

Visualization and Interpretability

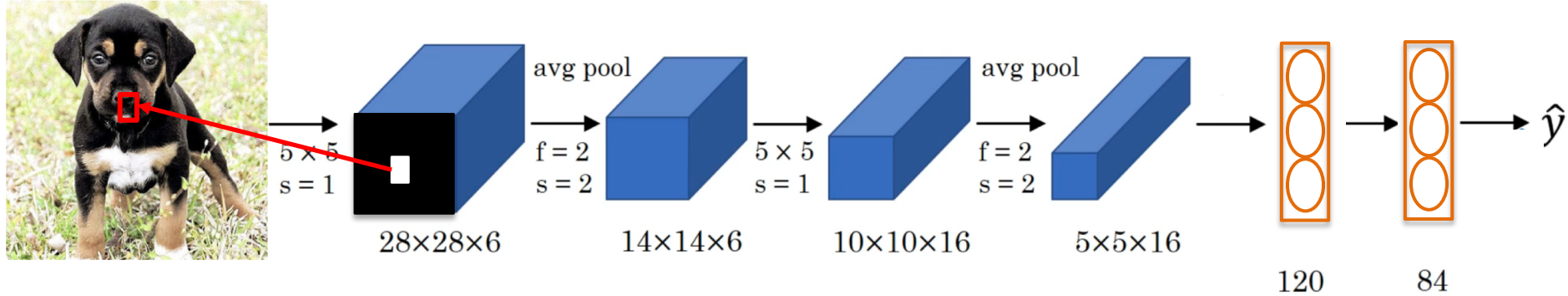
Visualization of ConvNets

- Visualization of Features
- Visualization of Activations
- Visualization of Gradients
- T-SNE Visualization
- DeepDream
-

Visualization is a great way for debugging!

Visualizing and understanding CNNs

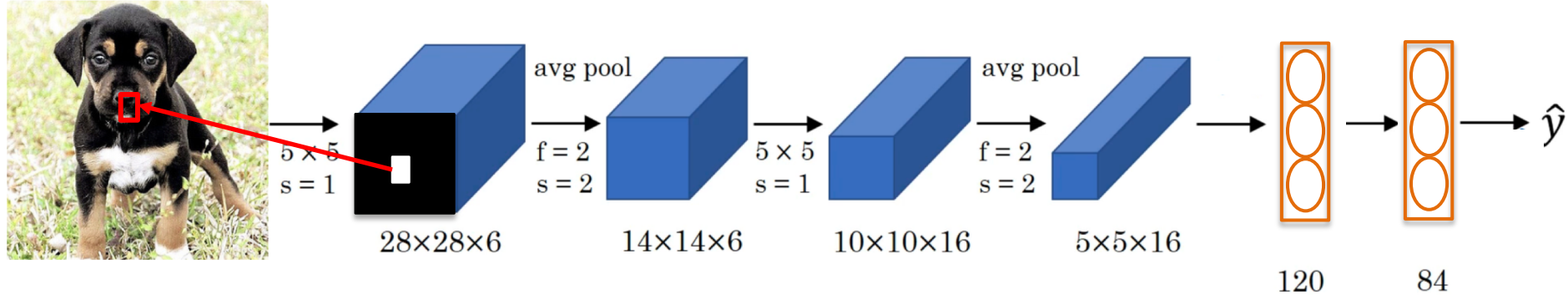
Visualizing in the image space



- Pick a unit in layer 1.
- Find the 9 image patches in your dataset that maximize the unit's activation.

Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

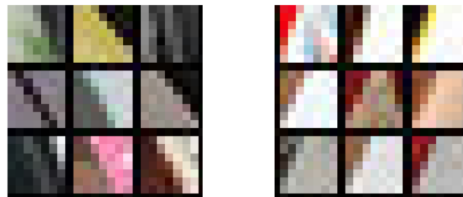
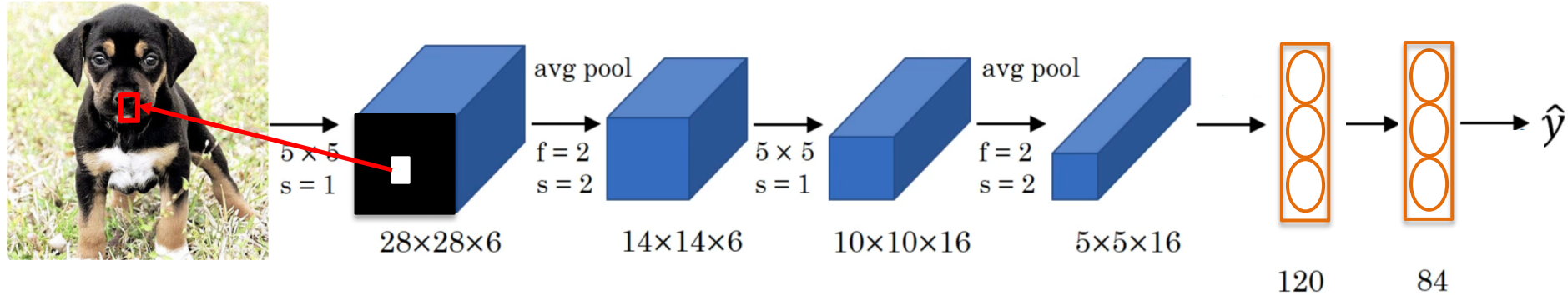
Visualizing in the image space



Feature map 1, layer 1, 9 image patches that provided the highest activation

Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing in the image space

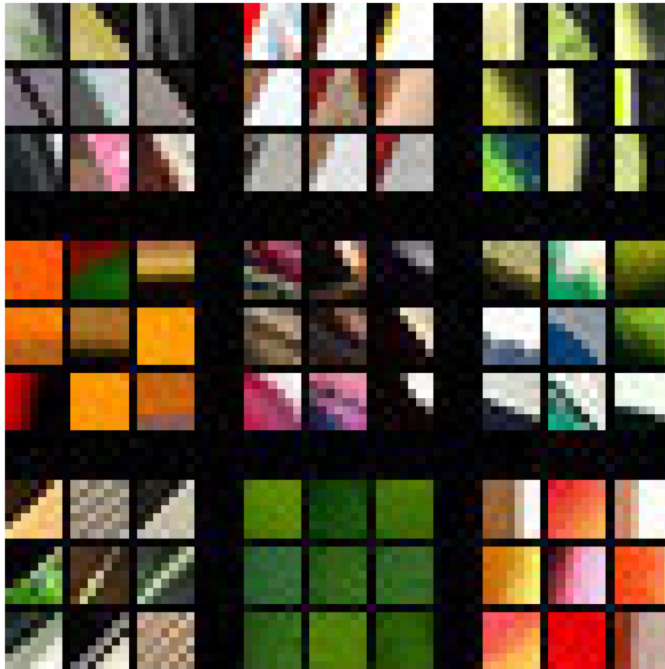


Feature map 2, layer 1, 9 image patches that provided the highest activation

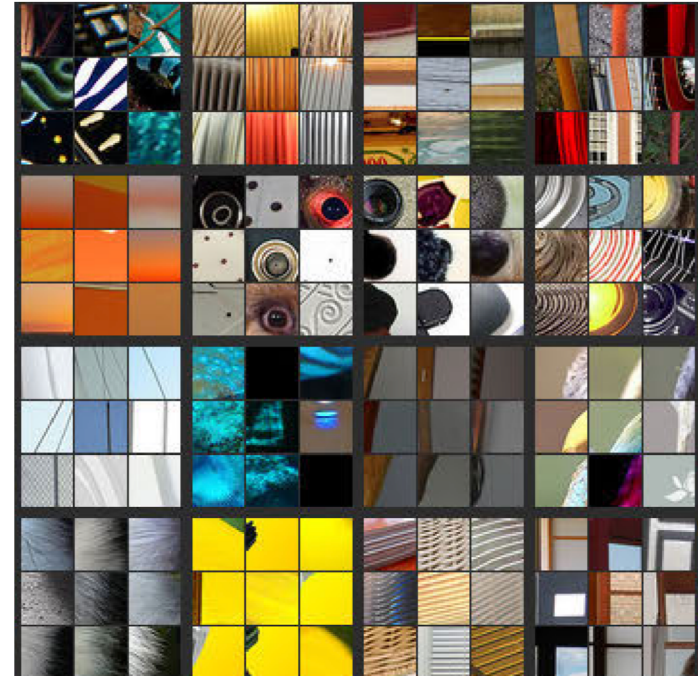
Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing in the image space

Layer 1



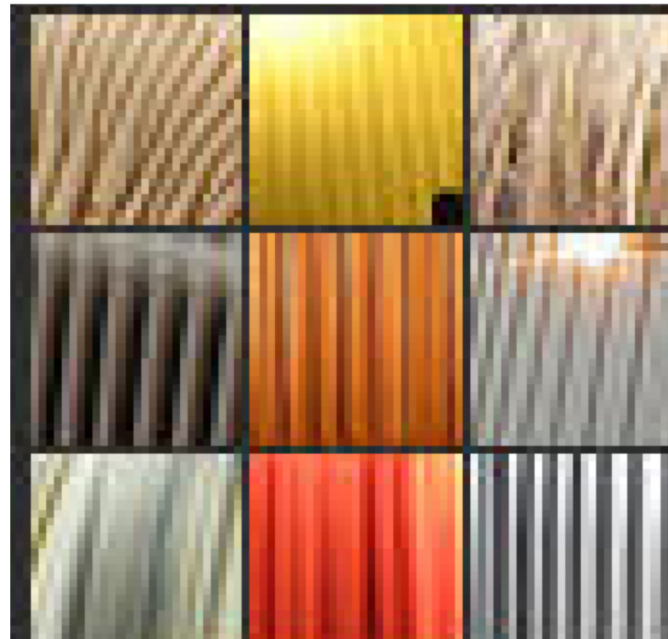
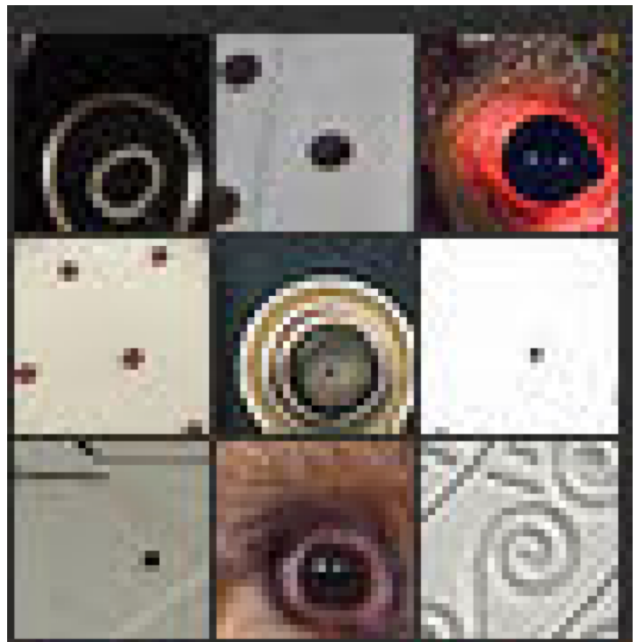
Layer 2



Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing in the image space

Zoom in, examples of Layer 2



Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing in the image space

Zoom in, examples of Layer 5



Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing in the image space

Zoom in, examples of Layer 5

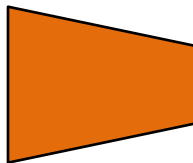


Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing importance

The occlusion experiment

- Block different parts of the image and see how the classification score changes

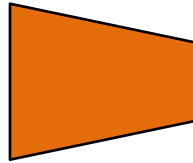


DOG 0.96

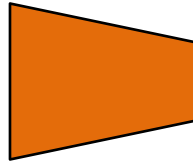
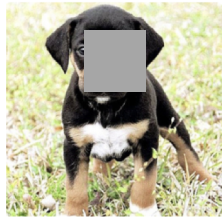
Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

The occlusion experiment

- Block different parts of the image and see how the classification score changes



