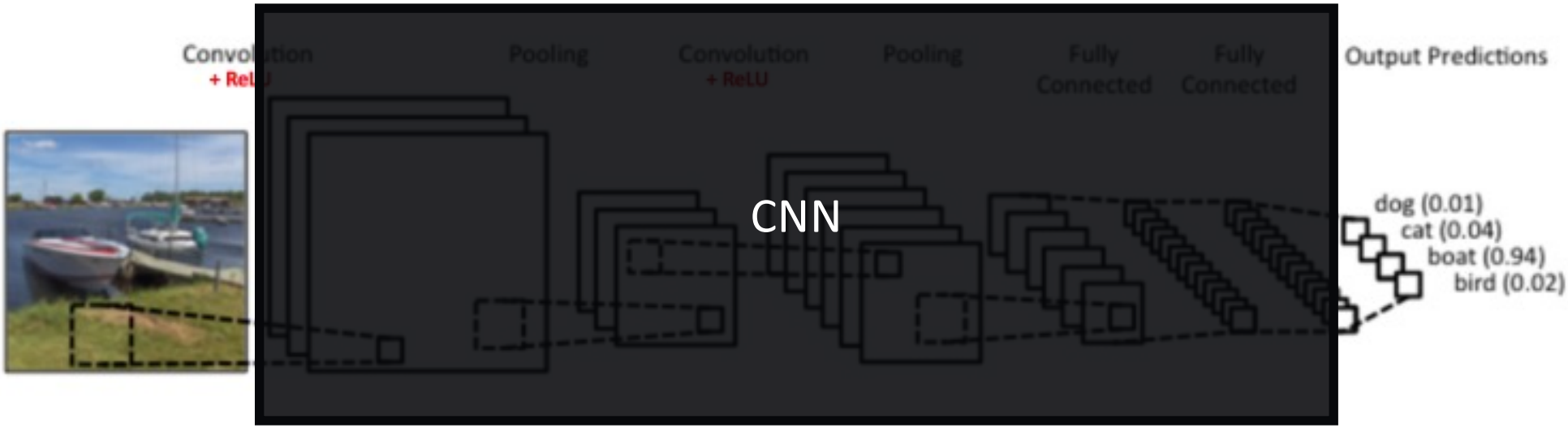
Week 08: Visualizing CNN

Dr. Hongping Cai
Department of Computer Science
University of Bath

Topic 1: Visualizing Activation Maps



Q: What does CNN learn?

A: Visualizing CNN

Different ways

Visualizing Filters

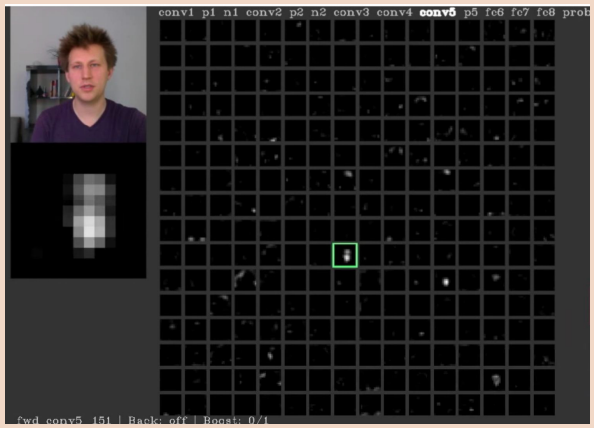
Display Filters directly

Find patches in a dataset

Synthesize patches

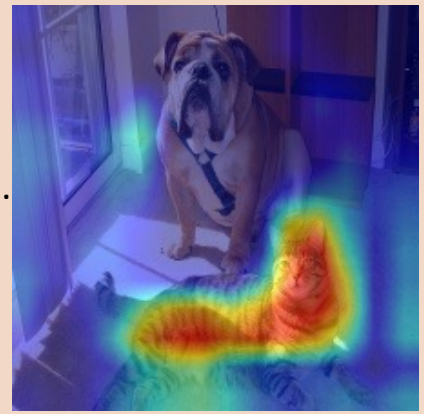
- Feature Inverse
- Texture synthesis
- Neural Style Transfer

Visualizing Activation Maps

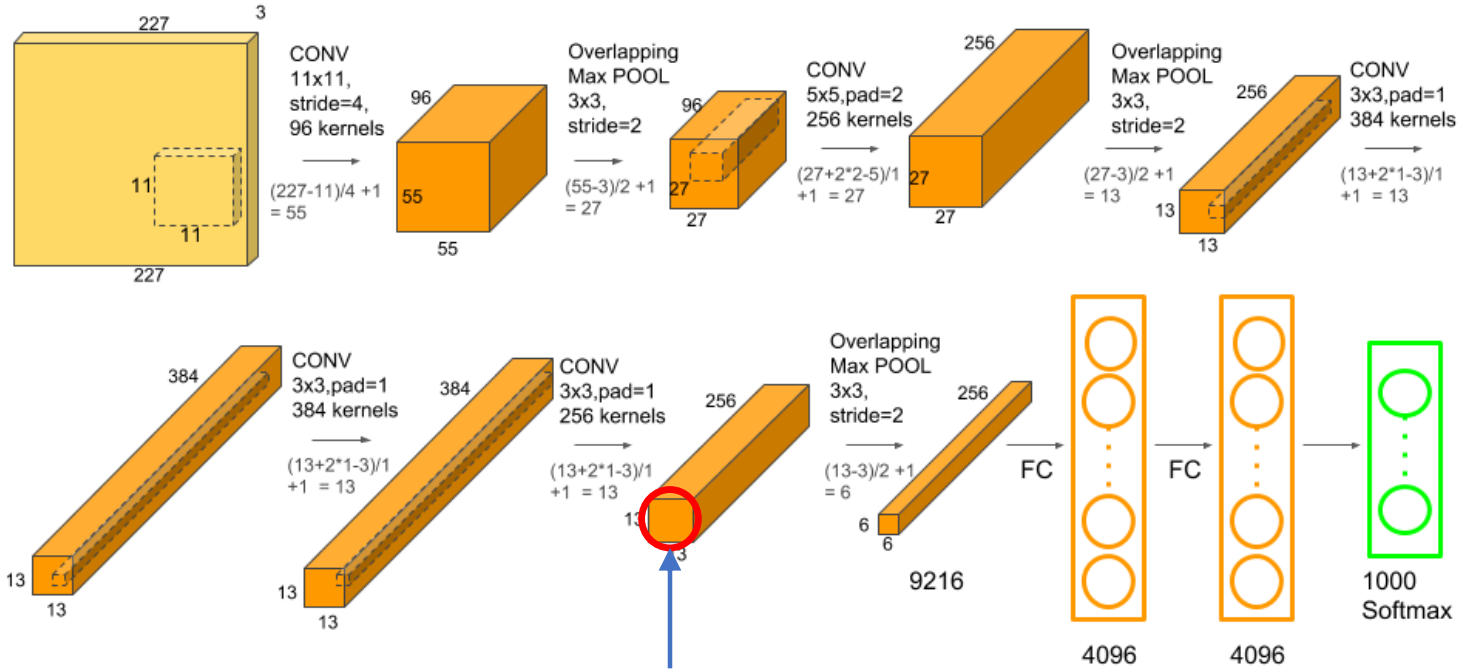


Visualizing Heatmaps

CAM, Grad-CAM, …



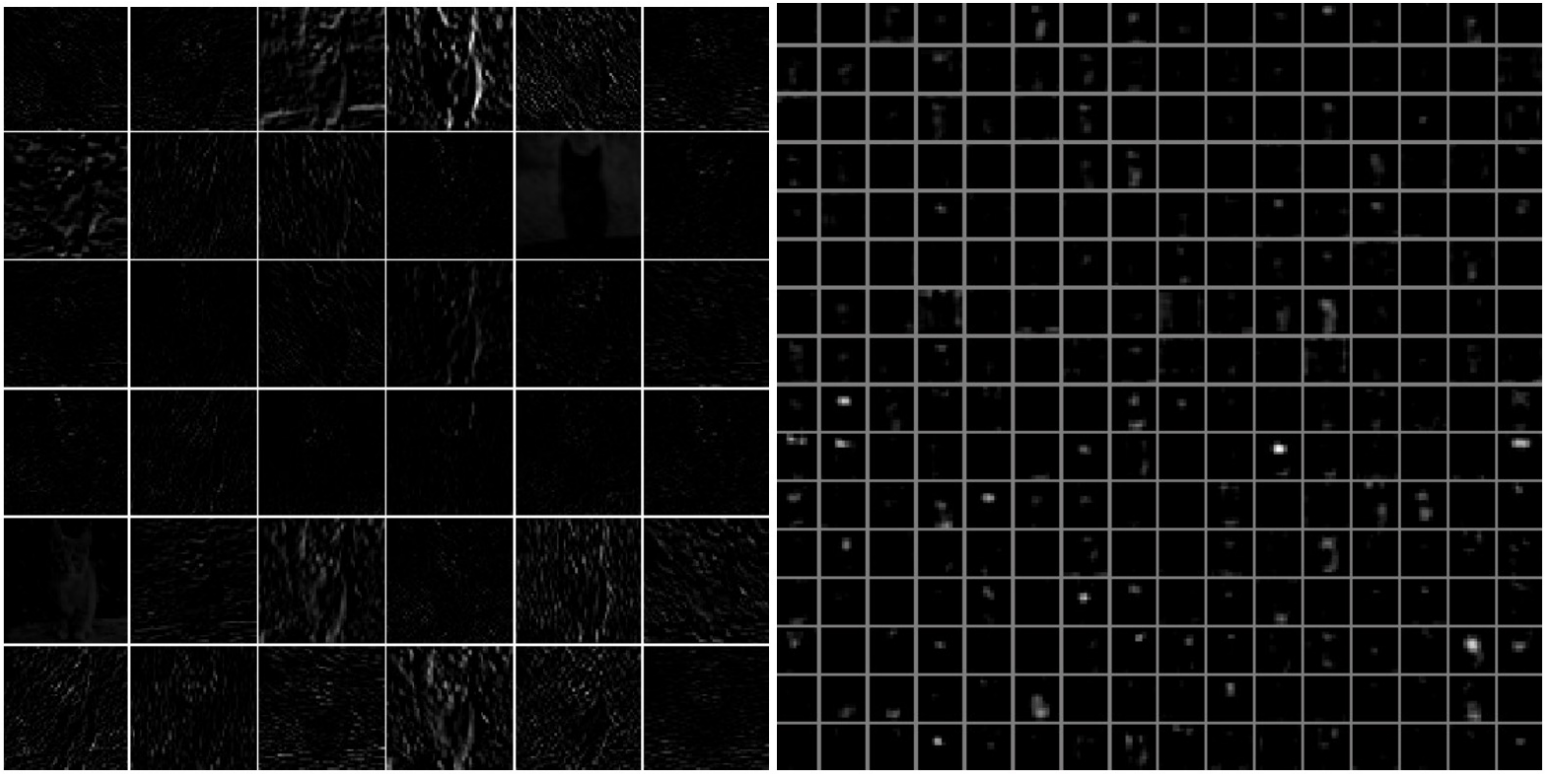
Visualizing the activation maps



Visualizing the activation maps

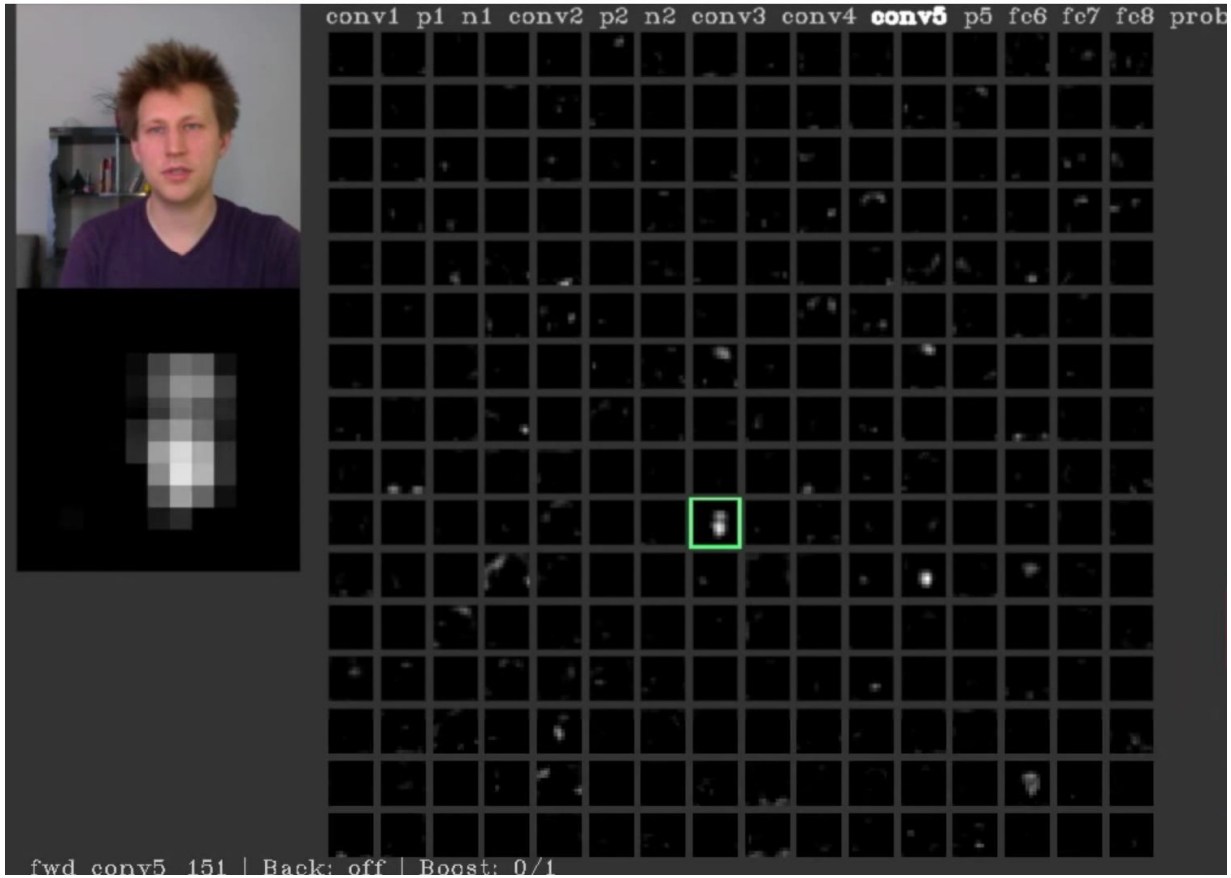
Image from: https://learnopencv.com/understanding-alexnet/

Visualizing the activation maps



Activation maps on the first CONV layer (left), and the 5th CONV layer (right)
of a trained AlexNet of a cat image

Visualizing the activation maps



256 Filters (each size: 13x13) in conv5 layer

Q: What does it mean that many higher-layer feature maps are blank?

A: The pattern encoded by these filters were not found in the input image.

For higher layers, the feature maps become less interpretable.

