Deep Learning for machine vision

Elad Hoffer

Technion Israel Institute Of Technology,
Computer-Science department

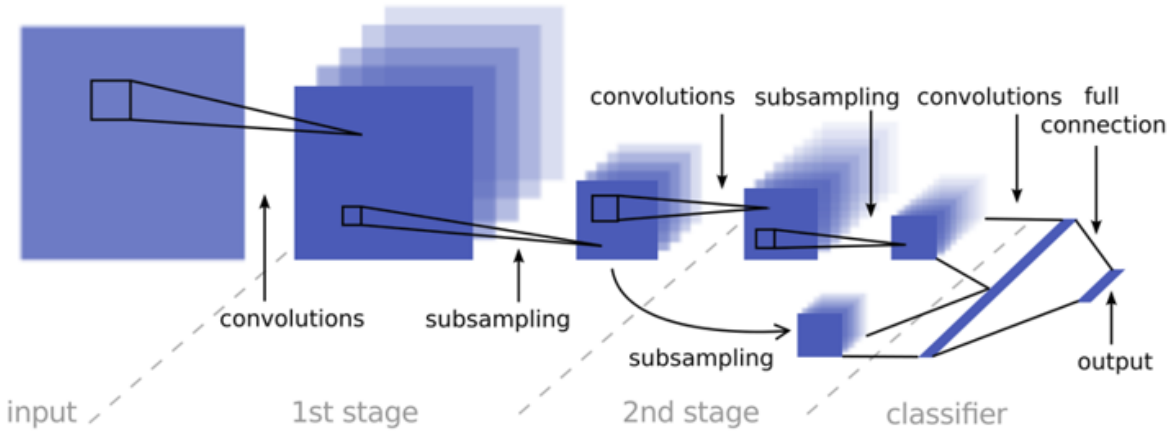
Overview

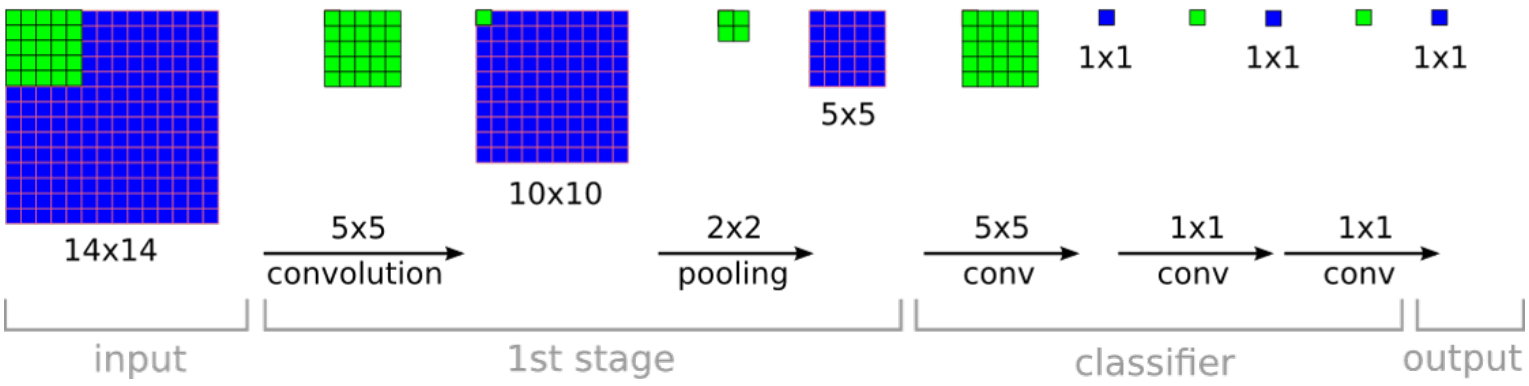
1. Background
2. CNN architectures
3. Visualizing CNNs
4. Adapting CNNs to scale
5. Localization, Detection & Segmentation
6. Generative Models

Background

ConvNets

A convolutional network (ConvNet/CNN), is composed of multiple layers of convolutions, pooling and non-linearities.





Convention and terminology

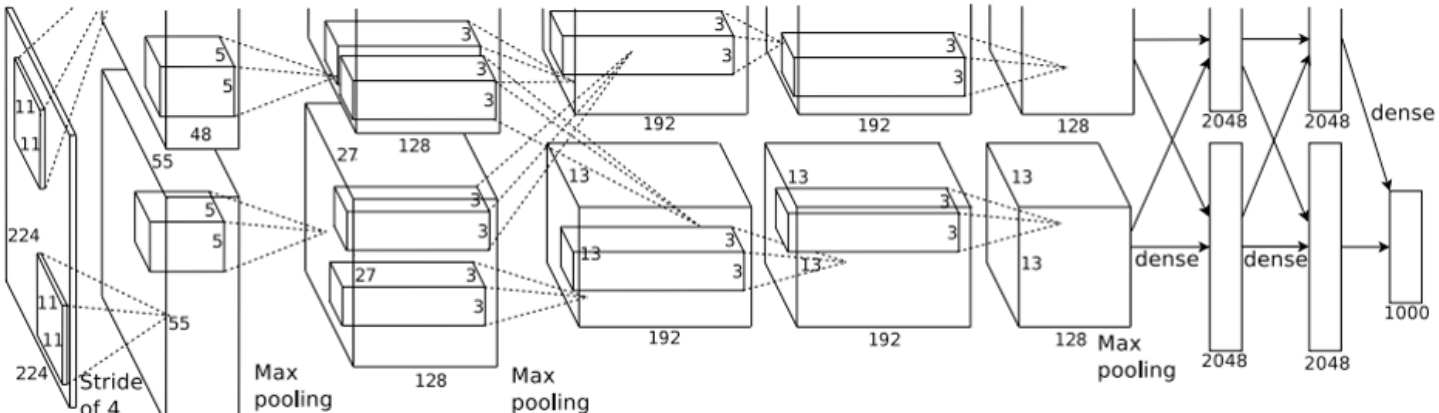
- $2d$ grids that represent image space are called *feature maps*. Each value in a feature map represent information about a specific location in the input image.
- The parameters learned for each convolution layer are also ordered as a $2d$ map - usually called “kernels” or “filters”
- Convolutions can have *strides* - moving the filters with a fixed step and *padded* - adding zero values to avoid spatial size decrease
- Bias values are added per-map (number of bias elements is the number of output feature maps)

CNN ARCHITECTURES

AlexNet

The first work that popularized Convolutional Networks in Computer Vision was the "AlexNet", developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton.

- AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% vs 26% error).
- The Network was deeper, bigger, than previous networks - $60M$ parameters, 8 trainable layers.

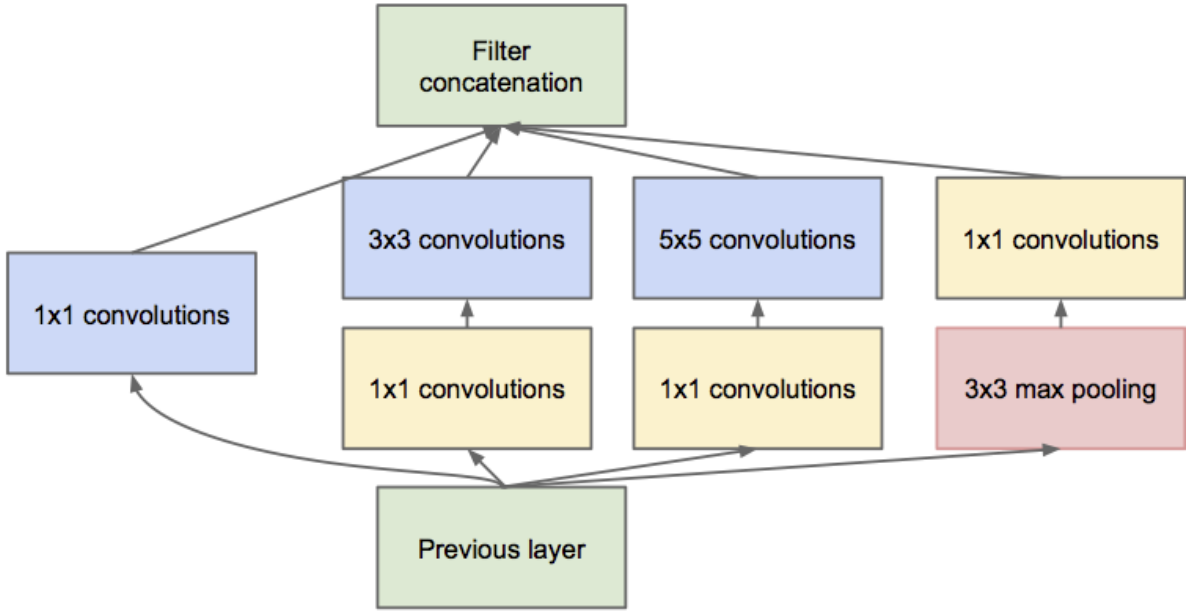


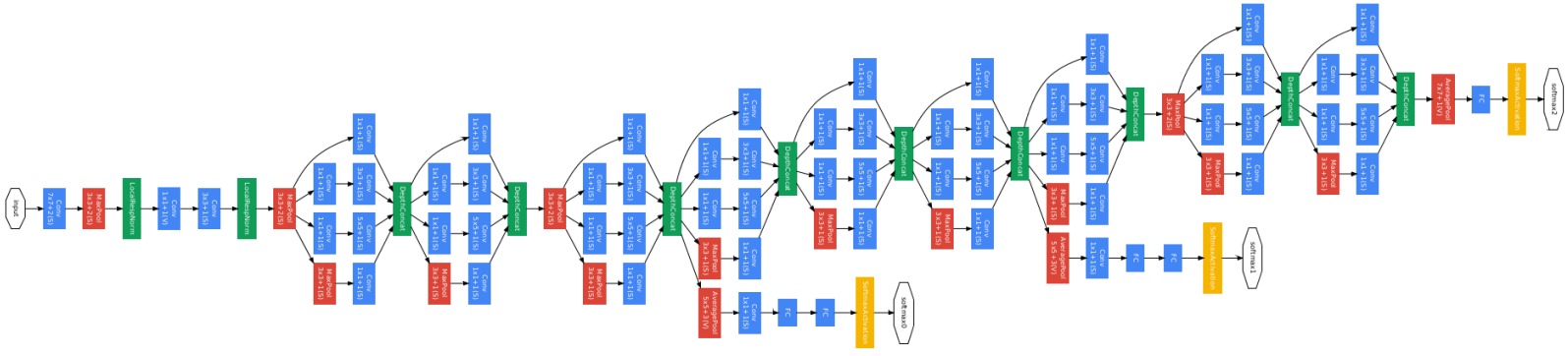
GoogLeNet

The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google called *GoogLeNet*. It featured some new ideas

- Inception Module - concatenation of several convolution sizes
- 1×1 convolutions + Average Pooling instead of Fully Connected layers (both introduced by Lin in "Network-in-network" paper from 2013).
- This dramatically reduced the number of parameters in the network (5M, compared to AlexNet with 60M).
- Google later improved this network by introducing batch-normalization + different inception configuration (Inception v2-v4)

