CS60010: Deep Learning

Spring 2023

Sudeshna Sarkar

CNN

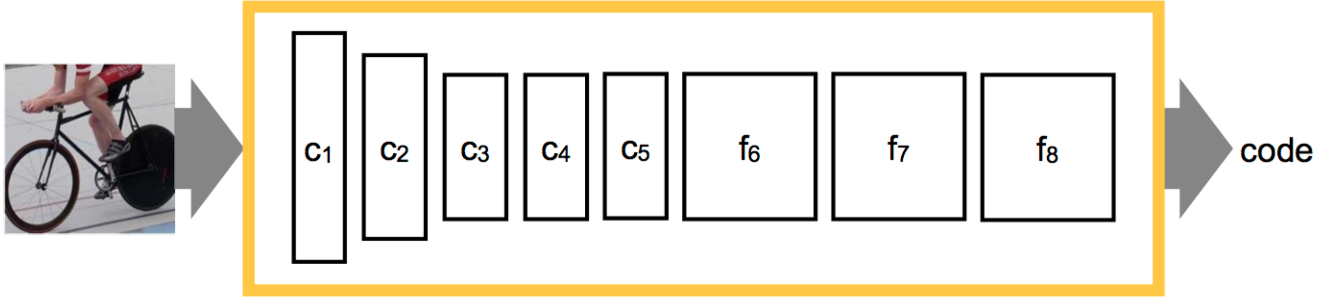
Part 4

10 Feb 2023

Features

CNN Features are Generic

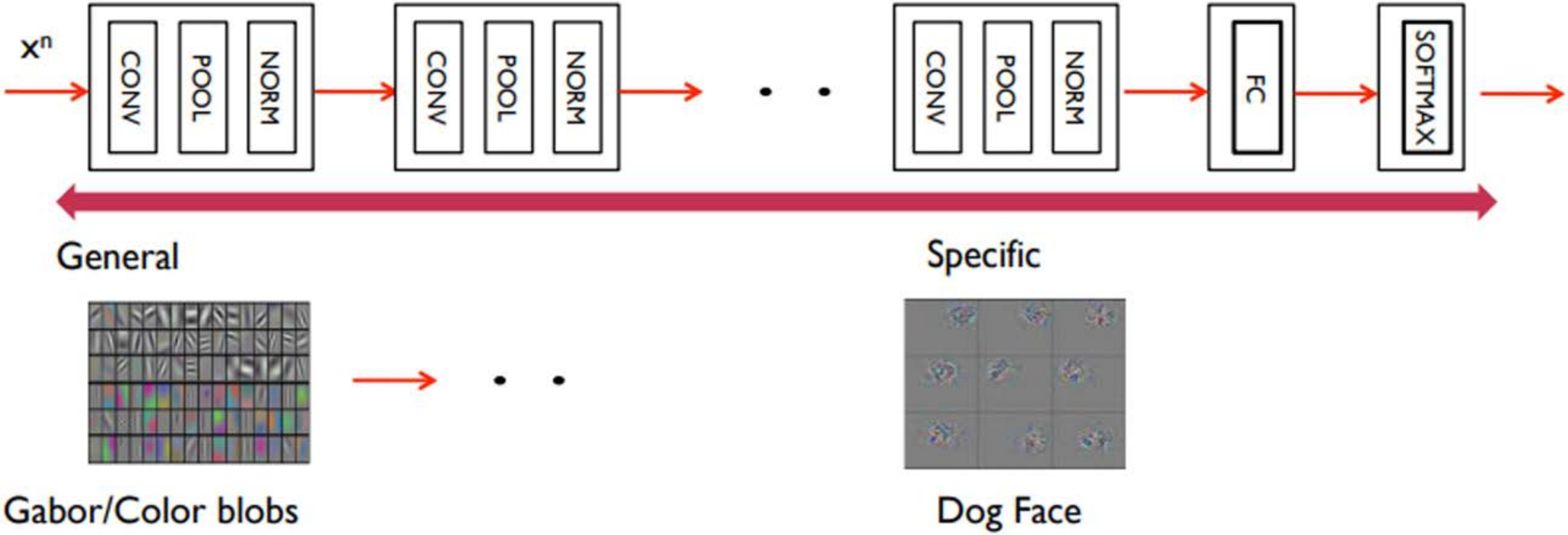
1. Train the CNN (deep network) on a very large database such as imagenet.
2. Reuse CNN to solve other problems
   1. Remove the last layer (classification layer)
   2. Output is the code/feature representation



