Week 14: Interpretability of deep learning - 1

Instructor: Ruixuan Wang
wangruix5@mail.sysu.edu.cn

School of Data and Computer Science
Sun Yat-Sen University

30 May, 2019

Are DL models really black boxes?

- DL models are often highly complex non-linear functions
- Should build 'transparent' models to explain decision making

Are DL models really black boxes?

- DL models are often highly complex non-linear functions
- Should build 'transparent' models to explain decision making

Motivations of model transparency and explanation:

- AI is weaker: helps researchers find failure causes
- AI is on par with humans: help establish trust with users
- AI is stronger: teach people how to make accurate predictions

Are DL models really black boxes?

- DL models are often highly complex non-linear functions
- Should build 'transparent' models to explain decision making

Motivations of model transparency and explanation:

- AI is weaker: helps researchers find failure causes
- AI is on par with humans: help establish trust with users
- AI is stronger: teach people how to make accurate predictions

How to improve transparency?

Ideas to interpret filters and neurons

Find and visualize visual patterns activating filters/neurons or influencing their activations

Interpretation of filters

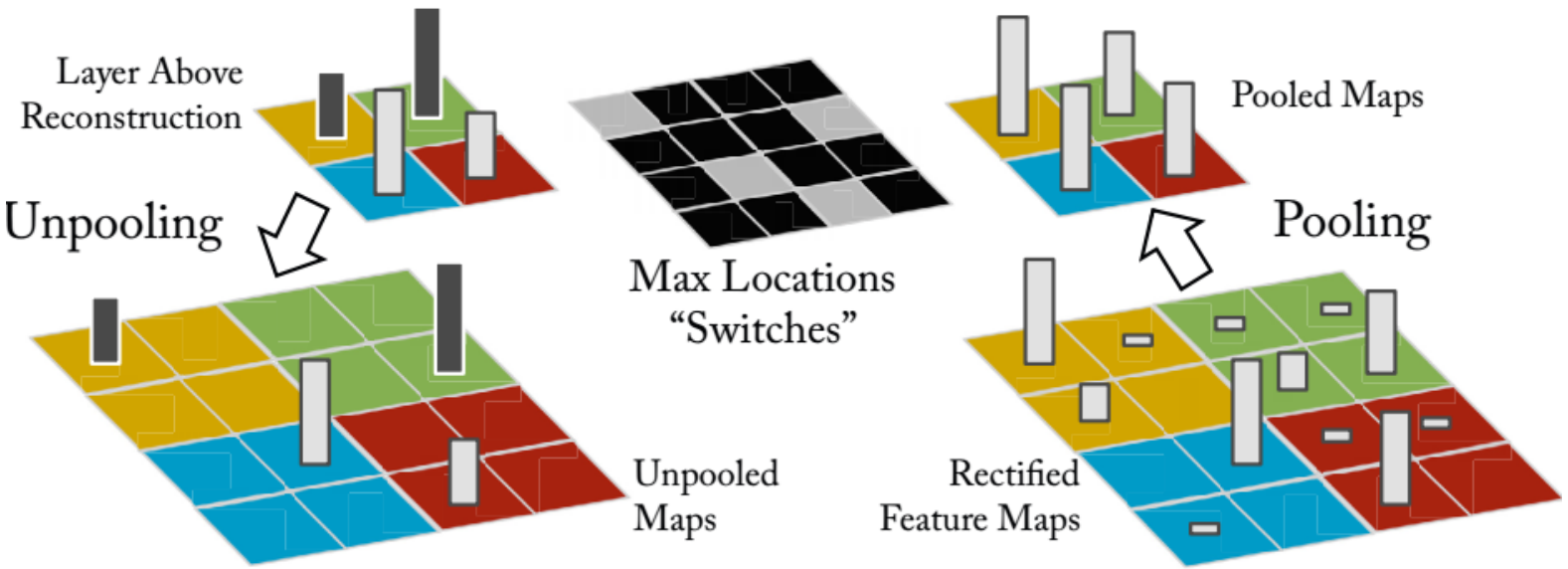
What is each filter representing?

Or: what is each filter looking for in an image?

Figures in next 4 slides from Zeiler, Fergus, "Visualizing and understanding convolutional networks", ECCV, 2014

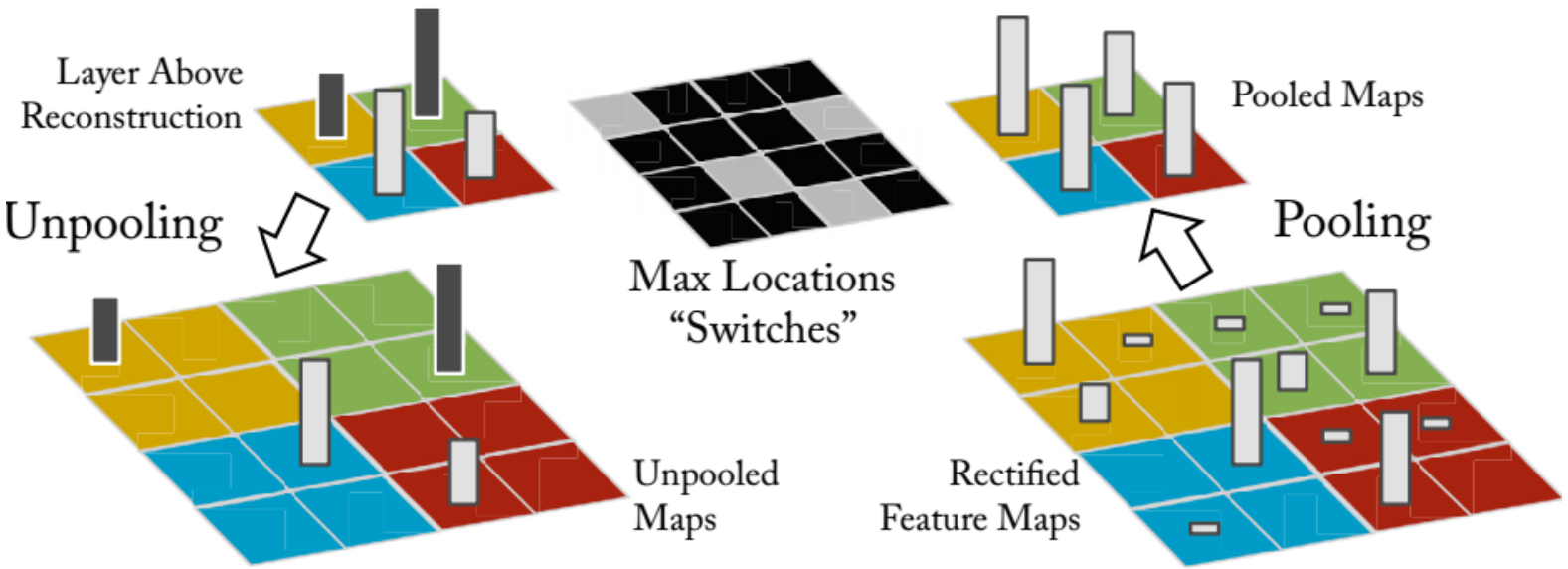
Deconvolution for visualization of feature map (kernel)

- 'Switches' record location of local max during pooling
- Unpooling reconstructs (approx) higher-resolution feature map



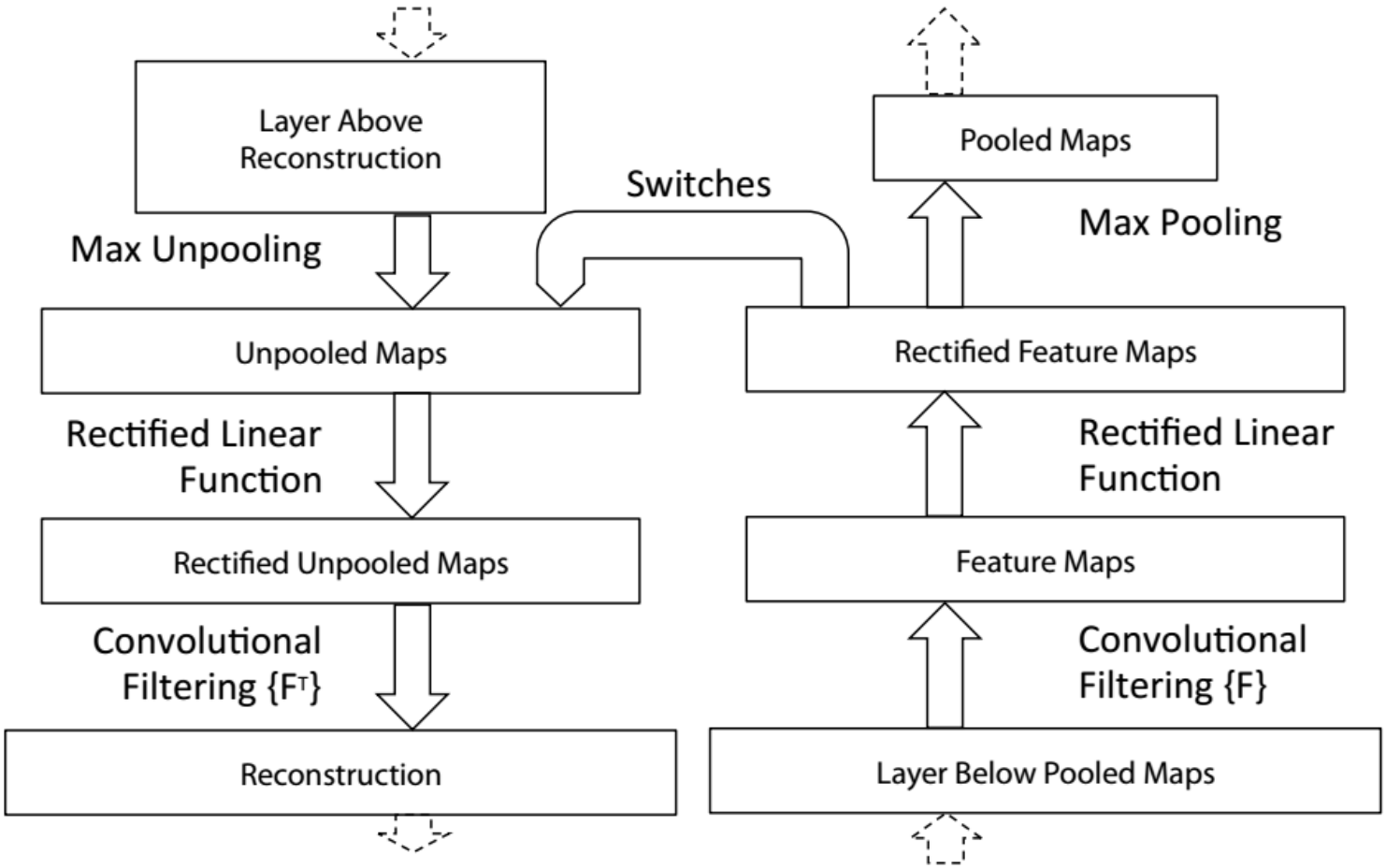
Deconvolution for visualization of feature map (kernel)

- 'Switches' record location of local max during pooling
- Unpooling reconstructs (approx) higher-resolution feature map
- How to visualize a feature map (kernel) at a conv-layer?
- 1^{st}: Select an image with strong activation at the feature map
- 2^{nd}: set all other channels to zero at the conv layer
- 3^{rd}: Unpool the feature map