DOG 0.95



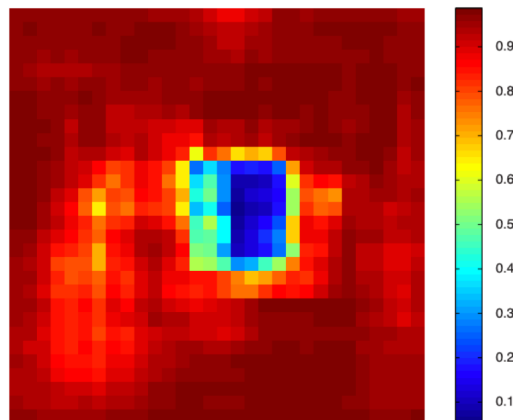
DOG 0.35

The face of the dog is more important for correct classification

Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

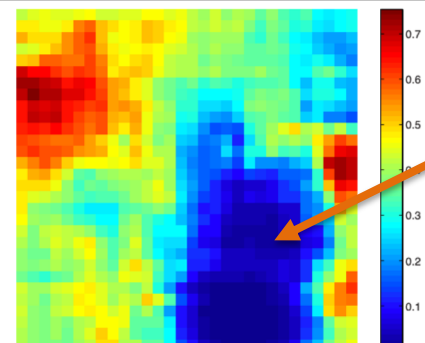
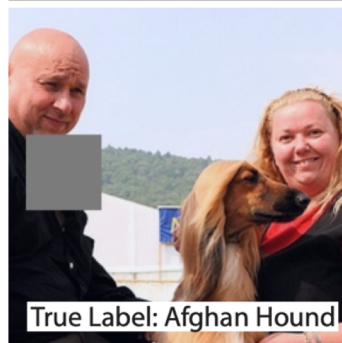
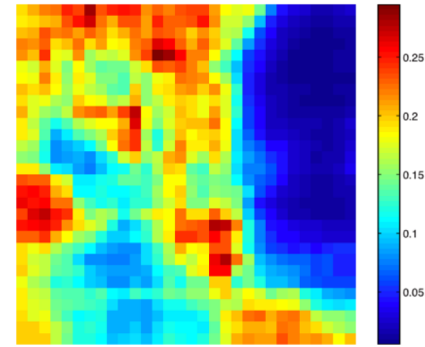
The occlusion experiment

- Create a map, where each pixel represents the classification probability if an occlusion square is placed in that region



Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

The occlusion experiment



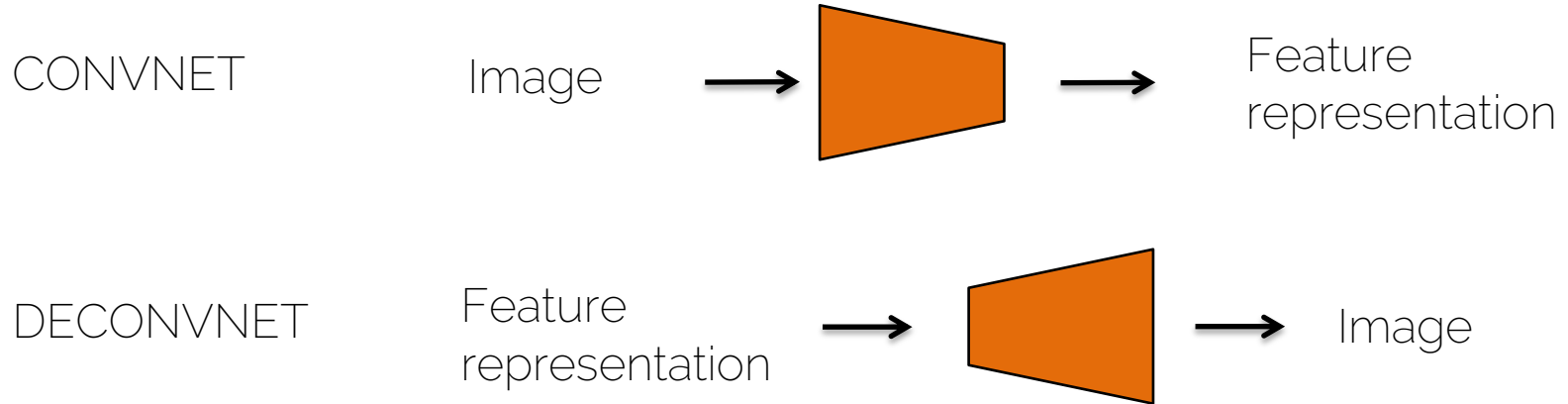
Most important
pixels for
classification

Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing features

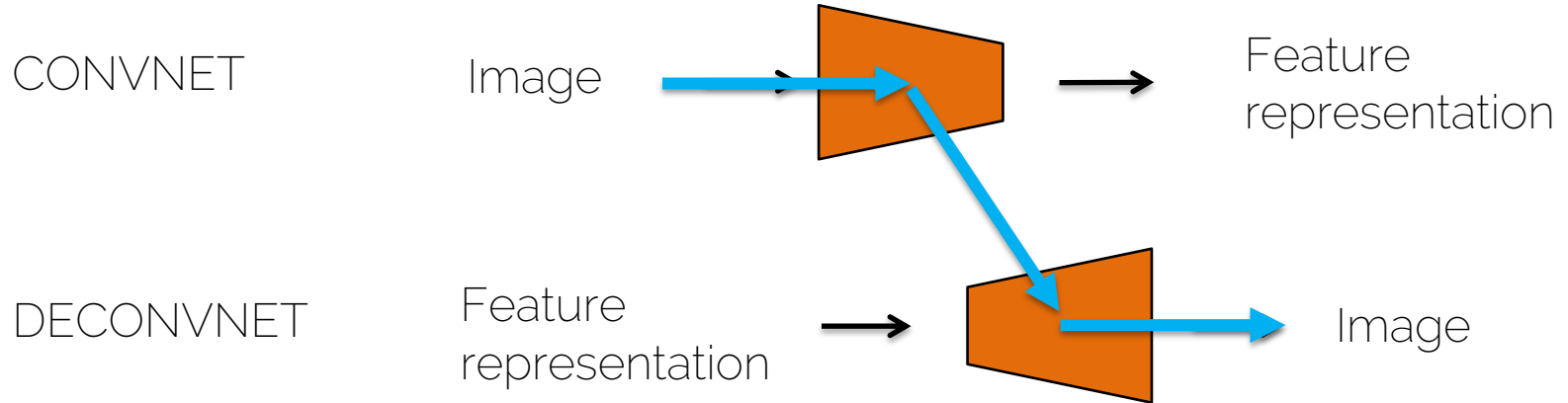
DeconvNet

- Map activations back to the image space



Visualizing features

- Use a DeconvNet to visualize a certain layer

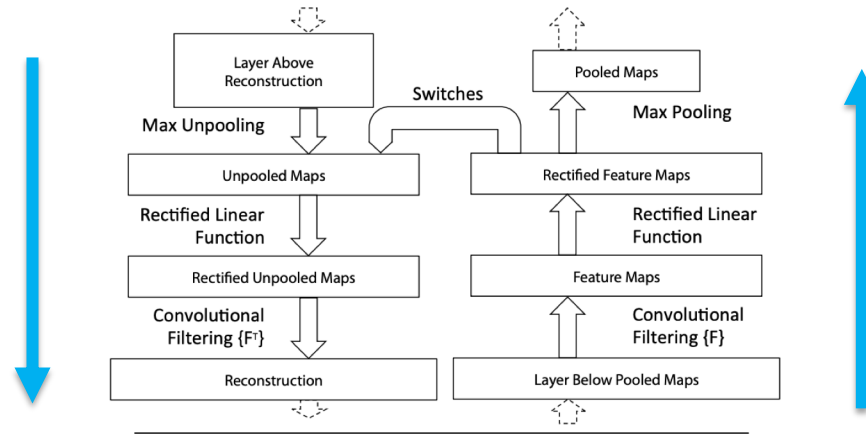


Visualizing features

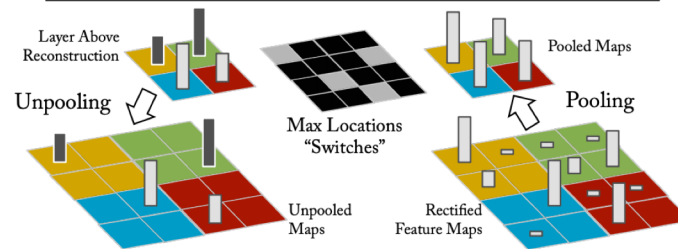
- Choose an input image
- Forward pass through the network.
- Observation: filter 15 of the 3rd layer is highly activated by this image
- Goal: visualize filter 15 of the 3rd layer
- Zero out all other filters
- Pass the layer through the DeconvNet

Visualizing features

DECONVNET



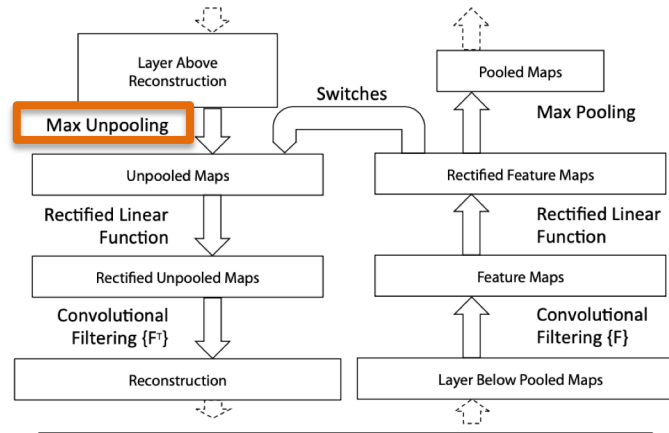
CONVNET



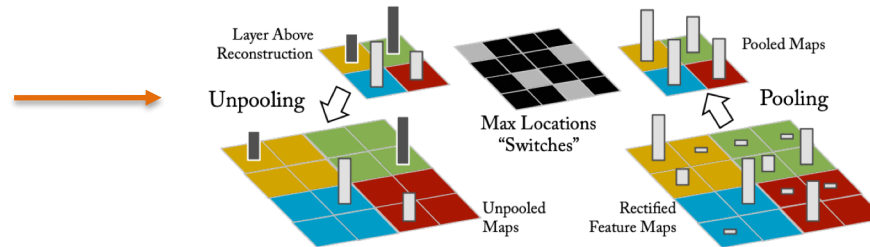
Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing features

- Unpooling



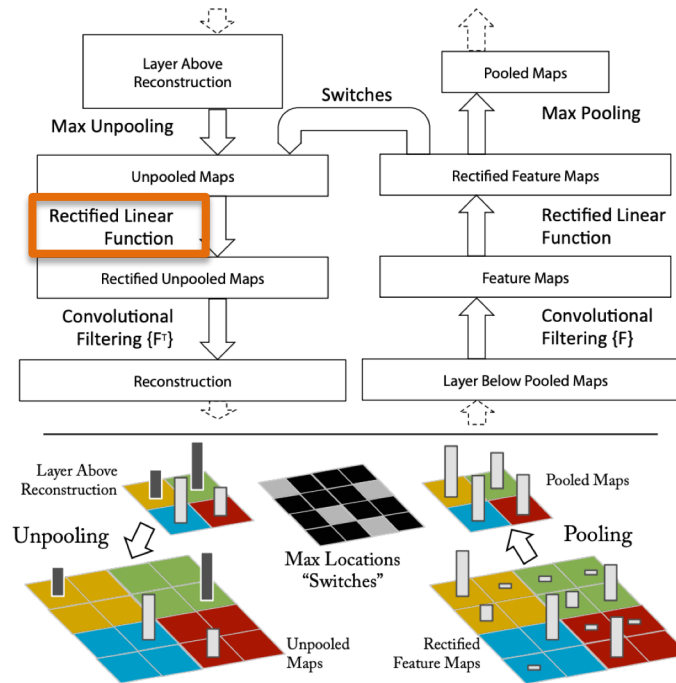
Keep the locations where the max came from



Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing features

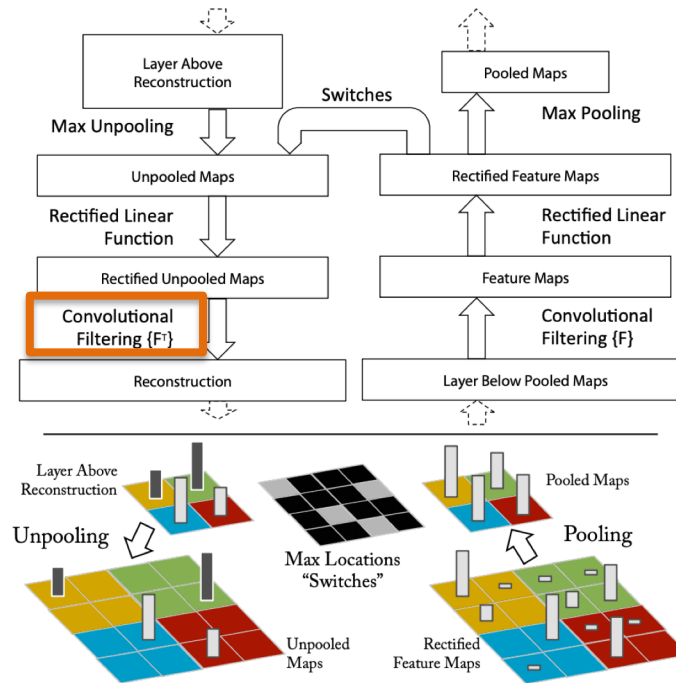
- ReLU: you are still interested in having positive features for visualization



Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing features

- Deconvolution operation
- In practice we convolve with the transpose of the learned filter



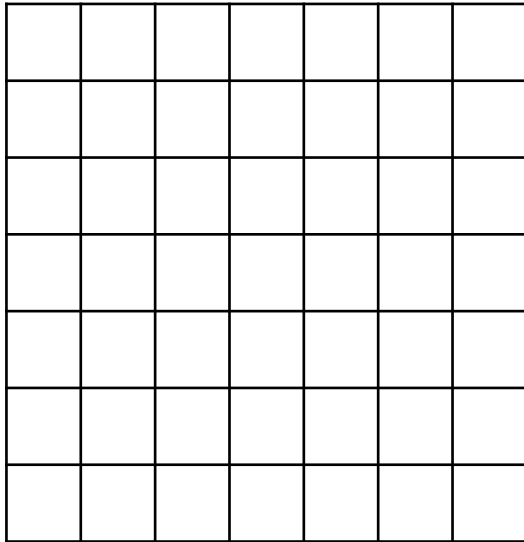
Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Why the transpose?