References

- Online Lecture: “Visualizing what ConvNets learn”. https://cs231n.github.io/understanding-cnn/
- Video lecture in University of Stanford. “Visualizing and Understanding”. 2017. https://www.youtube.com/watch?v=6wcs6szJWMY
- Blog by Jason Yosinski. “Understanding Neural Networks Through Deep Visualization”. 2015. https://yosinski.com/deepvis

Topic 2: Visualizing Filters

Visualizing Filters

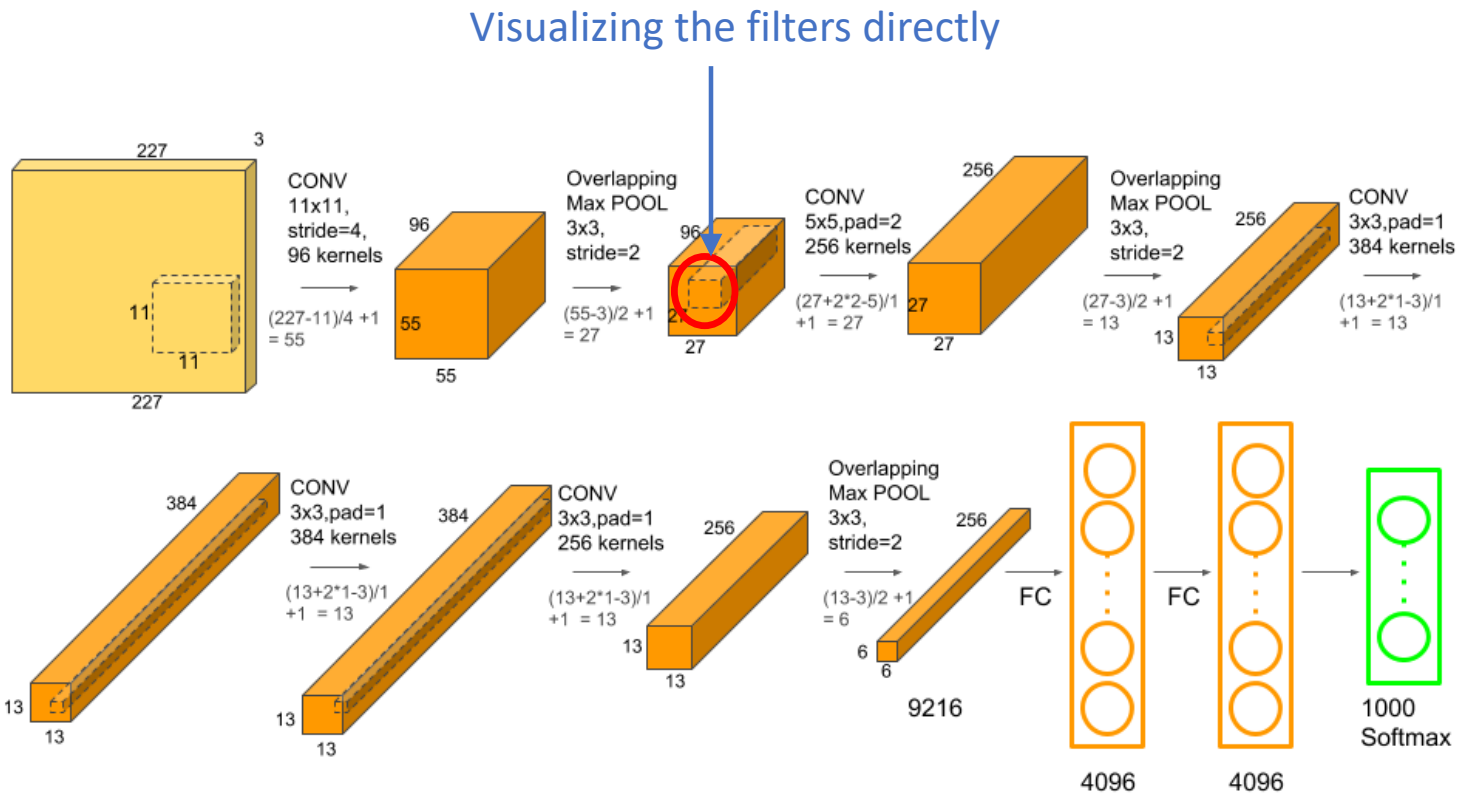
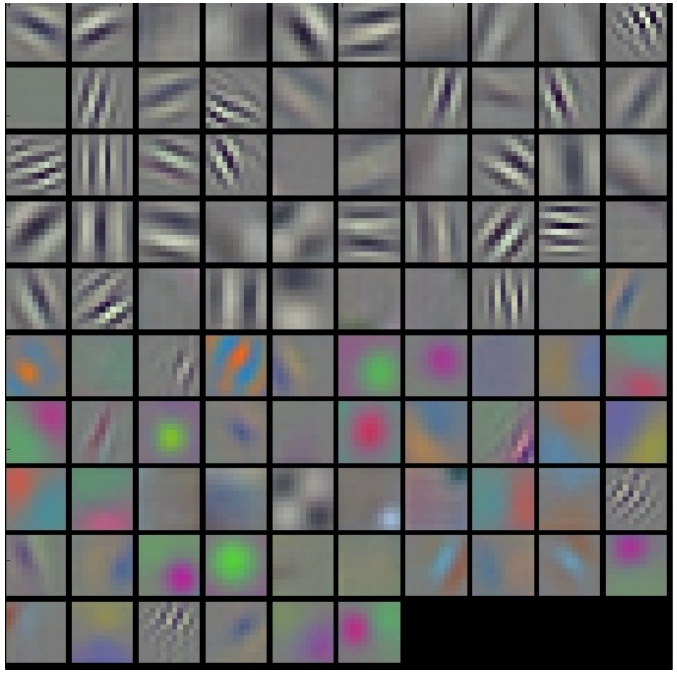


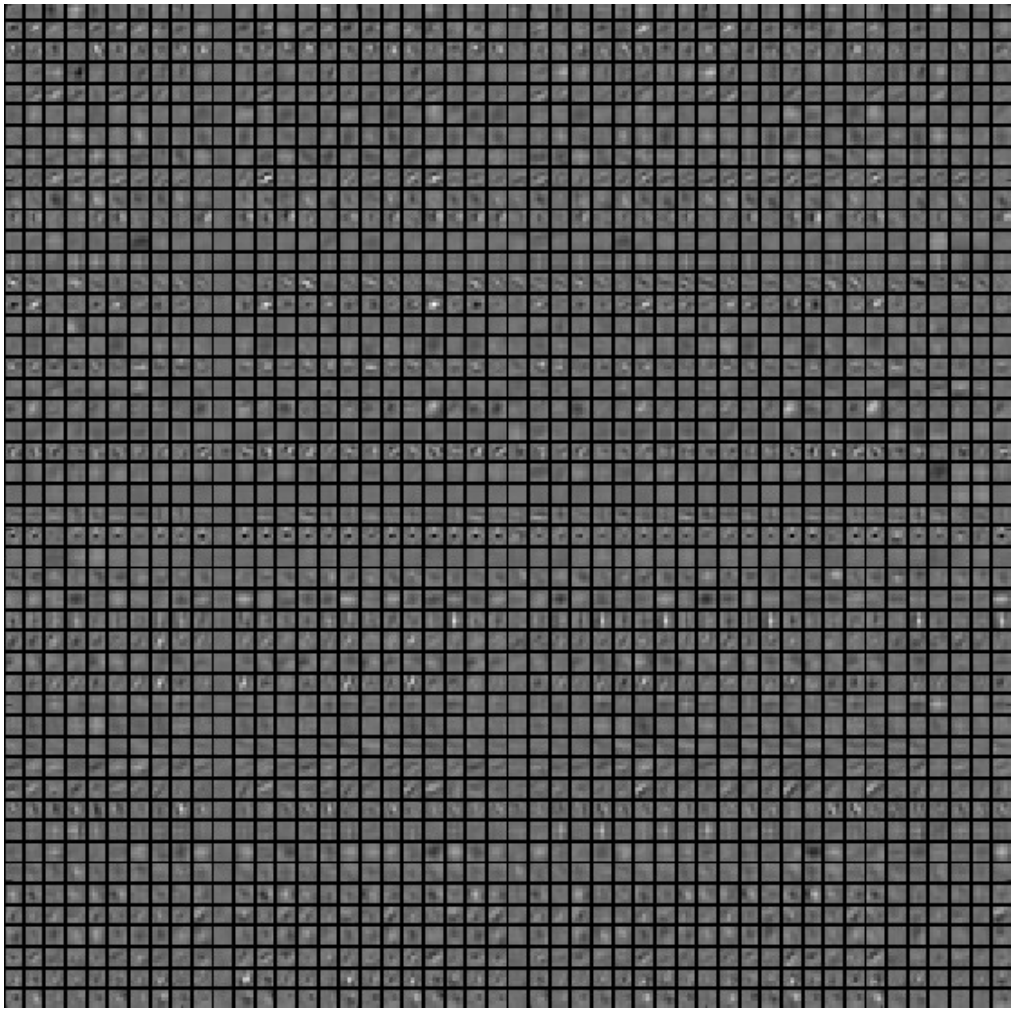
Image from: https://learnopencv.com/understanding-alexnet/

Visualizing Filters

1st Conv Layer of AlexNet



2nd Conv Layer of AlexNet

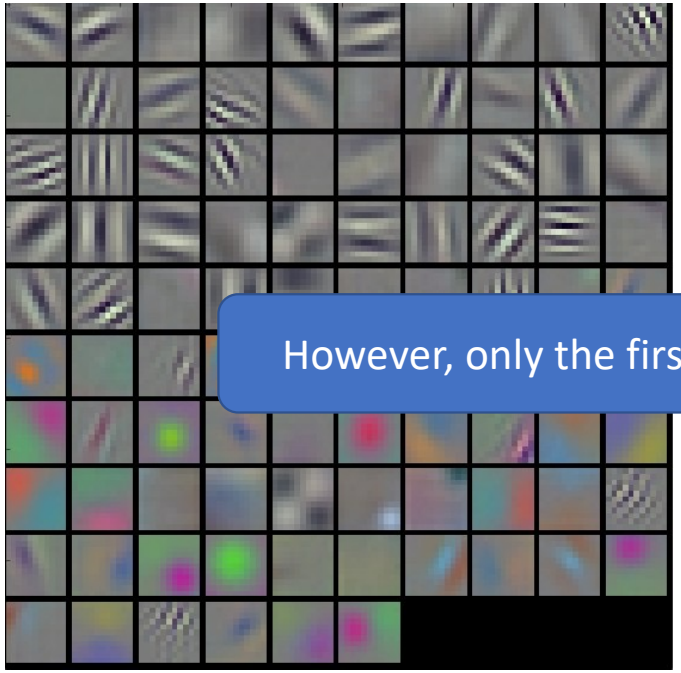


- Helpful to check if the network is trained properly (no noisy pattern)

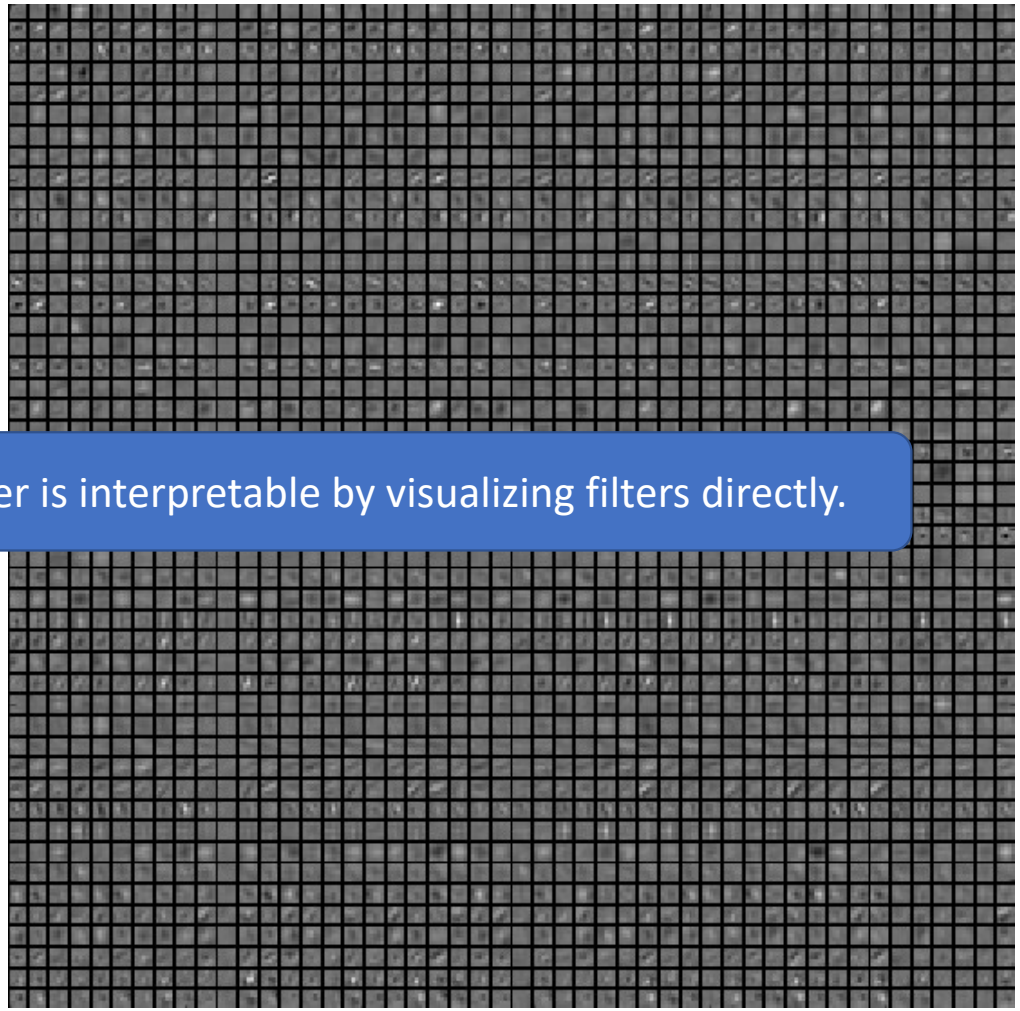
Image from: https://cs231n.github.io/understanding-cnn/

Visualizing Filters

1st Conv Layer of AlexNet



2nd Conv Layer of AlexNet

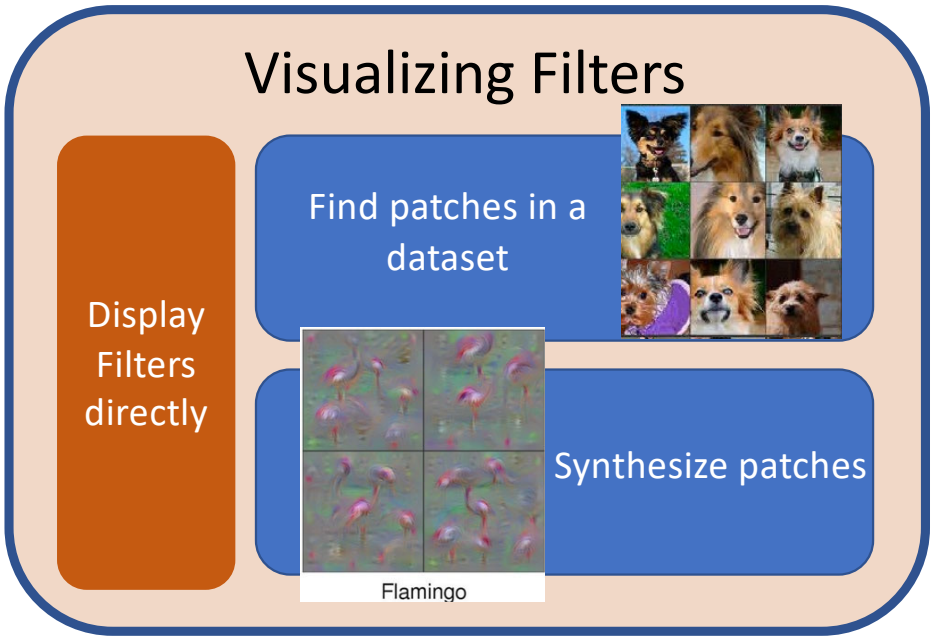


However, only the first layer is interpretable by visualizing filters directly.

- Helpful to check if the network is trained properly (no noisy pattern)

Image from: https://cs231n.github.io/understanding-cnn/

Visualizing Filters implicitly



- Instead of visualizing filters directly, visualize what sort of input maximizes the activation of one filter.
- We get an idea of what pattern each filter has learnt to extract.

Maximally Activating Patches

- Find patches produce maximal activation for a certain filter

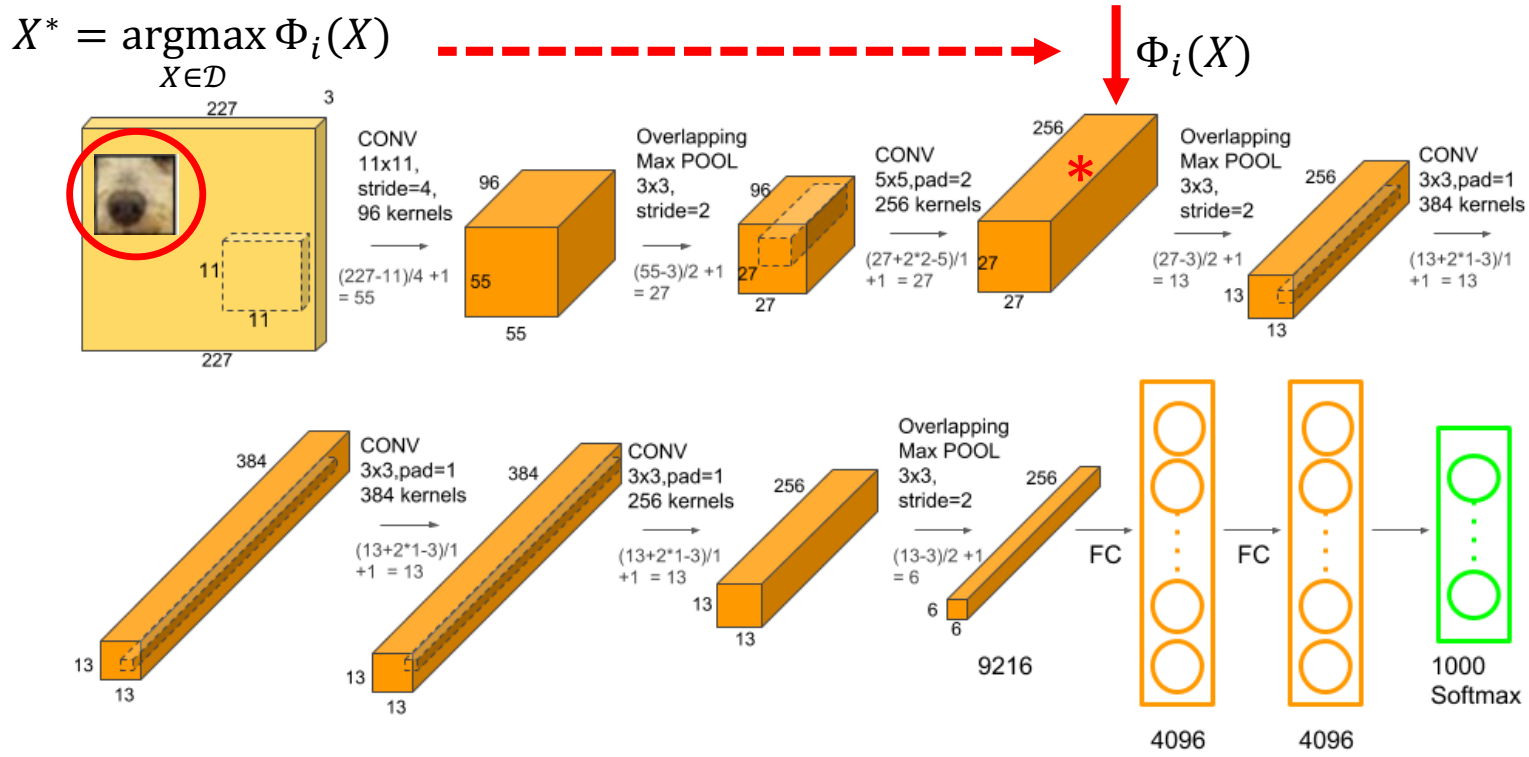
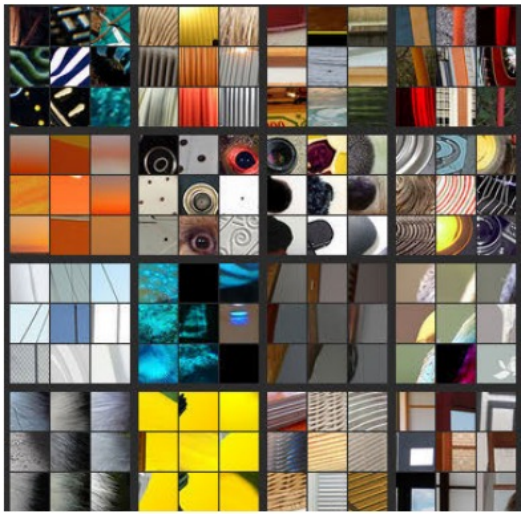