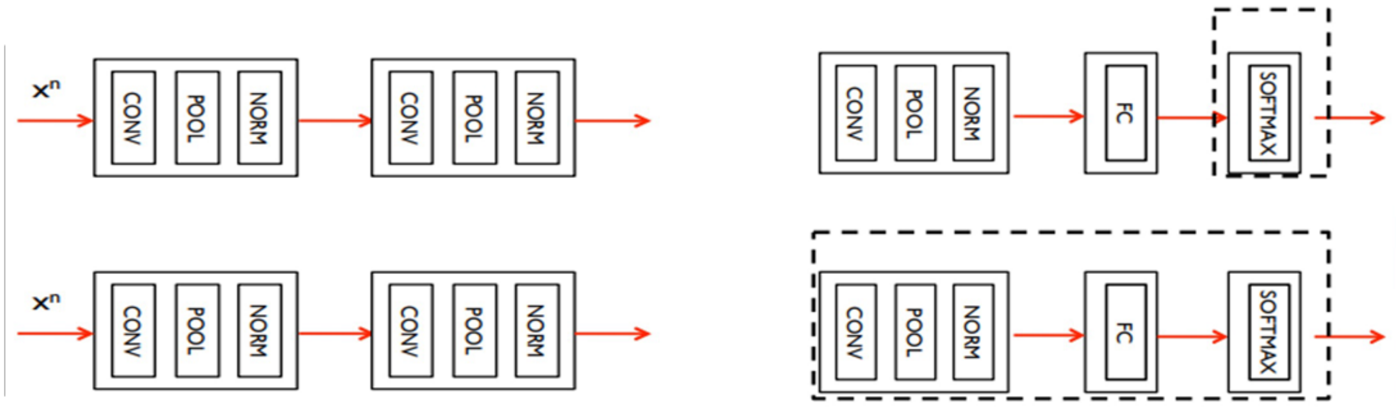
New Settings

- Extend to more classes
  - Extend from 1000 classes (say people) to another new 100
- Extend to new tasks
  - Extend from object classification to scene classification
- Extend to new data sets
  - Extend from imageNet to PASCAL (SLR to webcams)
- When we have a lesser amount of data.

Transfer Learning



Transfer Learning



Notes

If dataset is small retrain the softmax

If dataset is reasonable retrain larger portion with fine tuning of initial layers

Initializing a network with transferred features almost always gives better generalization

Interpreting and understanding a trained model

Plotting model architecture

model.summary()

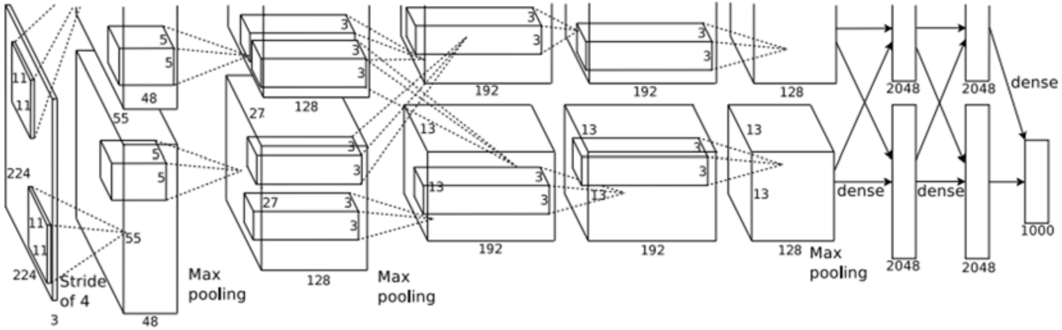
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
preds (Dense)                (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
```

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

- Slides are taken from CS231n at Stanford

What's going on inside ConvNets?

Input Image: 3 x 224 x 224



Class Scores: 1000 numbers

What are the intermediate features looking for?

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

Visualization and Understanding

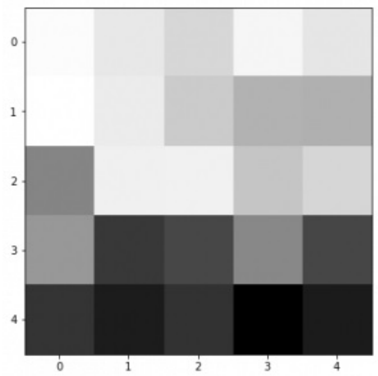
- Visualizing what models have learned:
    - Visualizing filters
    - Visualizing final layer features
    - Visualizing activations
- Understanding input pixels
    - Identifying important pixels
    - Saliency via backprop
    - Guided backprop to generate images
    - Gradient ascent to visualize features

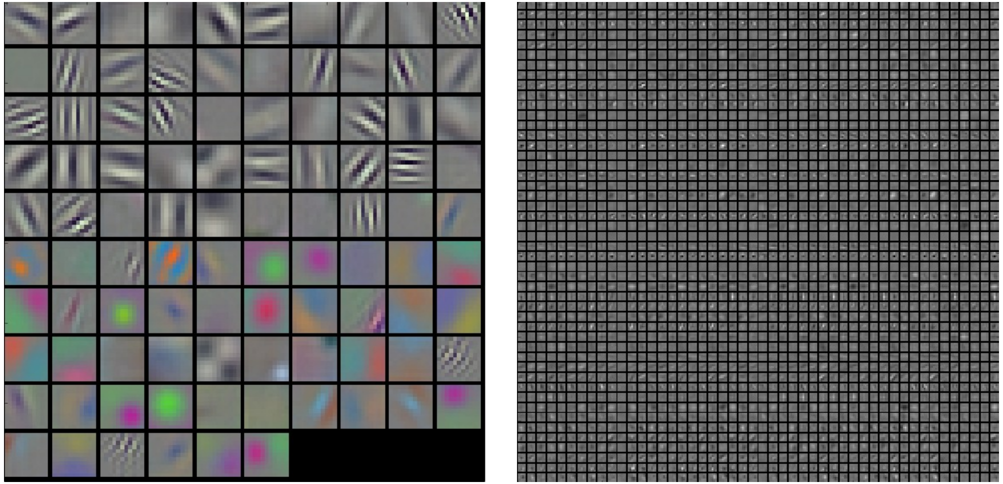
Visualize Filters

- Plot the filters of a trained model.
- For example, the first filter of the first layer looks like:

```
top_layer = model.layers[0]
plt.imshow(top_layer.get_weights()[0][:, :, :, 0].squeeze(), cmap='gray’)
```

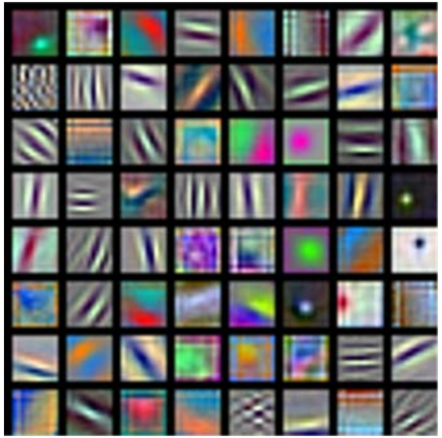
These are usually most interpretable on the first CONV layer.



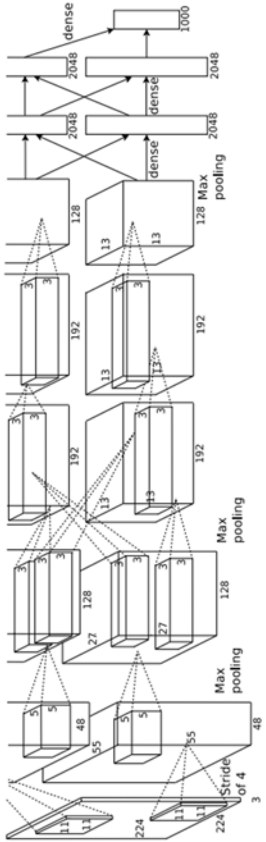


Typical-looking filters on the first CONV layer (left), and the 2nd CONV layer (right) of a trained AlexNet.

First Layer: Visualize Filters

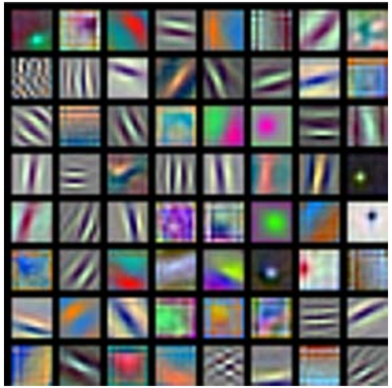


AlexNet: 64
x 3 x 11 x 11

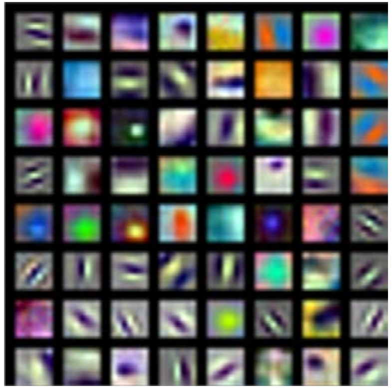


Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014 He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

First Layer: Visualize Filters

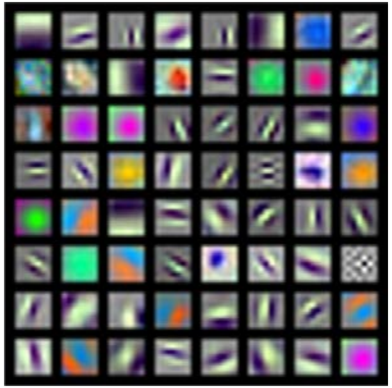


AlexNet: 64 x 3 x 11 x 11



ResNet-18: 64 x 3 x 7 x 7

ResNet-101: 64 x 3 x 7 x 7

