Convolutional neural networks

Fraida Fund

Contents

Motivation

People are good at recognizing objects in images.

Computers are bad at it! Why?

Scene conditions

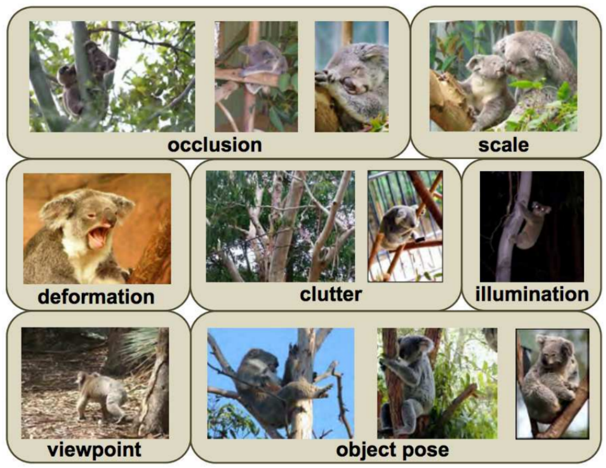


Figure 1: Difficult scene conditions: background clutter, occlusion...

Similarity and variability

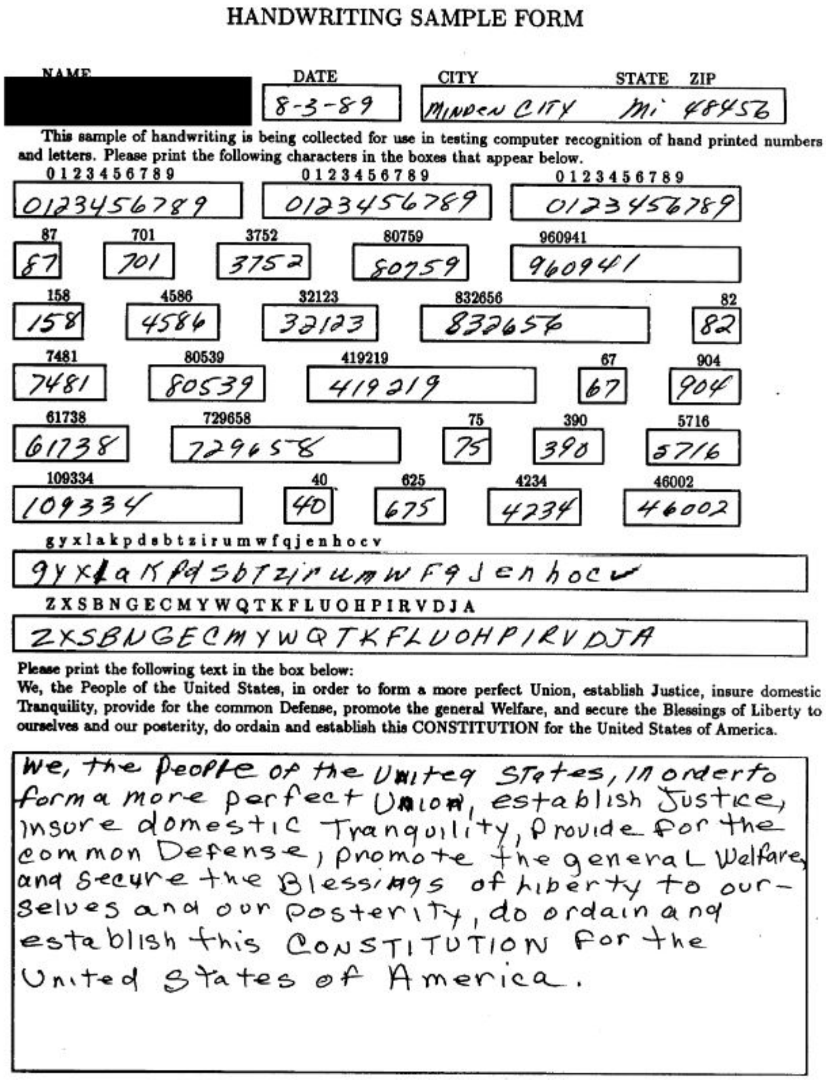
Figure 2: Must identify inter-class similarity, while accommodating intra-class variability.