GoogLeNet


Filter concatenation
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer



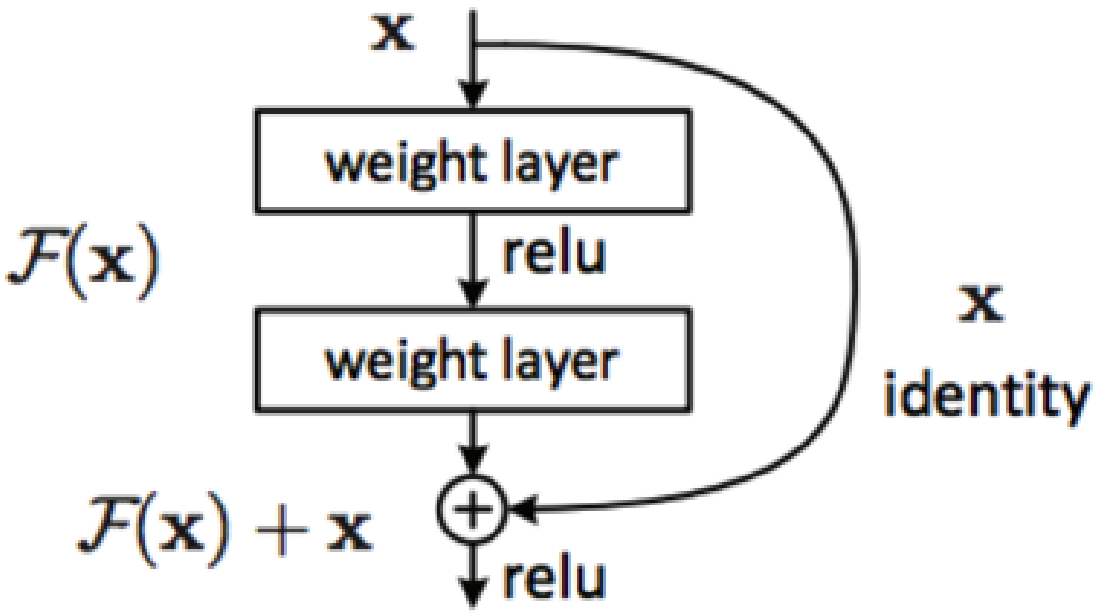
VGG Network

The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. Its main contribution was in showing that the depth of the network is a critical component for good performance.

- Their best network contains 16 trainable layers
- Features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end.
- Despite its slightly weaker classification performance, the VGG ConvNet features outperform those of GoogLeNet in multiple transfer learning tasks.
- Expensive to evaluate and uses a lot more memory and parameters (140M).

Residual networks

- Residual Network developed by Kaiming He et al. was the winner of ILSVRC 2015. It features a very deep architecture (152 layers) with special skip connections and heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network.



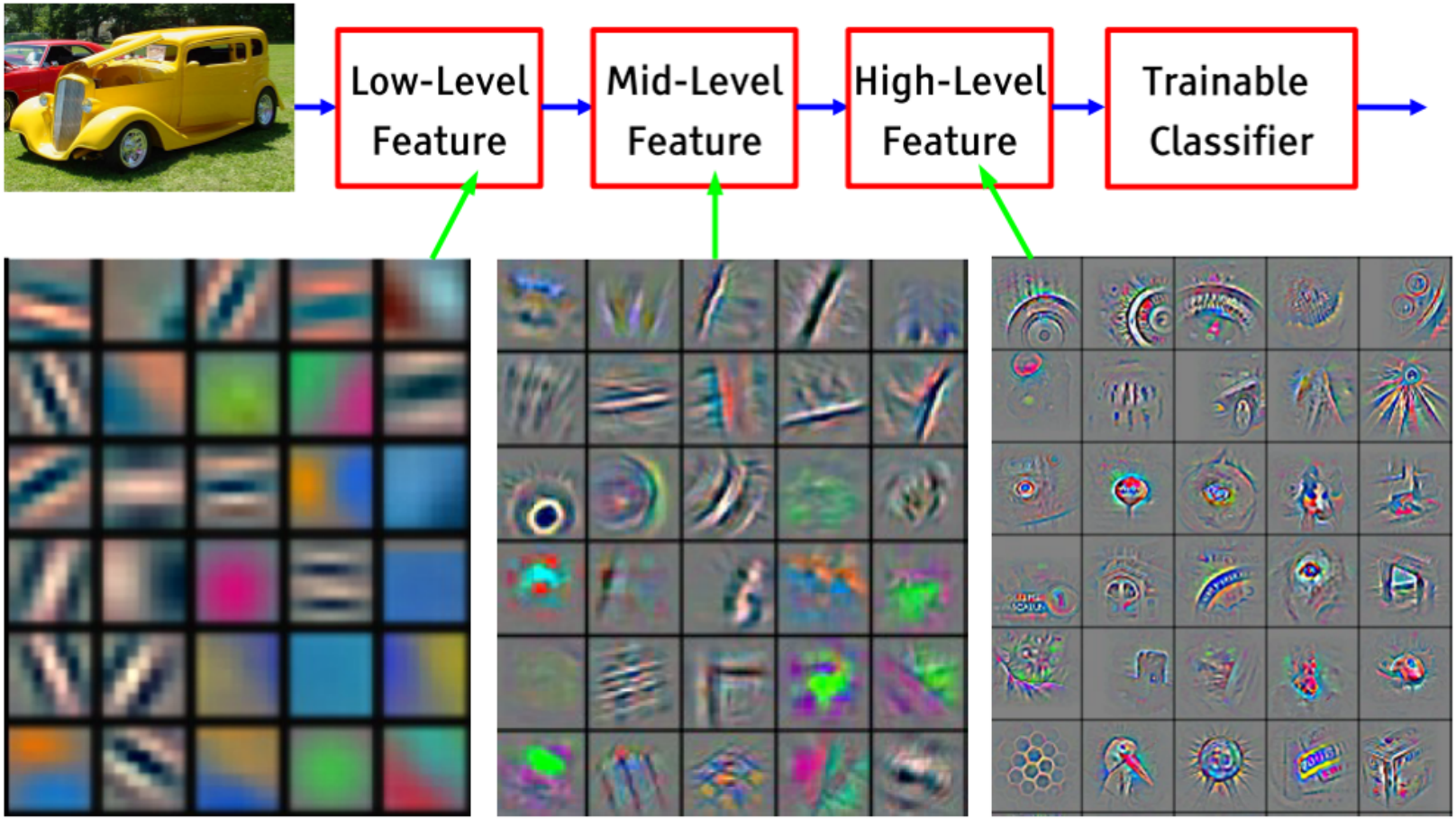
Trends in designing CNNs

Some noticeable trends in recent CNN architectures:

- Smaller convolution kernels - 3×3 is very dominant.
- Deeper - using smaller kernels is rectified by stacking more layers - effectively increasing the receptive field of the network.
- Less pooling - going deeper means we do not want to reduce spatial size too quickly. Modern network have small number (if any) of pooling layers.