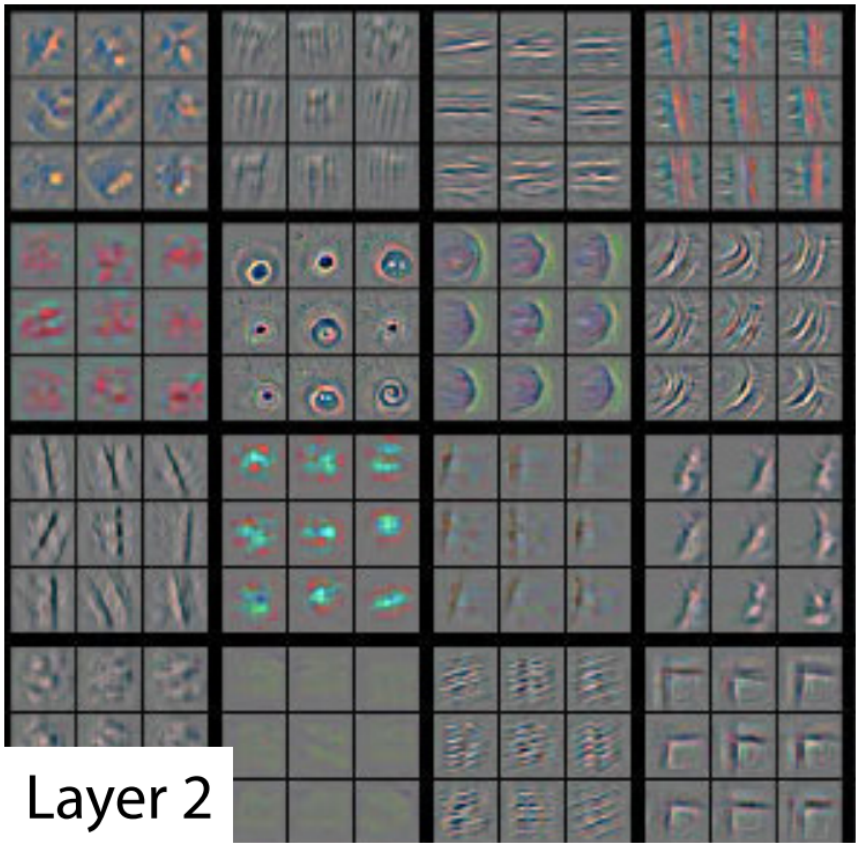
Deconvolution for visualization (cont')

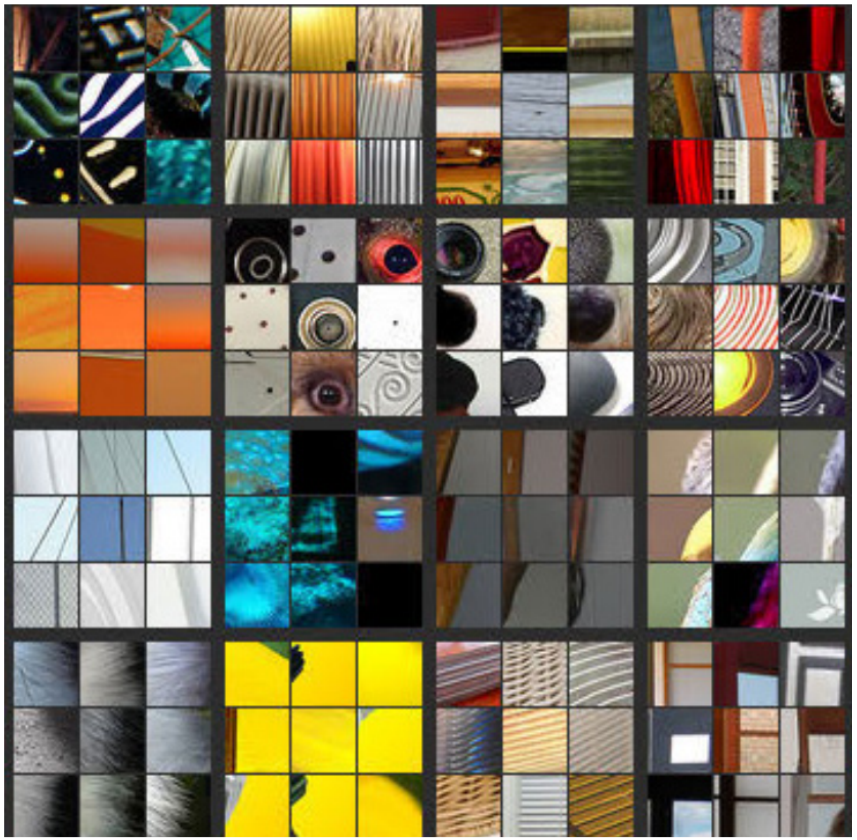
- 4^{th}: ReLU unpooled map, followed by transposed conv
- 5^{th}: Repeat steps 1-4 till the input layer
- Left path: deconvolution layer; Right path: convolution layer



Deconvolution for visualization: result

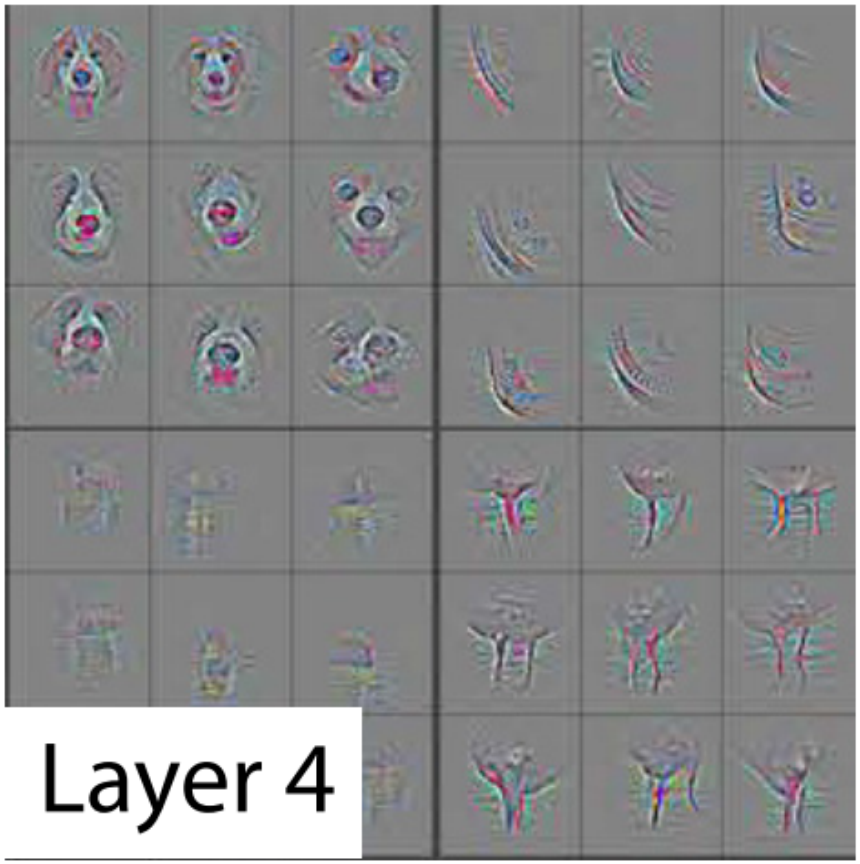
- Visualization of kernels in Layer 2 of trained CNN model
- Left: deconvolution results of 16 kernels * 9 examples
- Right: corresponding cropped patches with strong activations





Deconvolution for visualization: result

- Visualization of kernels in Layer 4 of trained CNN model
- Kernels at higher layers capture higher-level concepts



Interpretation of filters: further

Can we directly interpret each filter?

Network dissection to interpret filters

Basic idea:

- Step 1: Identify set of human-labeled visual concepts;
- Step 2: Align neurons' responses with visual concepts.

Network dissection to interpret filters

Basic idea:

- Step 1: Identify set of human-labeled visual concepts;
- Step 2: Align neurons' responses with visual concepts.

Step 1: Visual concepts were represented by Broaden dataset:

Figures here and in next 4 slides from Bau et al., "Network dissection: quantifying interpretability of deep visual representations", CVPR, 2017

Network dissection (cont')

Step 2: use neuron's response to segment region of each concept

- Given a trained CNN and an input image $\mathbf{x}$, denote by $A_k(\mathbf{x})$ the activation map of convolutional unit k.
- Segmentation: scale up (to the input image's resolution) and threshold $A_k(\mathbf{x})$ to get binary mask $M_k(\mathbf{x}) = S(A_k(\mathbf{x}) > T_k)$. Only 0.5% of locations in $M_k(\mathbf{x})$'s over all images are 1's.

Network dissection (cont')

Step 2: use neuron's response to segment region of each concept

- Given a trained CNN and an input image $\mathbf{x}$, denote by $A_k(\mathbf{x})$ the activation map of convolutional unit k.
- Segmentation: scale up (to the input image's resolution) and threshold $A_k(\mathbf{x})$ to get binary mask $M_k(\mathbf{x}) = S(A_k(\mathbf{x}) > T_k)$. Only 0.5% of locations in $M_k(\mathbf{x})$'s over all images are 1's.
- Evaluation of segmentation

$$\text{IoU}_{\text{set}}(c; M_k, s) = \frac{\sum_{\mathbf{x} \in X_{s,c}} |M_k(\mathbf{x}) \cap L_c(\mathbf{x})|}{\sum_{\mathbf{x} \in X_{s,c}} |M_k(\mathbf{x}) \cup L_c(\mathbf{x})|}$$

$L_c(\mathbf{x})$: human-annotated mask for concept c on image $\mathbf{x}$.
$\sum$: sum over images containing concept c. $s \in \{train, val\}$

Network dissection (cont')

Step 2: use neuron's response to segment region of each concept

- Given a trained CNN and an input image $\mathbf{x}$, denote by $A_k(\mathbf{x})$ the activation map of convolutional unit k.
- Segmentation: scale up (to the input image's resolution) and threshold $A_k(\mathbf{x})$ to get binary mask $M_k(\mathbf{x}) = S(A_k(\mathbf{x}) > T_k)$. Only 0.5% of locations in $M_k(\mathbf{x})$'s over all images are 1's.
- Evaluation of segmentation

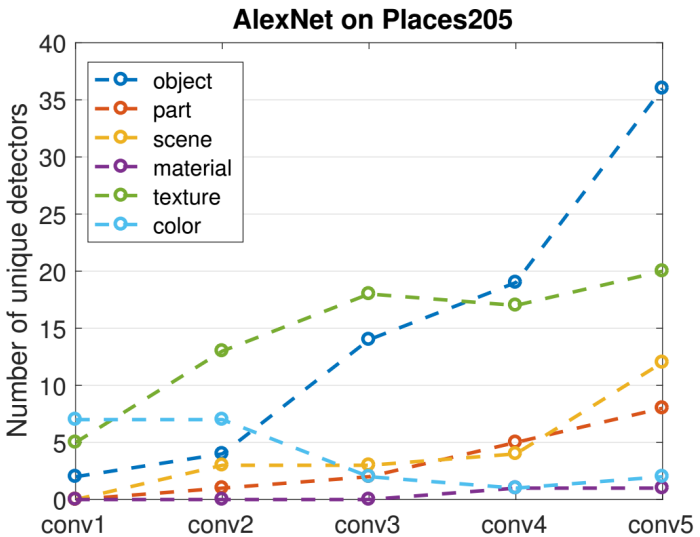
$$\text{IoU}_{\text{set}}(c; M_k, s) = \frac{\sum_{\mathbf{x} \in X_{s,c}} |M_k(\mathbf{x}) \cap L_c(\mathbf{x})|}{\sum_{\mathbf{x} \in X_{s,c}} |M_k(\mathbf{x}) \cup L_c(\mathbf{x})|}$$