Dimension

- Huge number of classes
- Images can have millions of pixels

For example, CIFAR-10: tiny images of size 32x32x3. One *fully-connected* neuron in a first hidden layer of a regular NN would have 3072 weights!

Object can be anywhere within image



HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE Mi ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789 0123456789 0123456789

87 701 3752 80759 960941

158 4586 32123 832656 82

7481 80539 419219 67 904

61738 729658 75 390 5716

109334 40 625 4234 46002

gyxlakpdsbtzirumwfqjenhocv

ZXSBNGECMYWQTKFLUOHPIRVDJA

Please print the following text in the box below:

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

Figure 3: MNIST sample. Find the “3” in the form?

Convolutional neural networks

Key idea

- Neuron is connected to a small part of image at a time (*locally connected*)
- By having multiple locally connected neurons covering the entire image, we effectively "scan" the image

Locally connected neurons: illustration

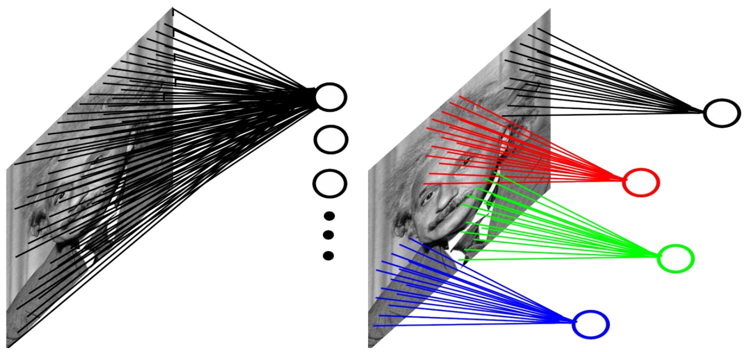


Figure 4: Example: 200x200 image. Fully connected network with 400,000 hidden units, 16 billion parameters. Locally connected network with 400,000 hidden units in 10x10 fields, 40 million parameters.

Spatial arrangement: conventional networks

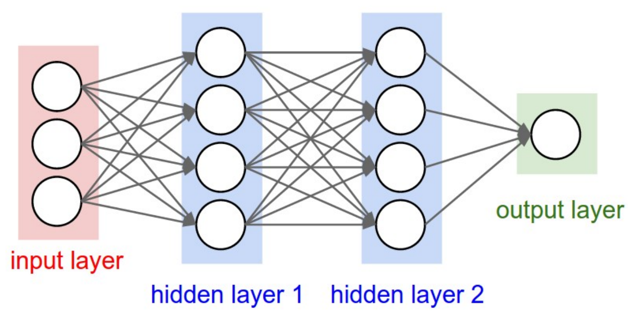


Figure 5: Conventional neural network: neurons don't have spatial arrangement.

Spatial arrangement: convolutional networks

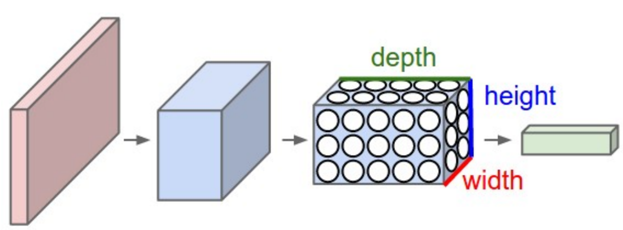


Figure 6: CNN: input and output of each layer is a *tensor*, a multidimensional array with width, height, and depth. Preserves spatial relationships.

Layers in CNN

Different layer types

- Convolutional Layer
- Pooling Layer
- Fully-Connected Layer

Each layer accepts an input 3D volume, transforms it to an output 3D volume.