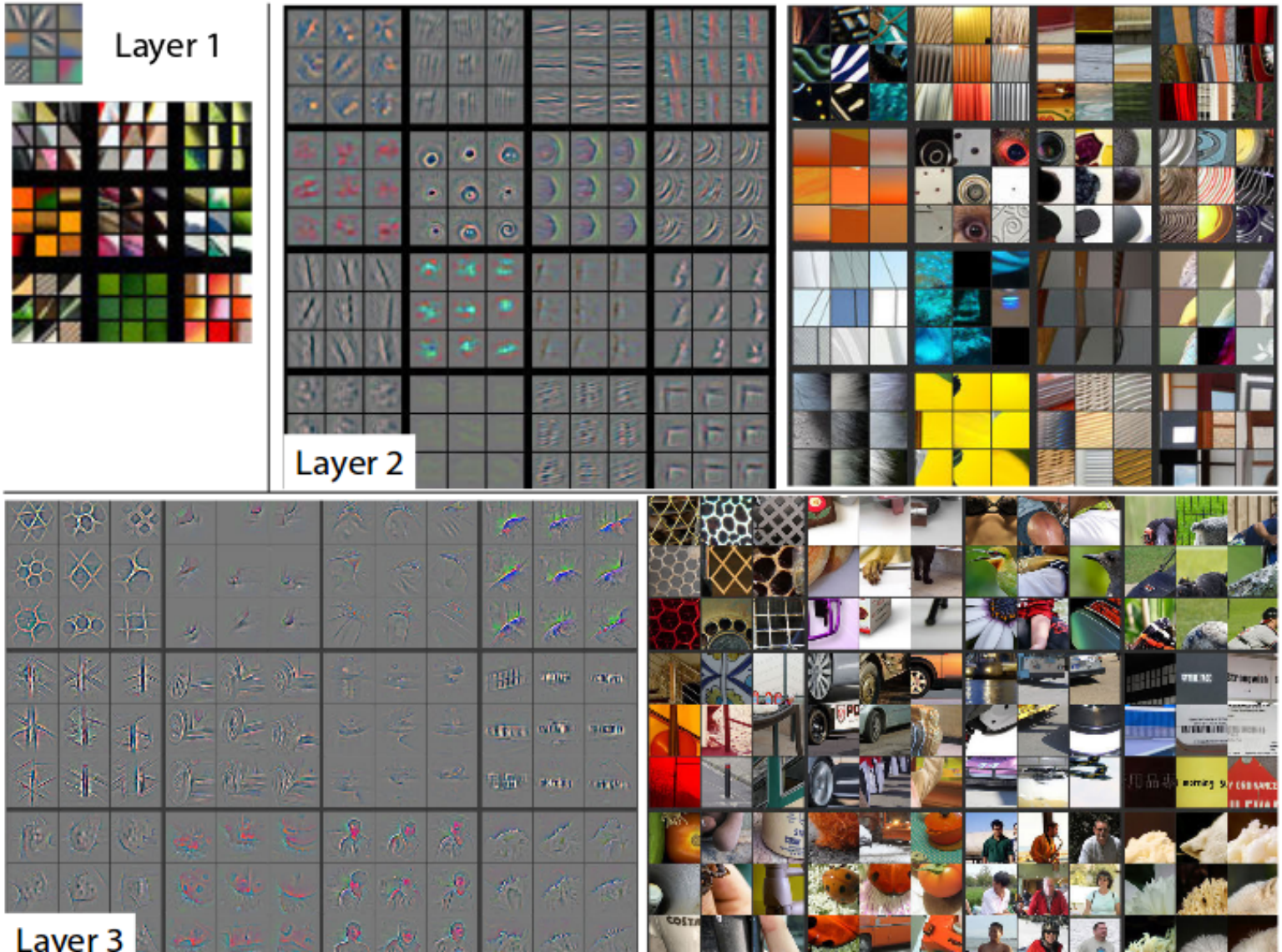
Visualizing CNNs

What do convolutional networks learn?