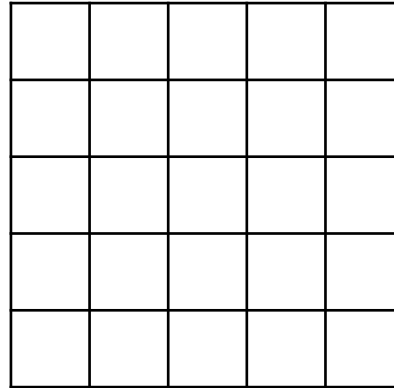
- You want to find out what inputs influenced your outputs and how much
- Blackboard!

Why the transpose?

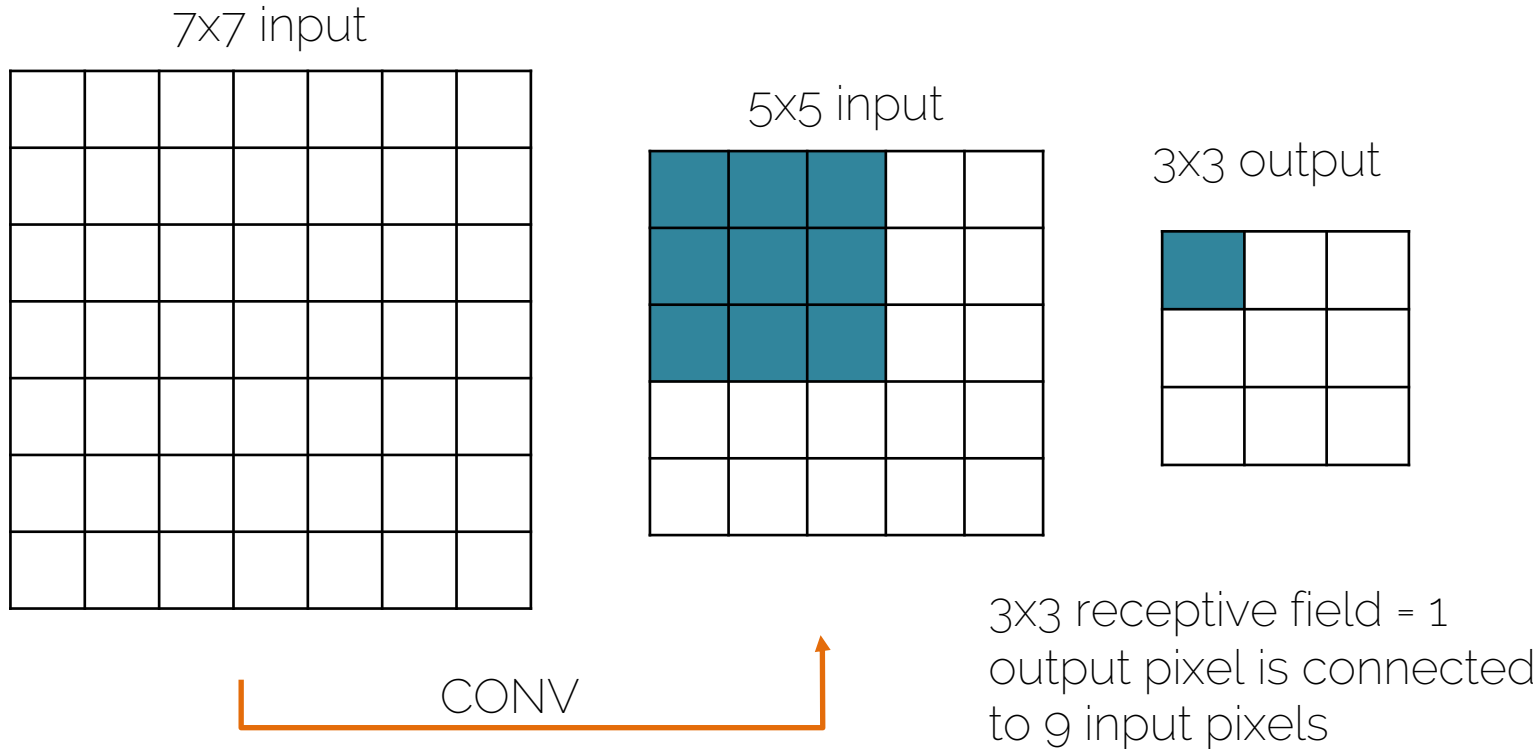
7x7 input



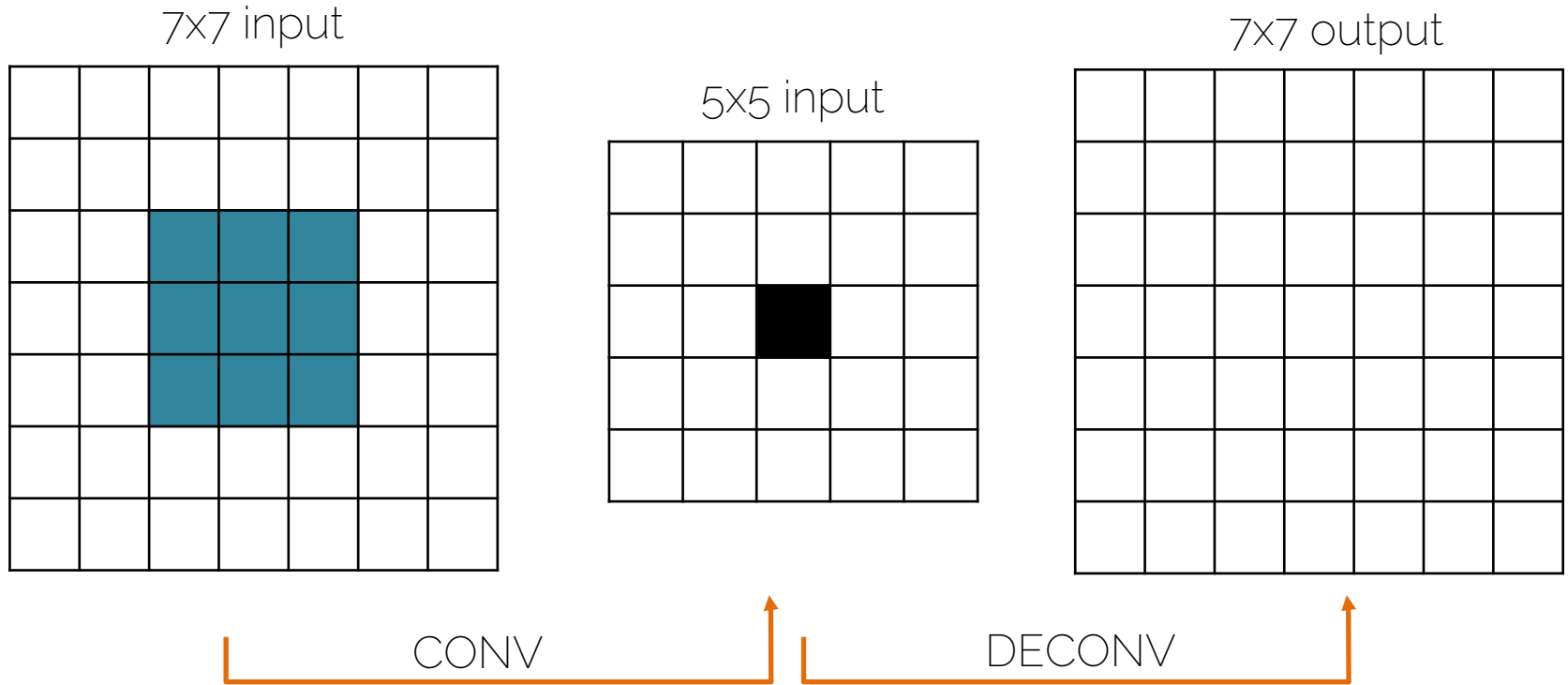
5x5 input



Why the transpose?

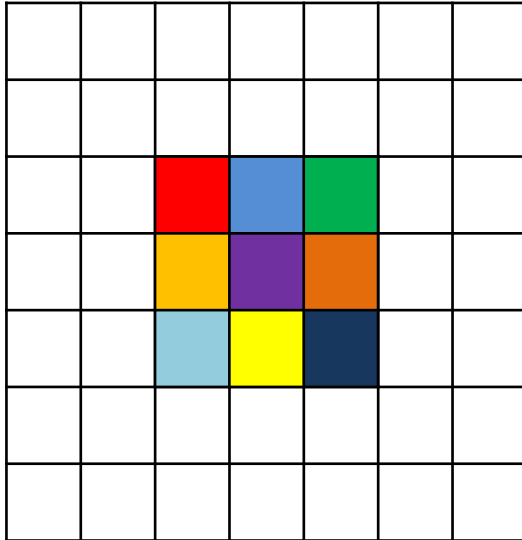


Why the transpose?

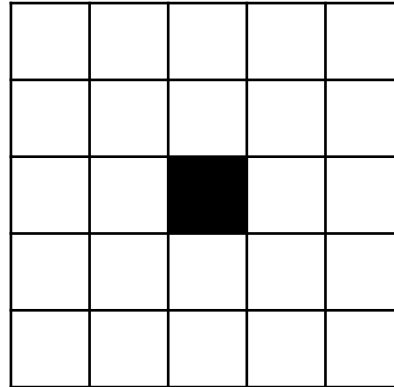


Why the transpose?