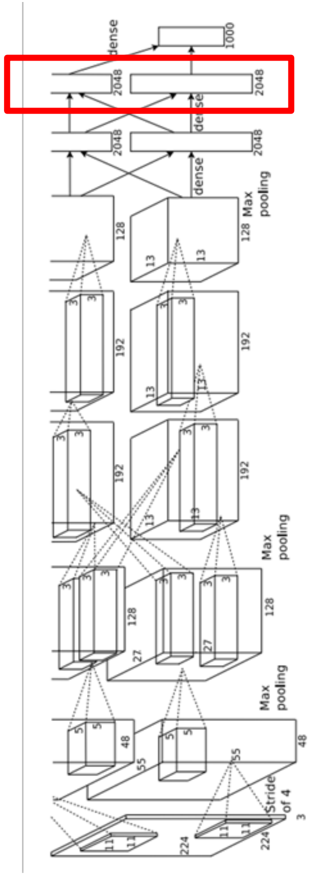


DenseNet-121: 64 x 3 x 7 x 7

Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014 He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Last Layer

FC7 layer

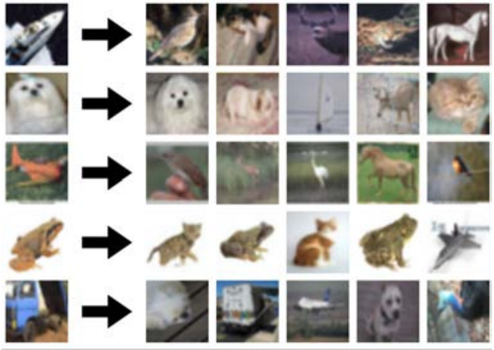


4096-dimensional feature vector for an image
(layer immediately before the classifier)

Run the network on many images, collect the feature vectors

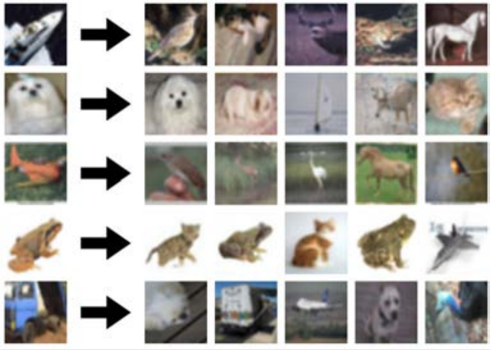
Last Layer: Nearest Neighbors

Recall: Nearest neighbors in pixel space



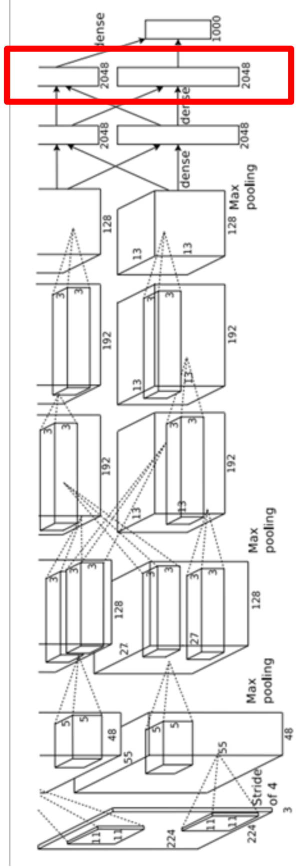
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

Last Layer: Nearest Neighbors



Test image

L2 Nearest neighbors in feature space



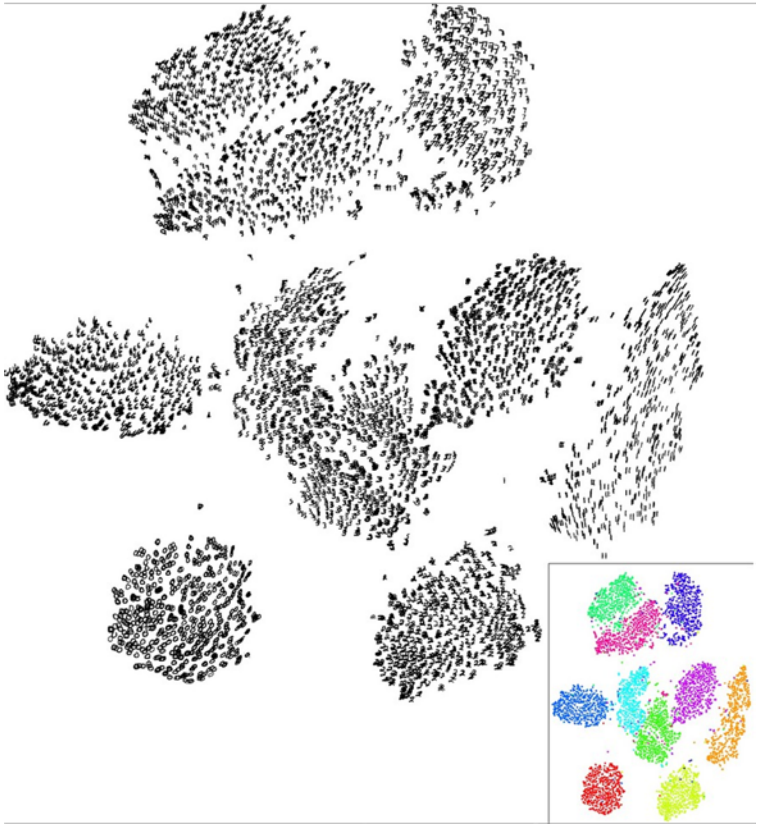
Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