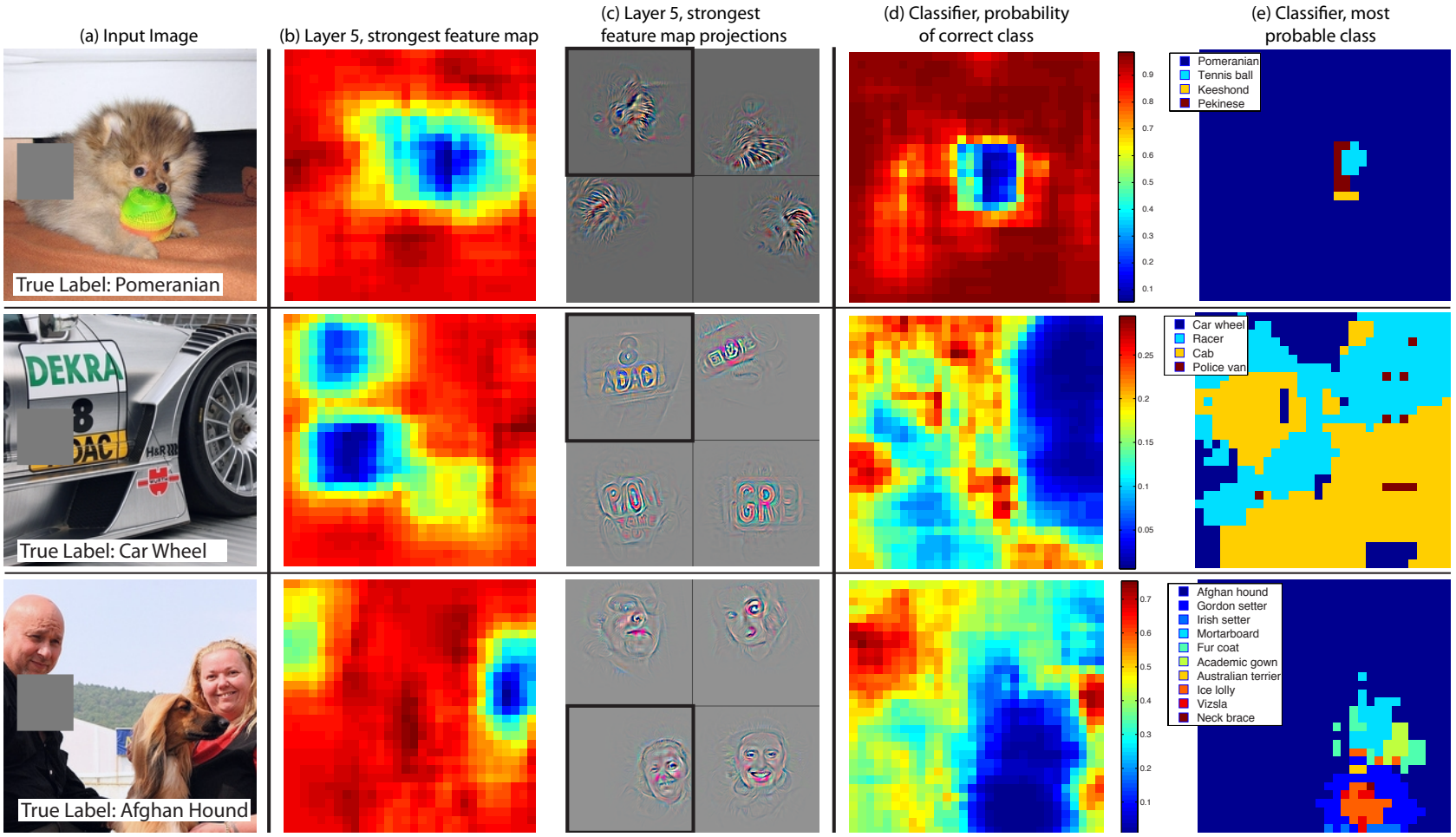


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

What do convolutional networks learn?



What do convolutional networks learn?



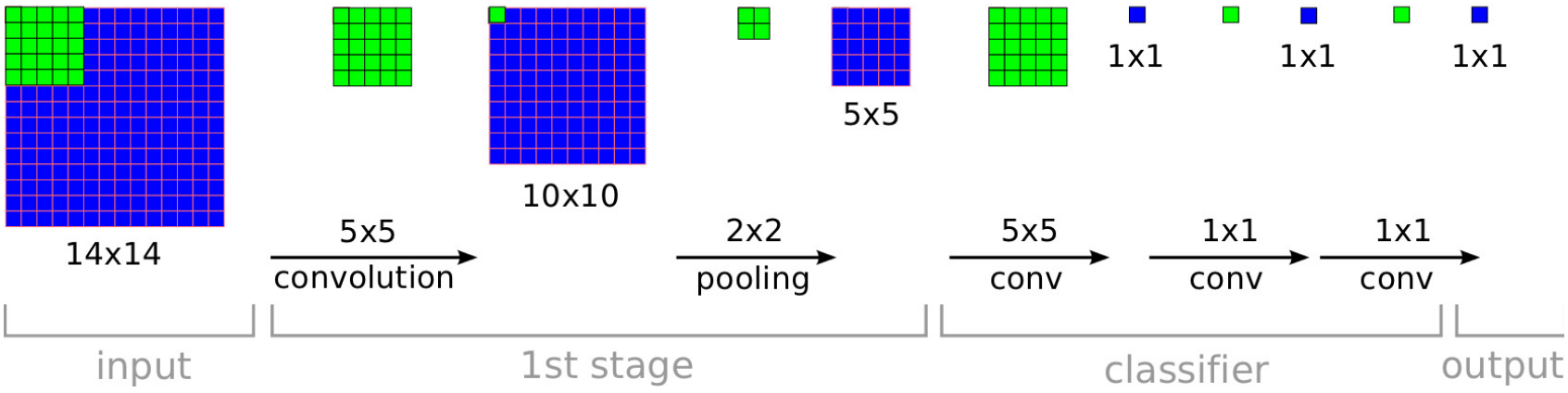
Adapting CNNs to scale

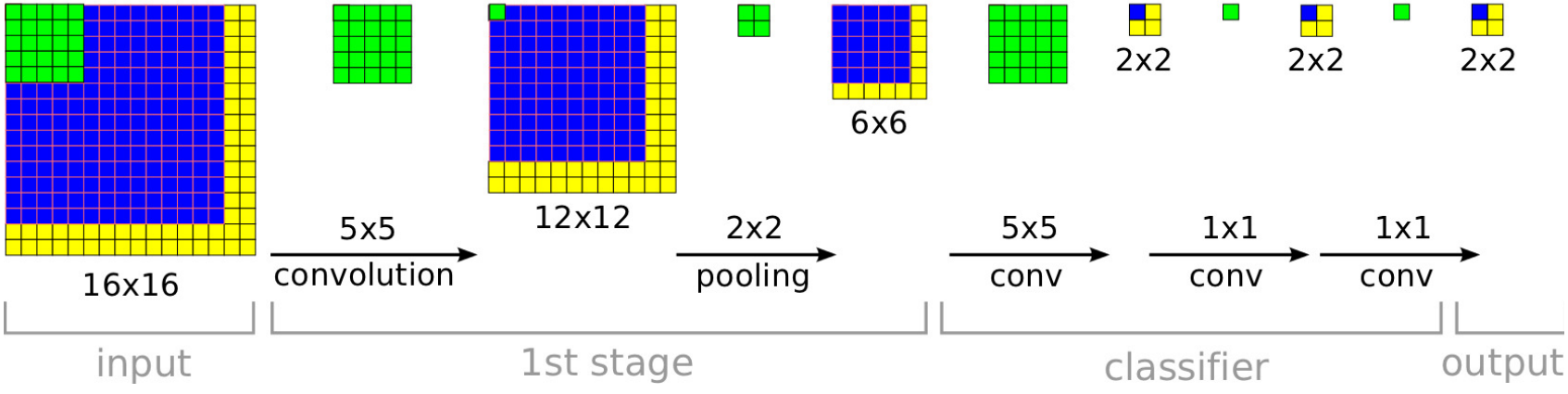
Spatial classification

In order to classify images bigger than the size we trained on, we can view fully-connected layers as convolutional kernels:

- Replace the first fully-connected layer that looks at spatial region (e.g 7×7) with a convolutional layer with kernel of the same size.
- Replace each subsequent layer with a convolutional layer with kernel of 1×1.
- Instead of a single classification, we will receive a class-map, where each spatial location is related to a different region in input image

Spatial classification

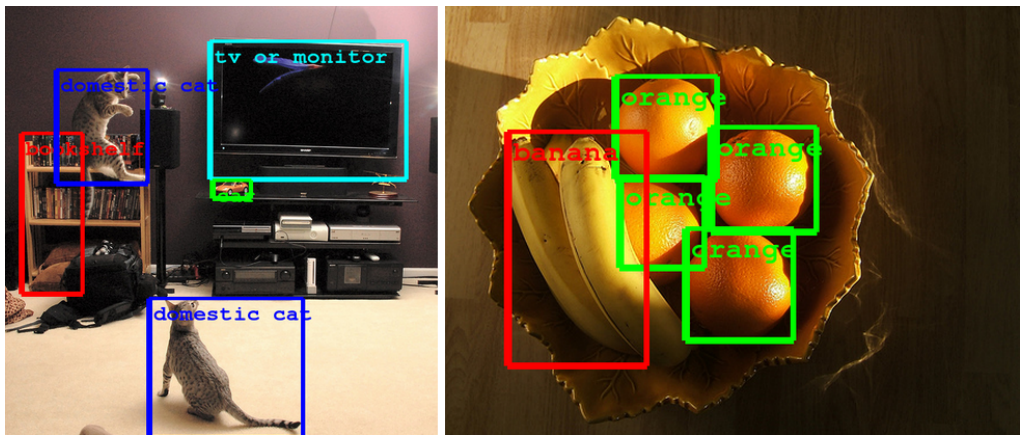




Localization, Detection & Segmentation

Localization + Detection

As we've seen, we can adapt a trained network for new sizes ands scales. A natural additional tasks in computer-vision is to try to *localize* and *detect* objects in an image.



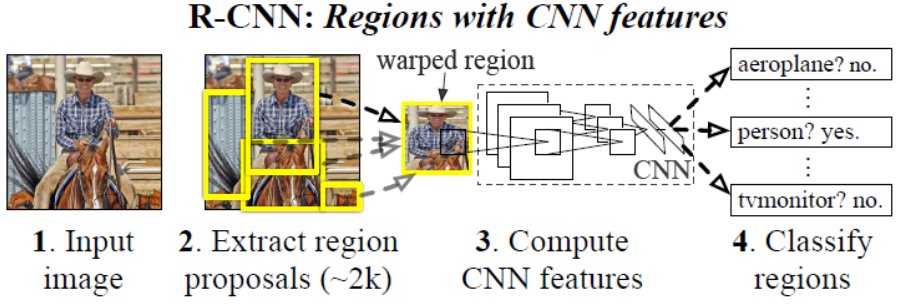
We need both to classify our object and to localize it

- For detection - we also need to handle multiple objects and reject a lot of false-positives (not-an-object)

Localization + Detection with CNNs

Some methods for detection using CNNs when we posses ground-truth bounding boxes

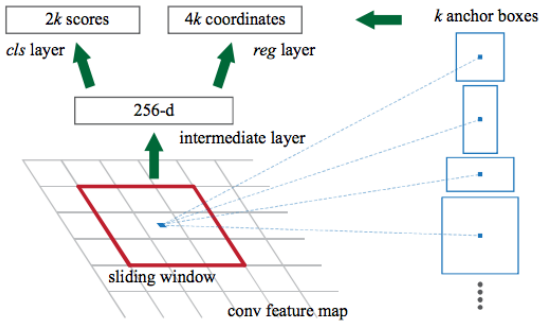
- Most naive approach is to add a regression objective - estimating the bounding boxes of each object (OverFeat - Sermanent 13').
- A more popular and effective method is to use *region-proposals* (often using "selective-search"), suggested in RCNN (Girshick 14')

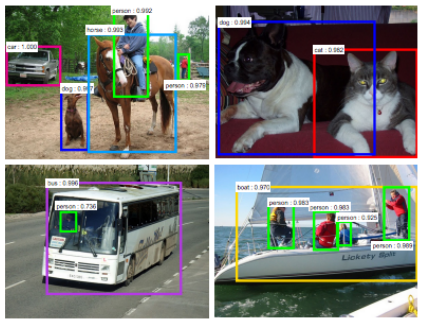


Localization + Detection with CNNs

The RCNN approach, although accurate, had a bottleneck - generating and evaluating on ~ 2000 proposals. To remedy this, future works suggested

- Evaluating most of the network on entire image, and pooling the proposals from feature space instead of image space (SPP, Fast-RCNN)
- Moved the region-proposal to be an inherent part of the network (Faster-RCNN)