Convolutional layer

- Layer has a set of learnable “filters”
- Each filter has small width and height, but full depth
- During forward pass, filter “slides” across width and height of input, and computes dot product
- Effectively performs “convolution”

Convolution example

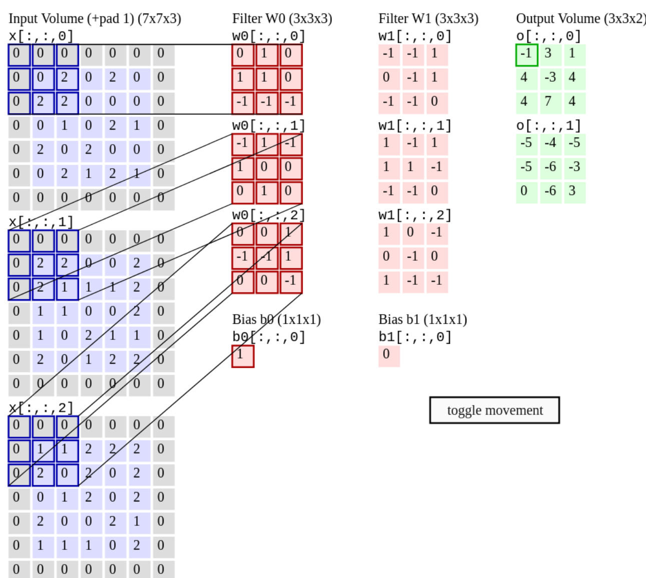


Figure 7: Animated demo at https://cs231n.github.io/assets/conv-demo/index.html

Feature localization via “convolution”

- Given large image X with dimensions $N_1 \times N_2$,
- small filter W with dimensions $K_1 \times K_2$

At each offset (j_1, j_2) we compute

$$Z[j_1, j_2] = \sum_{k_1=0}^{K_1-1} \sum_{k_2=0}^{K_2-1} W[k_1, k_2] X[j_1 + k_1, j_2 + k_2]$$

which is large if “matching” feature is present.

Feature localization via convolution (illustration)

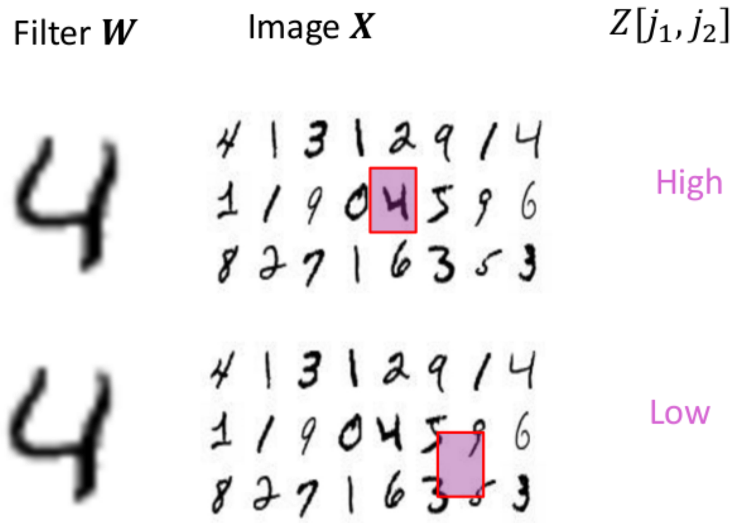


Figure 8: Finding features with convolution.

Local connectivity

- Each filter is fully connected along depth axis, but only locally connected along width and height.
- Example: For CIFAR-10 (32x32x3), a 5x5 filter will have weights to a (5x5x3) region in input volume.
- Parameter dimensions: 75 weights and 1 bias. (much smaller!)

Size of output volume

Size of output volume is determined by

- Input volume size W
- depth
- filter field size F
- stride S
- zero padding P

Size of output volume: depth

Output depth is a hyperparameter: corresponds to number of filters that should "look" at the same region of input at a time.

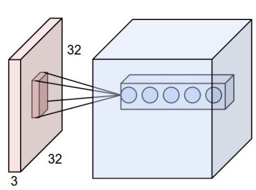


Figure 9: Example: this output depth is 5.

Size of output volume: stride

How many pixels do we slide the filter each time? This is called the *stride*.

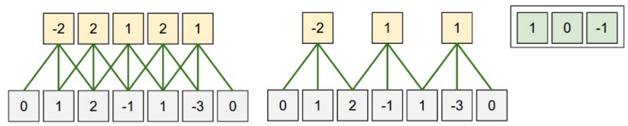


Figure 10: In this example there is one spatial dimension (x-axis), one neuron with F = 3, W = 5, and P = 1. Left: S=1. Right: S=2.

