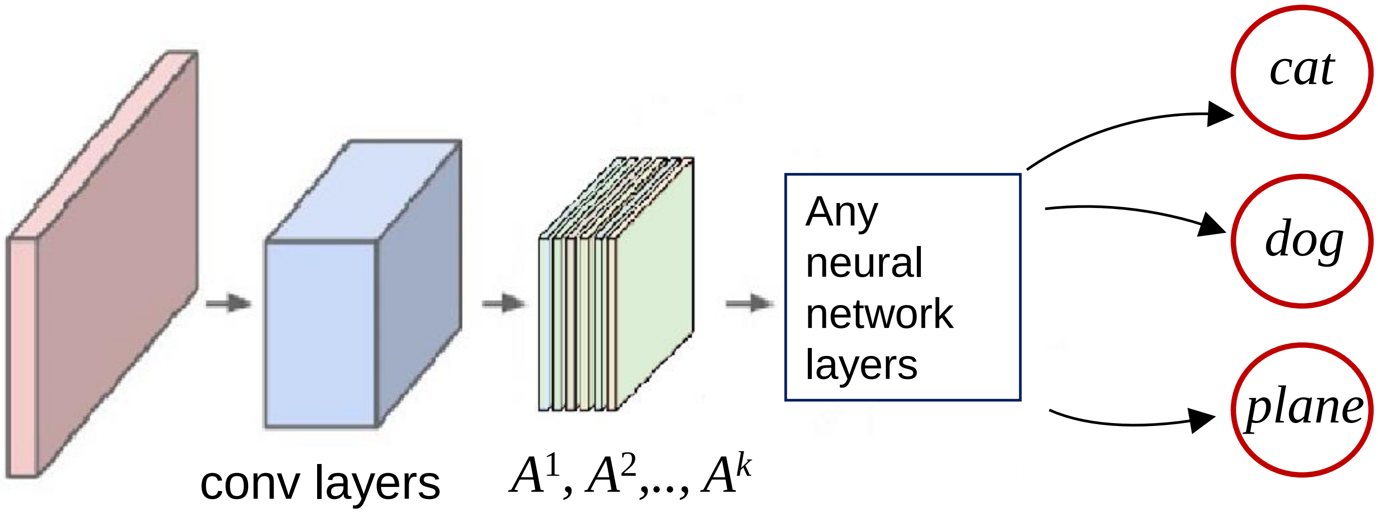


$$CAM^{cat} = w_1 A^1 + w_2 A^2 + \,..\, + w_k A^k = \Sigma_i\, w_i^{cat} A^i$$

* https://towardsdatascience.com/grad-cam-2f0c6f3807fe

Grad-CAM

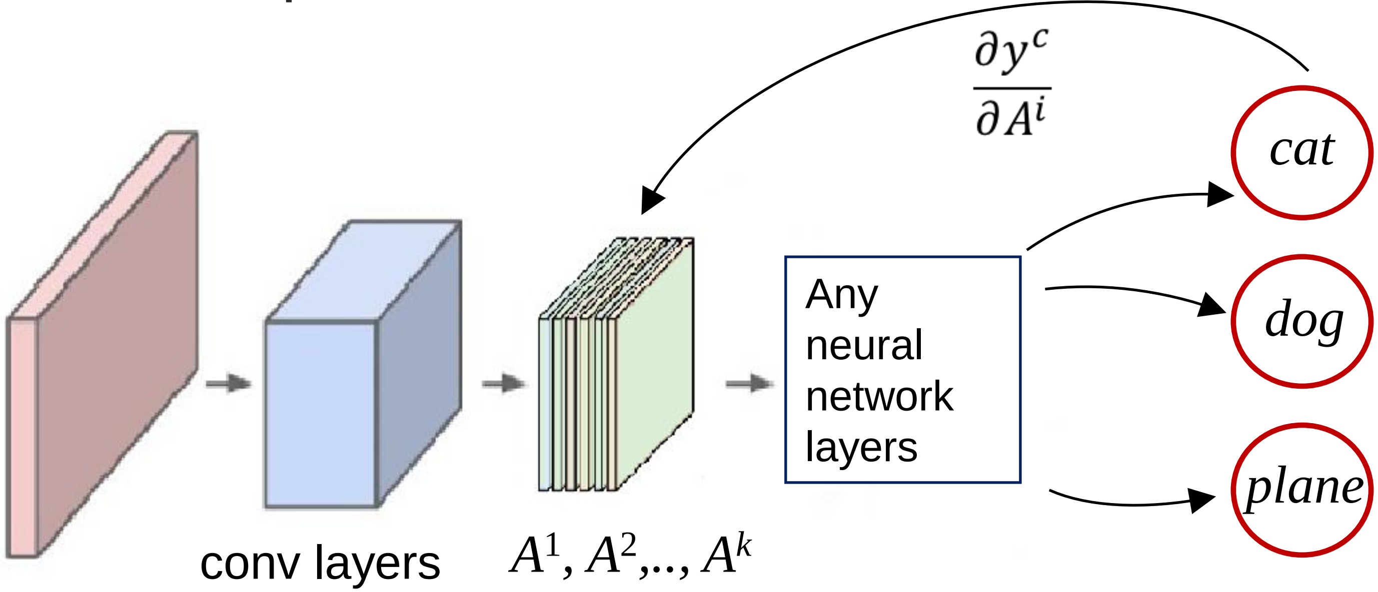
Grad-CAM can use any network between the last conv. layer and the classification layer as long as it is differentiable. We can calculate gradients through any kind of network. I.e. it does not require GAP.