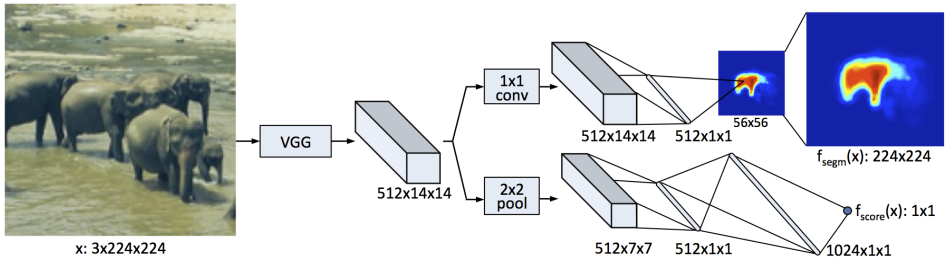




Segmentation using CNNs

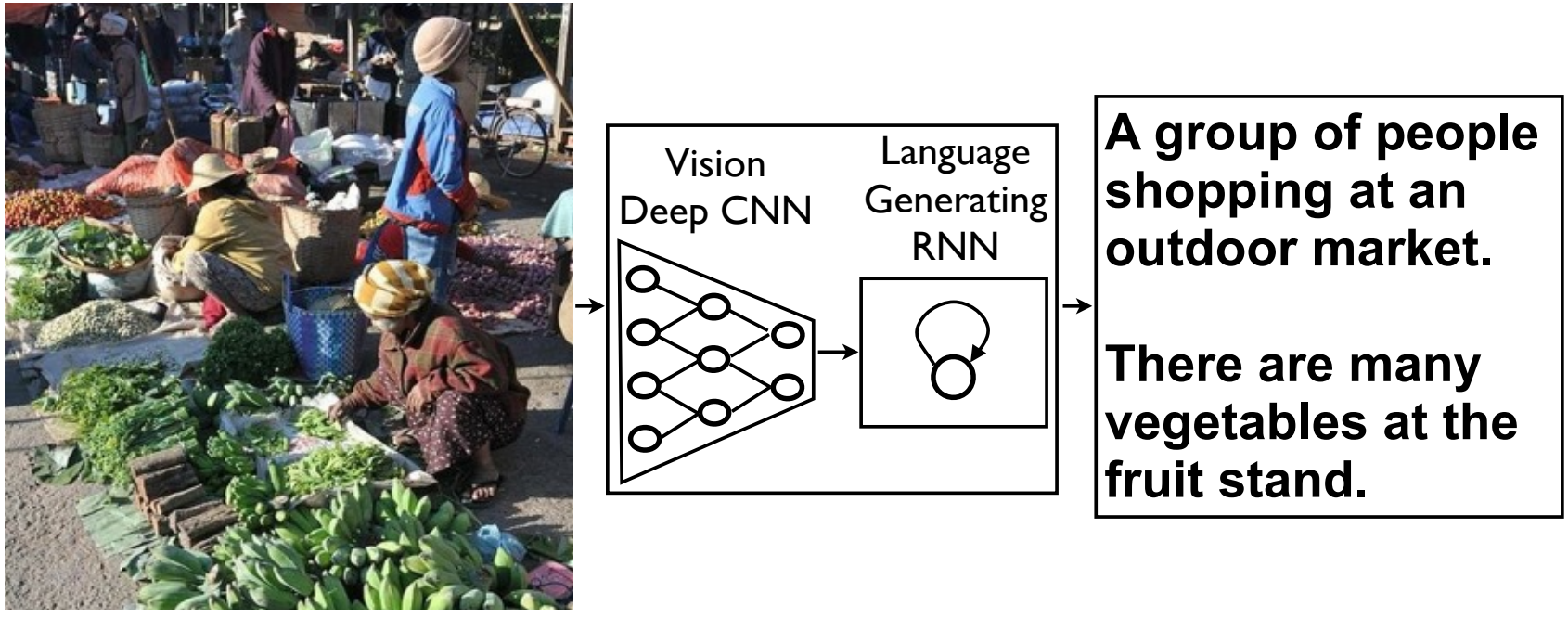
Another important computer-vision task is to segment whole images. This can be viewed as a classification at the pixel level.

- We might wish to incorporate both global, abstract features, as well as localized low-level features
- This means using multiple layers for final segmentation (Long 14')
- As with detection, trend is to build whole end-to-end models by creating object-candidates (Pinheiro 15')



Caption generation

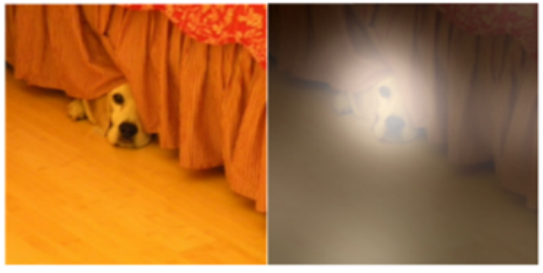
Another usage is transferring visual knowledge into language domain - such as generating captions or question answering. This is done by augmenting a convolutional network with a recurrent one.



Caption generation

This can be coupled with attention mechanisms - learn a state-dependent weighting over the input space in order to maximize an objective.

A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.

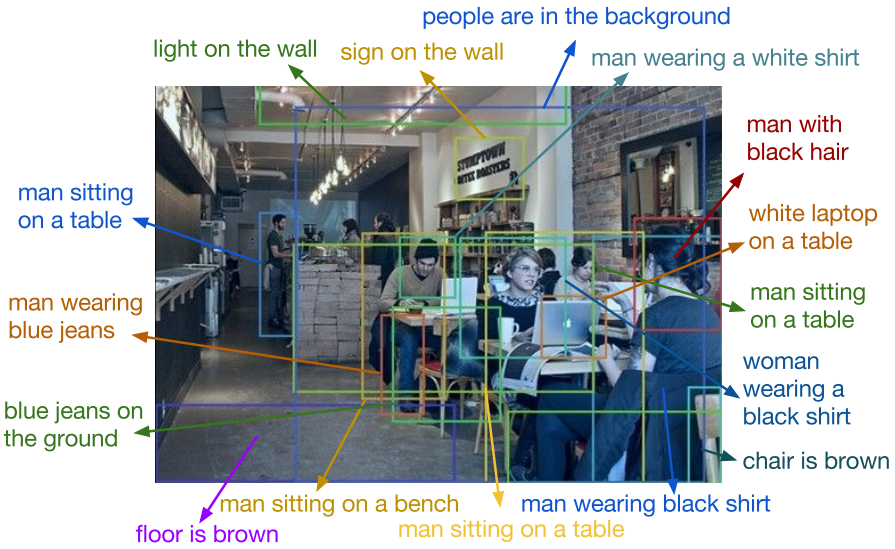
A stop sign is on a road with a mountain in the background.