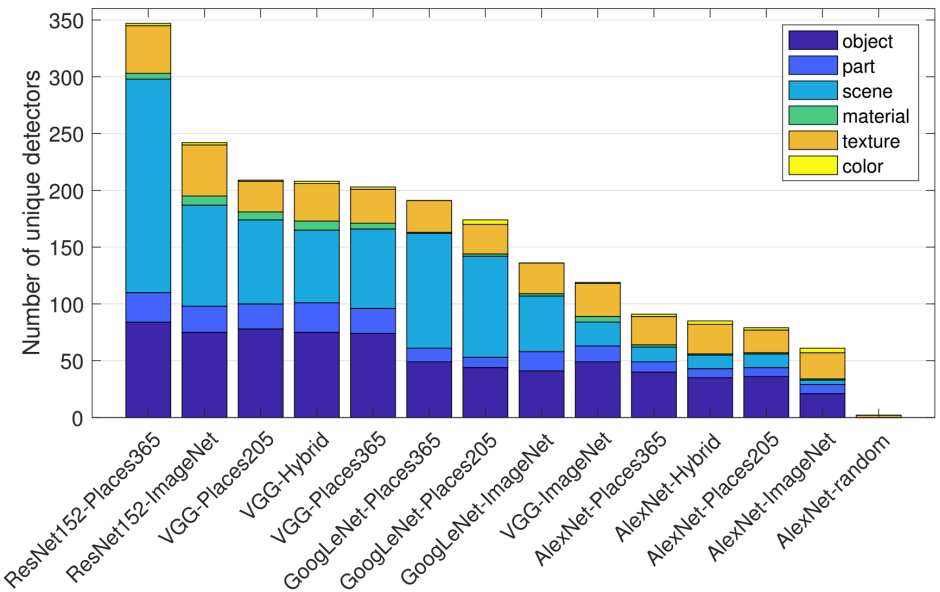
  $L_c(\mathbf{x})$: human-annotated mask for concept c on image $\mathbf{x}$.
  $\sum$: sum over images containing concept c. $s \in \{train, val\}$

- For each concept c, a few conv units k's with highest IoU_{set} (larger than a threshold) are considered relevant.
- For unit k, concepts c's with top IoU_{set} are associated.

Network dissection: result

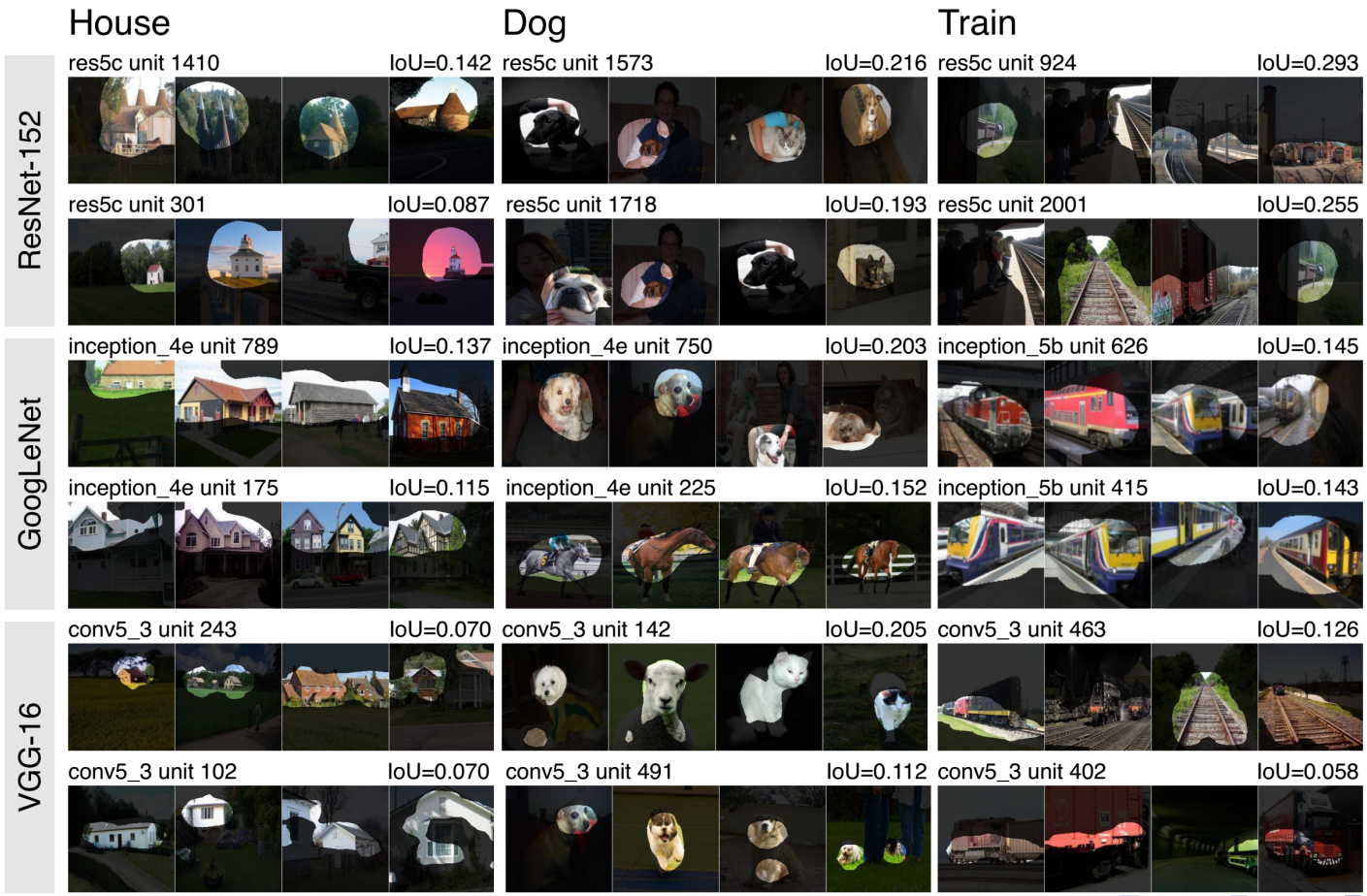
- Left figure: color and texture concepts dominate at lower layers, while object and part concepts emerge at conv layer 5.
- Right figure: deeper CNN models capture more concepts.





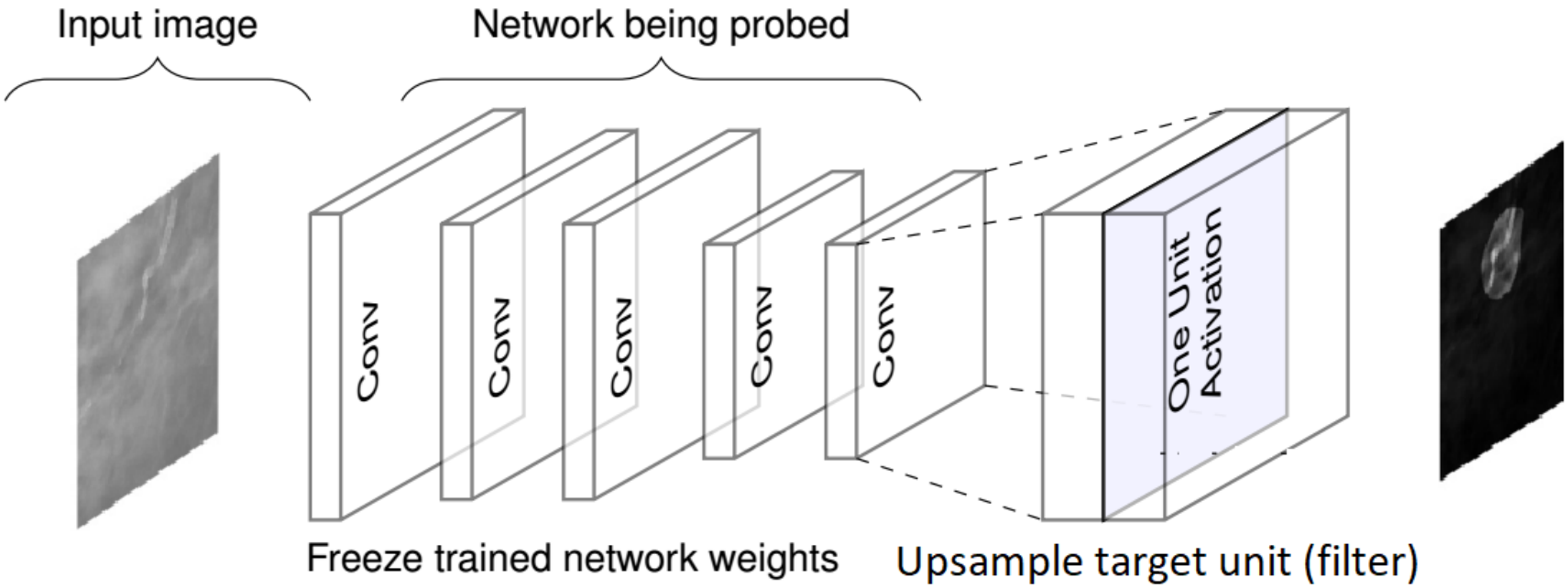
Network dissection: result

- For each CNN, 2 units with highest IoU's selected for each concept; 4 images with highest activation shown for each unit.
- units/filters did detect certain (even more than one) concepts.



Network dissection: medical application

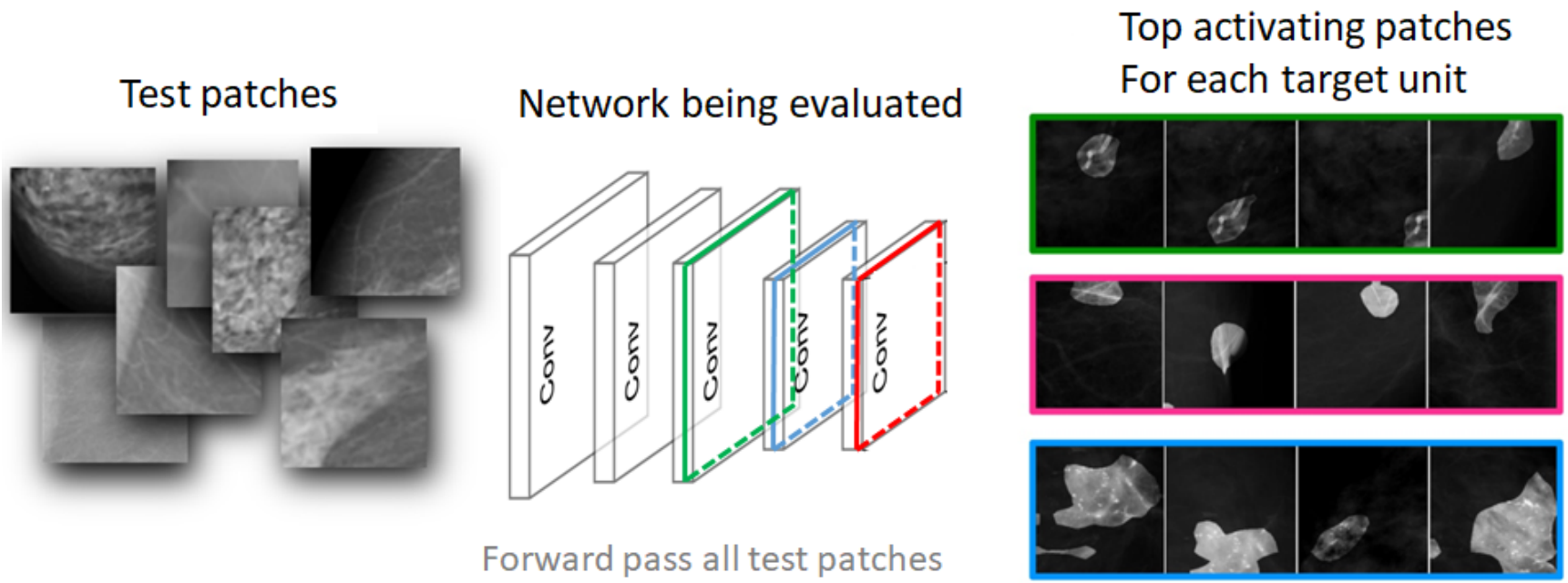
- For a filter (unit), choose top-k activation maps from all data
- Upsample activation map to the size of input image
- Segment out the regions with strong activations for evaluation