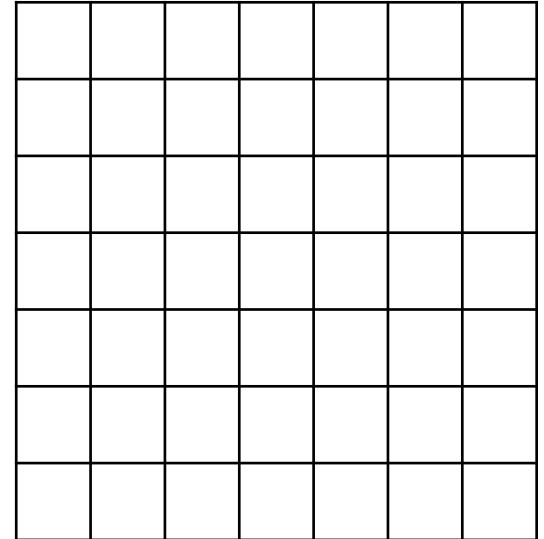
7x7 input



5x5 input

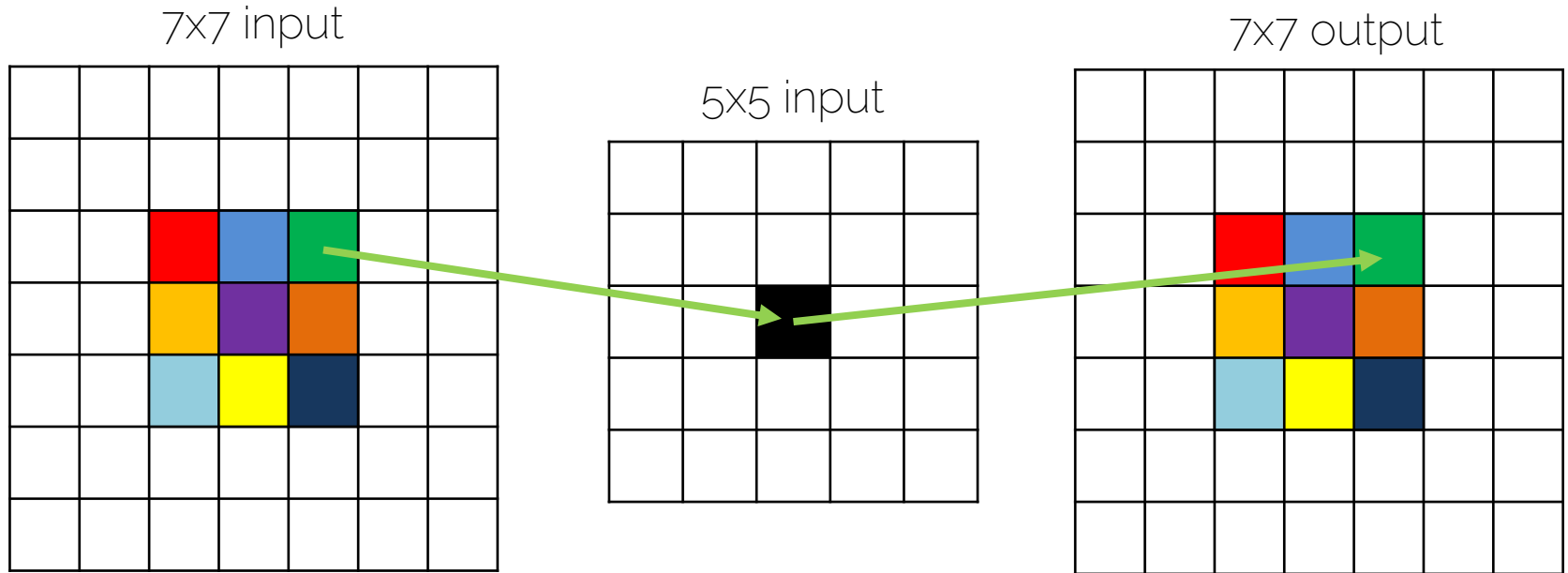


7x7 output



Each input pixel had a different contribution to the black output pixel

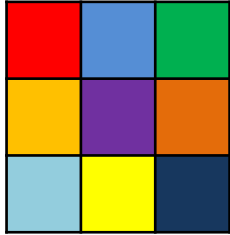
Why the transpose?



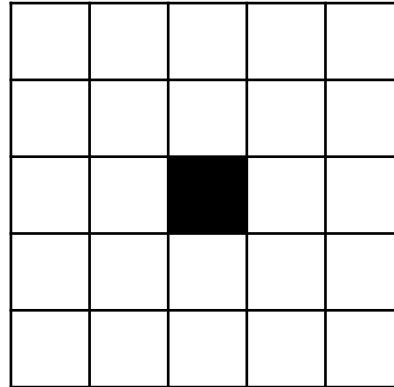
Goal: keep the contribution when we reconstruct the input (the contribution = weights)

Why the transpose?

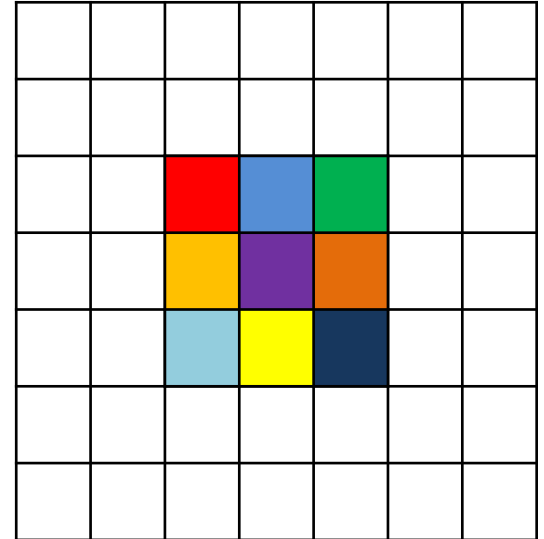
CONV 3x3 kernel



5x5 input



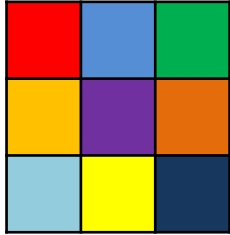
7x7 output



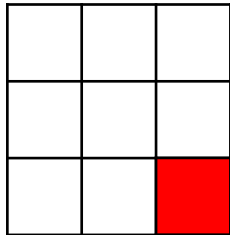
Goal: keep the contribution when we reconstruct the input (the contribution = weights)

Why the transpose?

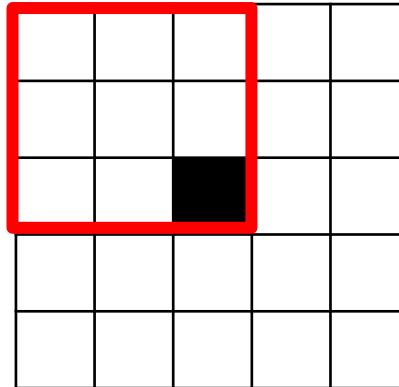
CONV 3x3 kernel



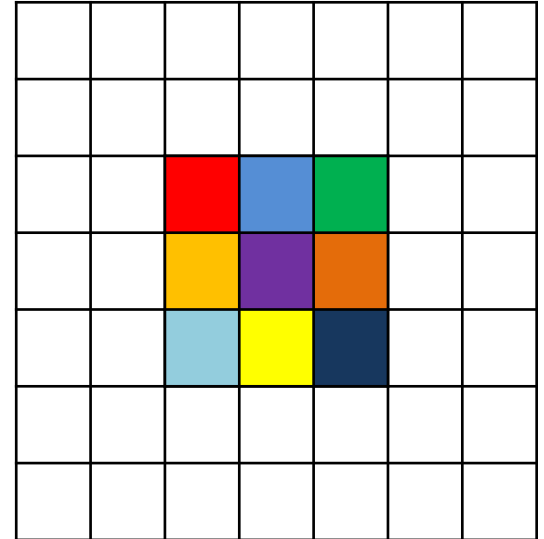
DECONV 3x3 kernel



5x5 input



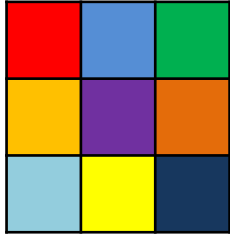
7x7 output



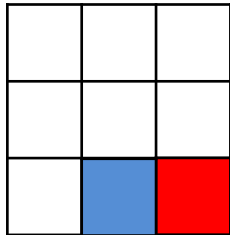
I want to express DECONV still as a convolution operation. To obtain the red pixel, where do I place the filter?

Why the transpose?

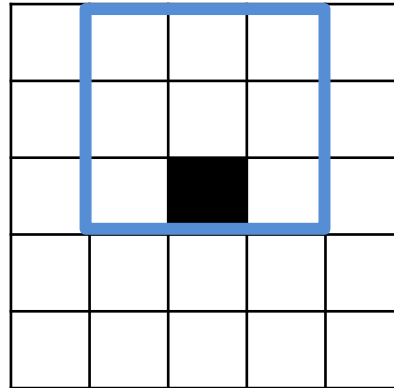
CONV 3x3 kernel



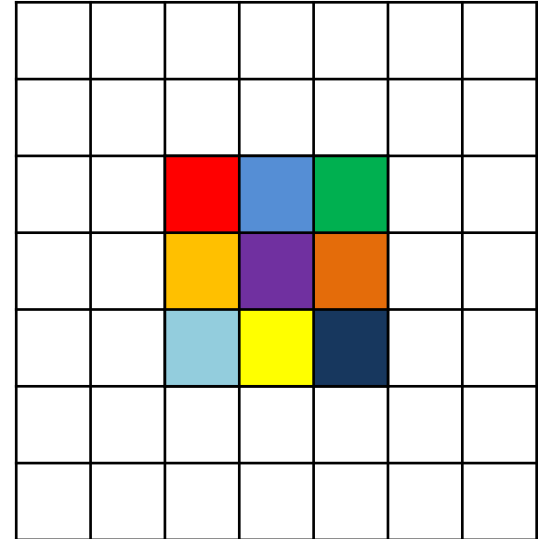
DECONV 3x3 kernel



5x5 input



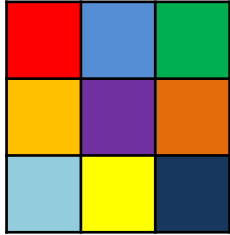
7x7 output



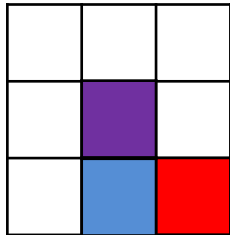
I want to express DECONV still as a convolution operation. To obtain the blue pixel, where do I place the filter?

Why the transpose?

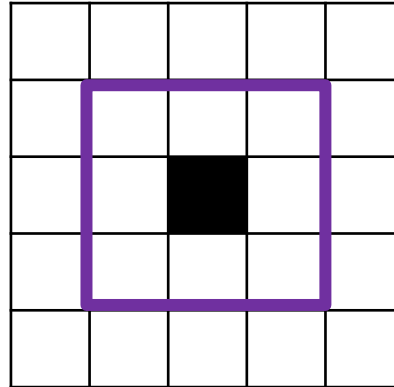
CONV 3x3 kernel



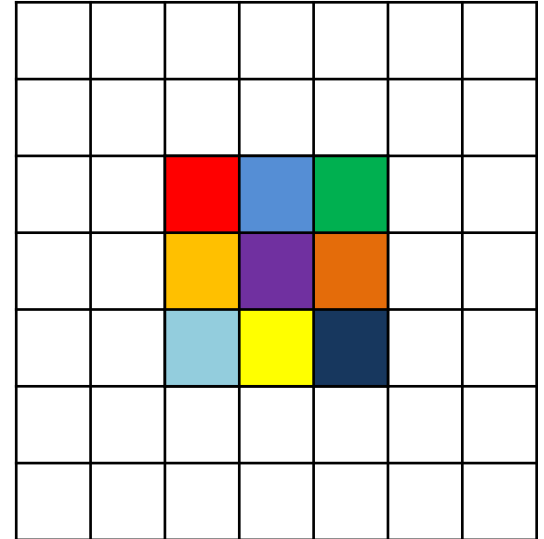
DECONV 3x3 kernel



5x5 input



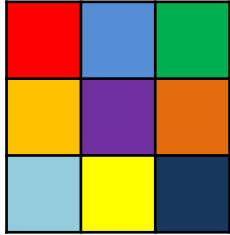
7x7 output



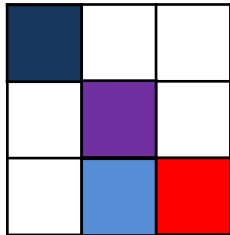
I want to express DECONV still as a convolution operation. To obtain the purple pixel, where do I place the filter?

Why the transpose?

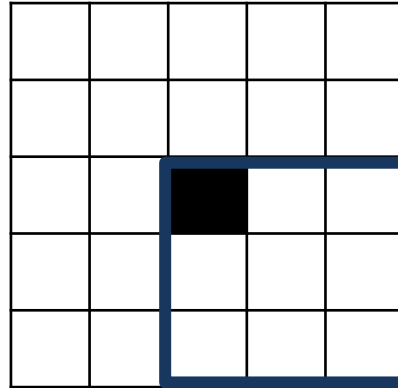
CONV 3x3 kernel



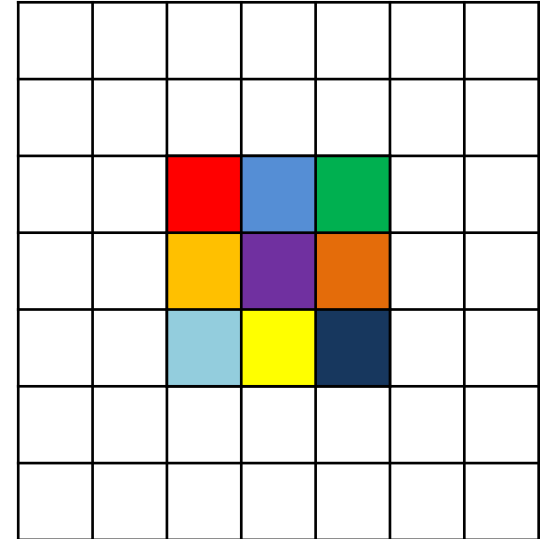
DECONV 3x3 kernel



5x5 input



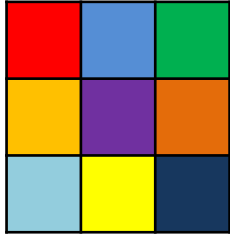
7x7 output



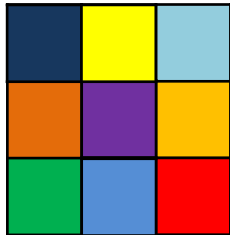
I want to express DECONV still as a convolution operation. To obtain the dark blue pixel, where do I place the filter?