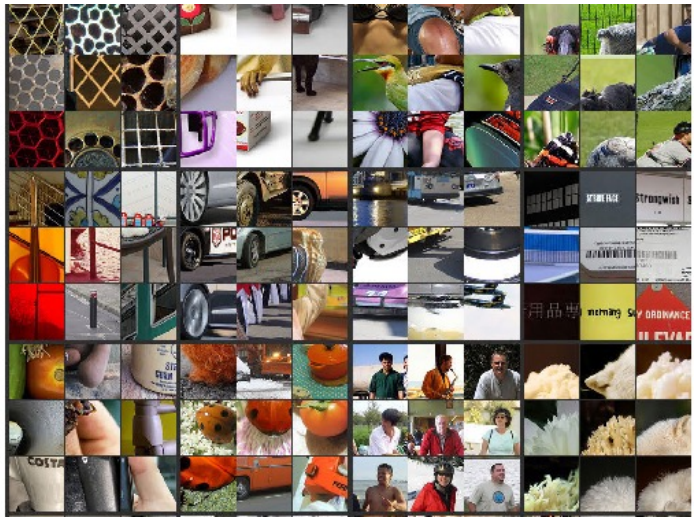
Image from: https://learnopencv.com/understanding-alexnet/

Maximally Activating Patches

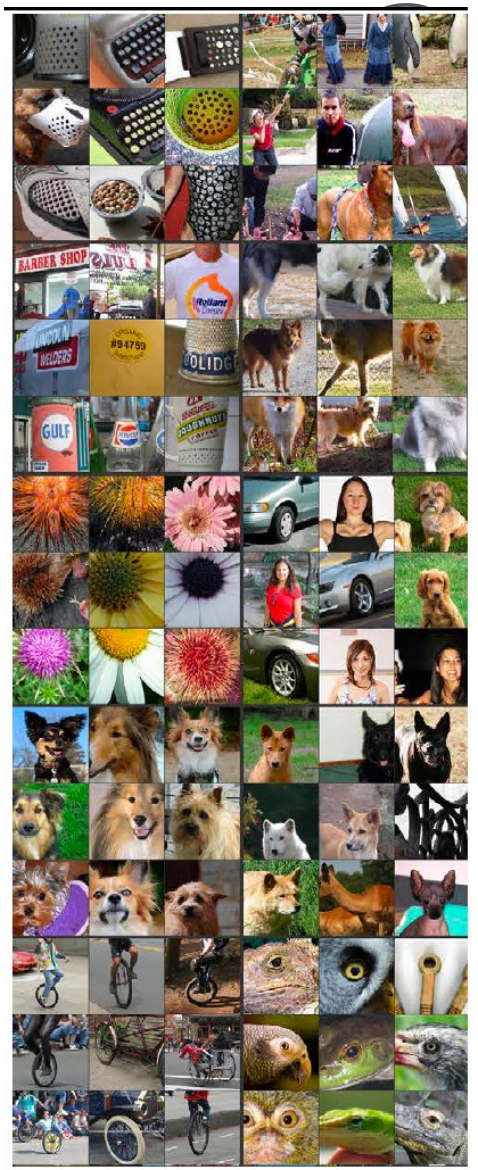
Top 9 image patches (from ImageNet) maximally activating each filter



Layer 2



Layer 3



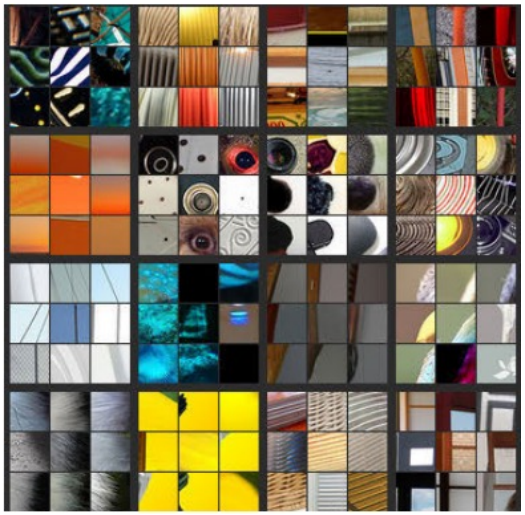
Layer 5

Zeiler, Fergus. “Visualizing and Understanding Convolutional Networks”. ECCV14

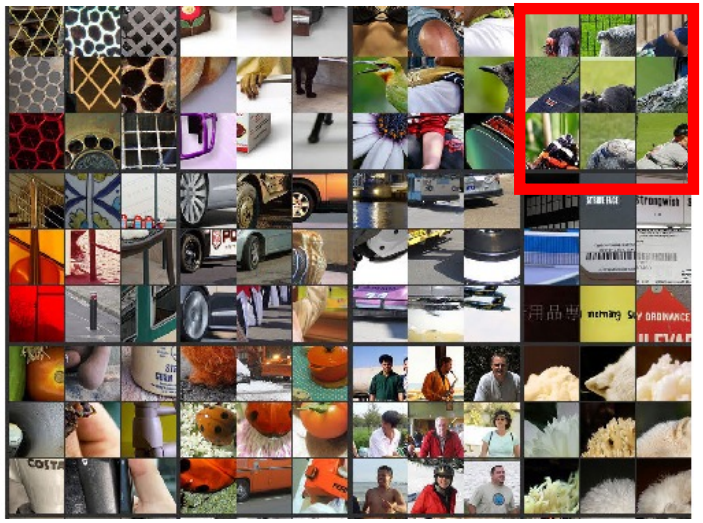
UNIVERSITY OF BATH

Maximally Activating Patches

Top 9 image patches (from ImageNet) maximally activating each filter



Layer 2



Layer 3

Layer 5

Q1: What are these 9 patches in common?

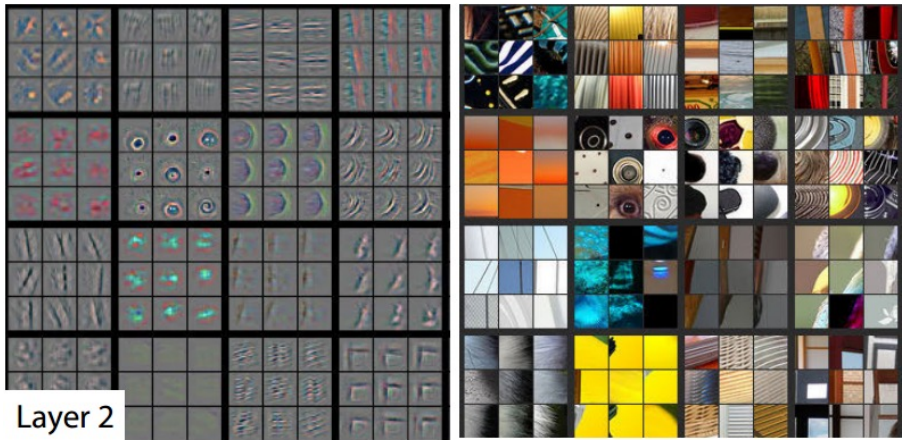
Q2: What is the discriminative part for these 9 patches?

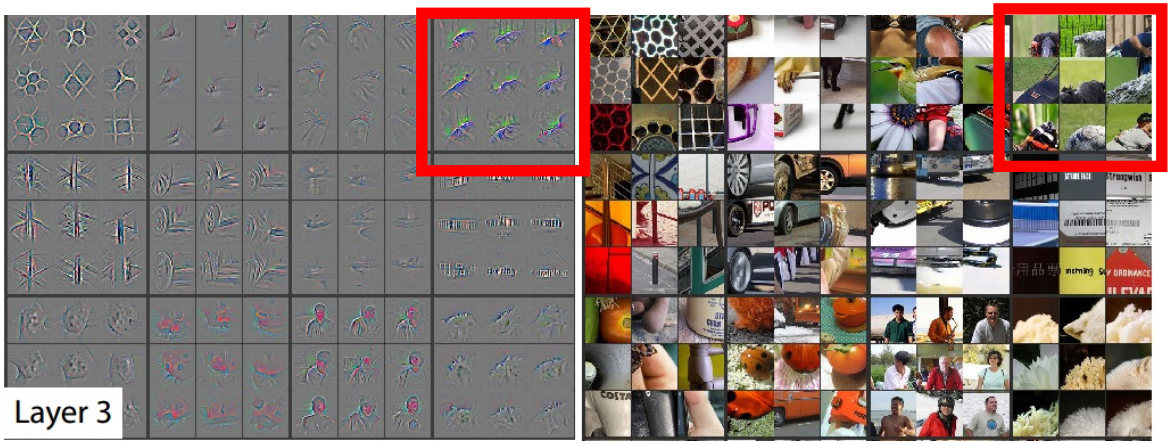
We need also visualize which pixels in the patch “excite” a given filter.
Solution: **Deconvnet**; **Guided-backpropagation**

Zeiler, Fergus. “Visualizing and Understanding Convolutional Networks”. ECCV14

Decovnet [Zeiler-ECCV14]

Deconvolutional Network (Decovnet) can be thought of as a ConvNet that uses the same components (conv, pooling) but in reverse, so instead of mapping pixels to features, deconvnet projects the feature activations back to the input pixel space.

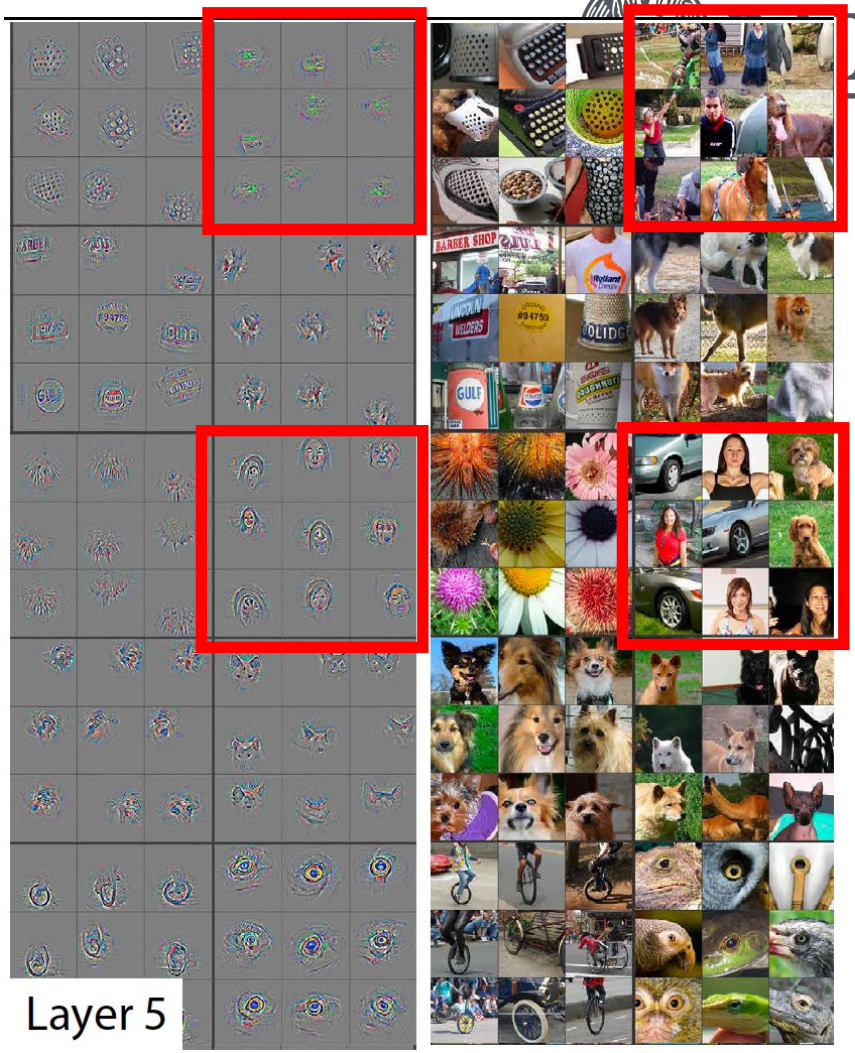




Zeiler, Fergus. “Visualizing and Understanding Convolutional Networks”. ECCV14

Decovnet [Zeiler-ECCV14]

- Strong grouping within each feature map
- Greater invariance at higher layers
- Exaggeration of discriminative parts of the patch



Zeiler, Fergus. “Visualizing and Understanding Convolutional Networks”. ECCV14

Synthesize an image to maximize the activation

$$X^* = \underset{X \in \mathbb{R}^{u \times v \times 3}}{\operatorname{argmax}} \Phi_i(X)$$

$\Phi_i(X)$

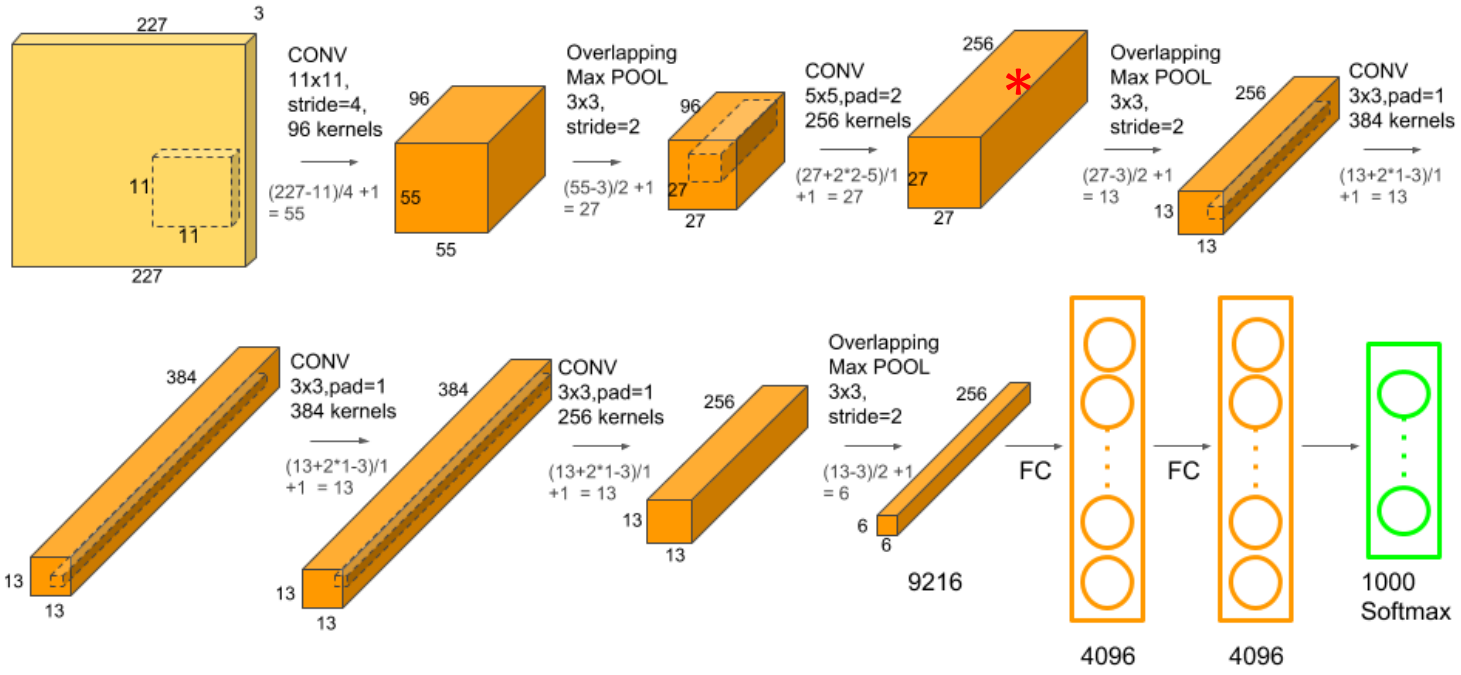


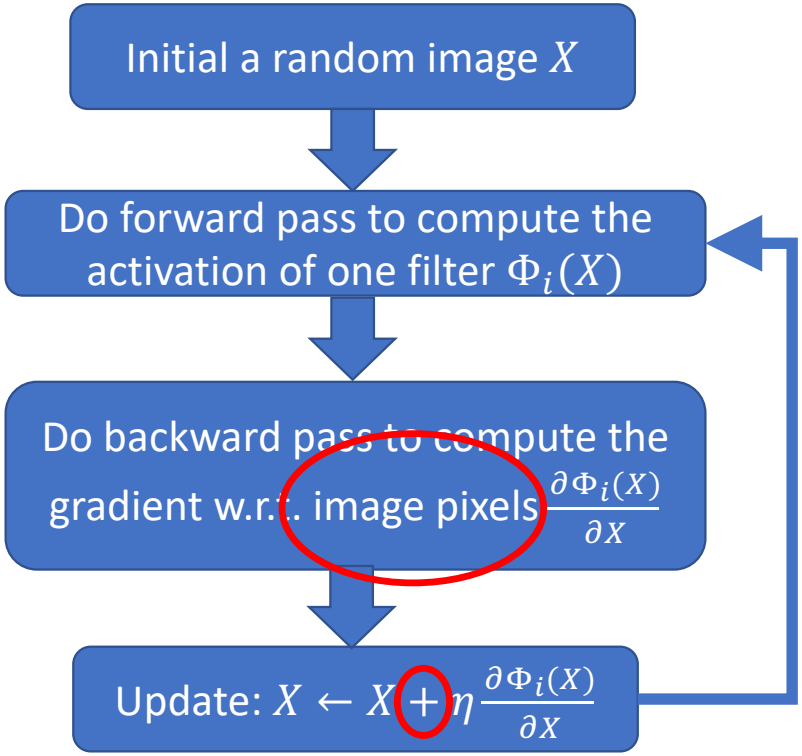
Image from: https://learnopencv.com/understanding-alexnet/