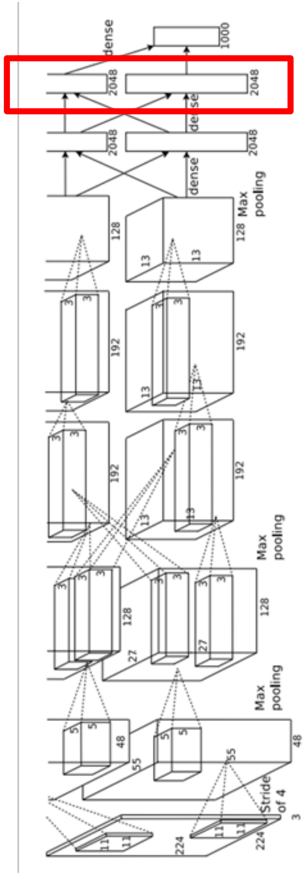
Last Layer: Dimensionality Reduction

Visualize the "space" of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principal Component Analysis (PCA)

More complex: **t-SNE**





Van der Maaten and Hinton, "Visualizing Data using t-SNE", JMLR 2008

Visualizing the activations

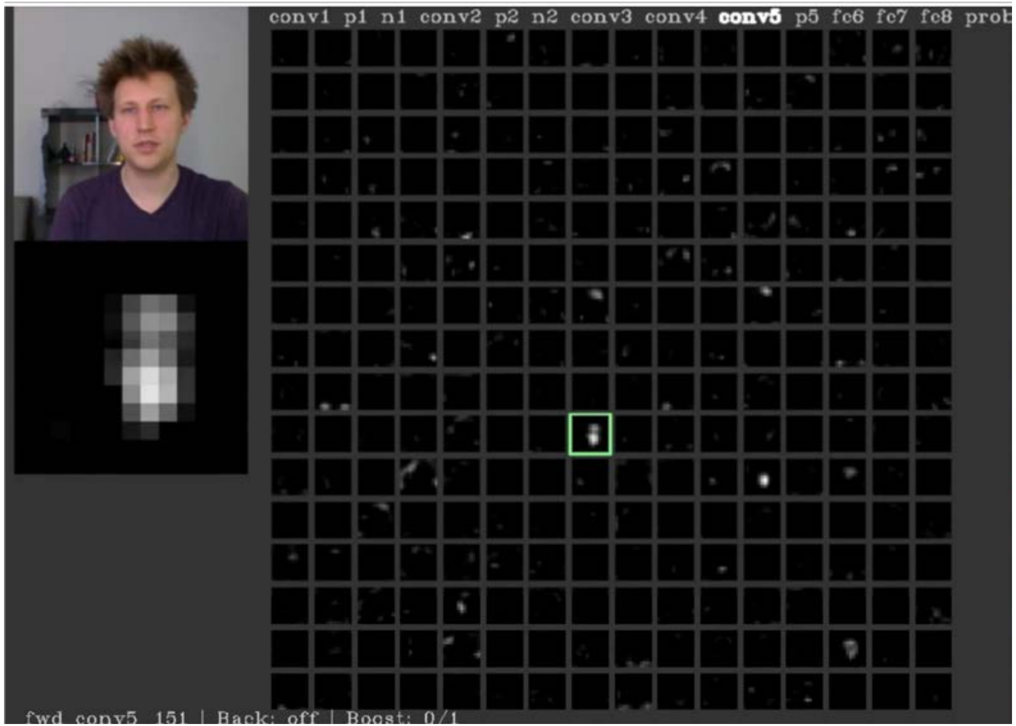
Activation Maps: Maximal Activations

- To see what our neural network is doing, we can apply the filters over an input image and then plot the output.
- This allows us to understand what sort of input patterns activate a particular filter. For example, there could be a face filter that activates when it gets the presence of a face in the image.

Visualizing Activations

conv5 feature map is 128x13x13;

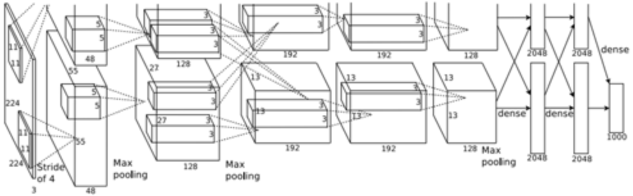