Why the transpose?

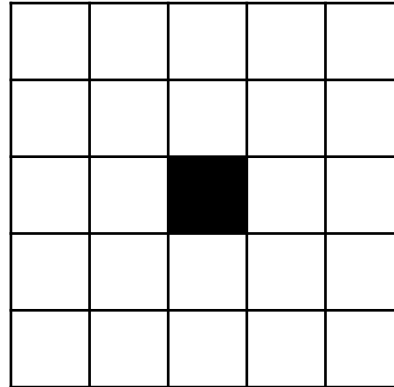
CONV 3x3 kernel



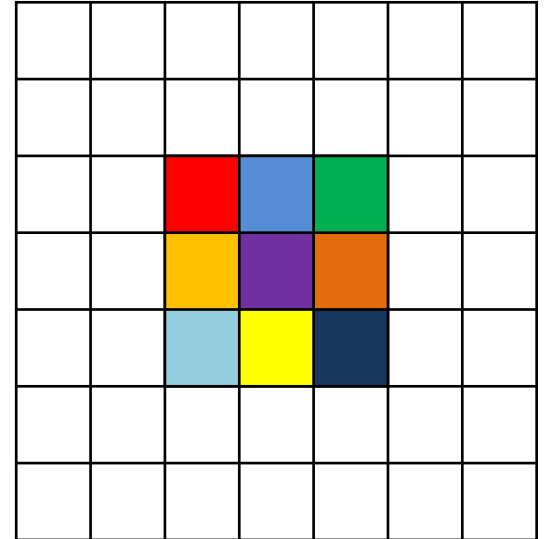
DECONV 3x3 kernel



5x5 input



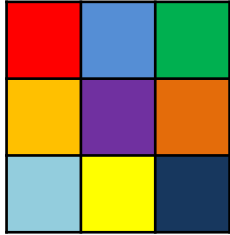
7x7 output



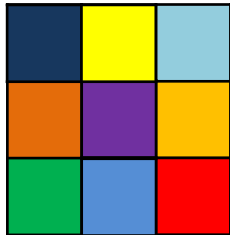
We obtain the transposed filter! We just convolve the 5x5 input with the transposed filter and obtain the “deconvolved” output.

Why the transpose?

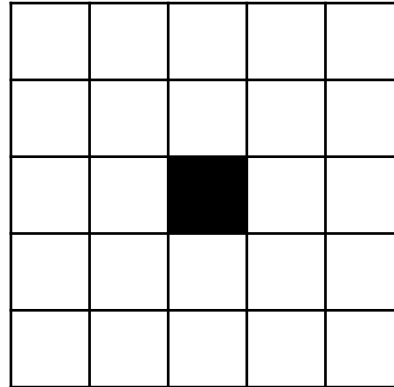
CONV 3x3 kernel



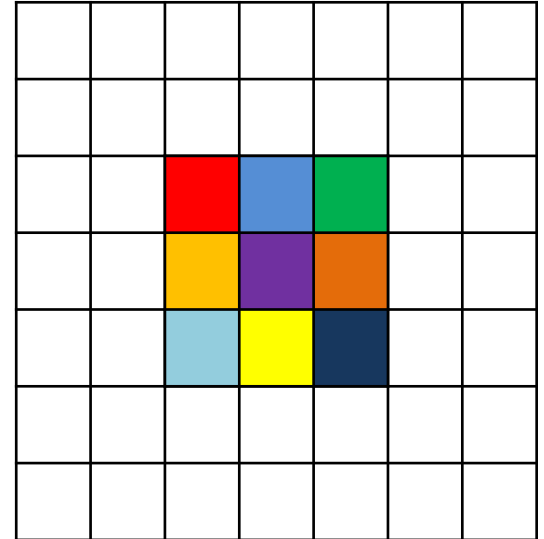
DECONV 3x3 kernel



5x5 input



7x7 output

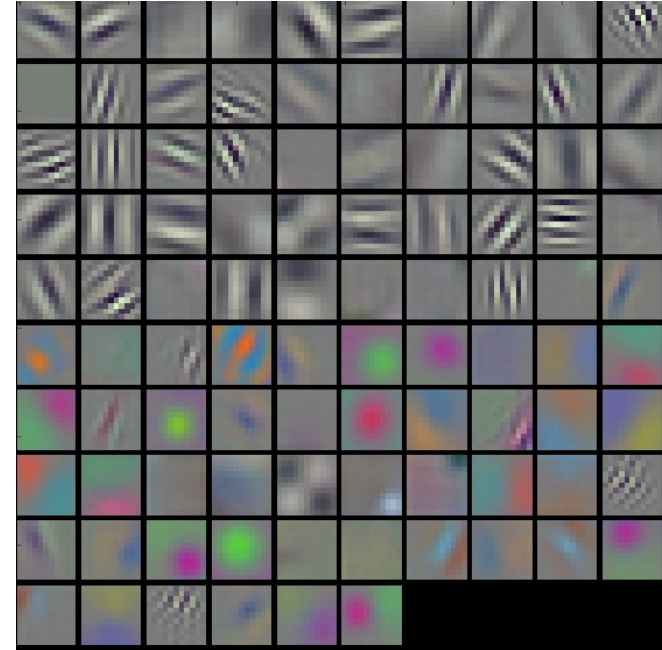
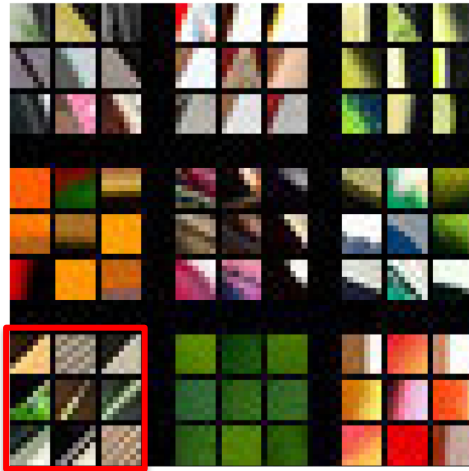


We obtain the transposed filter! We just convolve the 5x5 input with the transposed filter and obtain the "deconvolved" output.

Visualizing features



Layer 1



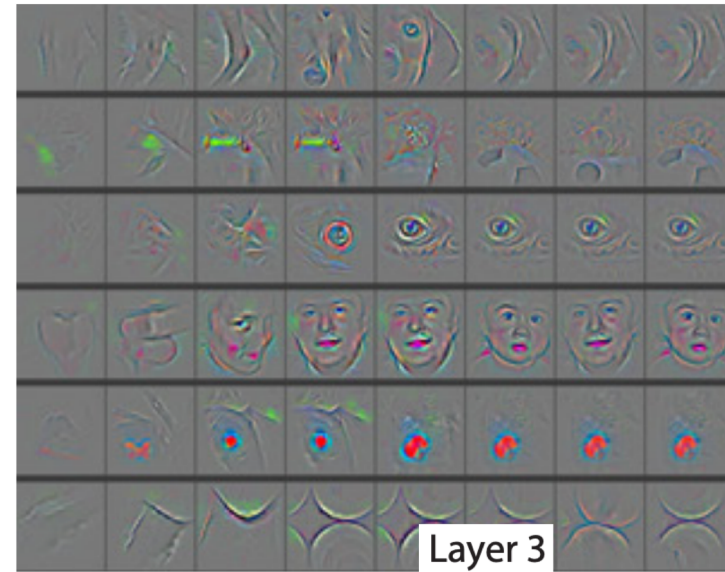
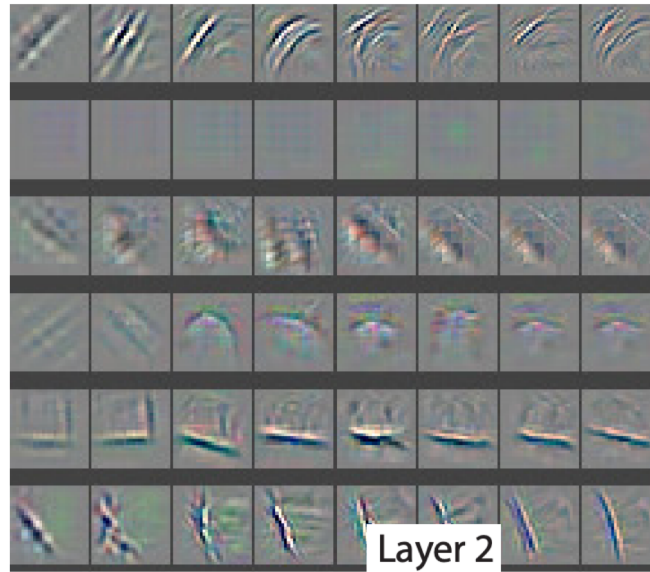
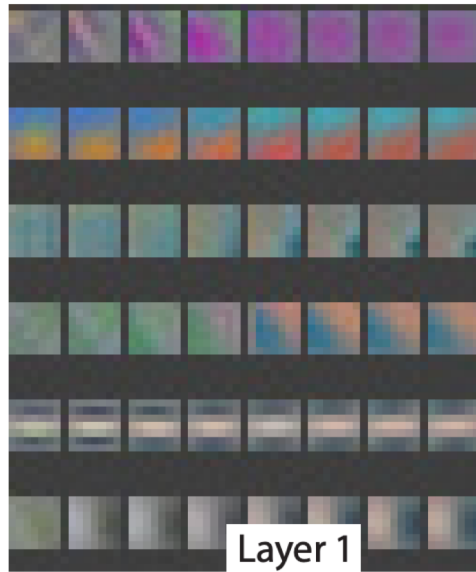
Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Visualizing features: evolution

Epochs

Epochs

Epochs

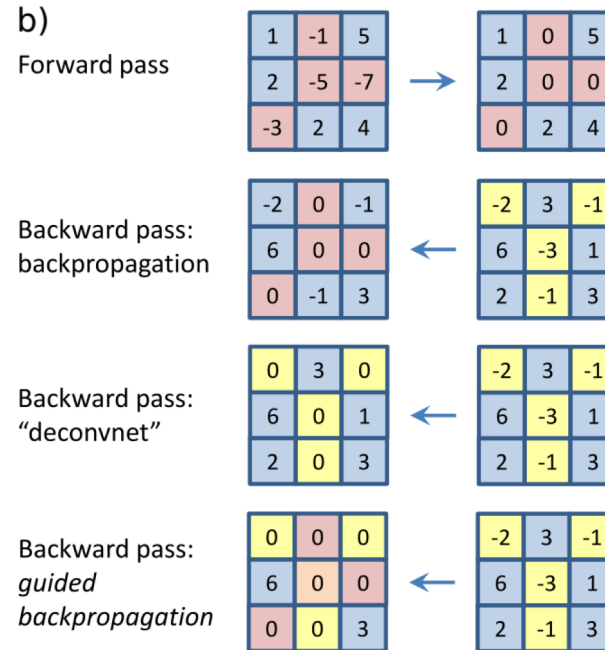


Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

Other ways of inverting ReLU

More info at: Springenberg et al. "Striving for simplicity: the all convolutional net". ICLR Workshop 2015