Synthesize an image to maximize the activation

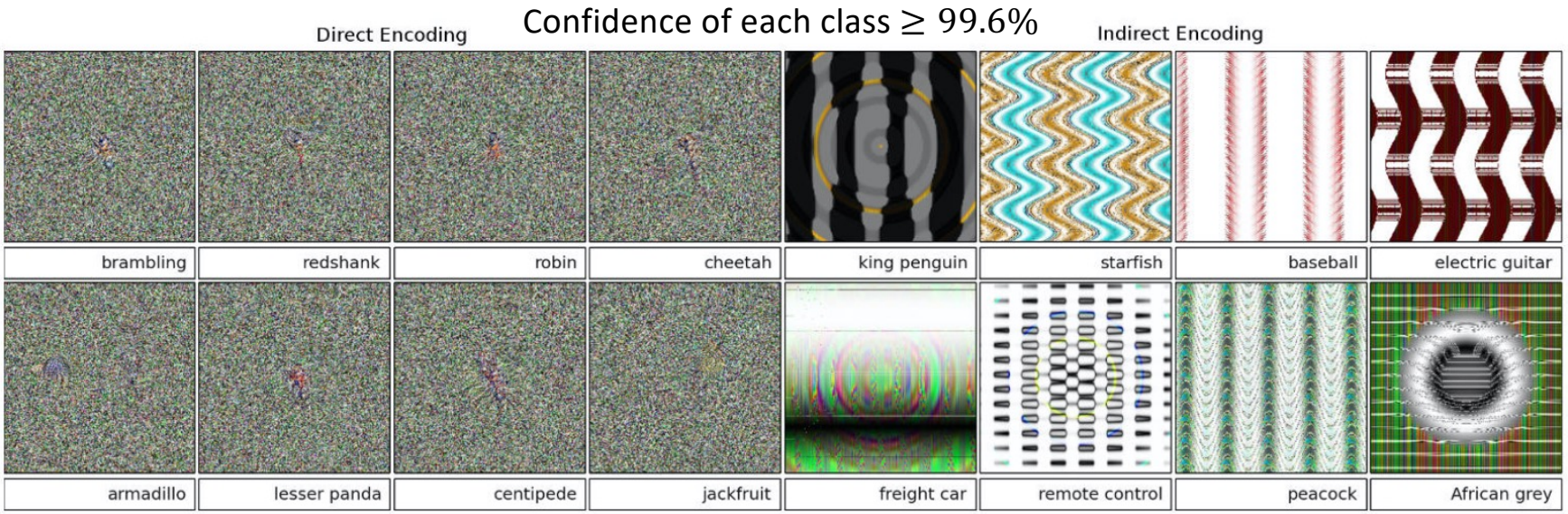
$$X^* = \underset{X \in \mathbb{R}^{u \times v \times 3}}{\operatorname{argmax}} \Phi_i(X)$$

Using **gradient ascent** to generate an image to maximize the activation of one filter.

Q: What is the difference from gradient descent of training neural networks?



Synthesize an image to maximize the activation



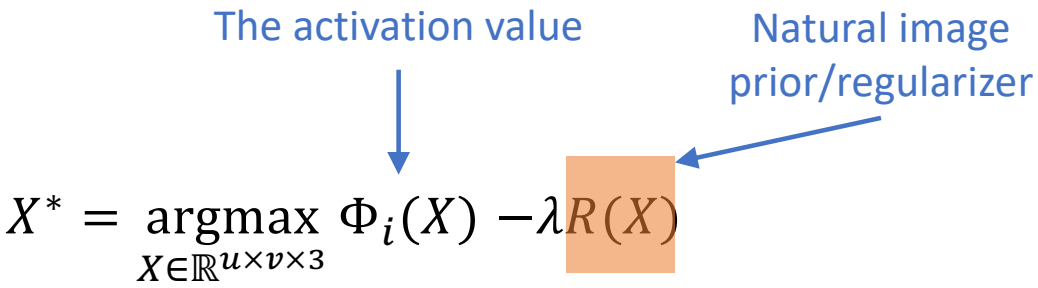
But, these unrecognizable images do not help much for understanding filters.

That's because it is lack of **natural image prior**.

Nguyen, Yosinski, Clune. Deep networks are easily fooled: high confidence predictions for unrecognizable images, CVPR2015

Visualizing filters with regularizations

- To produce more recognizable images, we need



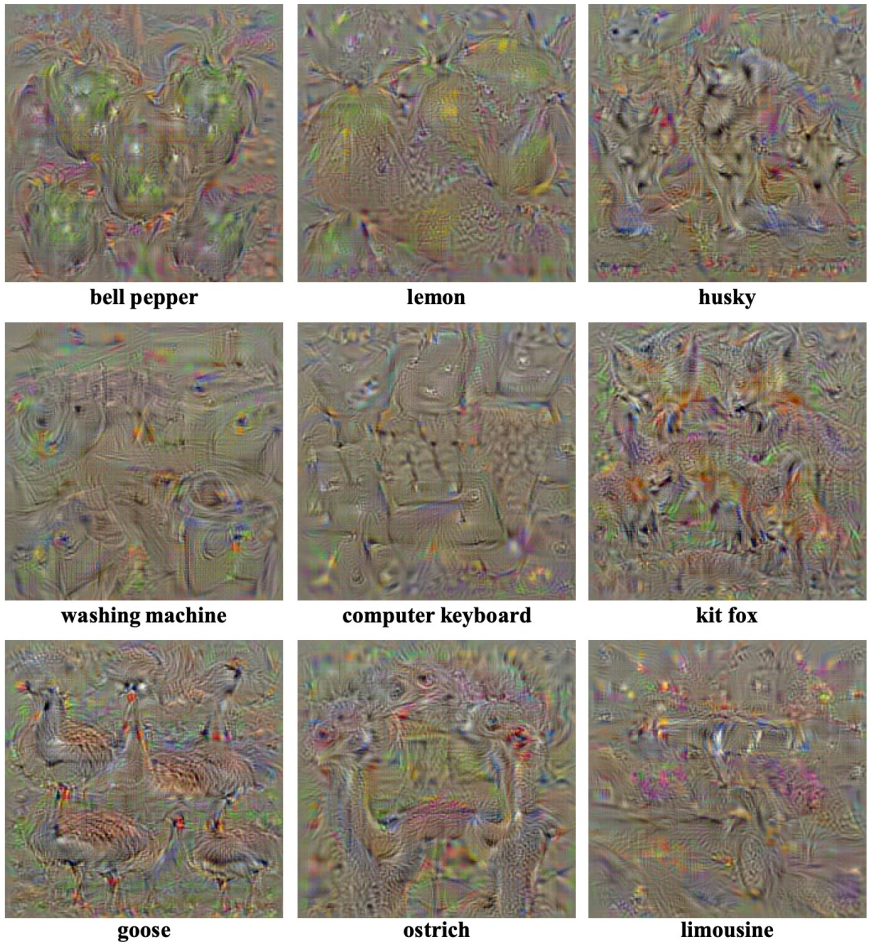
Visualizing filters with regularizations

- L_2-regularizer [Simonyan-2014]

The score of the class c (before softmax)

$$I^* = \underset{I \in \mathbb{R}^{H \times W \times 3}}{\operatorname{argmax}} \; S_c(I) - \lambda \|I\|_2^2$$

L_2-regularizer



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

Visualizing filters with regularizations

- Better regulariser [Yosinski-2015]

The activation value

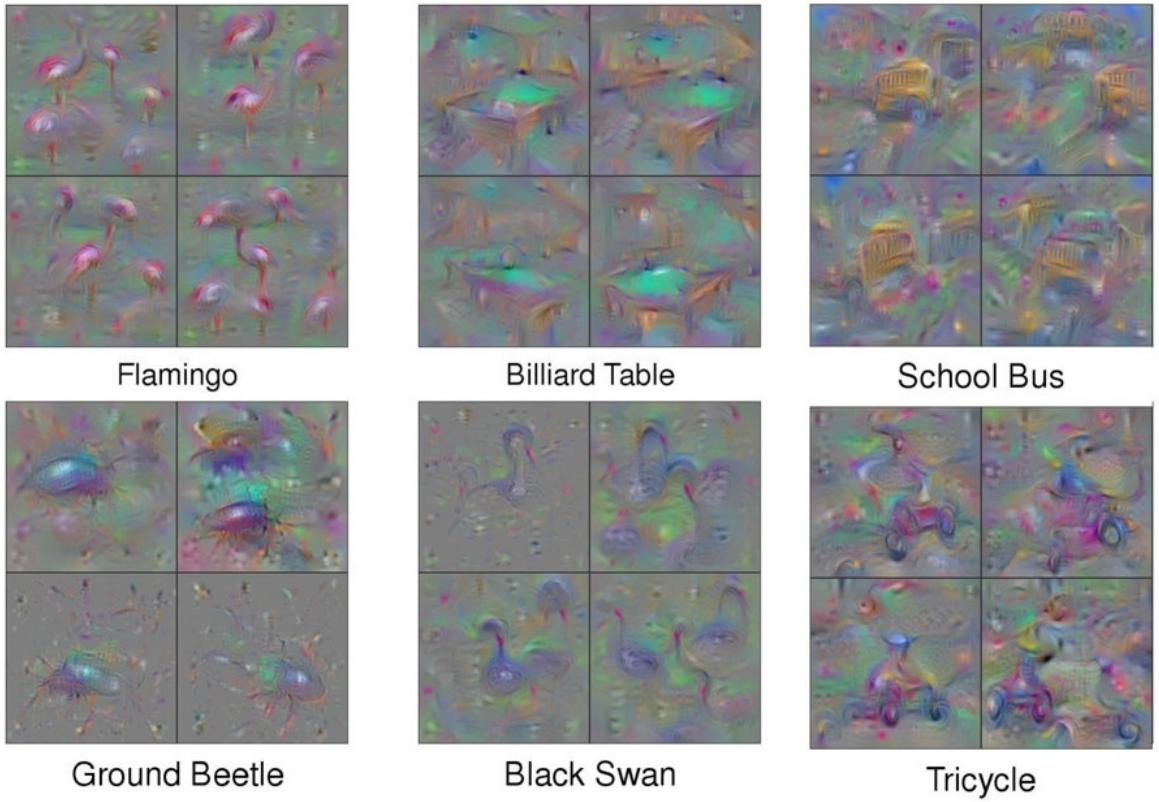
$$X^* = \underset{X \in \mathbb{R}^{u \times v \times 3}}{\operatorname{argmax}} \Phi_i(X) - \lambda \|X\|_2^2$$

L_2-regularizer

Not only penalize L2 norm of the image, also periodically:

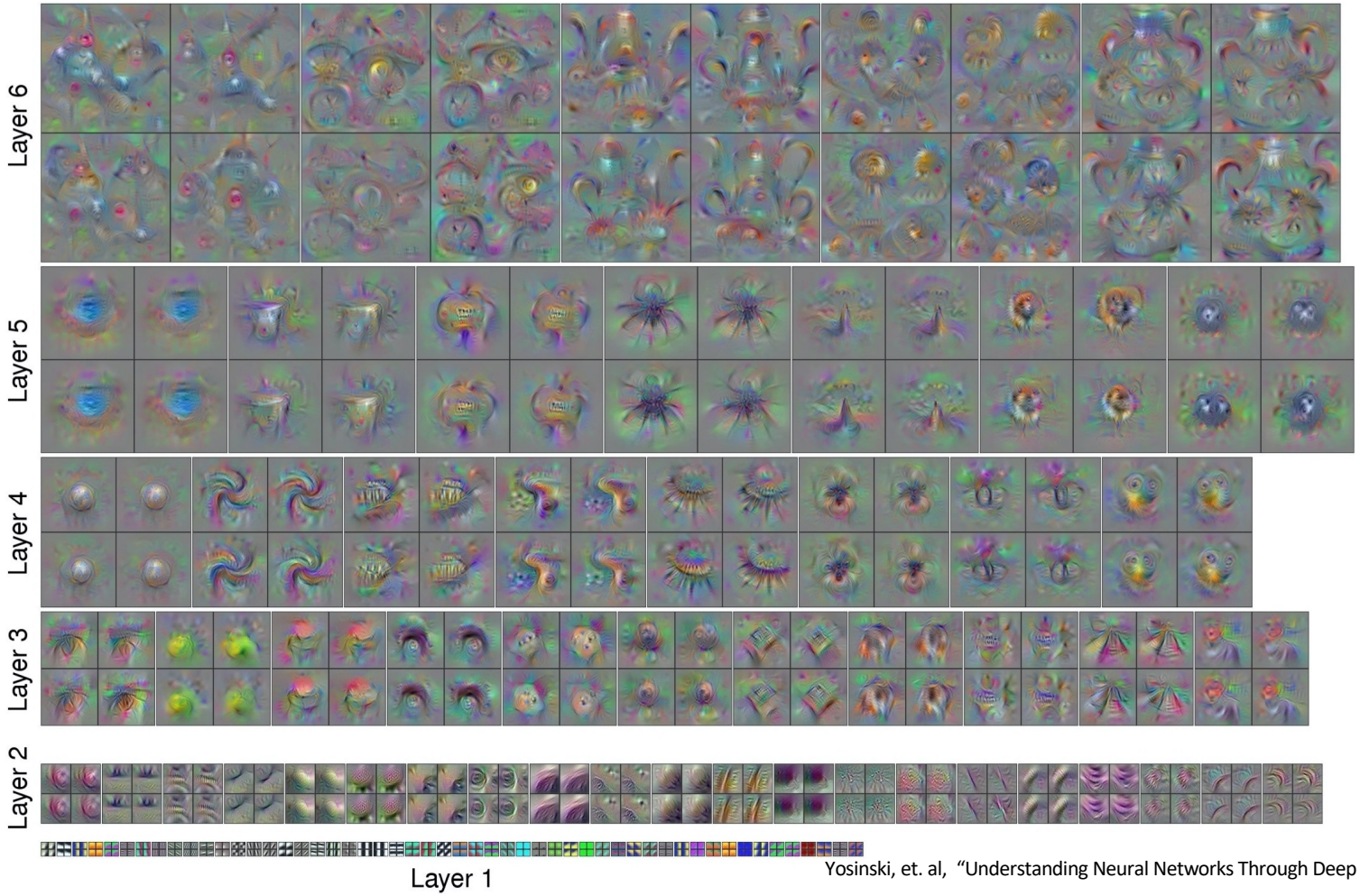
- Gaussian blue image
- Clip pixels with small values to 0
- Clip pixels with small gradients to 0

Layer fc8 of AlexNet



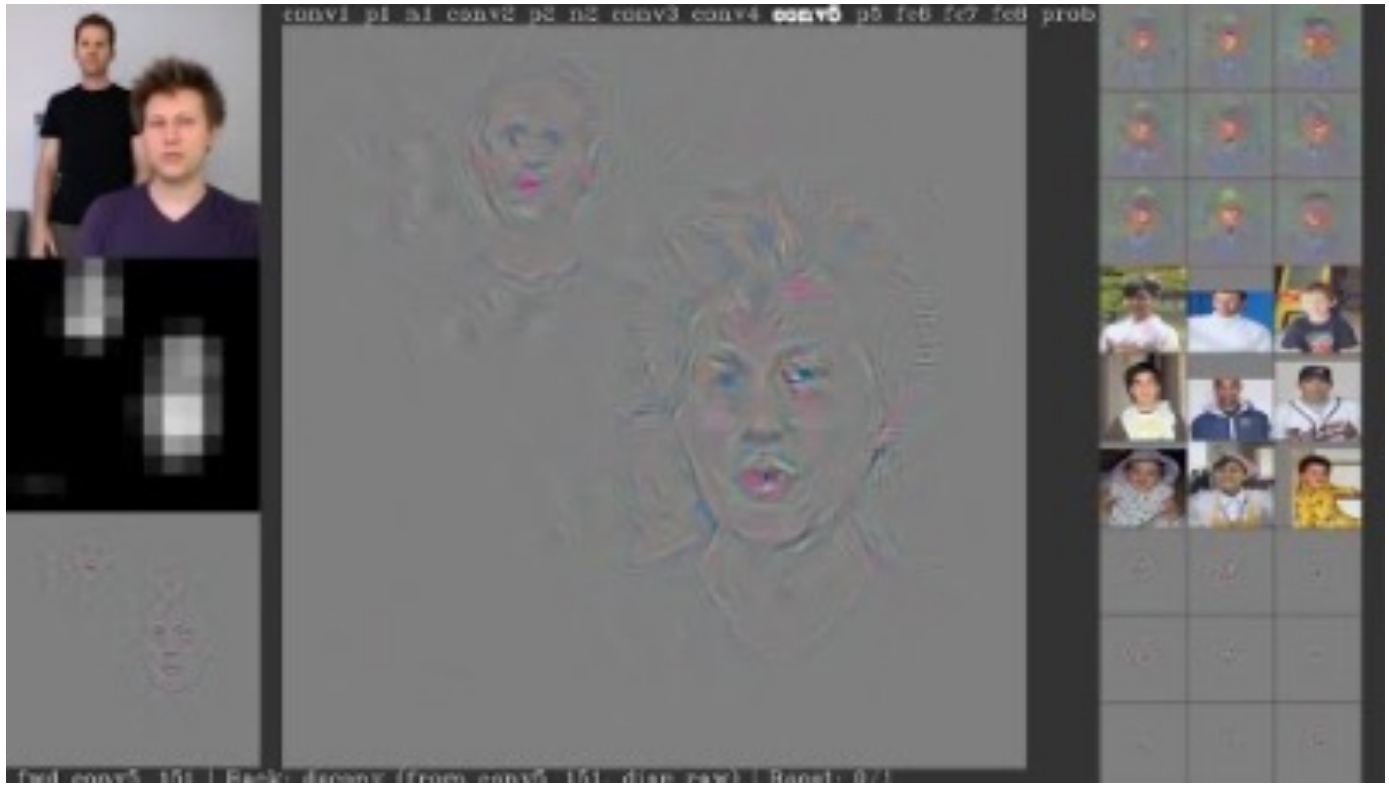
Yosinski, et. al, "Understanding Neural Networks Through Deep Visualization ", ICLR Workshop 2015.