- visualize as 128 13x13 grayscale images



Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

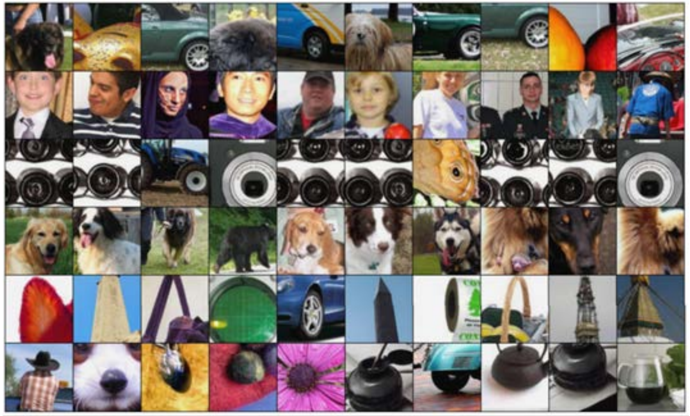
Maximally Activating Patches



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

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

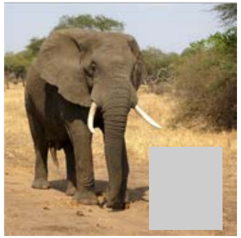
Visualize image patches that correspond to maximal activations

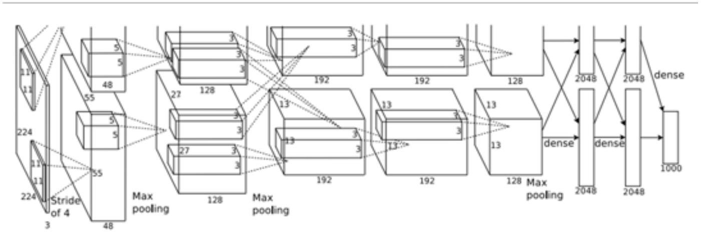


Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015

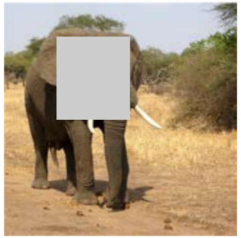
Which pixels matter: Saliency via Occlusion

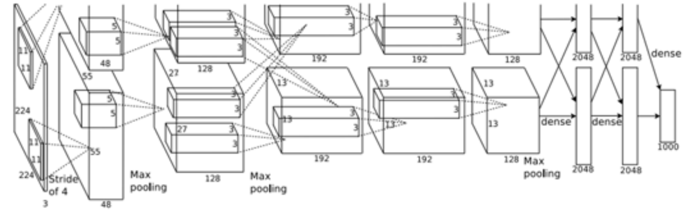
Mask part of the image before feeding to CNN, check how much predicted probabilities change





P(elephant) = 0.95



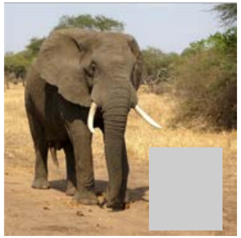


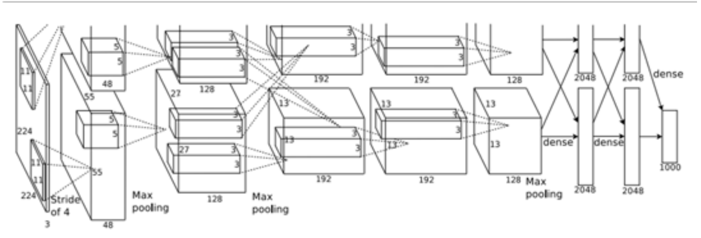