Size of output volume: zero-padding

Use zero padding on border-

- Without padding, size would shrink in each layer.
- Without padding, neurons "touch" the edges less often than the middle

To have output width and height the same as input, use $P = \frac{F-1}{2}$.

Summary of convolutional layer

- Accepts input volume $W_1 \times H_1 \times D_1$
- Four hyperparameters: number of filters K, filter size F, stride S, amount of zero padding P
- Produces volume of size

$$W_2 = \frac{W_1 - F + 2P}{S} + 1, H_2 = \frac{H_1 - F + 2P}{S} + 1$$
$$D_2 = K$$

- With parameter sharing: $F \cdot F \cdot D_1$ weights per filter, for $F \cdot F \cdot D_1 \cdot K$ weights and K biases
- Common setting: $F = 3, S = 1, P = 1.$

Parameter sharing

Basic insight:

- A particular filter with a set of weights represents a feature to look for
- If it is useful to look for a feature at position x, y, it is probably useful to look for the same feature at x', y'
- "Depth slice" = all the shifted versions of a filter. All neurons within a depth slice can share the same weights.

Greatly reduces number of parameters.

Example: AlexNet filters

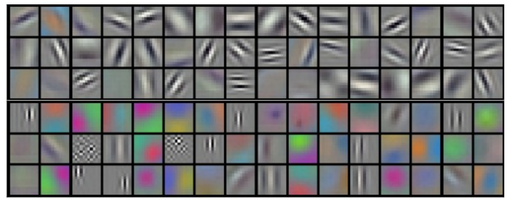


Figure 11: Each of the 96 filters shown here is of size 11x11x3, and each one is shared by the 55x55 neurons in one depth slice.

ReLU activation

- Convolutional typically followed by ReLU activation function

Pooling layer

- Reduces spatial size of image (reduce computation, prevent overfitting)
- Typical example: 2x2 filter size, stride of 2, downsamples by a factor of 2 along width and height
- Works independently on each depth slice
- Typically uses MAX operation

Pooling: illustration

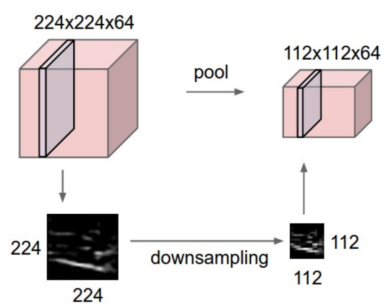


Figure 12: Input volume of size 224x224x64 is pooled with filter size 2, stride 2 into output volume of size 112x112x64 (with same depth).

Pooling: illustration of max operation

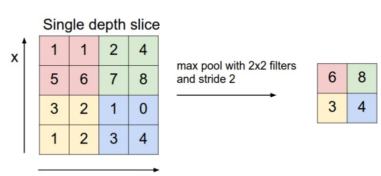


Figure 13: Each max is taken over a 2x2 square.

Summary of pooling layer

- Accepts input volume $W_1 \times H_1 \times D_1$
- Two hyperparameters: filter size F, stride S
- Produces volume of size

$$W_2 = \frac{W_1 - F}{S} + 1, H_2 = \frac{H_1 - F}{S} + 1, D_2 = D_1$$

- No parameters

Fully connected layer

- Reshape into matrix
- Output with matrix multiplication

$$Z[i,k] = \sum_j W[j,k]U[i,j] + b[k], \quad k = 0, \dots, N_O$$

Typical architecture

- Input
- Some number of convolutional + ReLU layers
- Occasional pooling layers
- Some number of fully connected + ReLU layers
- Fully connected output layer