Visualizing filters with regularizations



Yosinski, et. al, "Understanding Neural Networks Through Deep Visualization ", ICLR Workshop 2015.

Deep Visualization Toolbox



Video from: https://yosinski.com/deepvis

Summary

Find patches in the dataset

$$X^* = \underset{X \in \mathcal{D}}{\operatorname{argmax}} \Phi_i(X)$$

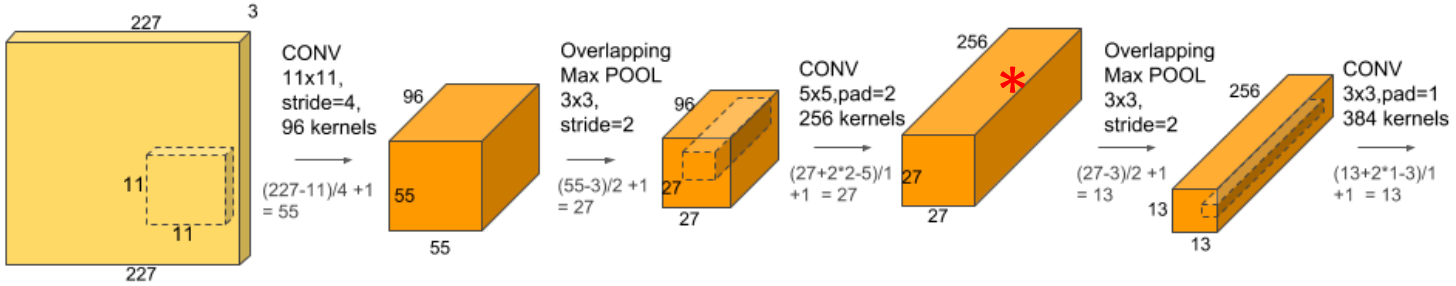
Synthesize patches

$$X^* = \underset{X \in \mathbb{R}^{u \times v \times 3}}{\operatorname{argmax}} \Phi_i(X)$$

Method: Gradient accent
Trick: Adding regularizer

$$X^* = \underset{X}{\operatorname{argmax}} \Phi_i(X)$$

$\Phi_i(X)$



Reference

- Online Lecture: “Visualizing what ConvNets learn”. https://cs231n.github.io/understanding-cnn/
- Video lecture in University of Stanford. “Visualizing and Understanding”. 2017. https://www.youtube.com/watch?v=6wcs6szJWMY
- Blog by Jason Yosinski. “Understanding Neural Networks Through Deep Visualization”. 2015. https://yosinski.com/deepvis

Topic 3: Neural Style Transfer

The activation value

Natural image regularizer/prior

$$X^* = \underset{X \in \mathbb{R}^{u \times v \times 3}}{\operatorname{argmax}} \Phi_i(X) - \lambda R(X)$$

Visualizing Filters

Visualize what sort of input maximizes the activation of each filter

$$X^* = \underset{X \in \mathbb{R}^{H \times W \times 3}}{\operatorname{argmin}} \ell\big(\Phi(X), \Phi(X_0)\big) + \lambda R(X)$$

Loss (matches the feature)

$\Phi(X)$: $\mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^d$ is an image representation of a certain layer (a feature vector)

Natural image regularizer/prior

Feature Inversion

Reconstruct an image whose feature representation best matches the original image

Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015

Feature Inversion

- Reconstruct an image whose feature representation best matches the original image's.

$$X^* = \underset{X \in \mathbb{R}^{H \times W \times 3}}{\operatorname{argmin}} \ \ell\big(\Phi(X), \Phi(X_0)\big) + \lambda R(X)$$

$$\ell\big(\Phi(X), \Phi(X_0)\big) = \| \Phi(X) - \Phi(X_0) \|^2$$

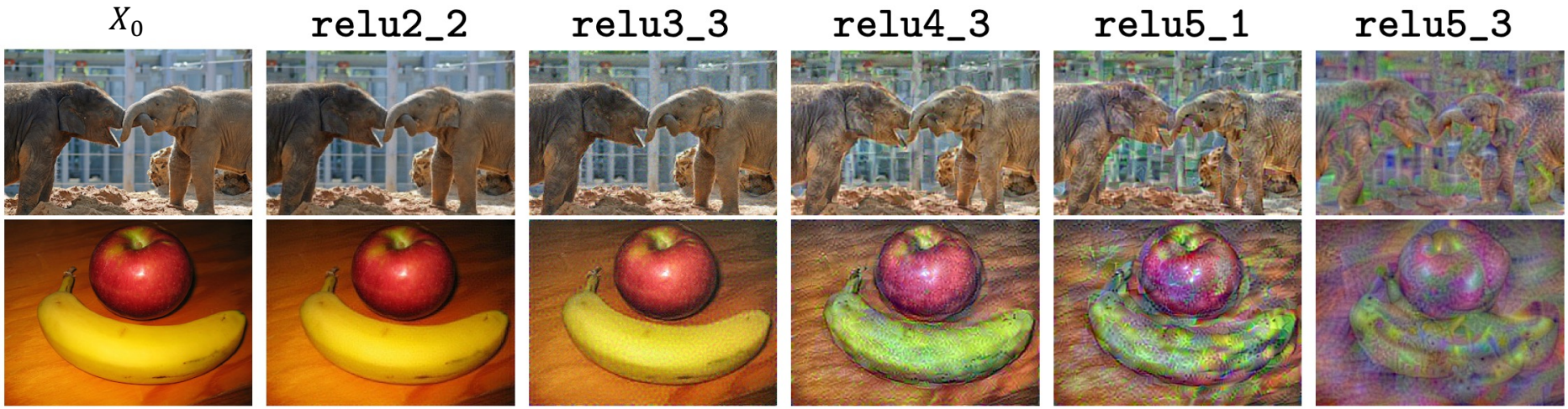
Feature distance

$$R_{V^\beta} = \sum_{i,j} \left(\left(x_{i,j+1} - x_{ij}\right)^2 + \left(x_{i+1,j} - x_{ij}\right)^2 \right)^{\frac{\beta}{2}}$$

Total Variation regularizer
(encourages spatial smoothness)

Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015

Feature Inversion



As we reconstruct from higher layers, image content and overall spatial structure are preserved, but color, texture, and exact shape are not.

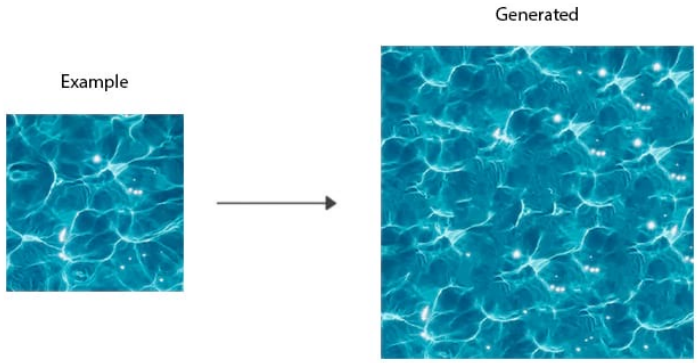
Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015
Image from: Perceptual Losses for Real-Time Style Transfer and Super-Resolution. CVPR 2016

Neural Texture Synthesis

$$X^* = \underset{X \in \mathbb{R}^{H \times W \times 3}}{\operatorname{argmin}} \ell(\Phi(X), \Phi(X_0))$$

$$X^* = \underset{X \in \mathbb{R}^{H \times W \times 3}}{\operatorname{argmin}} \ell(G(X), G(X_0))$$

Texture representation using CNN



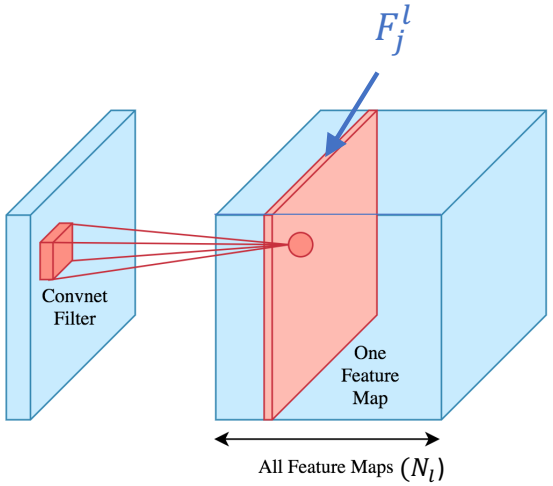
Neural Texture Synthesis

Image from: https://lib.rs/crates/texture-synthesis

Gram Matrix

F_j^l: j-th feature map at layer l

F_i^l: i-th feature map at layer l



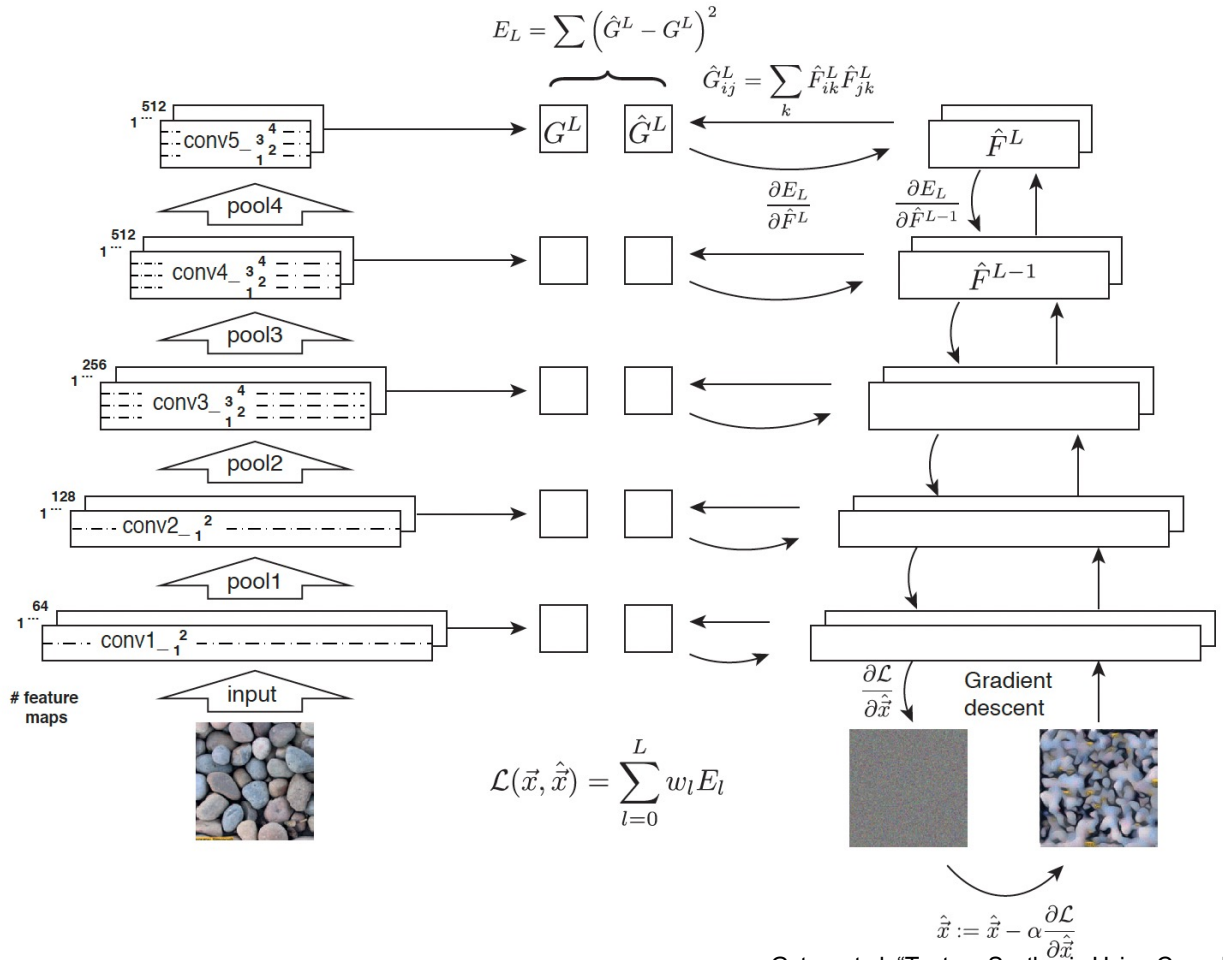
$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

the inner product between two feature maps.

Gram matrix: $G^l \in \mathbb{R}^{N_l \times N_l}$

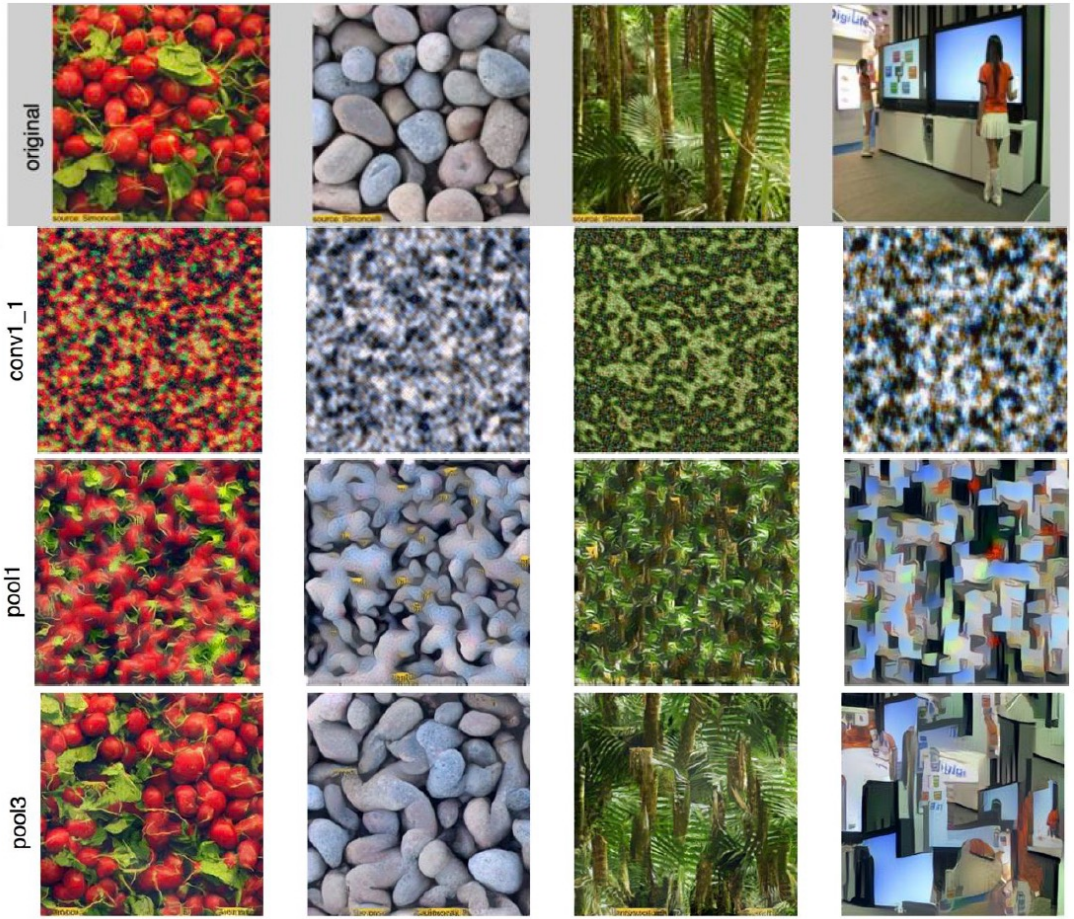
Texture features could be represented by a set of Gram Matrixes of conv layers: $\{G^1, G^2, \ldots, G^L\}$