P(elephant) = 0.75

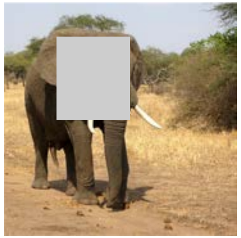
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

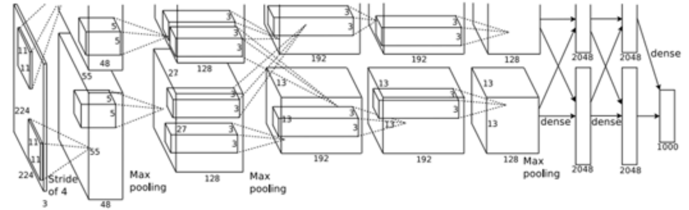
Which pixels matter: Saliency via Occlusion

Mask part of the image before feeding to CNN,
check how much predicted probabilities change

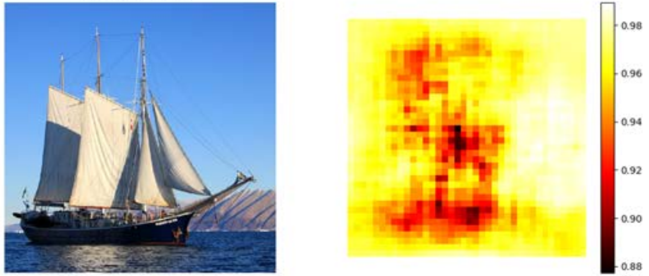




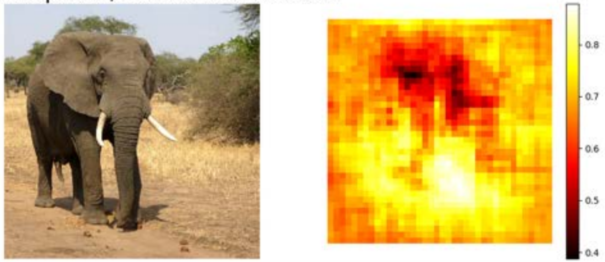




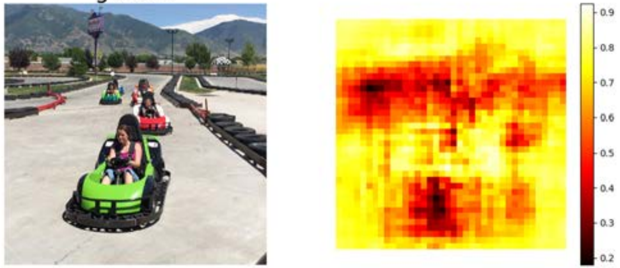
schooner



African elephant, Loxodonta africana



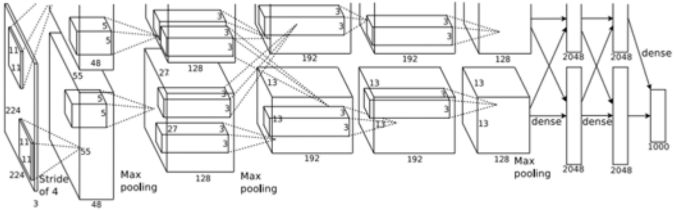
go-kart



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

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

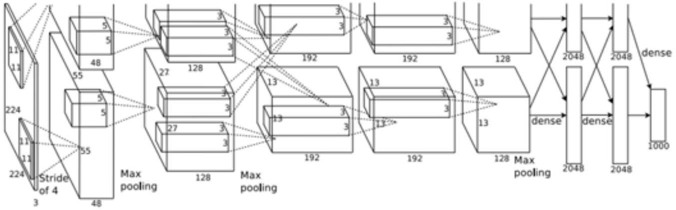


Dog

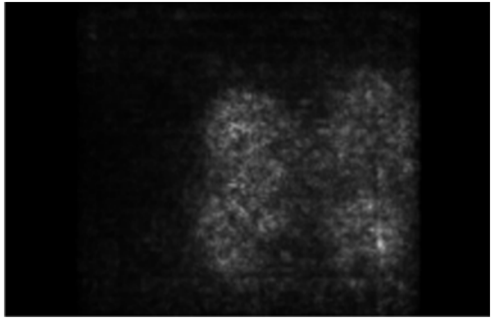
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