Figures here adapted from Bau et al., "Network dissection: quantifying interpretability of deep visual representations", CVPR, 2017

Network dissection: medical application

- For each segmented region, ask experts such as:
- 'Do these images show recognizable phenomena?'
- 'Please describe each of the phenomena you see'
- 'Indicate the phenomenon's association with breast cancer'

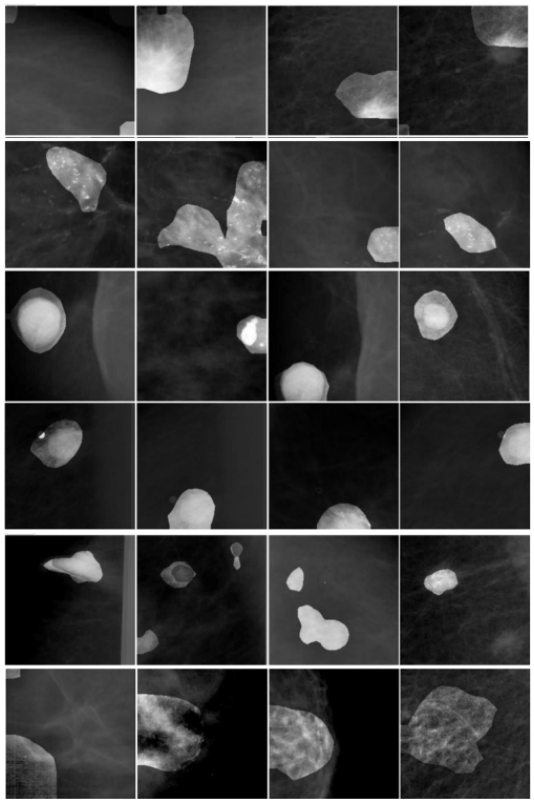


Figures here and in next slide adapted from Wu et al., "Expert identification of visual primitives used by CNNs during mammogram classification", SPIE, 2018

Network dissection: medical application

- CNN filters identify recognizable medical phenomena used by radiologists!

BI-RADS Lexicon Category	***Neuron Annotation***	**Network, Layer, Neuron**
Mass - Margin	*masses with spiculated edge*	Inception v3 mixed_7a unit 0371
Calcification	*calcifications, innumerable*	VGG 16 conv5_3 unit 0063
Breast Composition	*high density area, large calcifications*	AlexNet conv5 unit 0014
Mass	*advanced cancers*	VGG-16 conv5_3 unit 0283
Associated Features	*architectural distortion*	*ResNet 152 layer 4 unit 0183*
Breast Composition	*fatty breast texture*	ResNet 152 layer 4 unit 0005



But: observation from network dissection

- Clearly a conv unit encodes multiple concepts.

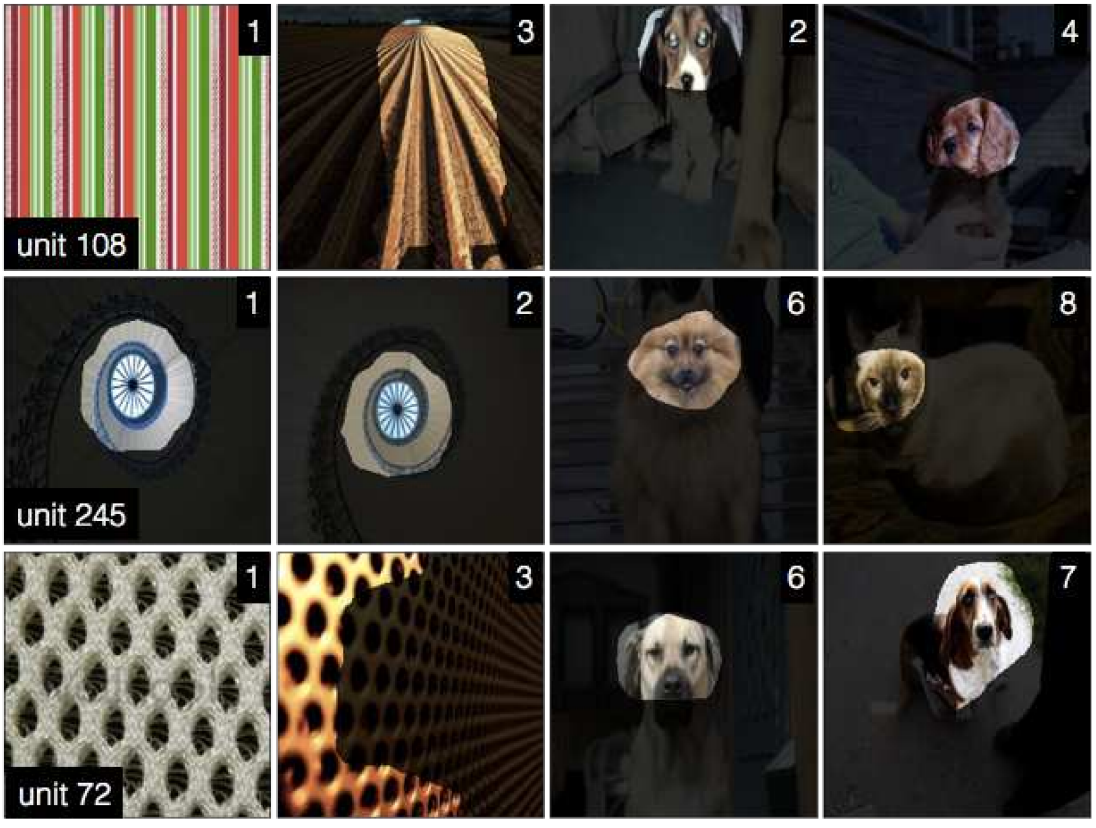


Figure (each row): corner n denotes the n-th most maximally activating image for a given filter.

One relevant question

Is each concept encoded by a single unit/filter?

Note: in brains, multiple neurons encode a concept.

Net2vec: quantifying how concepts are encoded by filters

- Basic idea: learn to use multiple filters' responses to segment regions of each concept.

Net2vec: quantifying how concepts are encoded by filters

- Basic idea: learn to use multiple filters' responses to segment regions of each concept.
- Learn weight $\mathbf{w} \in \mathbb{R}^K$ (K: number of filters in a layer) to linearly combine thresholded activations

$$M(\mathbf{x}; \mathbf{w}) = \sigma\left(\sum_k w_k \cdot \mathbb{I}(A_k(\mathbf{x}) > T_k)\right)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Net2vec: quantifying how concepts are encoded by filters

- Basic idea: learn to use multiple filters' responses to segment regions of each concept.
- Learn weight $\mathbf{w} \in \mathbb{R}^K$ (K: number of filters in a layer) to linearly combine thresholded activations

$$M(\mathbf{x}; \mathbf{w}) = \sigma\left(\sum_k w_k \cdot \mathbb{I}(A_k(\mathbf{x}) > T_k)\right)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

- For each concept c, learn $\mathbf{w}$ by minimizing a per-pixel binary cross entropy loss

$$\begin{aligned}\mathcal{L} = -\frac{1}{N_{s,c}} \sum_{\mathbf{x} \in X_{s,c}} & \alpha M(\mathbf{x}; \mathbf{w}) L_c(\mathbf{x}) \\ & + (1-\alpha)(1 - M(\mathbf{x}; \mathbf{w})(1 - L_c(\mathbf{x}))\end{aligned}$$

α is class weight to balance concept and background size.

Net2vec: result

- Evaluation on individual image with learned weight:

$$\mathrm{IoU}_{\mathrm{ind}}(\mathbf{x}, c; M) = \frac{|M(\mathbf{x}) \cap L_c(\mathbf{x})|}{|M(\mathbf{x}) \cup L_c(\mathbf{x})|}$$

Net2vec: result

- Evaluation on individual image with learned weight:

$$\text{IoU}_{\text{ind}}(\mathbf{x}, c; M) = \frac{|M(\mathbf{x}) \cap L_c(\mathbf{x})|}{|M(\mathbf{x}) \cup L_c(\mathbf{x})|}$$

- Combined filters outperform single filters for segmentation.

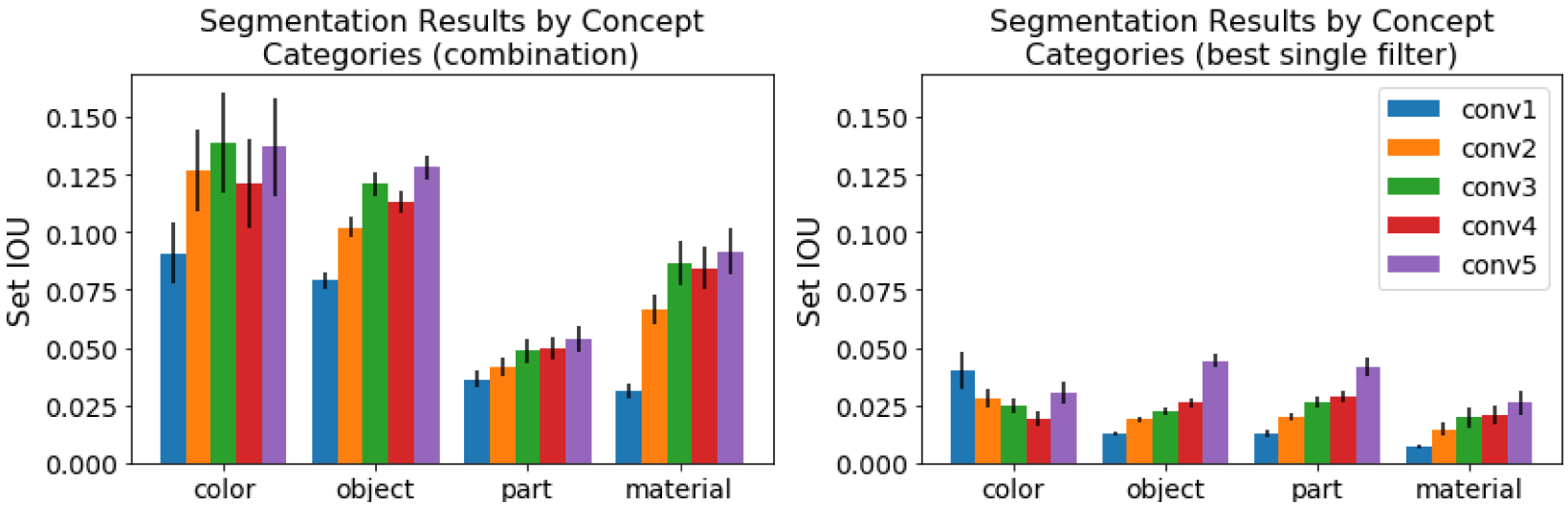


Figure: segmentation performance of using combined filters (left) versus using best single filter at each layer for each set of concepts.