Neural Texture Synthesis



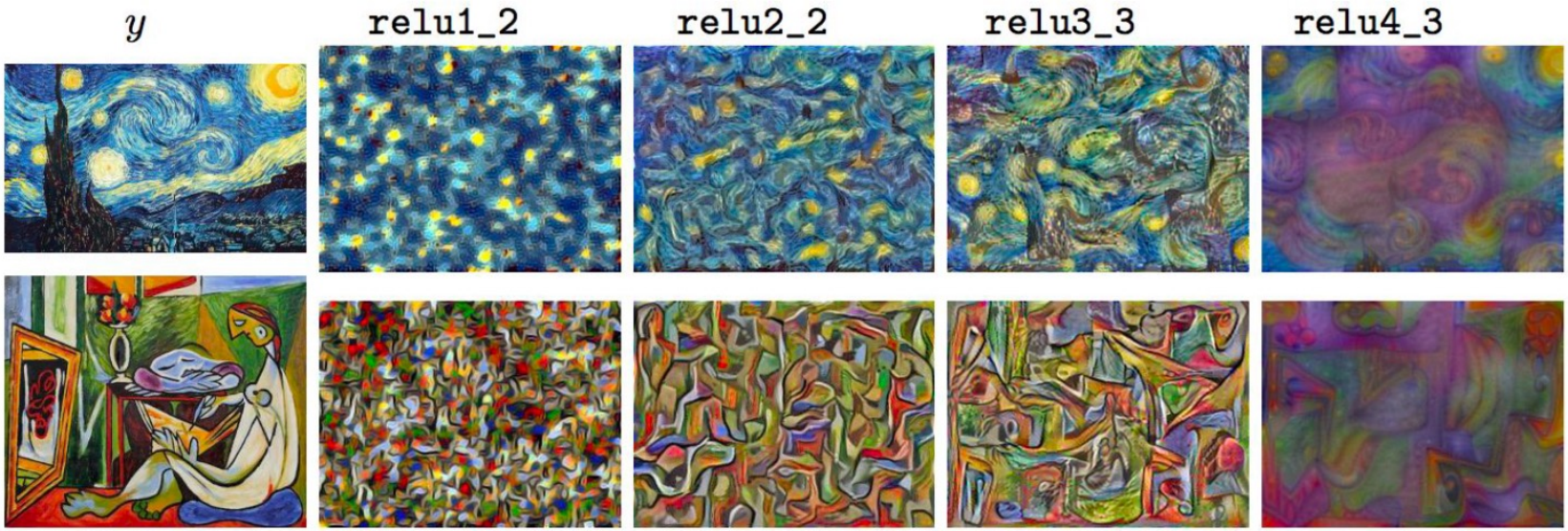
Gatys, et al. “Texture Synthesis Using Convolutional Neural Networks”. NIPS2015.

Neural Texture Synthesis



Gatys, et al. “Texture Synthesis Using Convolutional Neural Networks”. NIPS2015.

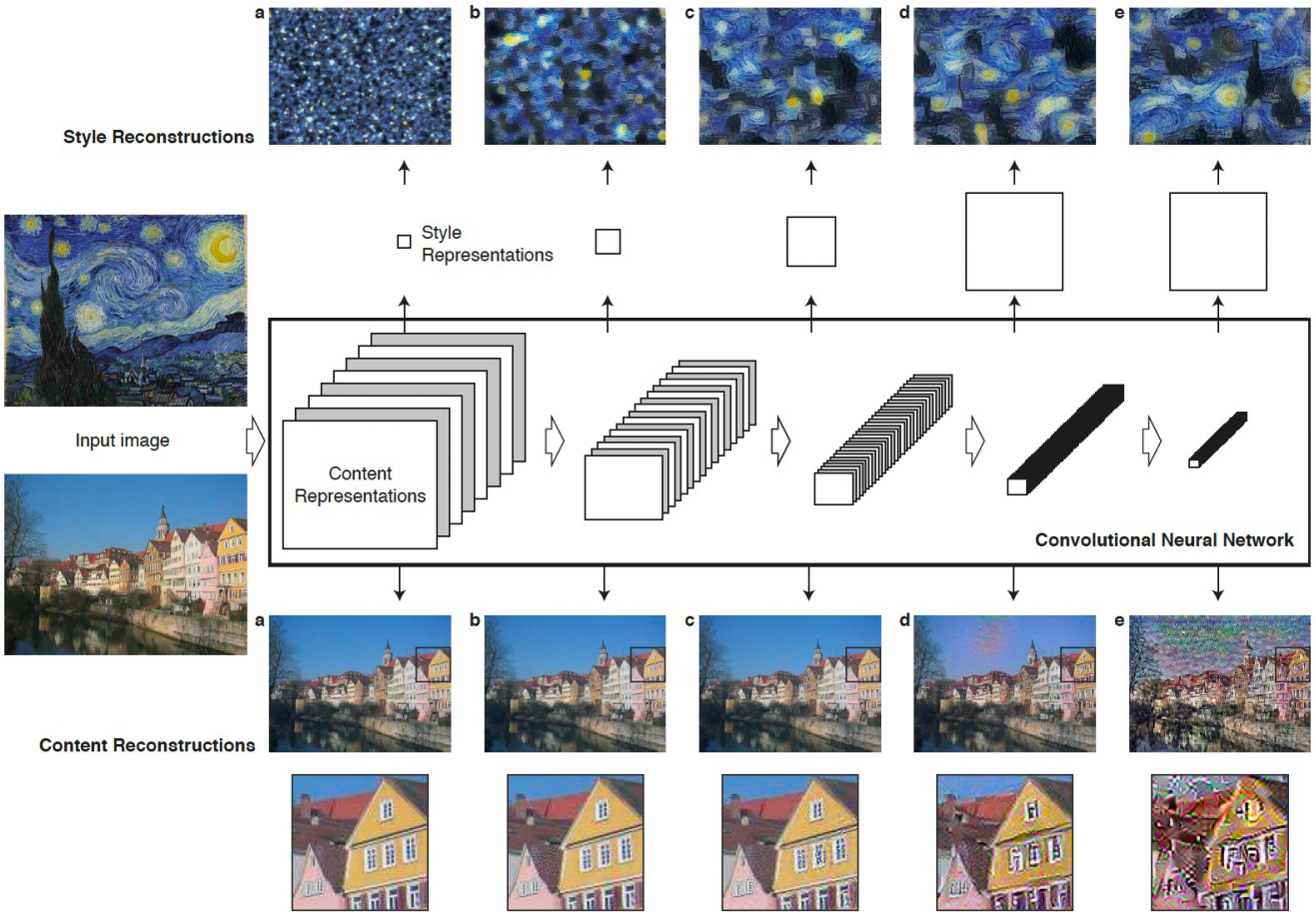
When texture synthesis applies to art …



Q: Can we transfer a particular art style to a new image?

Gatys, et al. “Image style transfer using convolutional neural networks”, CVPR 2016.
Gatys, et al. “Texture Synthesis Using Convolutional Neural Networks”. NIPS2015.

Neural Style Transfer



$$X^* = \underset{X \in \mathbb{R}^{H \times W \times 3}}{\operatorname{argmin}} \ell(G(X), G(X_0))$$

Texture synthesis

Reconstruct an image whose texture representation best matches the original image

$$X^* = \underset{X \in \mathbb{R}^{H \times W \times 3}}{\operatorname{argmin}} \ell(\Phi(X), \Phi(X_0))$$

Feature Inversion

Reconstruct an image whose feature representation best matches the original image

Neural Style Transfer

$\mathcal{L}_{\text{style}}$

A_{ij}^{l} G_{ij}^{l}

Content Image

Style Image

Style Transfer!

P_{ij}^{l} F_{ij}^{l}

$\mathcal{L}_{\text{content}}$

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{style}} + \beta\mathcal{L}_{\text{content}}$$

Match the texture (Gram-matrix):

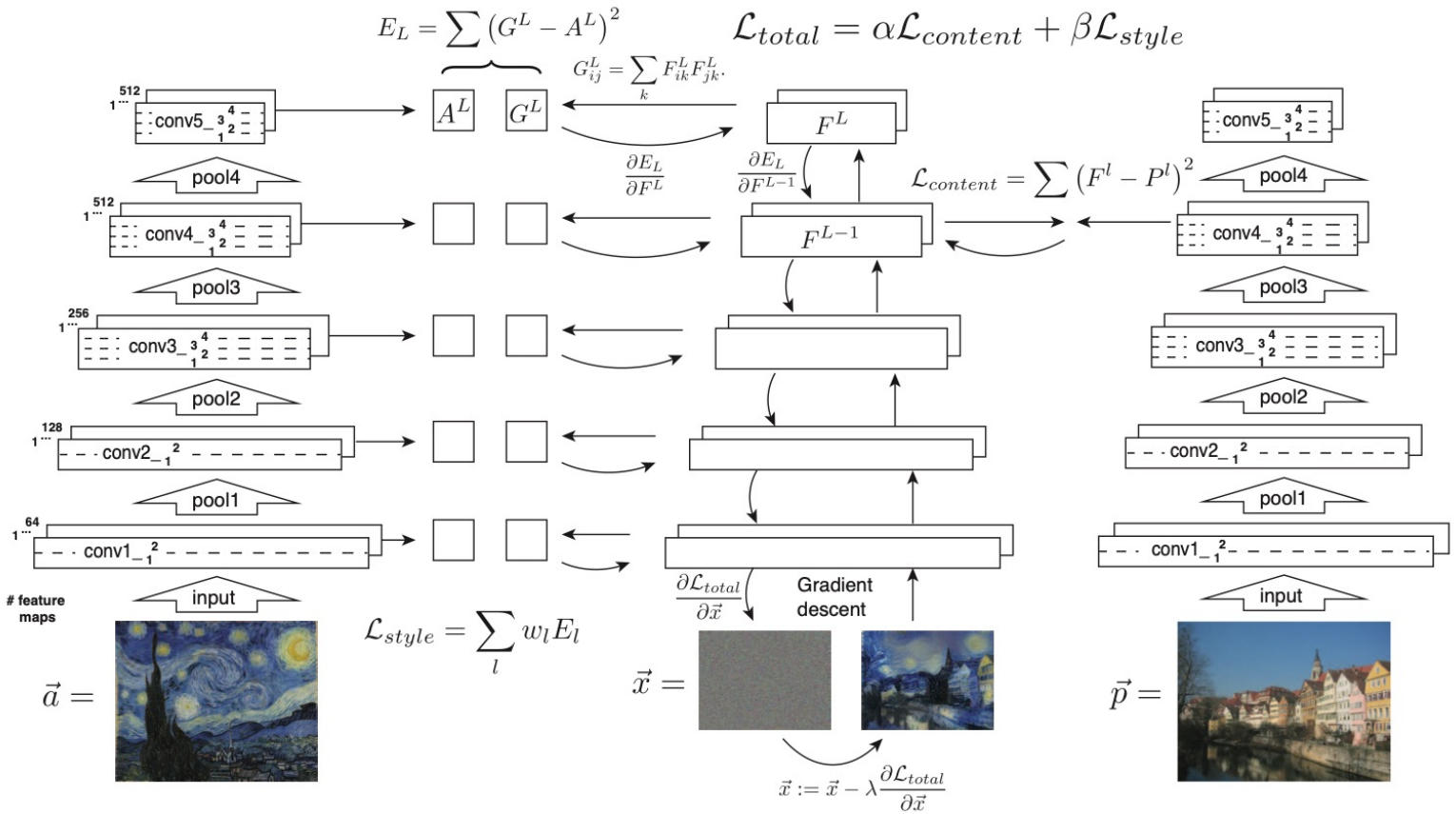
$$\sum_{l=1}^{L}\sum_{ij}(G_{ij}^{l} - A_{ij}^{l})^2$$

Match the feature:

$$\sum_{ij}(F_{ij}^{l} - P_{ij}^{l})^2$$

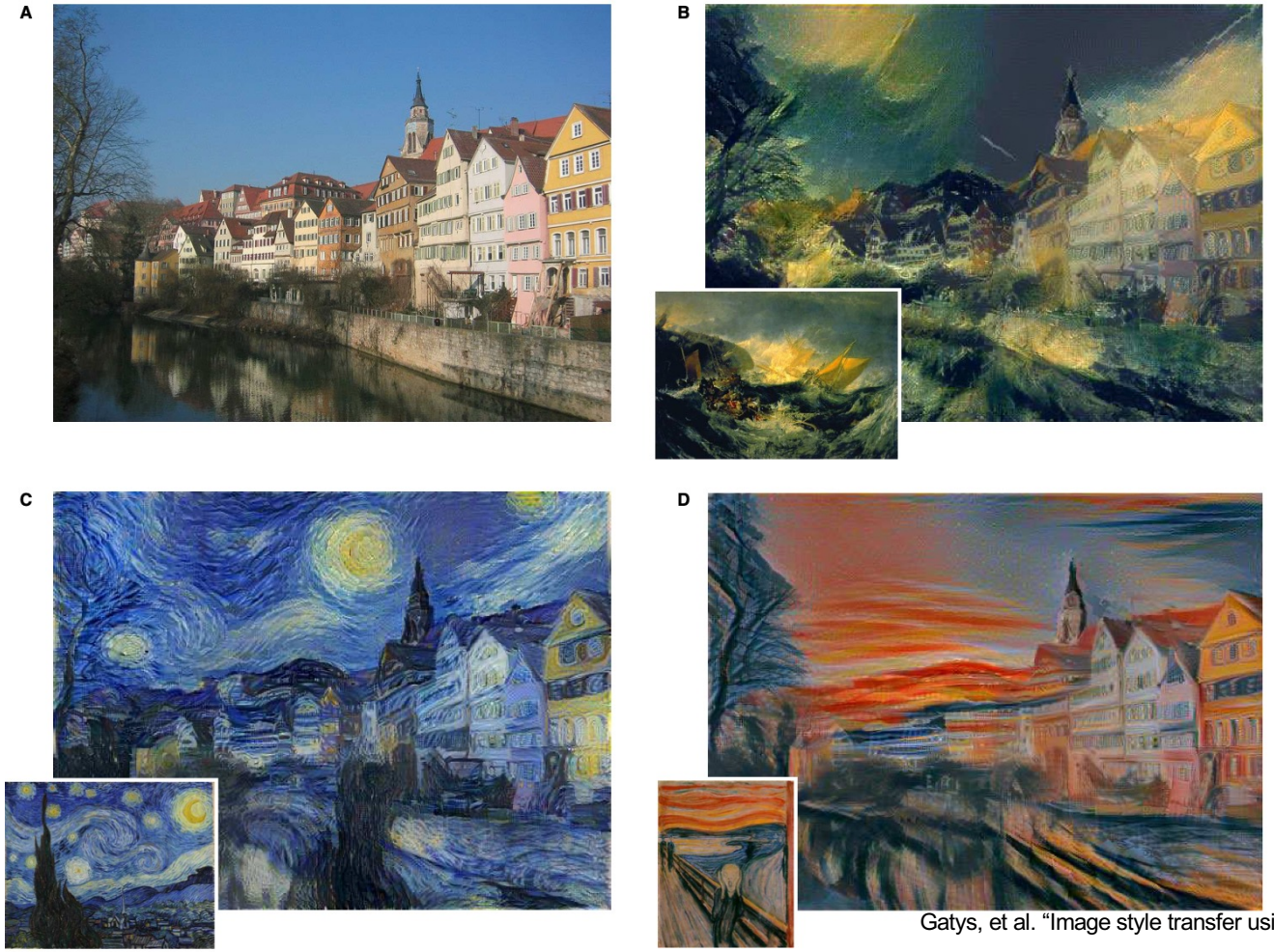
Gatys, et al. “Image style transfer using convolutional neural networks”, CVPR 2016.
Gatys, et al. “Texture Synthesis Using Convolutional Neural Networks”. NIPS2015.

Neural Style Transfer



Gatys, et al. “Image style transfer using convolutional neural networks”, CVPR 2016.