Grad-CAM

Grad-CAM has three steps:

1) Compute the gradient of scores (y^c) w.r.t the last conv layer activation maps



Grad-CAM

2) Calculate alphas by averaging gradients
Global average pool the gradients over the width x height

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

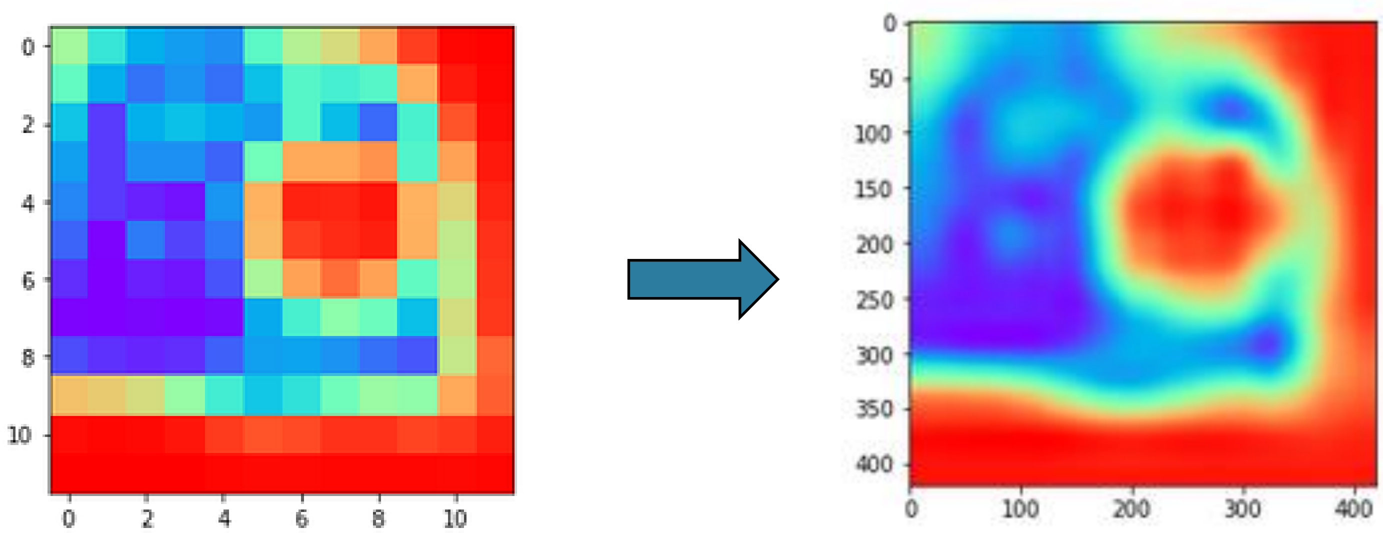
3) Perform a weighted combination of the activation maps where weights are calculated alphas (an additional RELU)

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

Grad-CAM

Q. Won’t the Grad-CAM heatmap be too small?
I.e. height and width of the last conv layer.

A. Smoothed while upsampling



Many variations of CAM*

Method	What it does
GradCAM	Weight the 2D activations by the average gradient
HiResCAM	Like GradCAM but element-wise multiply the activations with the gradients; provably guaranteed faithfulness for certain models
GradCAM++	Like GradCAM but uses second order gradients
XGradCAM	Like GradCAM but scale the gradients by the normalized activations
AblationCAM	Zero out activations and measure how the output drops (this repository includes a fast batched implementation)
ScoreCAM	Perbutate the image by the scaled activations and measure how the output drops
EigenCAM	Takes the first principle component of the 2D Activations (no class discrimination, but seems to give great results)
LayerCAM	Spatially weight the activations by positive gradients. Works better especially in lower layers

* https://jacobgil.github.io/pytorch-gradcam-book/introduction.html