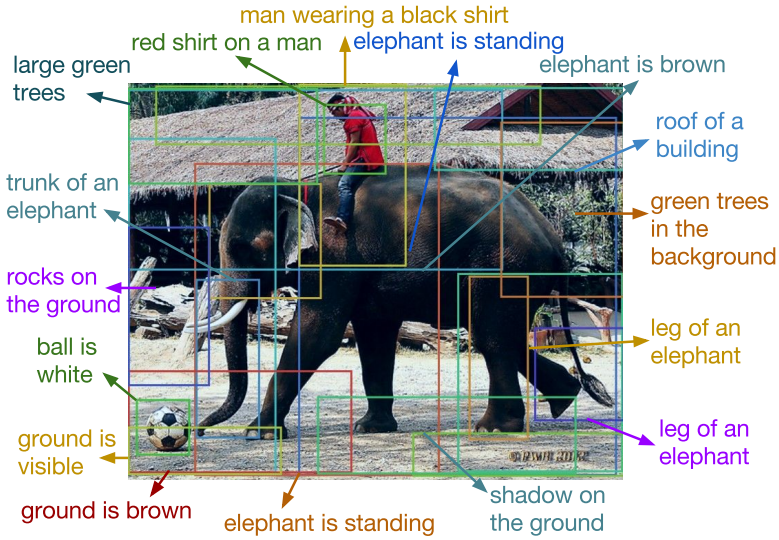
A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Caption generation





Generative Models

AutoEncoders

Probably the most popular application for unsupervised learning using neural networks, and ConvNets in particular. Autoencoder is composed of:

- An encoder $G(x; w_1)$ that maps an input to a hidden latent representation.
- A decoder $F(y; w_2)$, that maps the representation back into the input space.

The reconstrcuted output is therefor

$$\hat{x} = F(G(x; w_1); w_2)$$

The criterion used for training is usually the mean squared error (MSE) between the input and reconstruction.

$$min||x - \hat{x}||^2$$

Generative adversarial networks

One new and and exciting framework for unsupervised, generative models is the *Adversrial network* (Goodfellow 14'). It consists of two networks trained simultaneously:

- A *generative* model G that tries to capture the data distribution and generate new samples.
- A *discriminative* model D that estimates the probability that a sample came from the training data rather than G.

The training procedure for G is to maximize the probability of D making a mistake - it is "cheating" by using the gradients with respect to D.

Generative adversarial networks

"Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" (Alec Radford, Luke Metz, Soumith Chintala 15')

Generative adversarial networks

Style transfer

We can quantify an image "style" representation with the correlations between the different feature maps. These feature correlations are given by the Gram matrix $G^l \in \mathbb{R}^{N_l \times N_l}$, where for a given layer l:

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

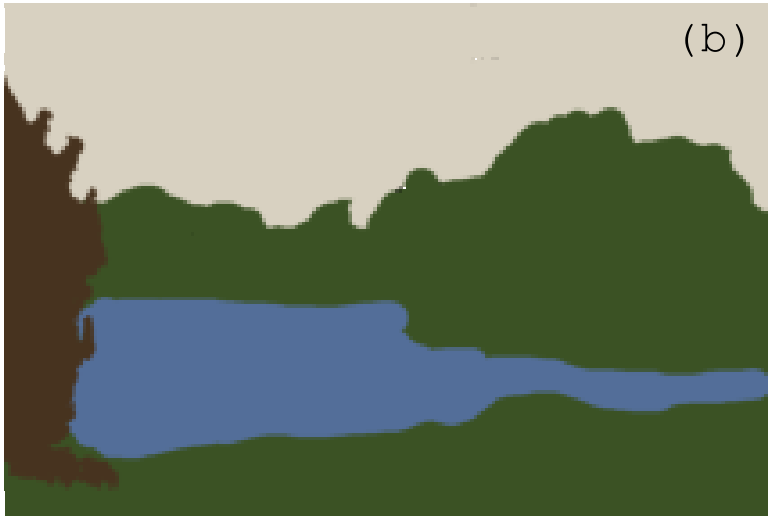
Thus we can change the target image x until it generates the same response in a certain layer of the CNN as the original image y.

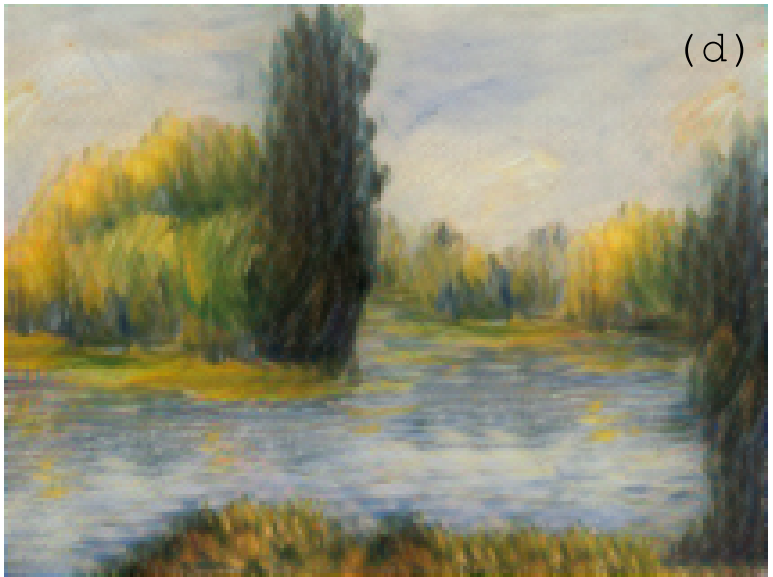
$$\mathscr{L}_{style} = \sum_{l=0}^{L} \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(S(x)_{ij}^l - S(y)_{ij}^l \right)^2$$

"A Neural Algorithm of Artistic Style" (Gatys et al)

Style transfer

Style transfer





Deep dreaming

Another interesting usage is to try and maximize the L_2 norm of intermediate layer activations, by optimizing an input image. This can help visualize the role each layer plays in the model.