<https://arxiv.org/pdf/1412.6806.pdf>

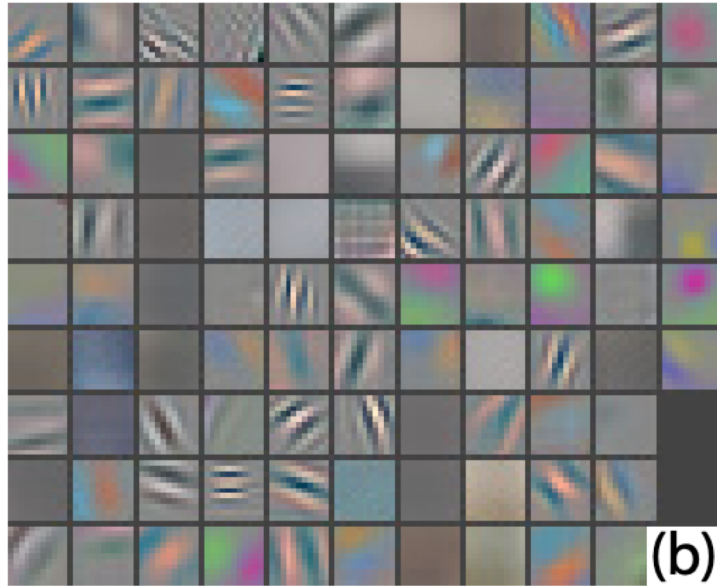


Visualization helps

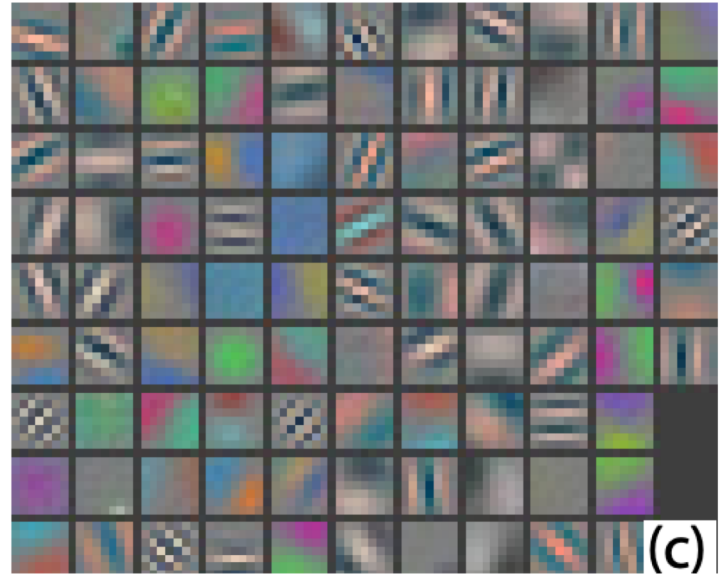
- Observations on AlexNet
- 1. First layer is a mix of low and high frequency information, no mid-frequencies are covered
- Solution: Change from 11x11 to 7x7

Visualization helps

11X11



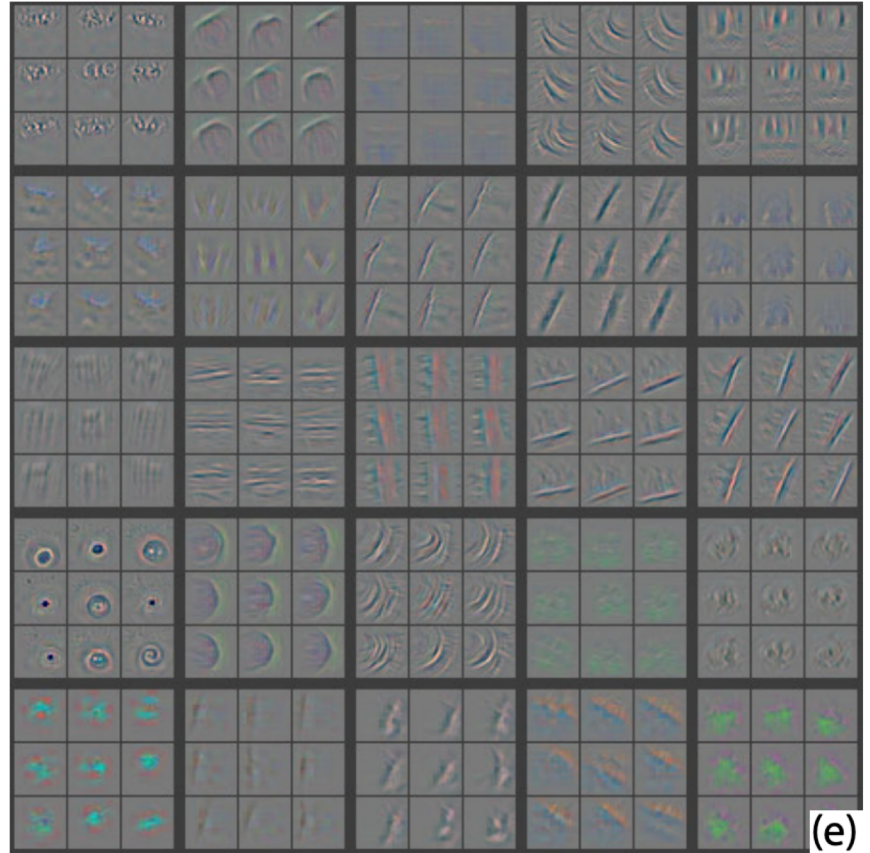
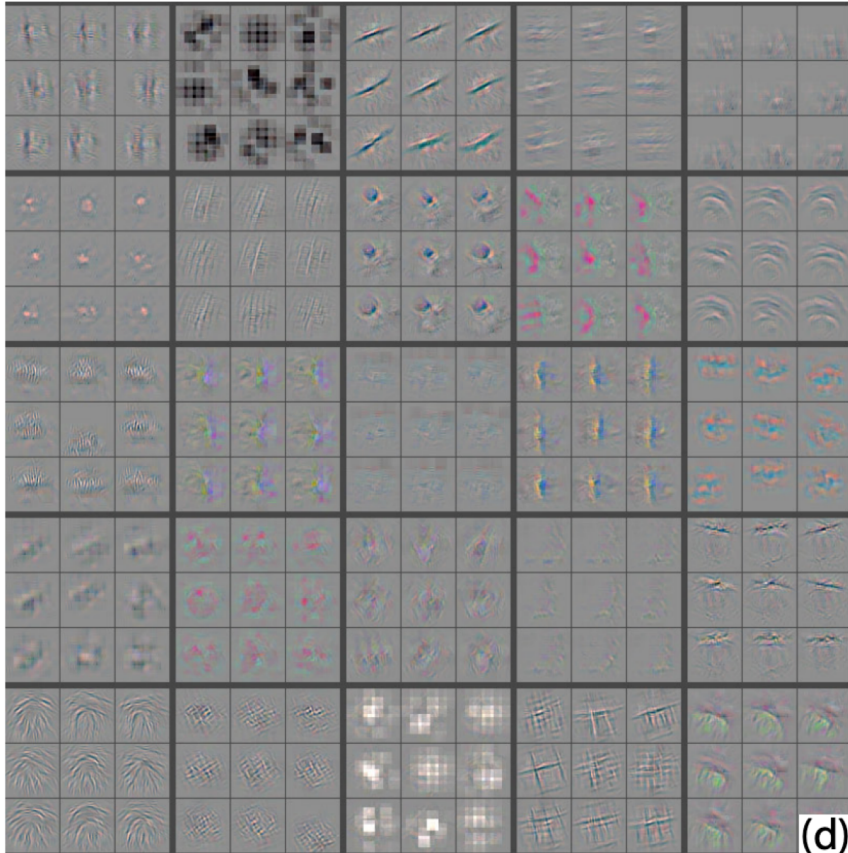
7x7



Visualization helps

- Observations on AlexNet
- 2. Aliasing artifacts in the 2nd layer caused by the large stride
- Solution: stride 4 changed to stride 2

Visualization helps



Visualization helps

- The best part: classification score improved by 2%!
- Actively use visualization to debug your CNNs

Visualizing features

- 1. DeconvNet: finding the parts of the image that the filter responds to
- 2. Gradient ascent: generate a synthetic image that maximally activates a filter

Simonyan et al. „Deep inside convolutional networks: visualizing image classification models and saliency maps“. ICLR Workshop 2014

Visualizing features 2

- Want to find an image that maximizes the score for a particular class

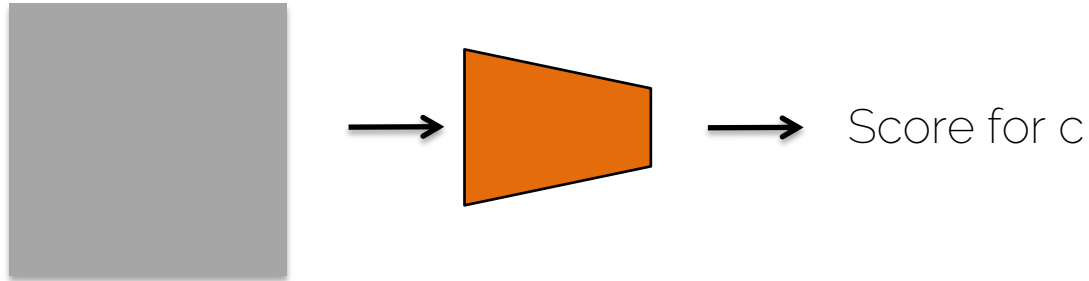
$$\arg \max_I S_c(I) + \lambda \|I\|_2^2$$

- The score is taken before the softmax layer. Direct output of the Fully Connected layer.
- L2 norm to avoid only a few very large pixels

Simonyan et al. „Deep inside convolutional networks: visualizing image classification models and saliency maps“. ICLR Workshop 2014

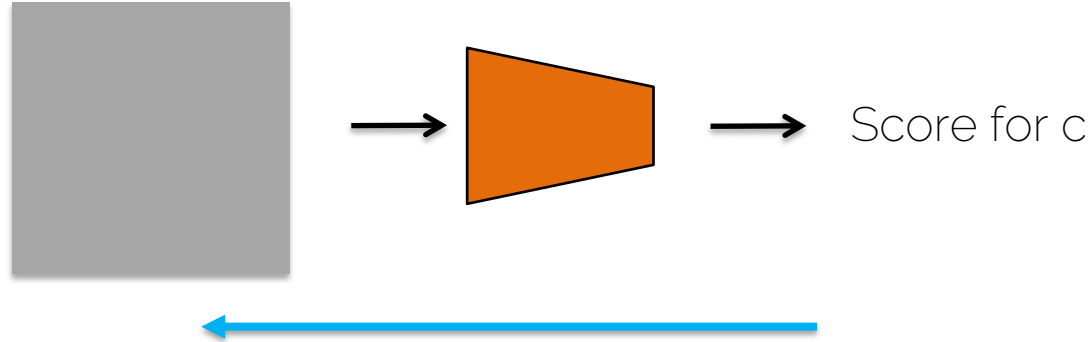
Visualizing features 2

- 1. Get a trained CNN (mean of the training images was subtracted to all images)
- 2. Pass a zero image through the CNN



Visualizing features 2

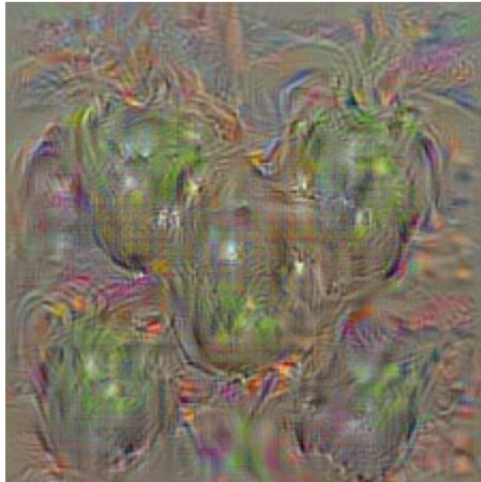
- 3. Maximize the score \rightarrow use gradient ascent and backpropagation