Figures here and in next 3 slides from Fong and Vedaldi, "Net2Vec: quantifying and explaning how concepts are encoded by filters in deep neural networks", CVPR, 2018

Net2vec: result

- Multiple filters required to encode a concept.
- 8 for materials and parts; 16 for objects; 128 for colors.

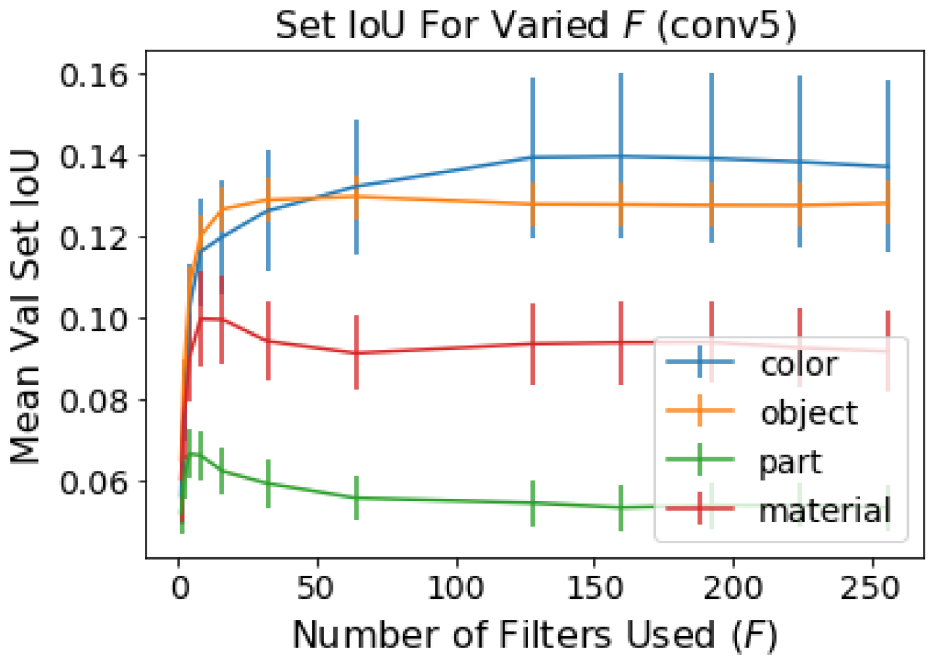


Figure: Segmentation performance with respect to number of top conv5 filters used to encode single concept.

Net2vec: result

- Many filters selected as best single filters by multiple concepts

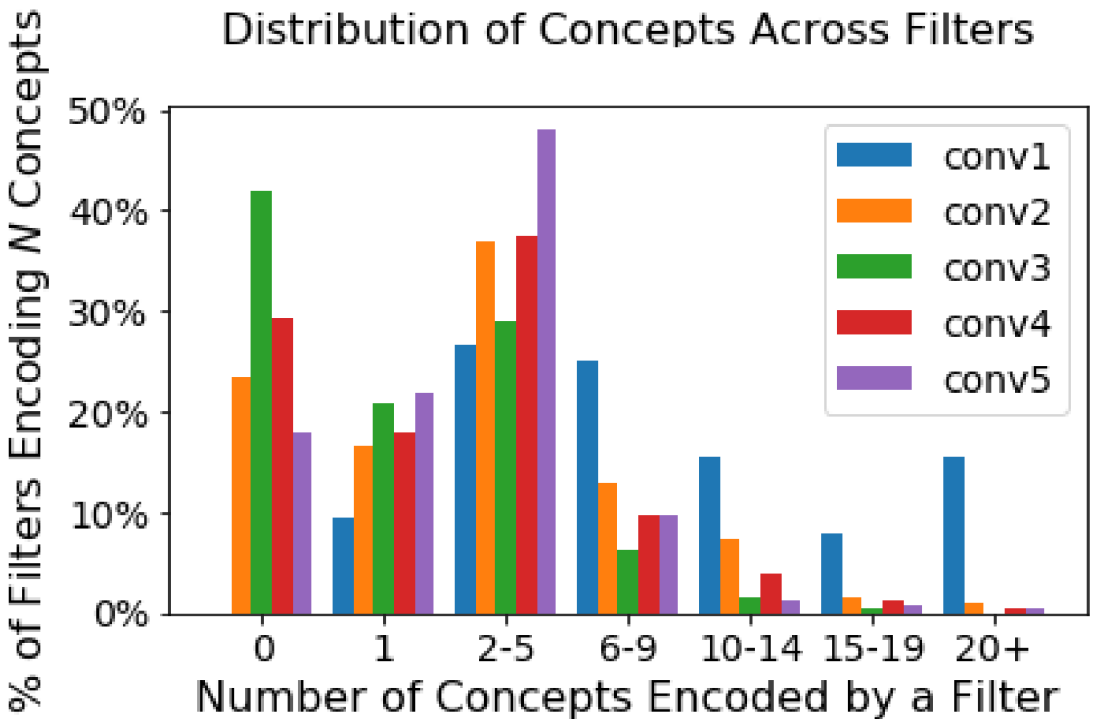


Figure: distribution of how many filters (y-axis) encode how many concepts (x-axis)

Net2vec: result

- With single filter (blue hist): IoUs =0 for many 'dog' images.
- Single filter doesn't consistently fire strongly on given concept.

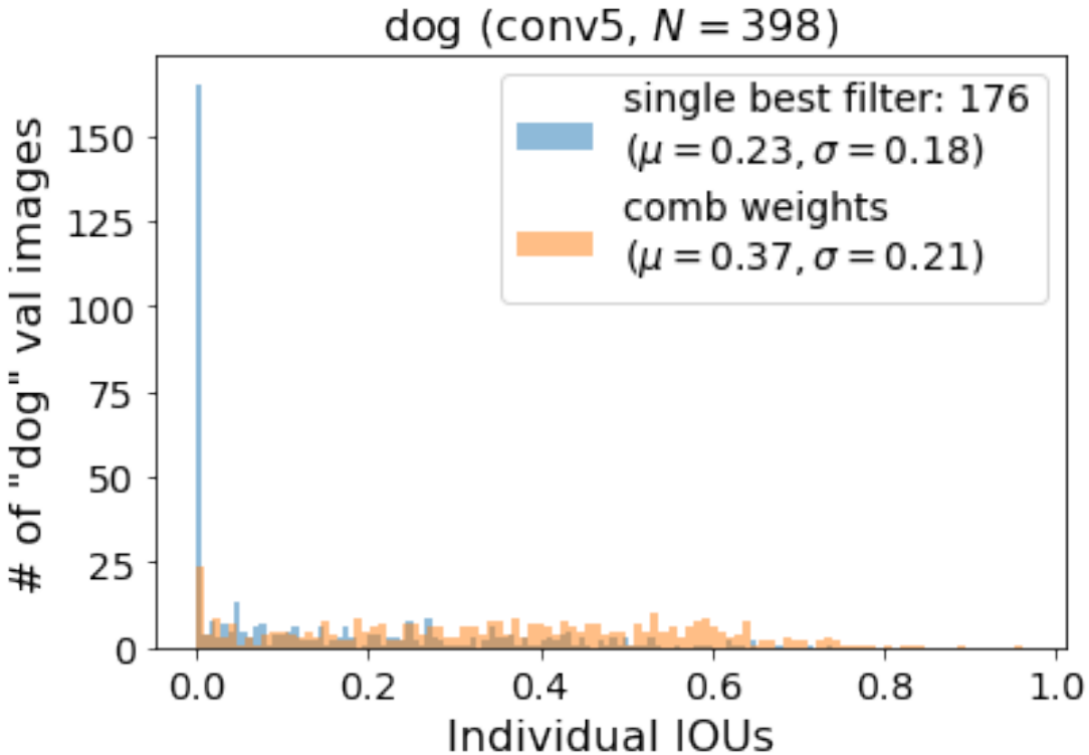


Figure: distribution of IoUs when using single best filter (blue) and the learned weights (orange) for 'dog' concept.

Can we force each neuron to encode one object part?

- New loss terms: (1) Each filter encodes an object part from a single catetory; (2) the filter is activated by a single region.

Can we force each neuron to encode one object part?

- New loss terms: (1) Each filter encodes an object part from a single catetory; (2) the filter is activated by a single region.

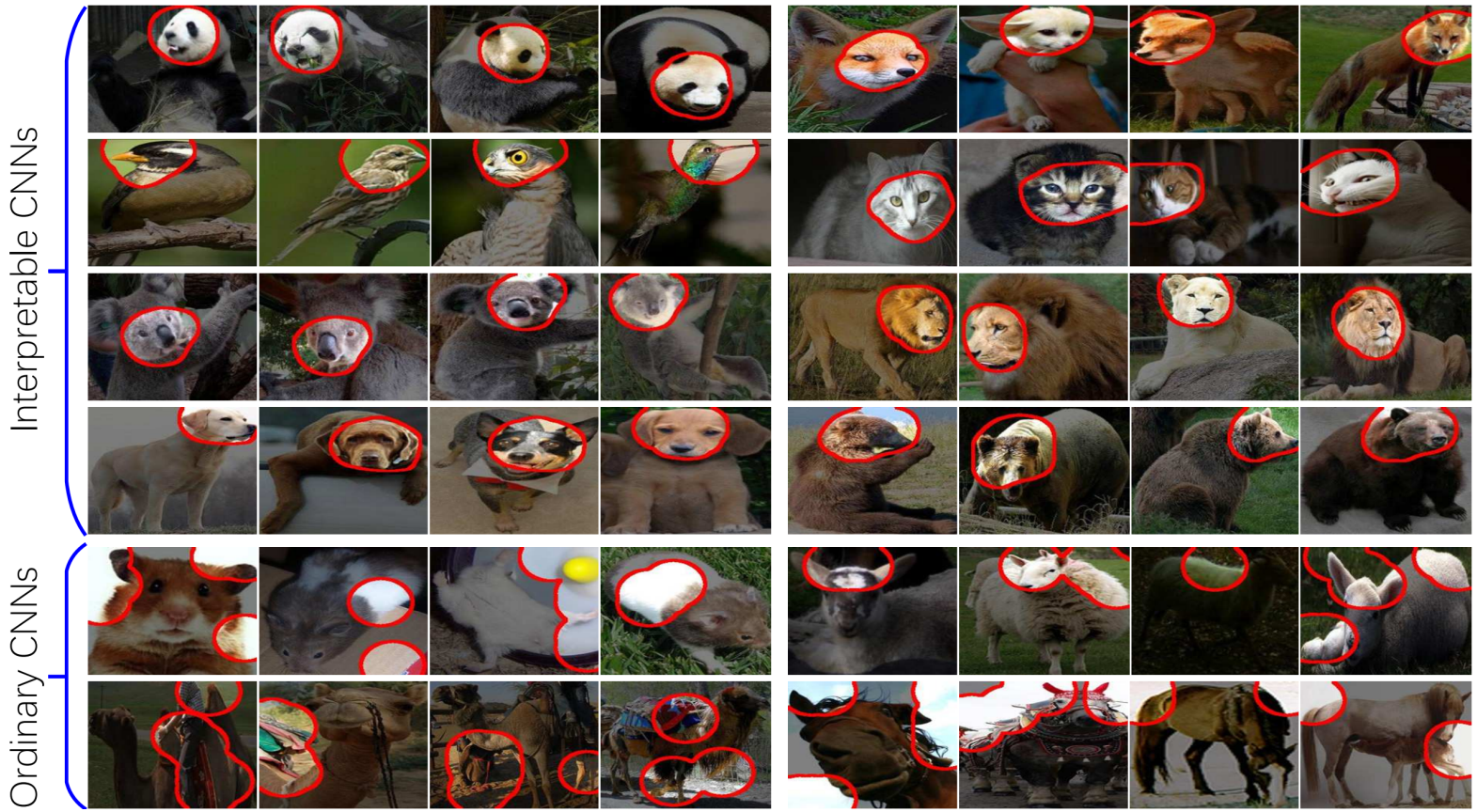


Figure from Zhang et al., "Interpretable convolutional neural networks", CVPR, 2018

Interpretation of classifier output neurons

What is each output neuron representing?