Neural Style Transfer



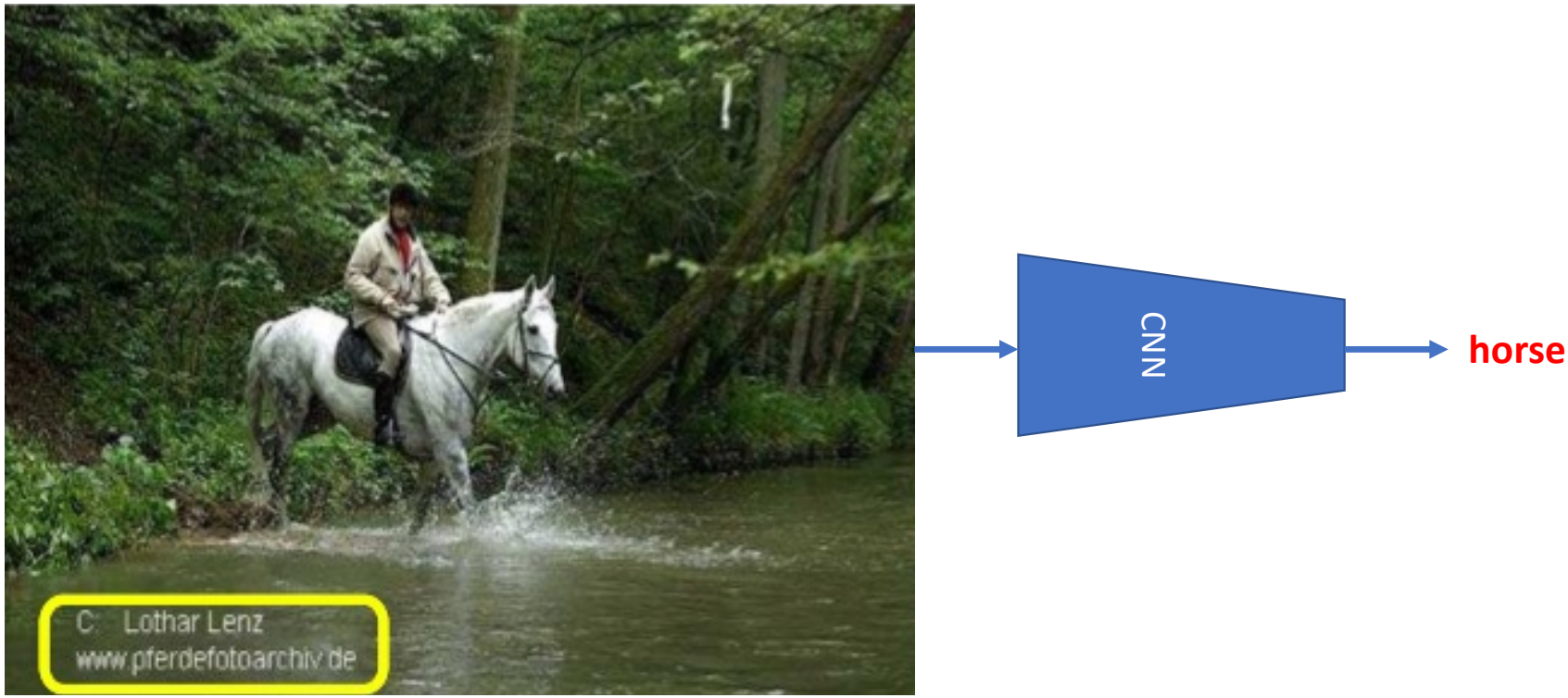
Gatys, et al. “Image style transfer using convolutional neural networks”, CVPR 2016.

References

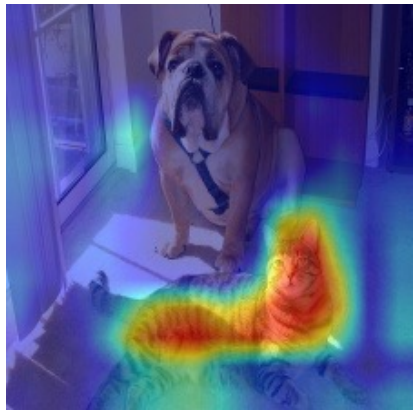
- Video lecture in University of Stanford. “Visualizing and Understanding”. 2017. https://www.youtube.com/watch?v=6wcs6szJWMY

Topic 4: Visualizing Class-Discriminative Heatmap

Q: When a CNN model outputs a class for an image, which parts of the image contribute high to the decision?



Visualizing heat maps



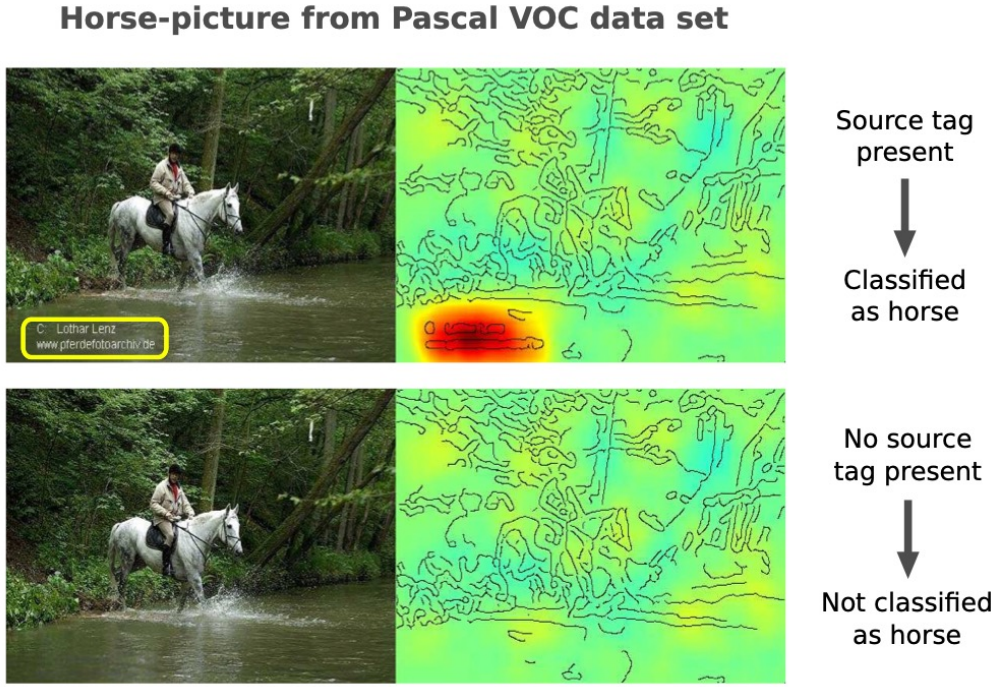
CAM for the cat class

- **Class Activation Mapping (CAM)** is to produce heat maps to highlight class specific region of images, to indicate how important each location is with respect to the given class.

Image from: https://glassboxmedicine.com/2019/06/11/cnn-heat-maps-class-activation-mapping-cam/

Why does this matter?

- Help investigate the traits or features of the input images which had major contribution in producing a certain class
- Identify the weak-points of our models
- Identify dataset bias



Visualizing heat maps

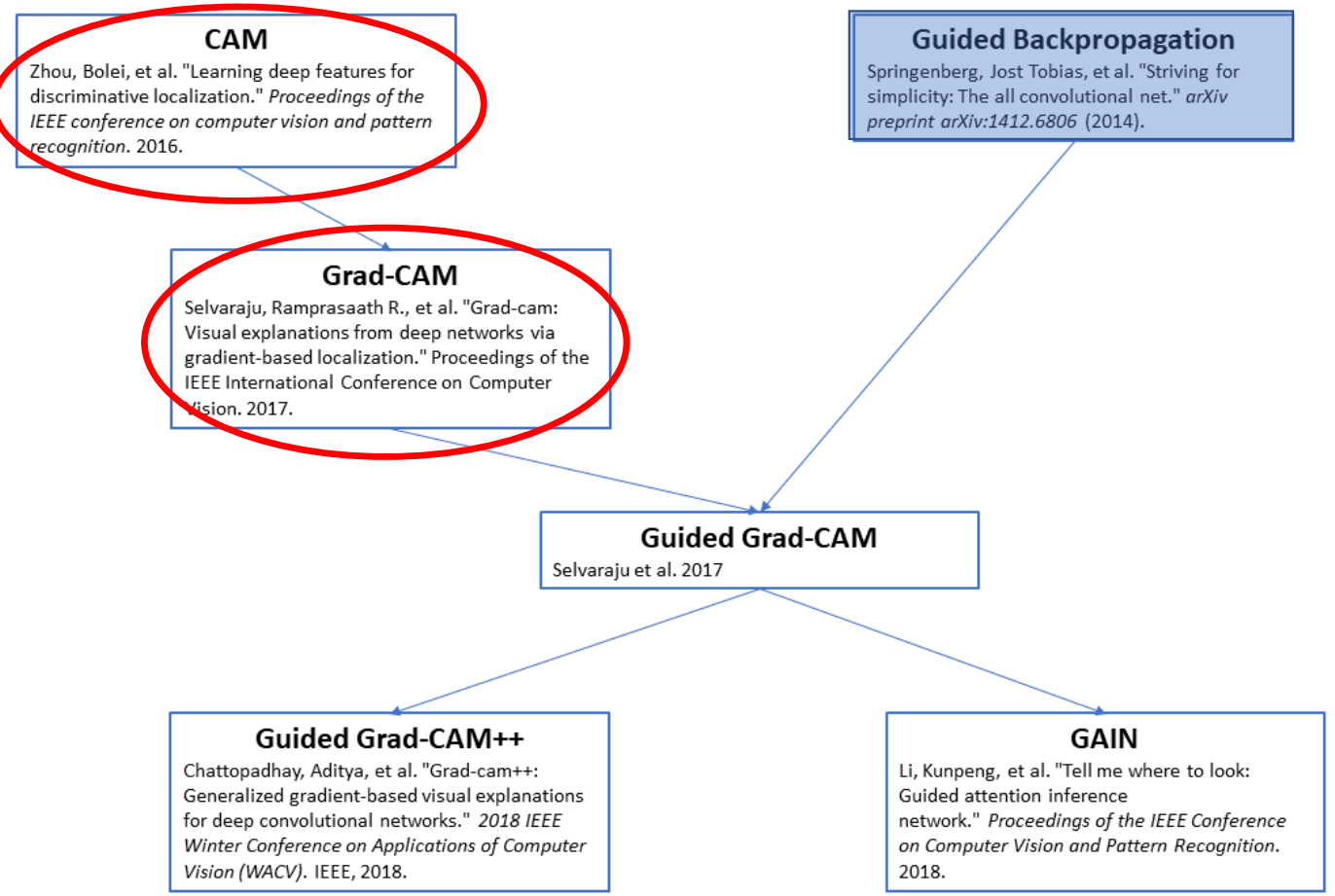
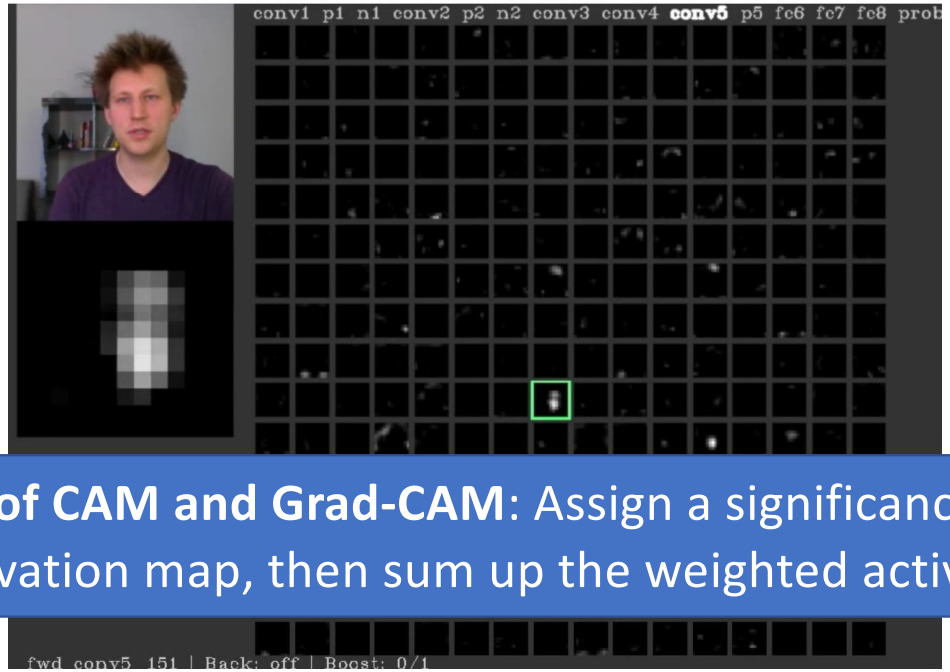
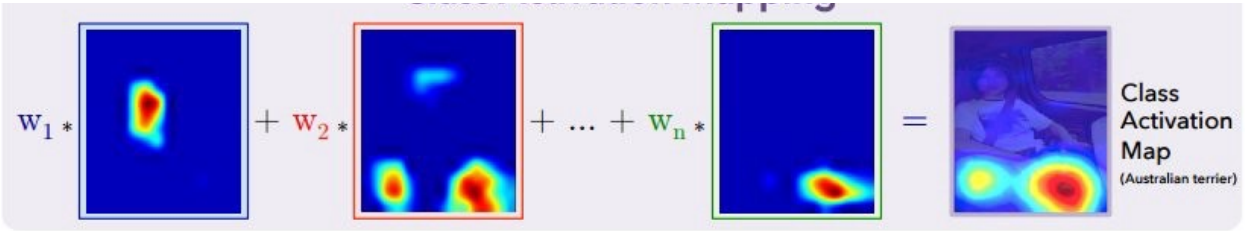


Image from: https://glassboxmedicine.com/2019/06/11/cnn-heat-maps-class-activation-mapping-cam/



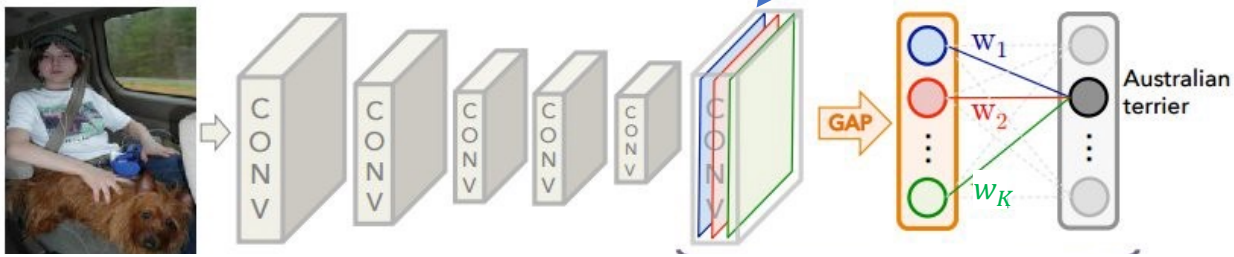
Main principle of CAM and Grad-CAM: Assign a significance value (weight) to each activation map, then sum up the weighted activation maps.



Q: How to calculate the weights?

CAM [Zhou-CVPR16]

K feature maps:
$A^k \in \mathbb{R}^{u \times v}, 1 \leq k \leq K$



GAP (Global Average Pooling):

$$\alpha_k = \frac{1}{uv} \sum_{i=1}^{u} \sum_{j=1}^{v} A_{i,j}^k$$

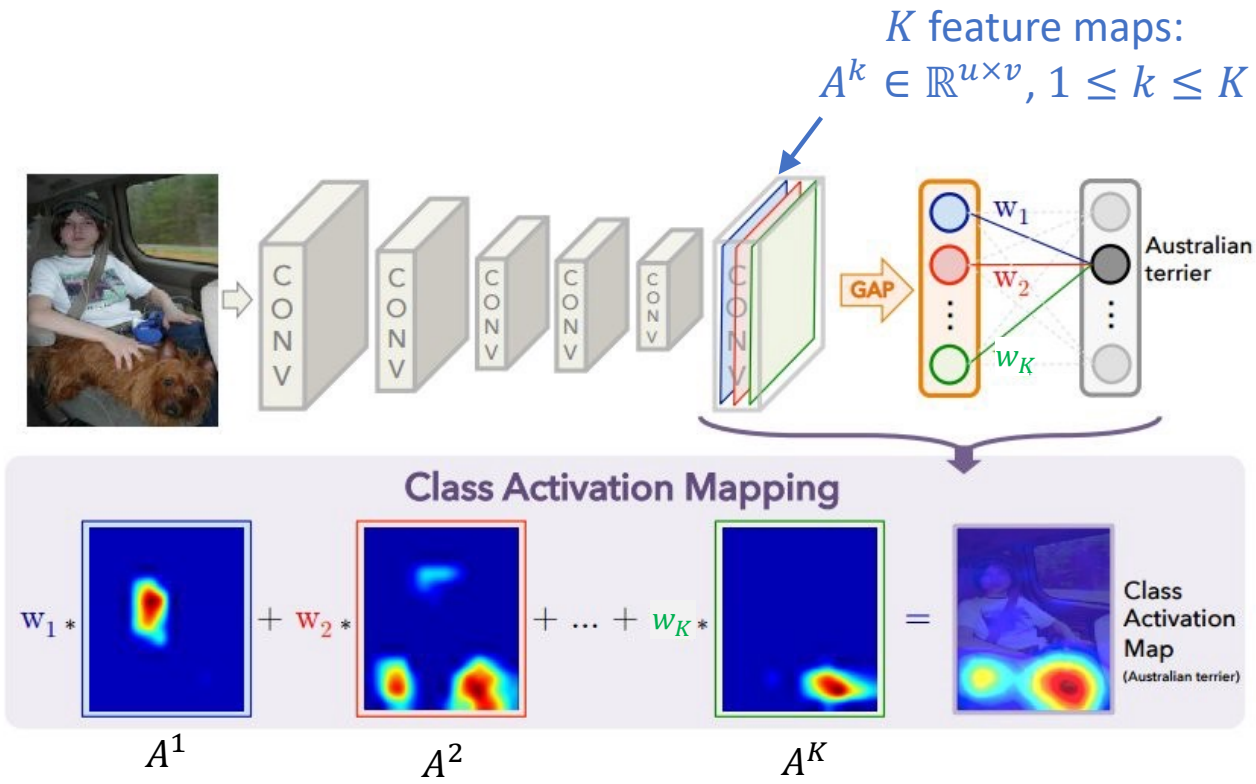
Class score for c–th class:

$$\hat{y}^c = \sum_{k=1}^{K} w_k^c \alpha_k$$

w_k^c is learnt by backpropagation

Zhou et. Al. “Learning deep features for discriminative localization” CVPR 2016.