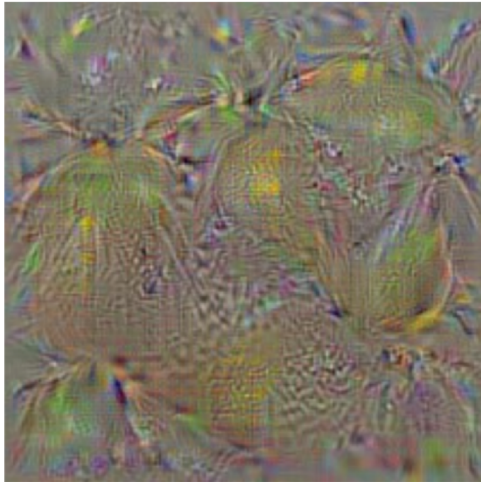
- 4. Make a small update on the image

Visualizing features 2

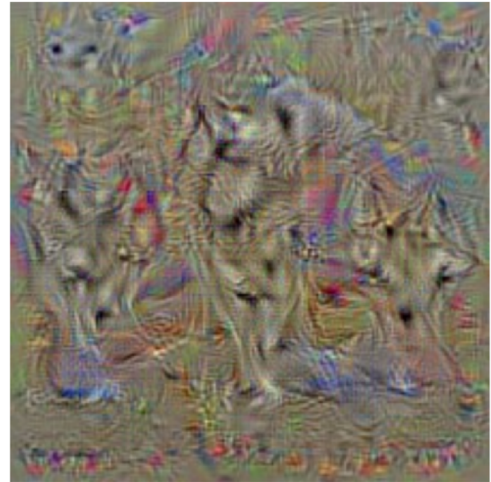
- 5. Repeat
- 6. Visualize by adding the training mean image



bell pepper



lemon



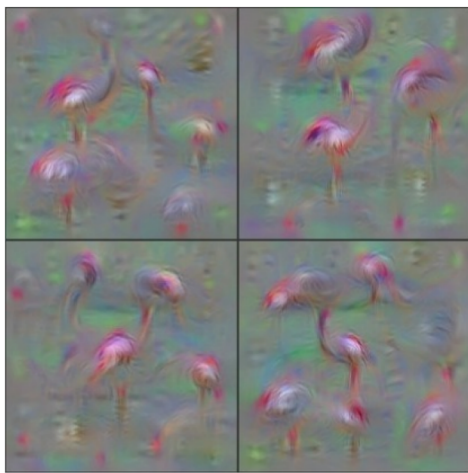
husky

Visualizing features 2.1

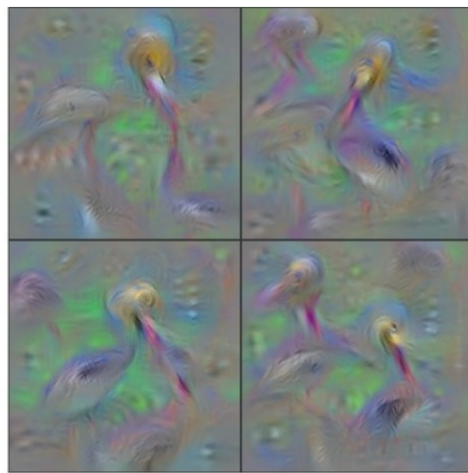
- Improve visualization with a better regularization

$$\arg \max_I S_c(I) + \lambda \|I\|_2^2$$

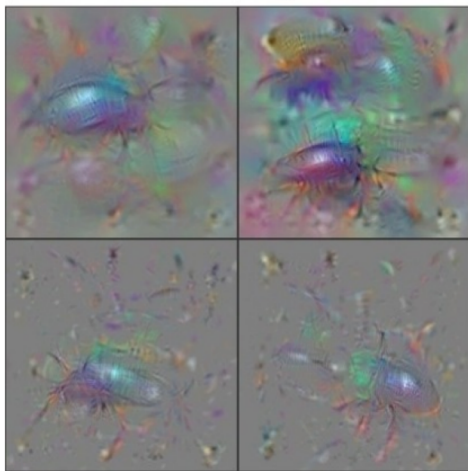
- Propose different regularizations: using a gaussian blur on the image, clipping pixels with small value to 0, clipping gradients with small value to zero



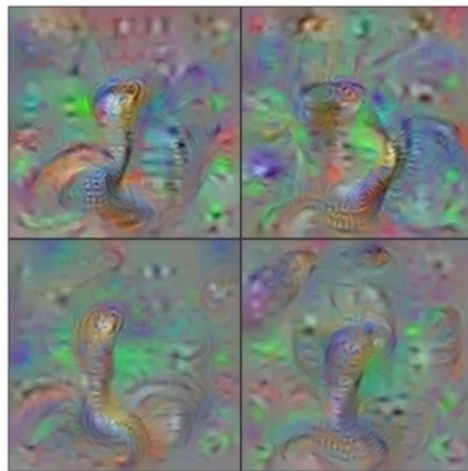
Flamingo



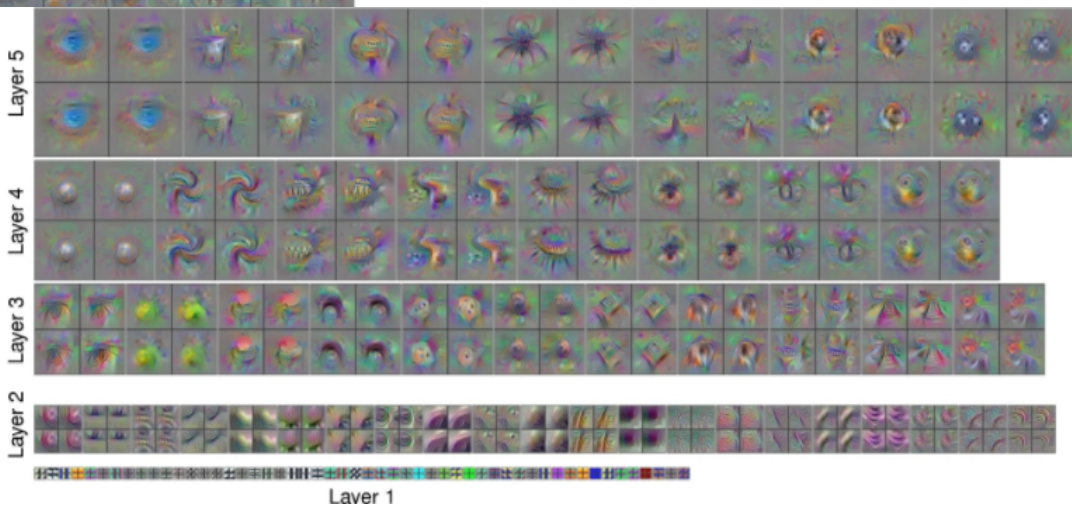
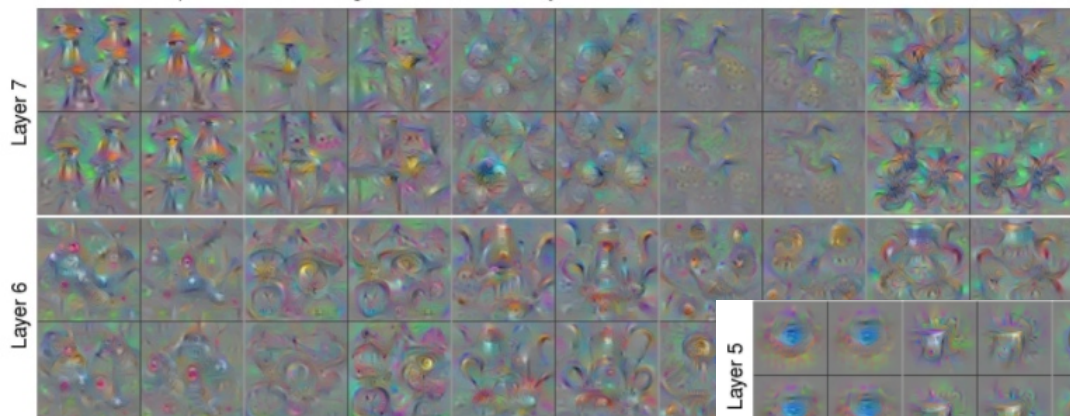
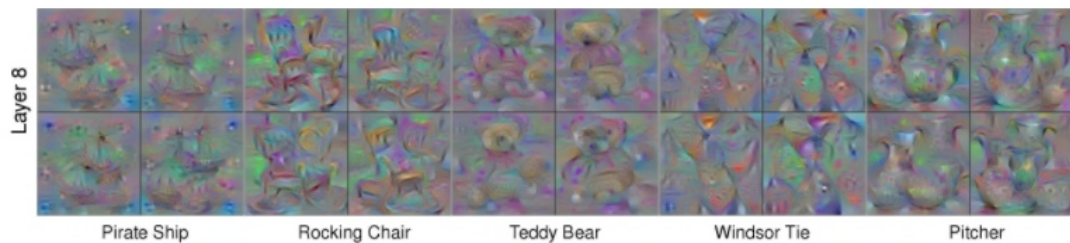
Pelican



Ground Beetle



Indian Cobra



You can also visualize the features
 DeepVis [Yosinski et al. 15]
<http://yosinski.com/deepvis>

DeepDream

- Until now: Synthesize an image to maximize a specific feature
- Now: Amplify the feature activations at some layer in the network

DeepDream

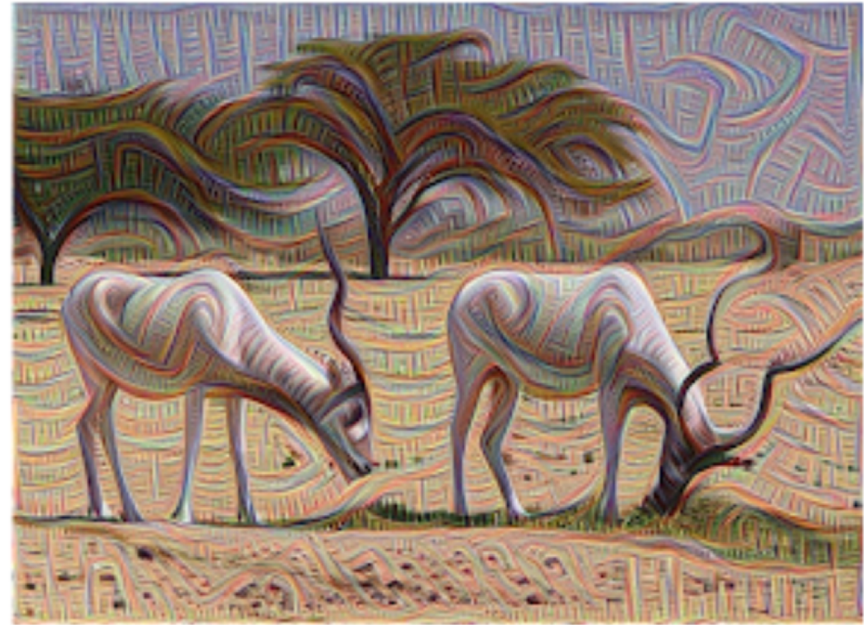
- 1. Feed an image to a network
- 2. Choose a layer and ask the network to enhance whatever was detected → If you see dogs, show me more dogs!

DeepDream

- 1. Forward pass of the image up to layer L
- 2. Set the gradient of the layer = activations
 - Large activations for the *dog* filter will create large gradients
 - The image will be changed to „show more dogs“
- 3. Backpropagate
- 4. Update the image