Idea: finding patterns maximizing neuron activations

Activation maximization for class-specific neuron

- Numerically generate a representative image $\mathbf{I}$ for class c

$$\arg\max_{\mathbf{I}} f_c(\mathbf{I}) - \lambda \|\mathbf{I}\|^2$$

where $f_c(\mathbf{I})$ is score of class c based on pre-trained CNN.

Activation maximization for class-specific neuron

- Numerically generate a representative image $\mathbf{I}$ for class c

$$\arg\max_{\mathbf{I}} f_c(\mathbf{I}) - \lambda \|\mathbf{I}\|^2$$

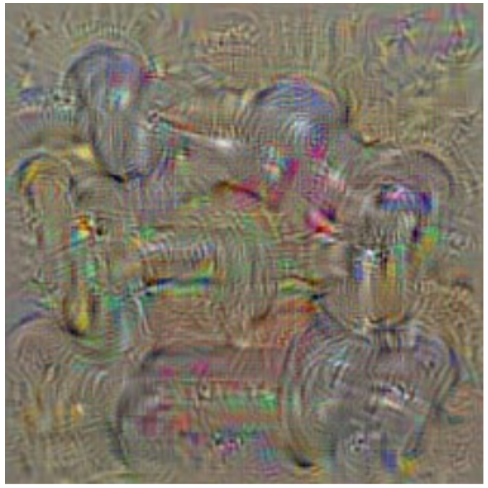
  where $f_c(\mathbf{I})$ is score of class c based on pre-trained CNN.
- Use backprop to find locally optimal $\mathbf{I}$, not CNN weights.
- $f_c(\mathbf{I})$ is (unnormalised) pre-softmax score rather than final posterior score; maximizing final posterior of the class can be achieved by minimizing posteriors of other classes.

Activation maximization for class-specific neuron

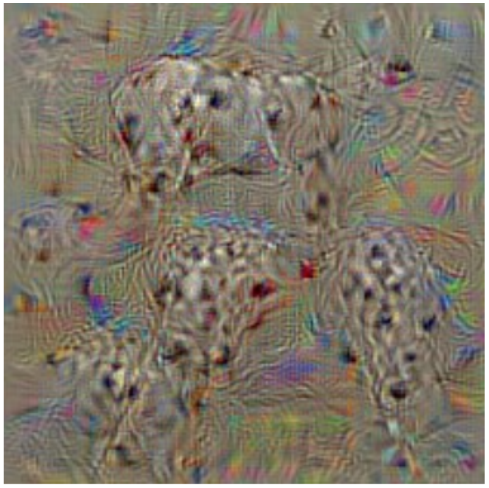
- Numerically generate a representative image $\mathbf{I}$ for class c

$$\arg\max_{\mathbf{I}} f_c(\mathbf{I}) - \lambda \|\mathbf{I}\|^2$$

  where $f_c(\mathbf{I})$ is score of class c based on pre-trained CNN.
- Use backprop to find locally optimal $\mathbf{I}$, not CNN weights.
- $f_c(\mathbf{I})$ is (unnormalised) pre-softmax score rather than final posterior score; maximizing final posterior of the class can be achieved by minimizing posteriors of other classes.


dumbbell


cup


dalmatian

Activation maximization: extension

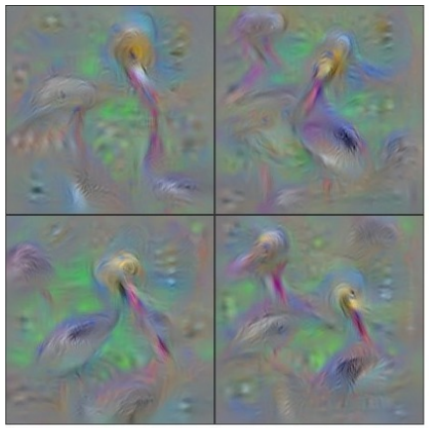
Besides L2 regularizer (reduce high pixel values), also added:

- 1) Gaussian blur: avoid high frequency information
- 2) Clip pixels with small values (norms) to 0: suppress pixels belong to background
- 3) Clip pixels with small gradients to 0: suppress pixels with small contributions to neuron activation

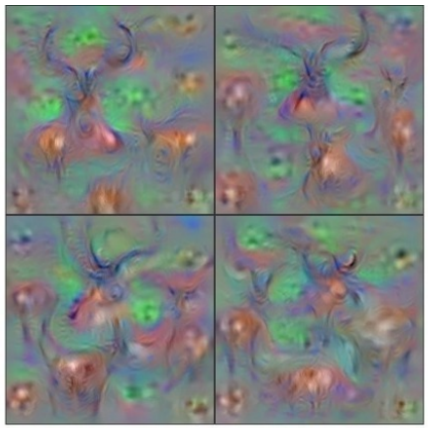
Activation maximization: extension

Besides L2 regularizer (reduce high pixel values), also added:

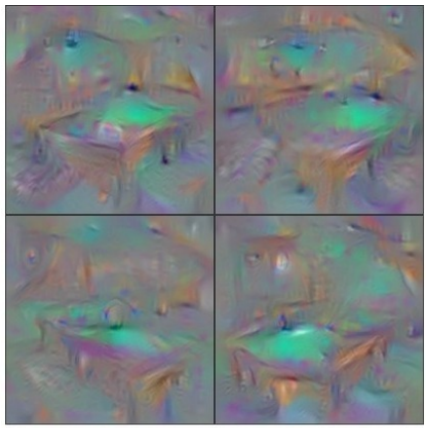
- 1) Gaussian blur: avoid high frequency information
- 2) Clip pixels with small values (norms) to 0: suppress pixels belong to background
- 3) Clip pixels with small gradients to 0: suppress pixels with small contributions to neuron activation
- However, visualizations are fragmented, with parts of objects



Pelican



Hartebeest



Billiard Table

Activation maximization: refinement with multifacets

- Good initialization: cluster dataset based on projected feature vectors; optimization starts from each cluster centre (15 closest images averaged)
- More regularizers: total variations & jitter (in later slides)

Reconstructions of multiple feature types (facets) recognized by the same “grocery store” neuron

Corresponding example training set images recognized by the same neuron as in the "grocery store" class

Figures here, in prev 3 and next 2 slides from Simonyan et al., "Deep inside convolutional networks: visualising image classification models and saliency maps", ICLR workshop, 2014
Yosinski et al., "Understanding neural networks through deep visualization", ICML workshop, 2015
Nguyen et al., "Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks", ICML workshop, 2016

Multifacet feature visualization: result

- More natural color palette and globally consistent; output layer neurons also encode global structures, details, context

Multifacets for different layers' filters

- Neurons at all levels are multifacets
- Higher-level neurons are more multifaceted than lower levels

Another idea to visualize neurons/filters

Alternatively, find neuron-specific saliency regions

Gradient-based method

- Given image $\mathbf{I}_0$, a class c, a CNN with class score function $f_c(\mathbf{I})$, determine influence of each pixel in $\mathbf{I}_0$ on score $f_c(\mathbf{I}_0)$.

Gradient-based method

- Given image $\mathbf{I}_0$, a class c, a CNN with class score function $f_c(\mathbf{I})$, determine influence of each pixel in $\mathbf{I}_0$ on score $f_c(\mathbf{I}_0)$.
- Consider a simplified linear score model for class c,

$$f_c(\mathbf{I}) = \mathbf{w}_c^T \mathbf{I} + \mathbf{b}_c$$

$\mathbf{I}$ is in vectorized form; Magnitude of elements of $\mathbf{w}_c$ indicates the importance of corresponding pixels of $\mathbf{I}$ for class c.

Gradient-based method

- Given image $\mathbf{I}_0$, a class c, a CNN with class score function $f_c(\mathbf{I})$, determine influence of each pixel in $\mathbf{I}_0$ on score $f_c(\mathbf{I}_0)$.
- Consider a simplified linear score model for class c,

$$f_c(\mathbf{I}) \quad = \quad \mathbf{w}_c^T \mathbf{I} + \mathbf{b}_c$$

  $\mathbf{I}$ is in vectorized form; Magnitude of elements of $\mathbf{w}_c$ indicates the importance of corresponding pixels of $\mathbf{I}$ for class c.
- For a deep CNN, $f_c(\mathbf{I})$ is highly non-linear!
- However, given an image $\mathbf{I}_0$, first-order Taylor expansion gives

$$f_c(\mathbf{I}) \quad \approx \quad \mathbf{w}^T \mathbf{I} + \mathbf{b}$$

  where $\mathbf{w}$ is the derivative of f_c w.r.t. image $\mathbf{I}$ at the point $\mathbf{I}_0$:

$$\mathbf{w} \quad = \quad \frac{\partial f_c}{\partial \mathbf{I}}\Big|_{\mathbf{I}_0}$$

Gradient-based method

- Larger magnitude of elements $\mathbf{w}$ indicates the corresponding pixels of image $\mathbf{I}$ affects the class score more
- Since $\mathbf{I}$ is very close to $\mathbf{I}_0$, above indication also holds for $\mathbf{I}_0$.

Gradient-based method

- Larger magnitude of elements $\mathbf{w}$ indicates the corresponding pixels of image $\mathbf{I}$ affects the class score more
- Since $\mathbf{I}$ is very close to $\mathbf{I}_0$, above indication also holds for $\mathbf{I}_0$.

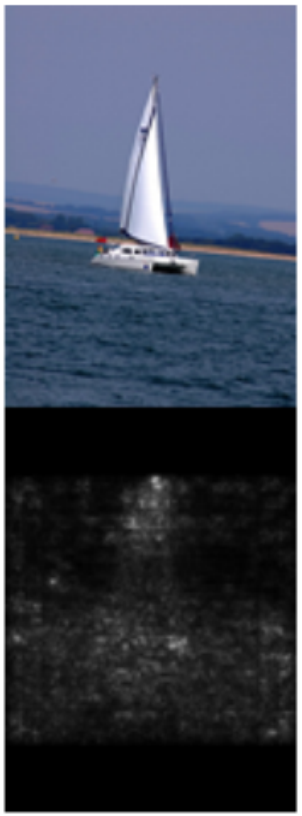
Image-specific saliency map: highlights image regions constituting most evidence for (or against) activation of a class output.