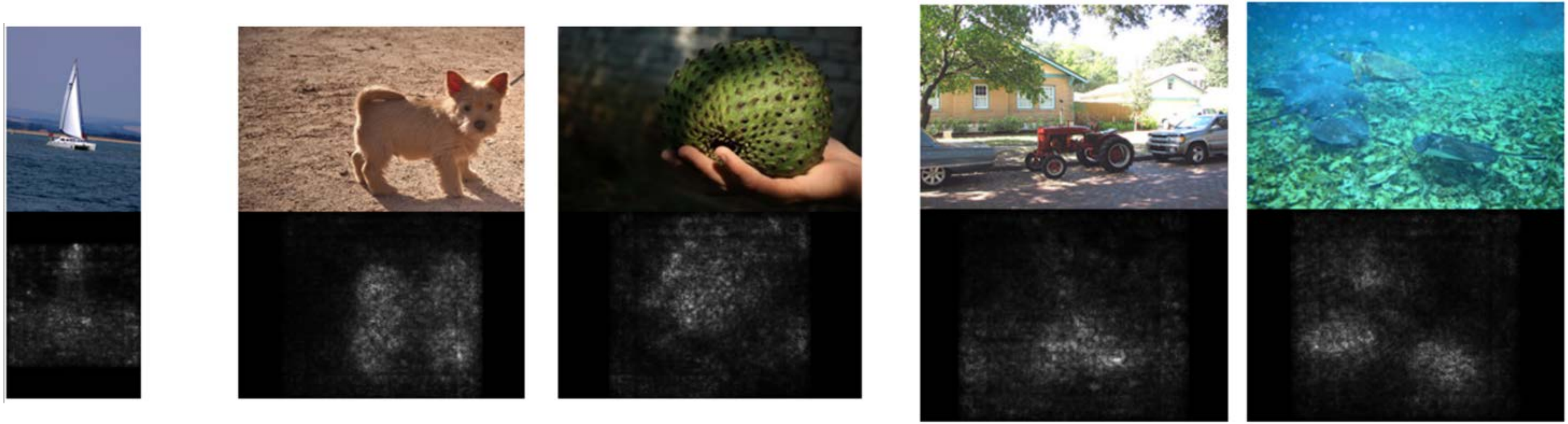
Dog



Compute gradient of **(unnormalized) class score** with respect to image pixels, take absolute value and max over RGB channels

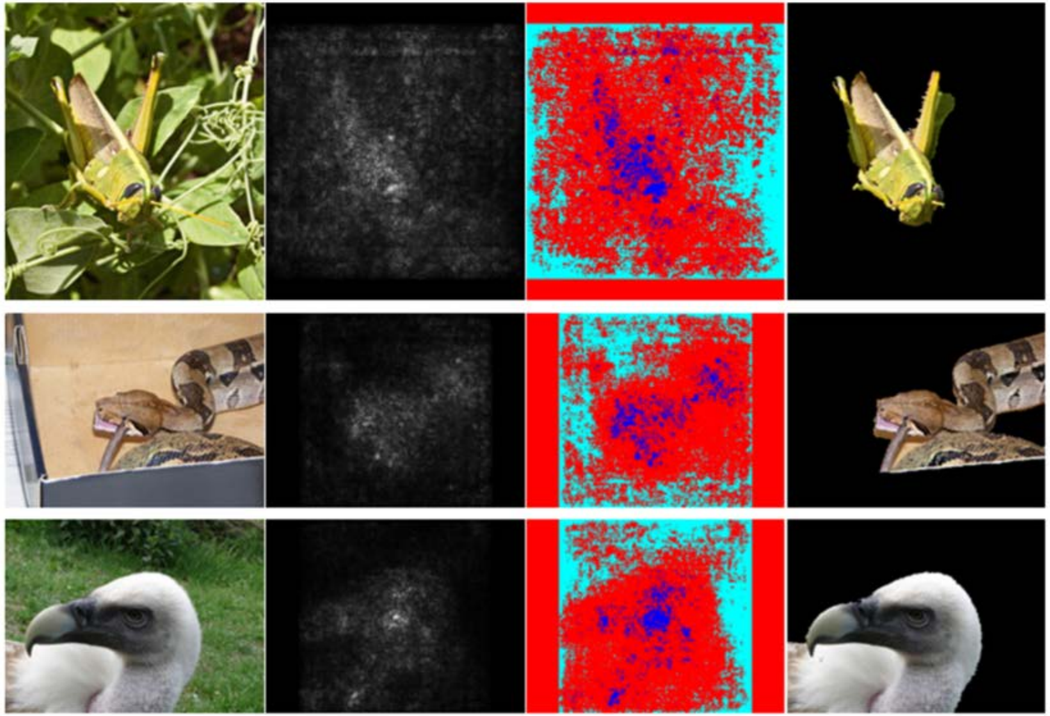
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Saliency Maps



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

Saliency Maps: Segmentation without supervision



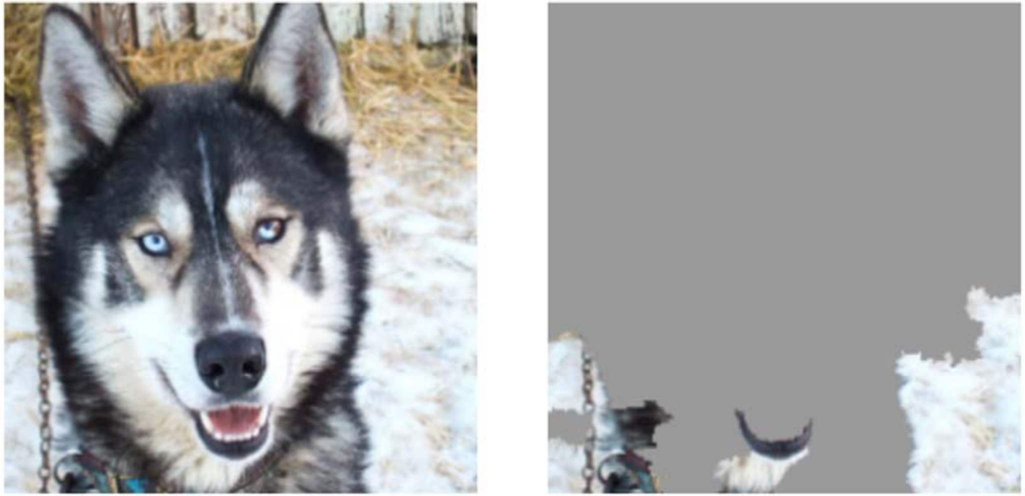
Use GrabCut on saliency map

Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission. Rother et al, "Grabcut: Interactive foreground extraction using iterated graph cuts", ACM TOG 2004

Saliency maps: Uncovers biases

Such methods also find biases

wolf vs dog classifier looks is actually a snow vs no- snow classifier



(a) Husky classified as wolf (b) Explanation