DeepDream

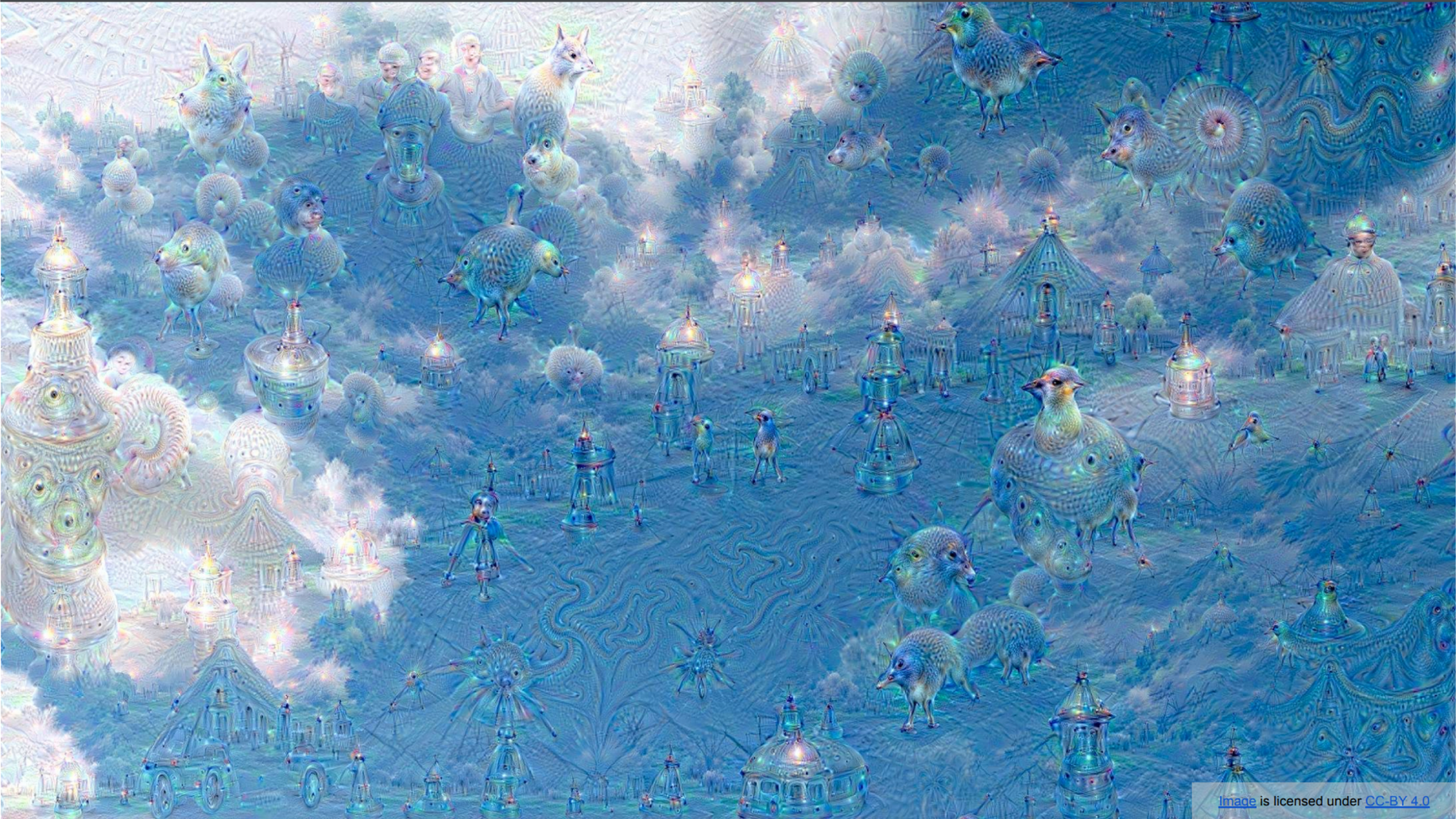
- Low layers: basic features



DeepDream

- Deep layers: we start to see whole objects



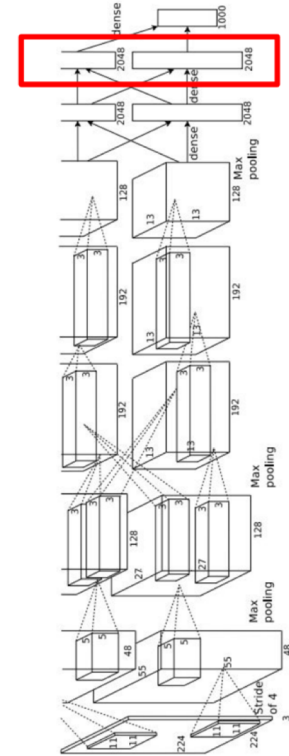




t-SNE

Intuition

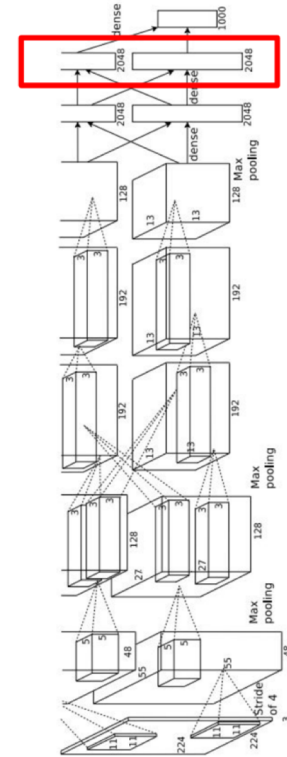
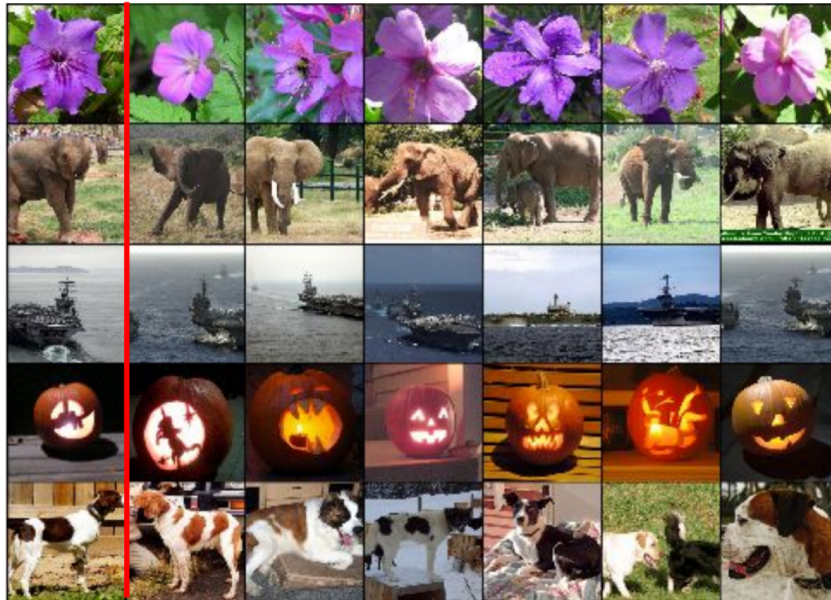
- We want to visualize the last FC layer of AlexNet which dimension 4096
- We do a forward pass of all the images and get their 4096 representations



Intuition

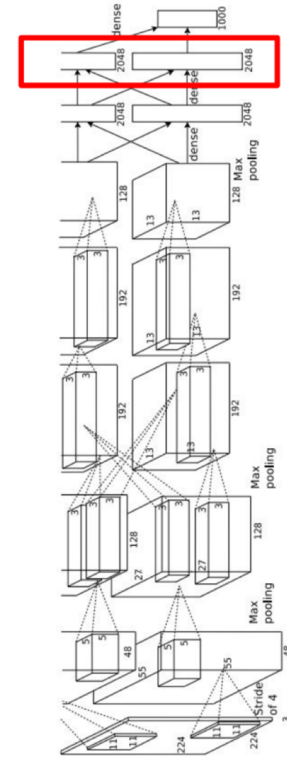
- Nearest neighbor visualization

Test image L2 Nearest neighbors in feature space

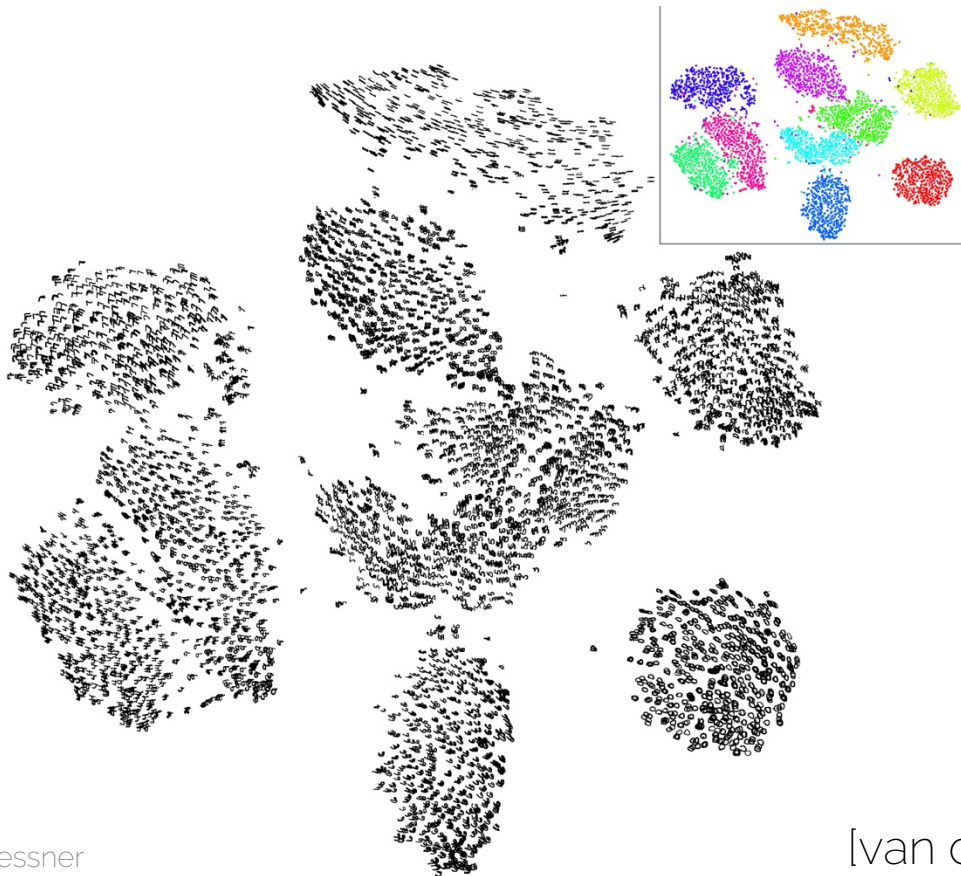


Intuition

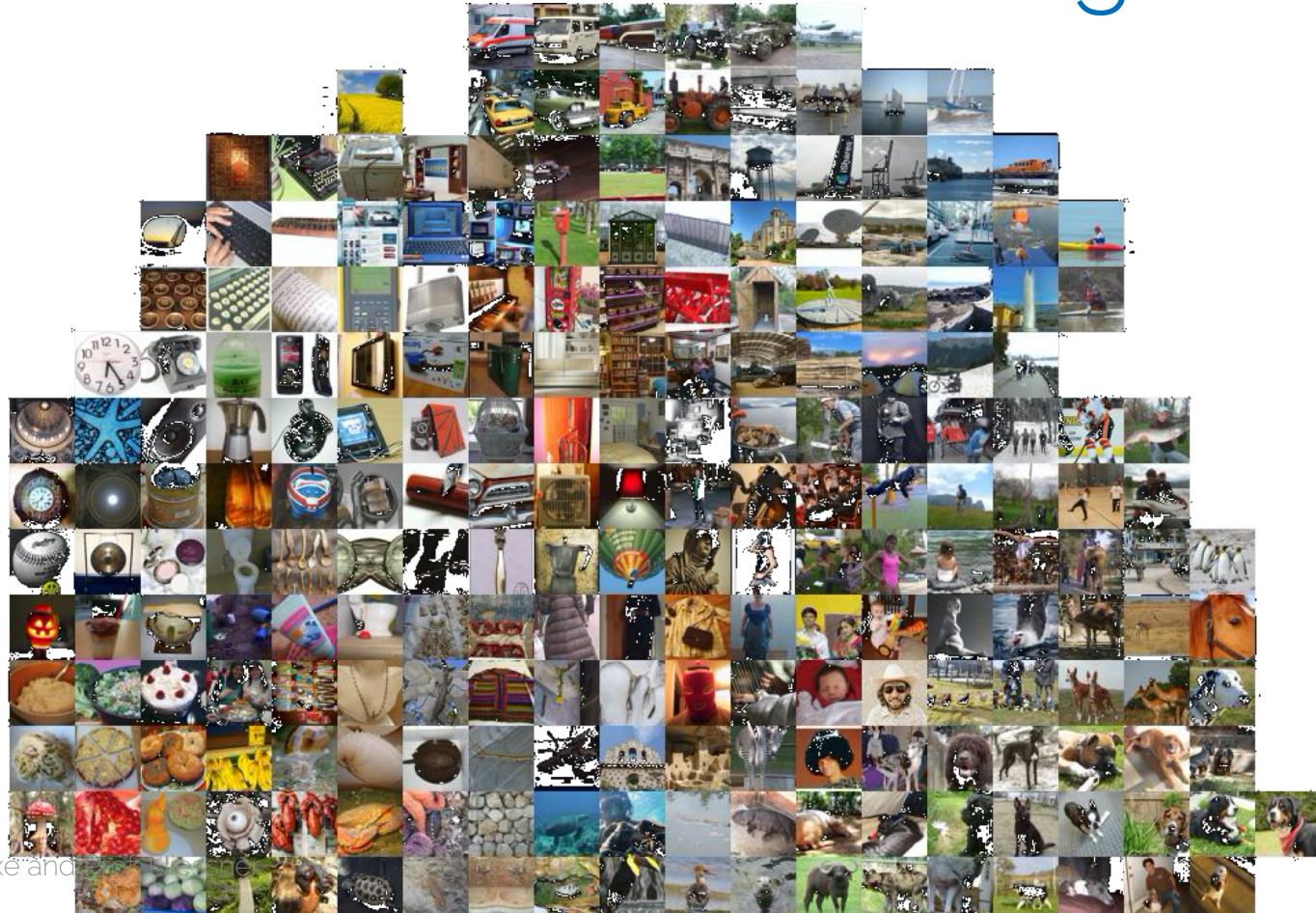
- How can I visualize these clusters in feature space?
- Map high-dimensional embedding to 2D map which preserves the pairwise distance of the points
- This mapping is done by t-SNE



t-SNE Visualization: MNIST



t-SNE Visualization: ImageNet



t-SNE Visualization: ShapeNet



When is t-SNE worth using?

- You can use it to debug your network
- Good for visualizing the clusters created by a Siamese network

More visualizations

- Saliency visualization: Simonyan et al. „Deep inside convolutional networks: visualizing image classification models and saliency maps“. ICLR Workshop 2014
- [Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization](#)
Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, Dhruv Batra

Next lecture

- Next lecture on Monday - Autoencoders
- Make sure you are working on your projects!