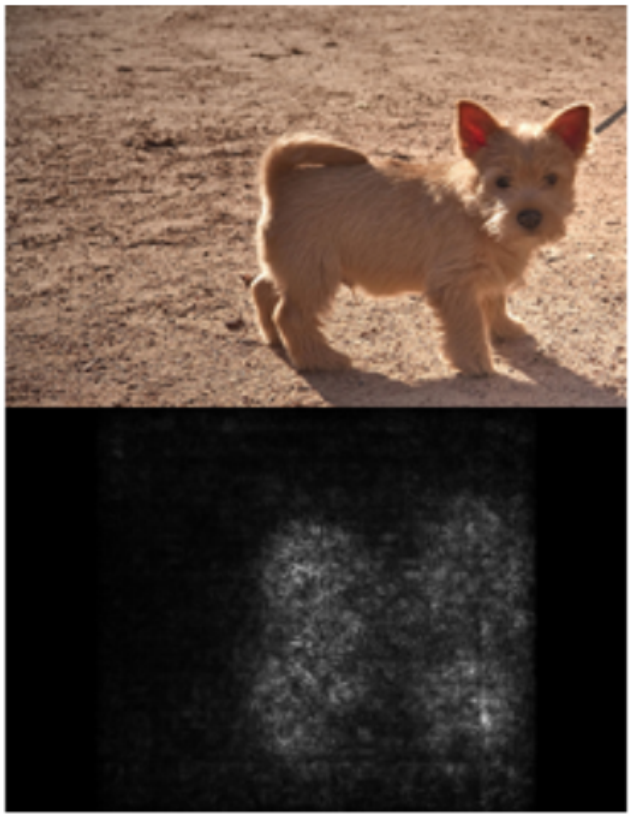
- 1^{st}: compute gradient of (unnormalized) class score w.r.t. image pixels
- 2^{nd}: take absolute value and max over RGB channels

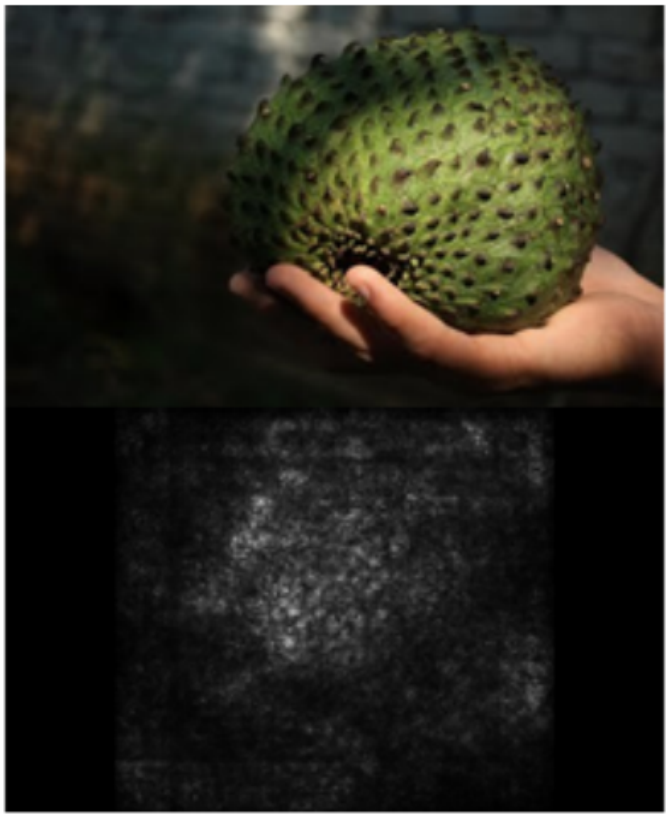
Figures in next 3 slides from Simonyan et al., "Deep inside convolutional networks: visualising image classification models and saliency maps", ICLR workshop, 2014
Springenberg et al., "Striving for simplicity: the all convolutional net", ICLR workshop, 2015

Gradient-based method

- Saliency maps for the highest-scoring class on each image
- Pixels with larger saliency contributed more to class score







Another way to interpret output neurons

Idea: feel importance by imagining losing it!

Occlusion for saliency region

- Occlude one patch per time, obtain classifier output
- If output drops more, the patch contributes more

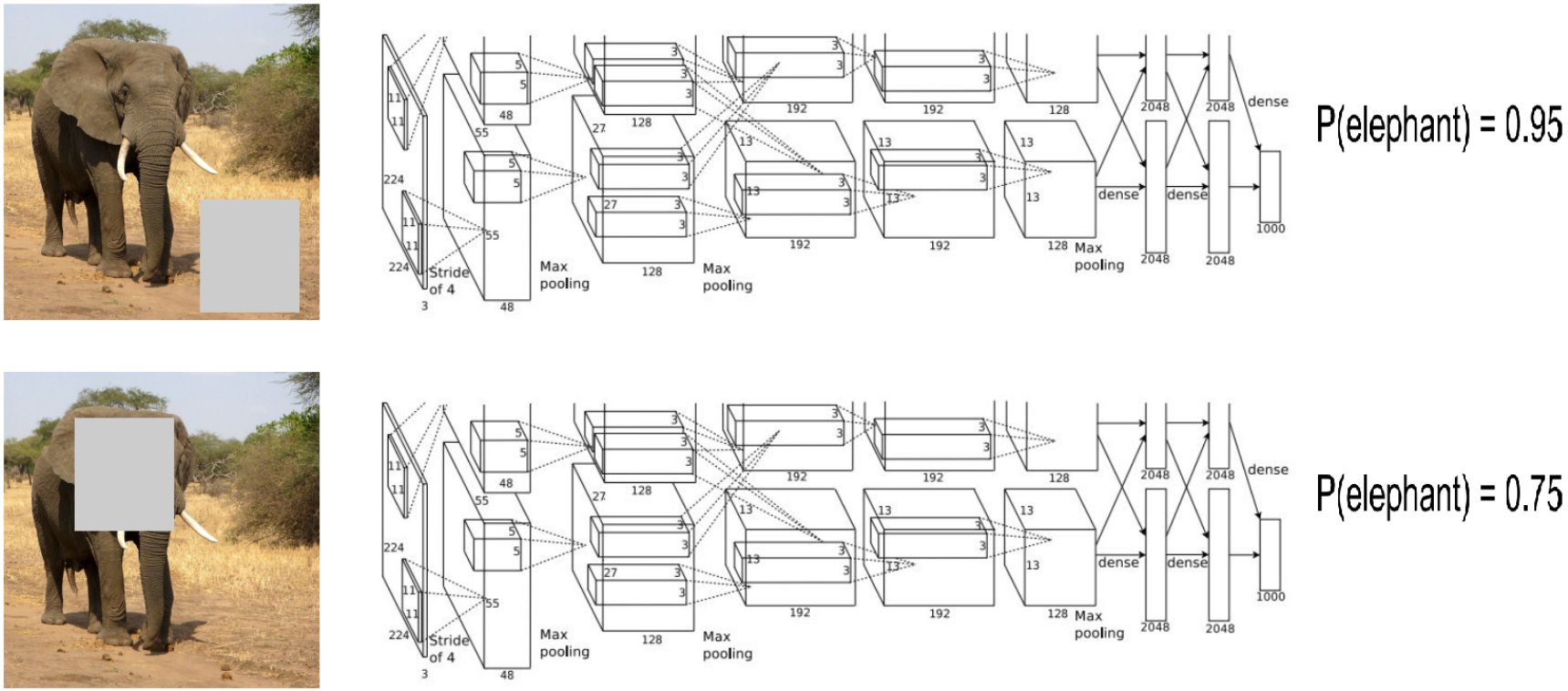


Figure from Stanford CS231n Lecture 13, "Visualizing and understanding", 2018

Occlusion for saliency map

- Large classifier output drop indicates important regions

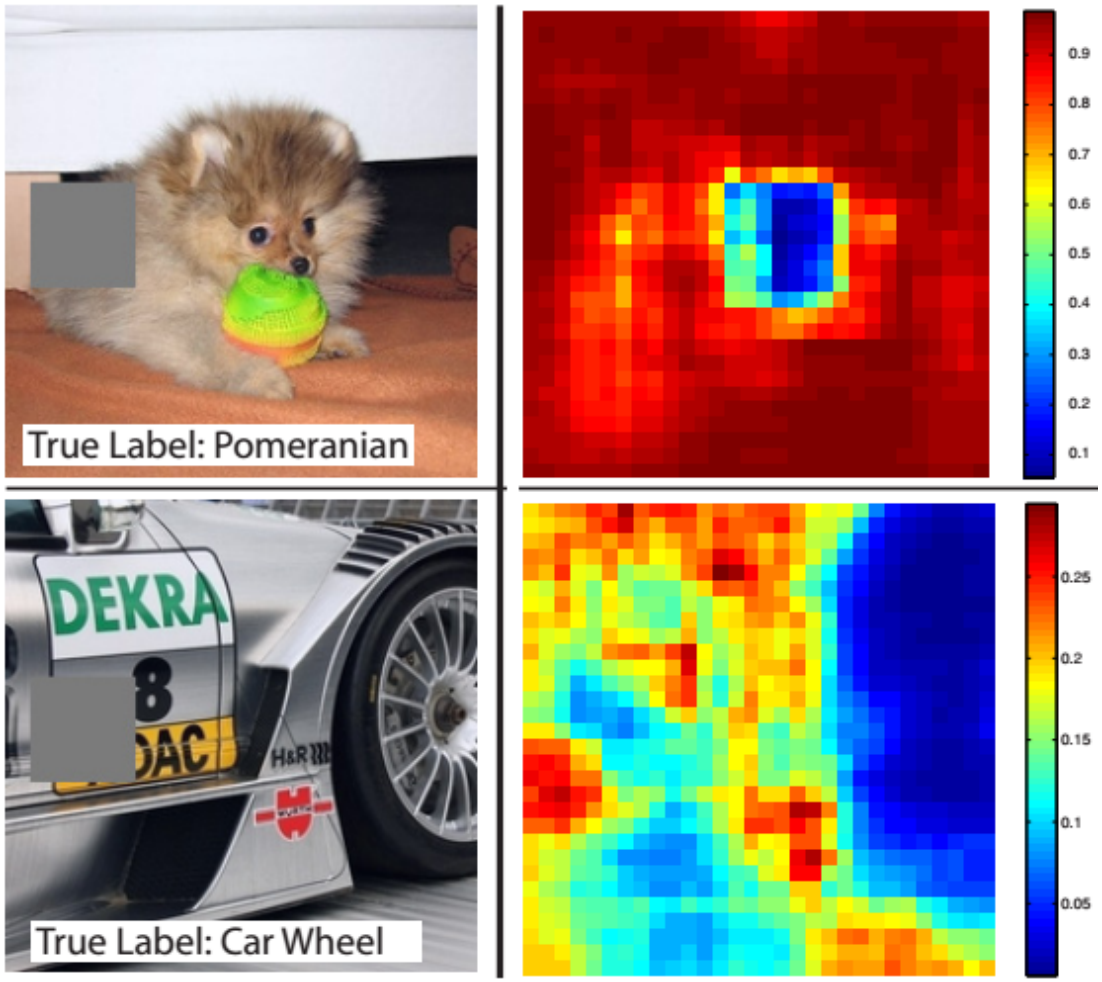
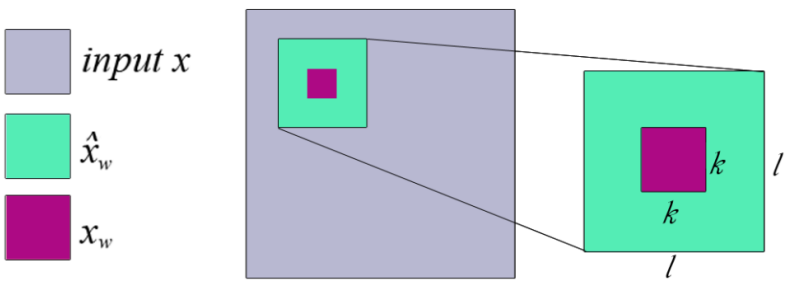


Figure from Zeiler, Fergus, "Visualizing and understanding convolutional networks", ECCV, 2014

Better way to fill occluded region: conditional sampling

- 1^{st}: Use surrounding neighborhood region (green) to generate multiple smaller patches
- 2^{nd}: Fill occluded region (purple) with each generated patch
- 3^{rd}: Obtain CNN output score for each filled image
- 4^{th}: Average these output scores



Better way to fill occluded region: conditional sampling

- 1^{st}: Use surrounding neighborhood region (green) to generate multiple smaller patches
- 2^{nd}: Fill occluded region (purple) with each generated patch
- 3^{rd}: Obtain CNN output score for each filled image
- 4^{th}: Average these output scores
- Right: red regions contribute evidence *for* the class prediction, blue regions being evidence *against* the class.