Example

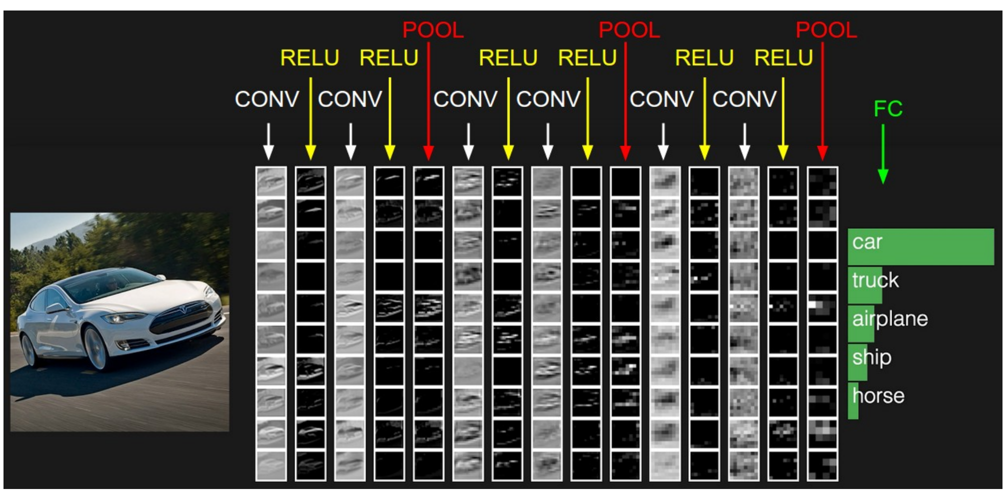


Figure 14: Example of a convolutional network architecture. Live demo link.

Reference

Source of most images here, and excellent set of notes on convolutional neural networks: https://cs231n.github.io/convolutional-networks/

Transfer learning

Using pre-trained networks

- State-of-the-art networks involve millions of parameters, huge datasets, and days of training on GPU clusters
- Idea: share pre-trained networks (network architecture and weights)
- Some famous networks for image classification: Inception, ResNet, and more
- Can be loaded directly in Keras

Transfer learning from pre-trained networks

Use pre-trained network for a different task

- Use early layers from pre-trained network, freeze their parameters
- Only train small number of parameters at the end

Transfer learning illustration (1)

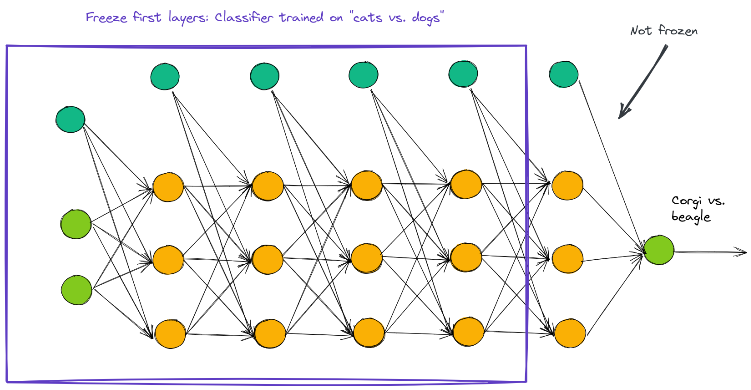


Figure 15: When the network is trained on a very similar task, even the abstract high-level features are probably very relevant, so you might tune just the classification head.

Transfer learning illustration (2)

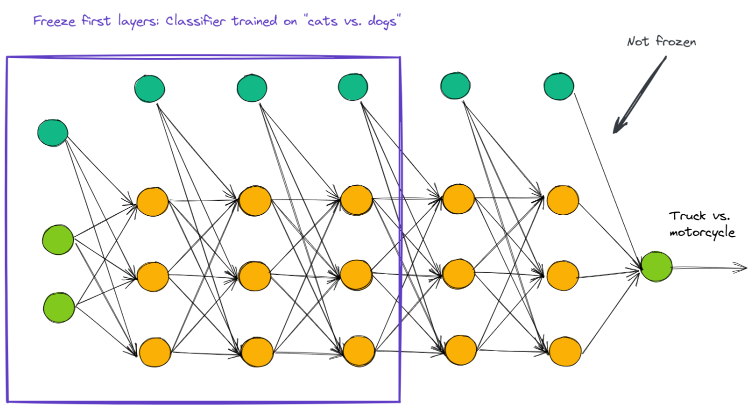


Figure 16: If the original network is not as relevant, may fine-tune more layers.