CAM [Zhou-CVPR16]



GAP (Global Average Pooling):

$$\alpha_k = \frac{1}{uv} \sum_{i=1}^{u} \sum_{j=1}^{v} A_{i,j}^k$$

Class score for c–th class:

$$\hat{y}^c = \sum_{k=1}^{K} w_k^c \alpha_k$$

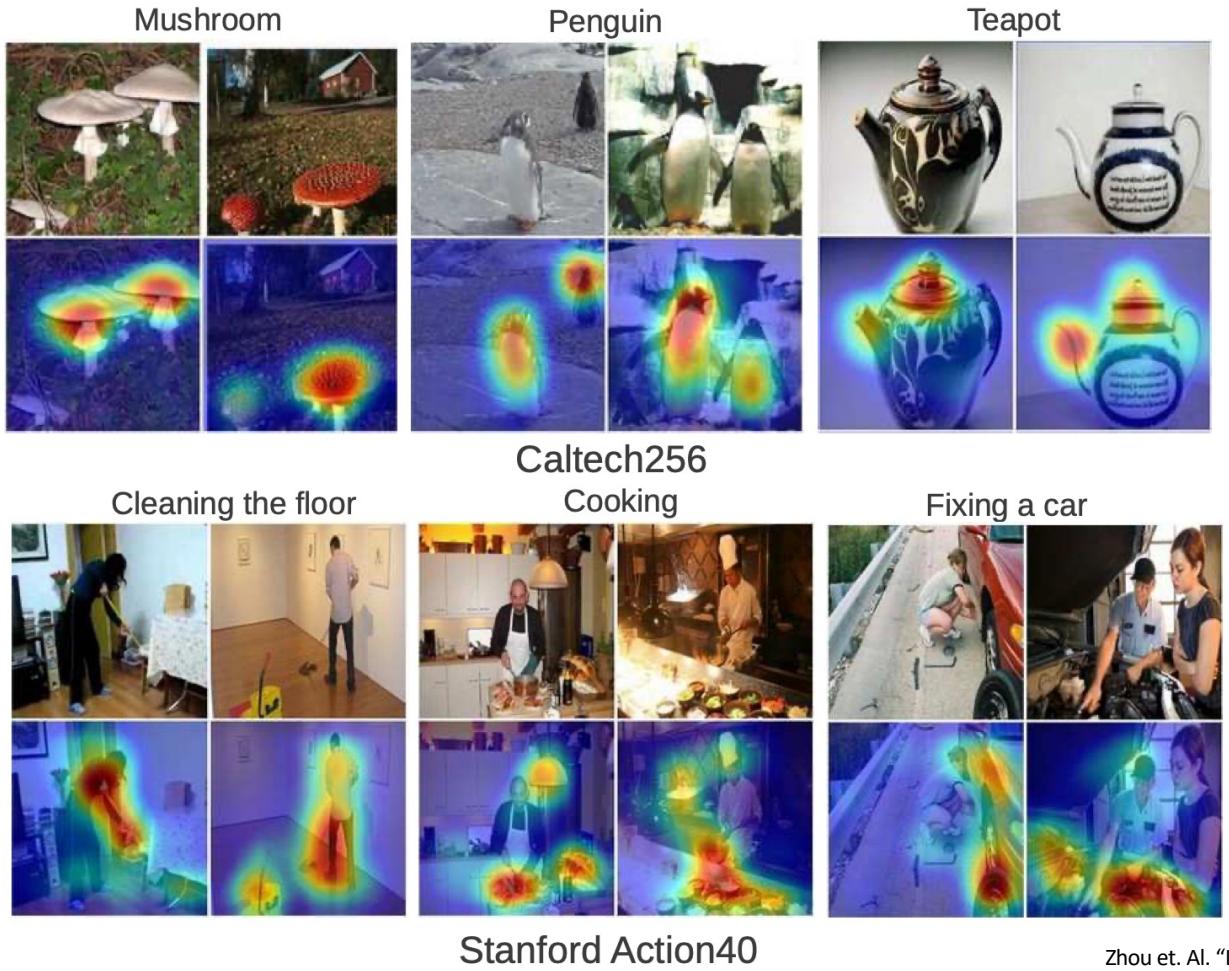
w_k^c is learnt by backpropagation

Class Activation Map for c–th class:

$$L_{\text{CAM}}^c = \sum_{k=1}^{K} w_k^c A^k$$

Zhou et. Al. "Learning deep features for discriminative localization" CVPR 2016.

CAM [Zhou-CVPR16]



Zhou et. Al. “Learning deep features for discriminative localization” CVPR 2016.

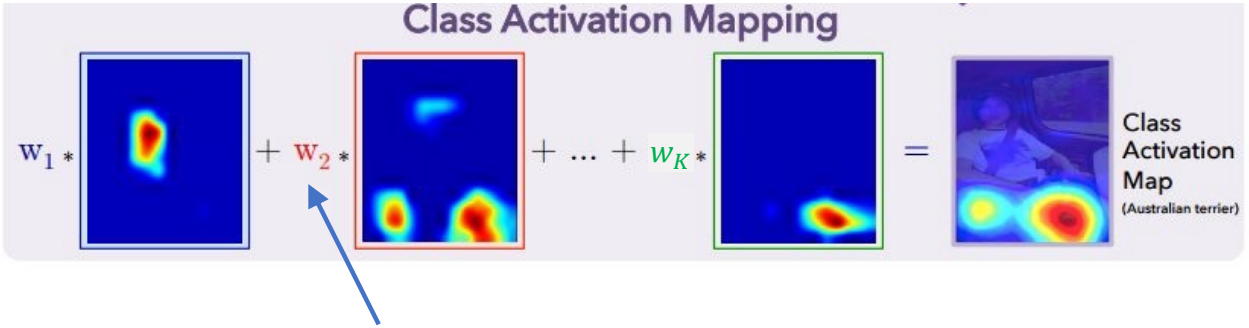
Limitations of CAM

- Need to have a GAP layer in the model architecture
  - Only applicable to this particular kind of CNN architecture (conv layers → GAP → FC layer (with Softmax)), leading to inferior accuracies.
- Can only visualize the final layer heatmap

Solution: A generalization of CAM: **Grad−CAM**
Compatible with any kind of CNN architecture

Grad-CAM [Selvaraju-ICCV17]

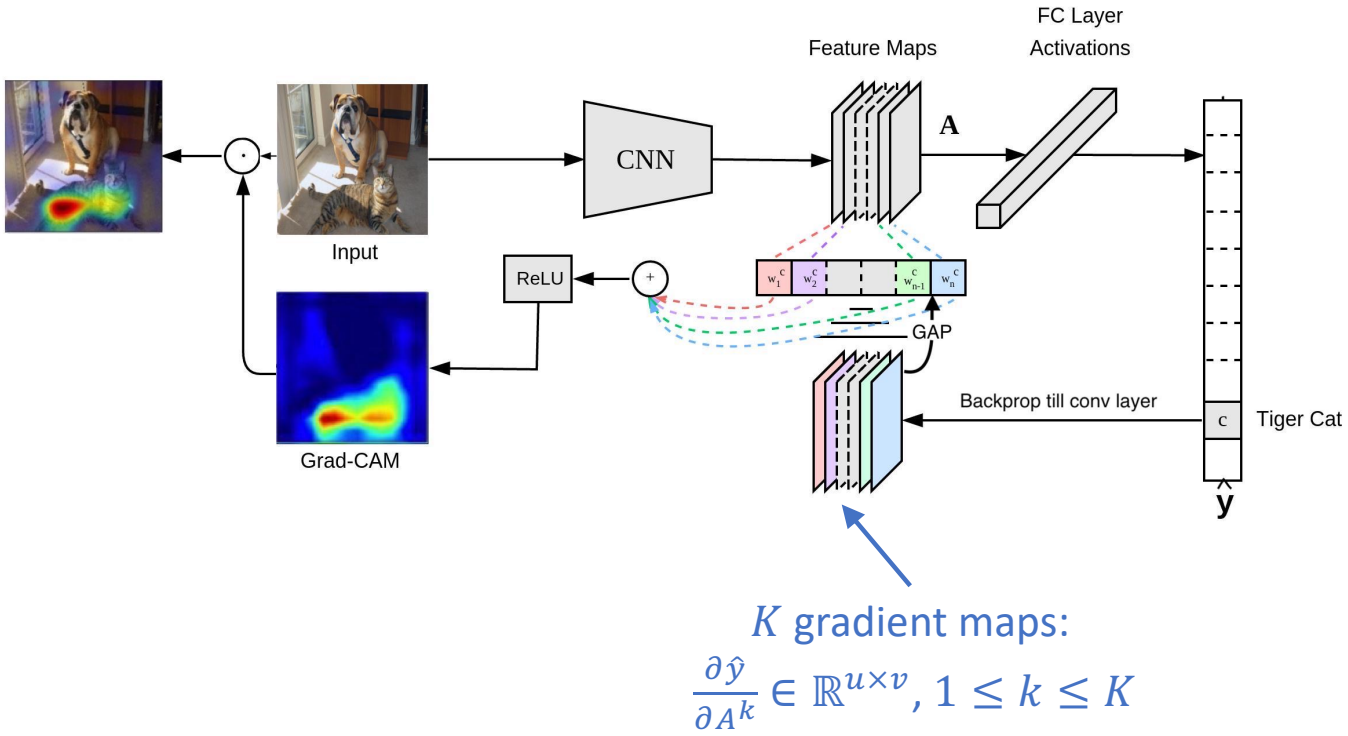
- **Intuition**: Gradients indicates its importance for making a decision by a model.



Q: How to calculate w_k?

A: Using average of gradients w.r.t. a certain feature map.

Grad-CAM [Selvaraju-ICCV17]



K gradient maps:

$$\frac{\partial \hat{y}}{\partial A^k} \in \mathbb{R}^{u \times v}, 1 \leq k \leq K$$

Gradient via backpropagation

$$\frac{\partial \hat{y}^c}{\partial A^k_{i,j}}$$

GAP (Global Average Pooling):

$$w_k^c = \frac{1}{uv} \sum_{i=1}^{u} \sum_{j=1}^{v} \frac{\partial \hat{y}^c}{\partial A^k_{i,j}}$$

Grad-CAM for c–th class:

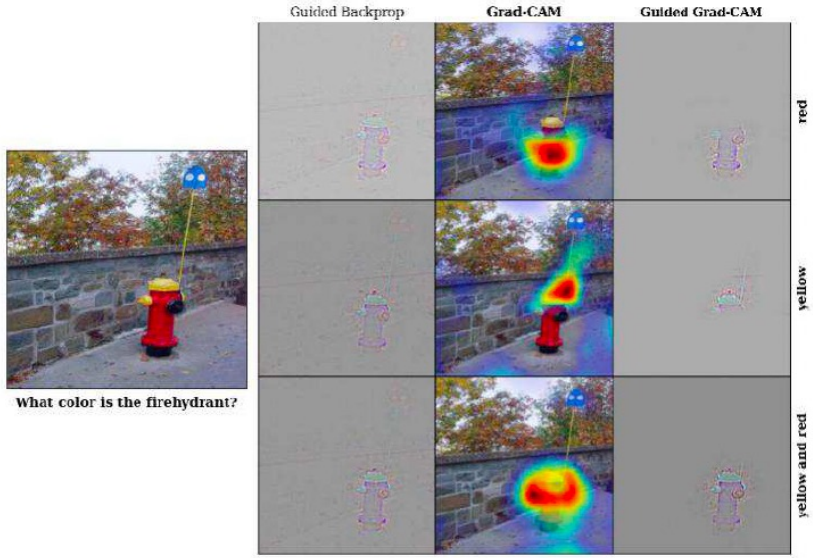
$$L^c_{\text{Grad-CAM}} = ReLU\left(\sum_{k=1}^{K} w_k^c A^k\right)$$

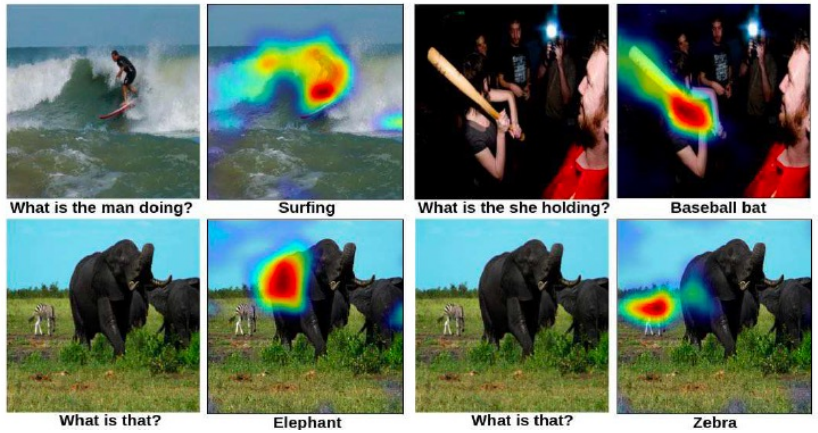
Beyond classification

- Grad-CAM can be also used for other CNN-based models.

Image captioning

VQA (Visual Question Answering)





More on Grad-CAM Demo: http://gradcam.cloudcv.org

UNIVERSITY OF BATH

Visualizing Filters

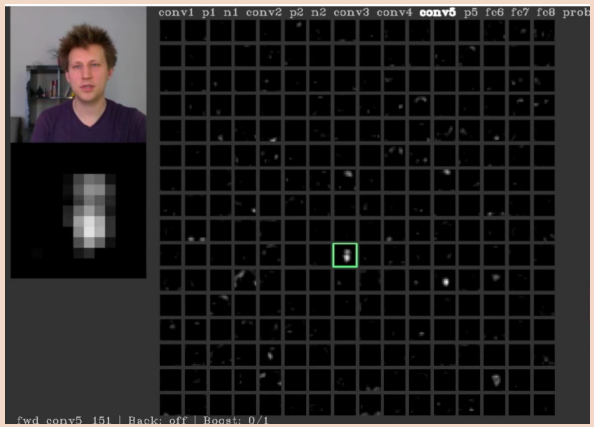
Display Filters directly

Find patches in a dataset

Synthesize patches

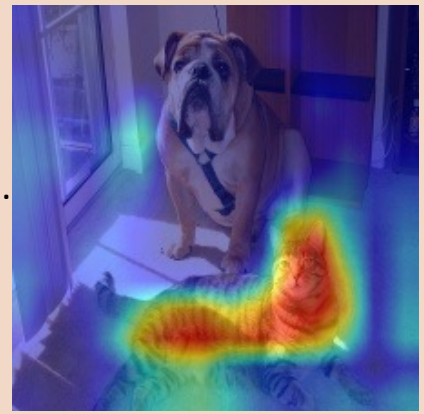
- Feature Inverse
- Texture synthesis
- Neural Style Transfer

Visualizing Activation Maps



Visualizing Heatmaps

CAM, Grad-CAM, …



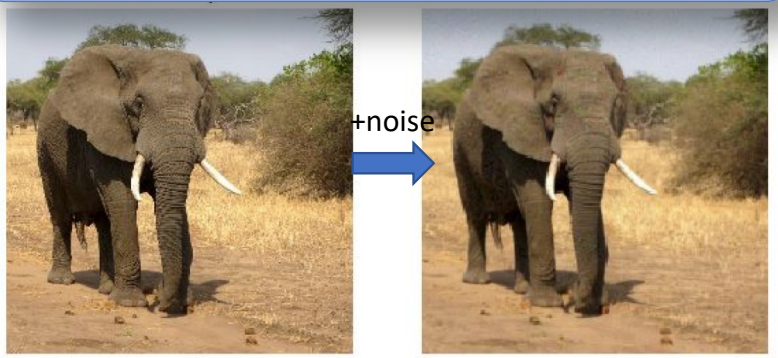
Benefits of visualizing CNN

- Helps to understand why a model they build works or does not work.
- Helps to debug models and improve training and testing strategies.
- Helps to do architectural changes or build new models
- Helps to identify dataset bias

UNIVERSITY OF BATH

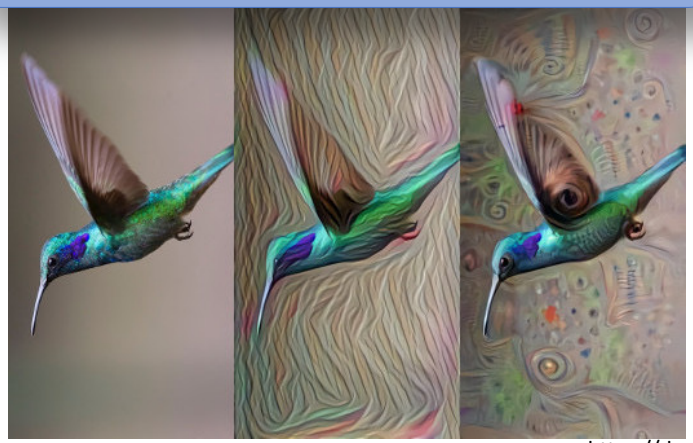
Style transfer

Fooling images



Prediction: Koala

Deep Dream



https://deepdreamgenerator.com

More than just visualization …

References

- Blog by Rachel Draelos. “CNN Heat maps: class activation mapping (GAM)”. https://glassboxmedicine.com/2019/06/11/cnn-heat-maps-class-activation-mapping-cam/
- Blog by Anil Matcha. “Class activation maps: visualizing neural network decision-making”, 2019. https://heartbeat.comet.ml/class-activation-maps-visualizing-neural-network-decision-making-92efa5af9a33
- Blog by Shubham Panchal. “Grab-CAM: A camera for your model’s decision”, 2021. https://towardsdatascience.com/grad-cam-camera-for-your-models-decision-1ef69aae8fe7