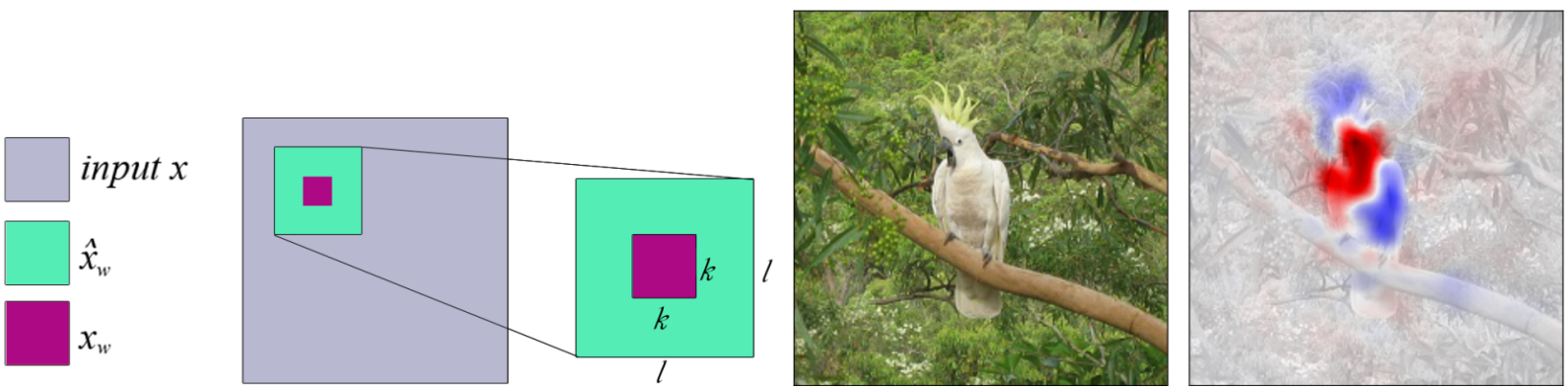
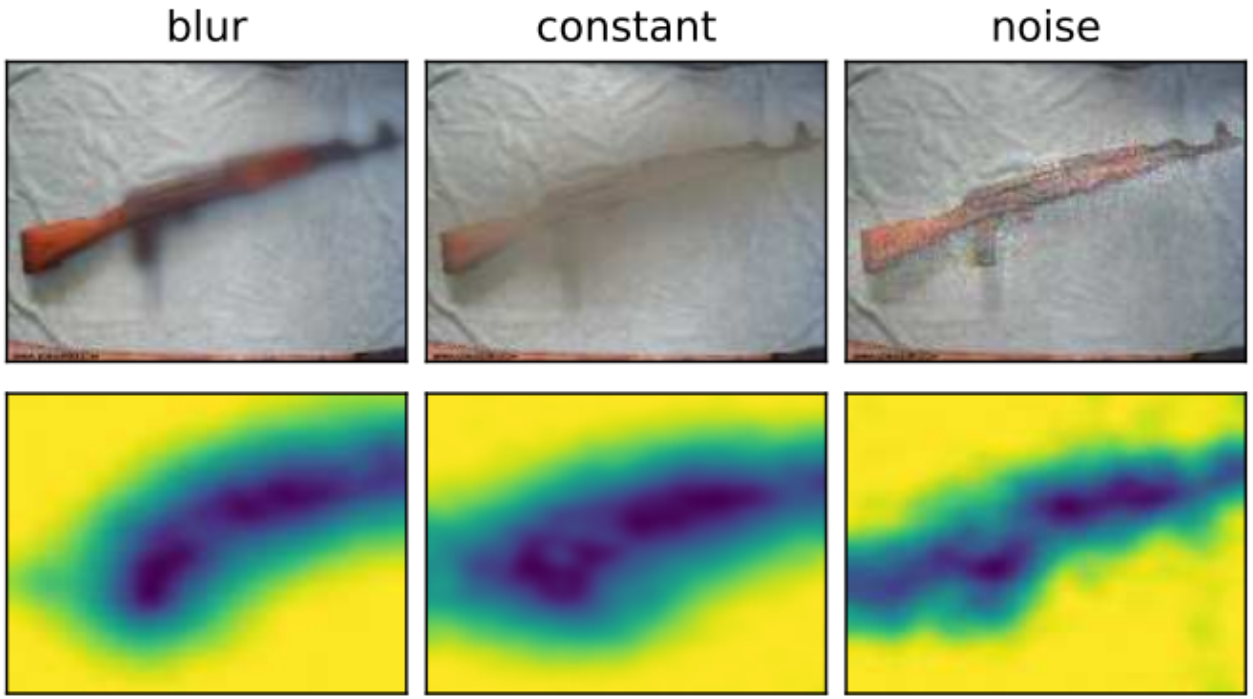


Figure from Zintgraf et al., "Visualizing deep neural network decisions: prediction difference analysis", ICLR, 2017

More ways to perturb regions

- With mask m, change region with constant, noise, blur

$$[\Phi(x_0; m)](u) = \begin{cases} m(u)x_0(u) + (1 - m(u))\mu_0, & \text{constant}, \\ m(u)x_0(u) + (1 - m(u))\eta(u), & \text{noise}, \\ \int g_{\sigma_0 m(u)}(v - u)x_0(v)\, dv, & \text{blur}, \end{cases}$$



Deletion and preservation

- Goal: find smallest mask with large class score drop

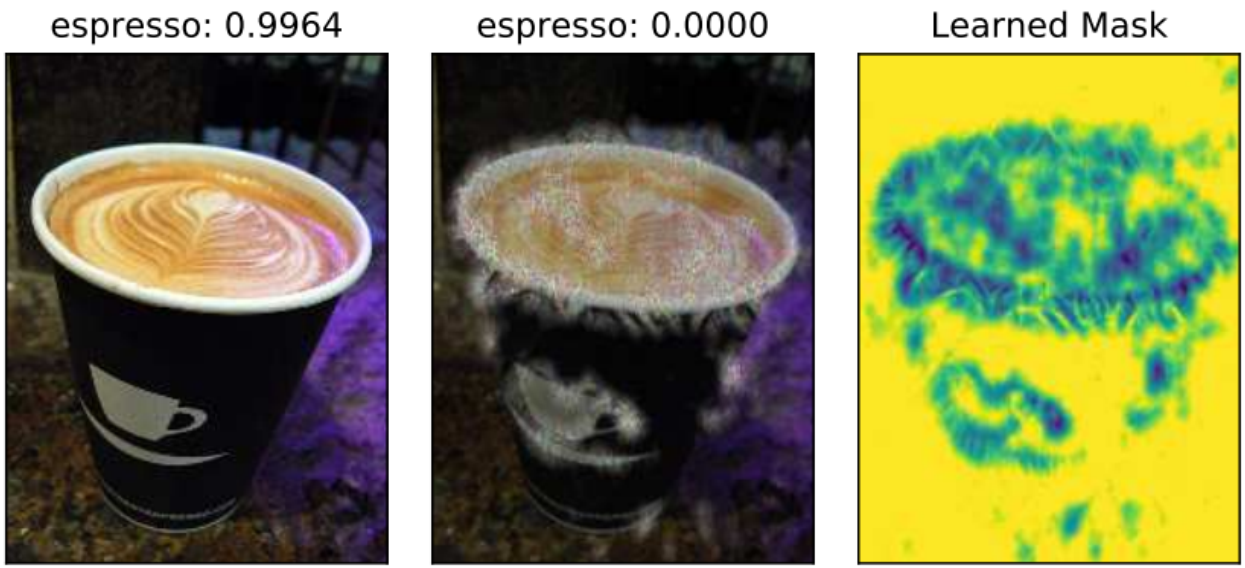
$$m^* = \underset{m \in [0,1]^\Lambda}{\operatorname{argmin}} \lambda \|\mathbf{1} - m\|_1 + f_c(\Phi(x_0; m))$$

Deletion and preservation

- Goal: find smallest mask with large class score drop

$$m^* = \underset{m \in [0,1]^\Lambda}{\operatorname{argmin}} \lambda \|\mathbf{1} - m\|_1 + f_c(\Phi(x_0; m))$$

- However, learned mask may contain large artefacts



Refined loss with prior

- Prior 1: mask is simple with regular structure; total variation should be small
- Prior 2: mask is robust to noise/jittering; jittering mask still causes large class score drop

Refined loss with prior

- Prior 1: mask is simple with regular structure; total variation should be small
- Prior 2: mask is robust to noise/jittering; jittering mask still causes large class score drop

$$\min_{m\in[0,1]^{\Lambda}} \lambda_1 \|\mathbf{1} - m\|_1 + \lambda_2 \sum_{u\in\Lambda} \|\nabla m(u)\|_{\beta}^{\beta} + \mathbb{E}_{\tau}[f_c(\Phi(x_0(\cdot - \tau), m))]$$

Refined loss with prior

- Prior 1: mask is simple with regular structure; total variation should be small
- Prior 2: mask is robust to noise/jittering; jittering mask still causes large class score drop

$$\min_{m \in [0,1]^{\Lambda}} \lambda_1 \|\mathbf{1} - m\|_1 + \lambda_2 \sum_{u \in \Lambda} \|\nabla m(u)\|_{\beta}^{\beta} + \mathbb{E}_{\tau}[f_c(\Phi(x_0(\cdot - \tau), m))]$$

Figures here and in prev 2 slides from Fong, Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation", ICCV, 2017

Summary

- Visualization of neurons/filters helps understand DL models
- Relation between neurons and concepts: many-to-many.
- DL models are becoming more transparent

Further reading:

- Shrikumar et al., "Learning important features through propagating activation differences", ICML, 2017
- Zhang et al., "MDNet: a semantically and visually interpretable medical image diagnosis network", CVPR, 2017