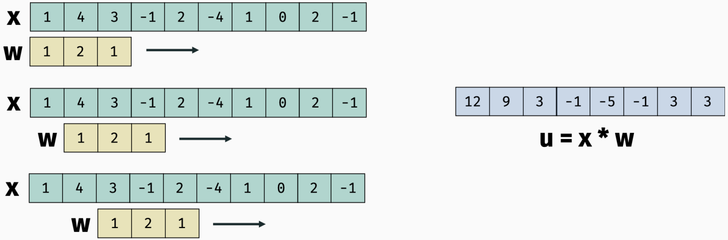
CS-UY 4563: Lecture 19
Convolutional Neural Networks

NYU Tandon School of Engineering, Prof. Christopher Musco

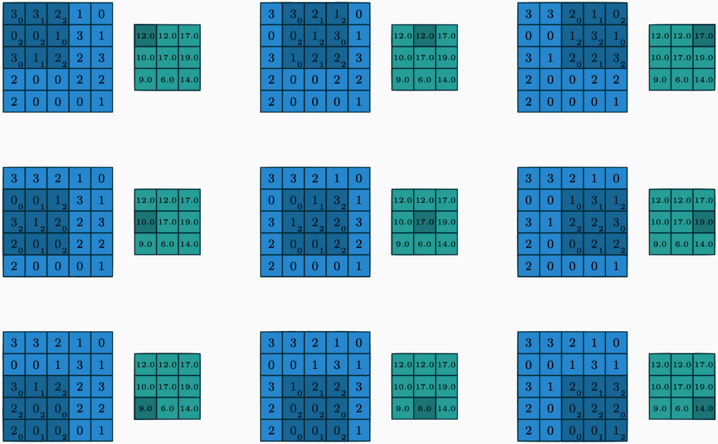
- I will be reviewing project proposals over the next few days. If you need feedback sooner, please email separately to set up a meeting, or stop by office hours.
- Written Homework 4 due **next Monday**.
- Work through `demo_convolutions.ipynb`, now on course website.

CONVOLUTIONAL FEATURE EXTRACTION

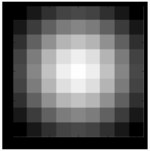
Last lecture: Convolution in 1, 2, and 3D is a powerful generic tool for designing natural **feature transformations** for time-series data, audio data, or images.

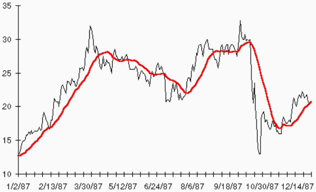


$$\mathbf{w} = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

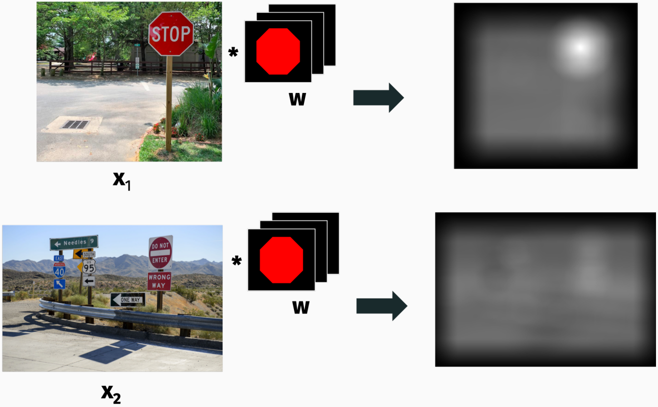


A uniform or Gaussian filter can be used to smooth input data:





Convolution can be used to find local patterns in images:



SIDE NOTE

Good question from Piazza: Won't this filter yield lots of false positives?

SIDE NOTE

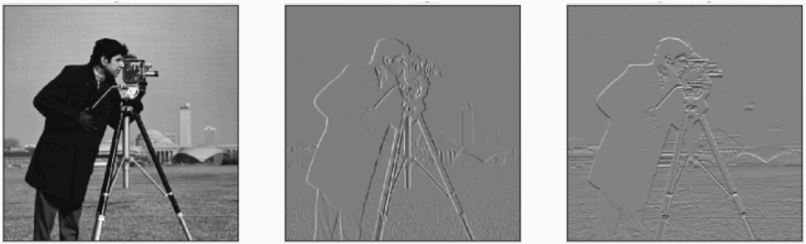
Possible fix that doesn't require image normalization:

APPLICATIONS OF CONVOLUTION

Application 3: Edge detection.

Consider a 2D edge detection filter:

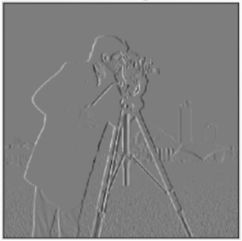
$$W_1 = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad W_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$



x * ?   x * ?

Sobel filter is more commonly used:

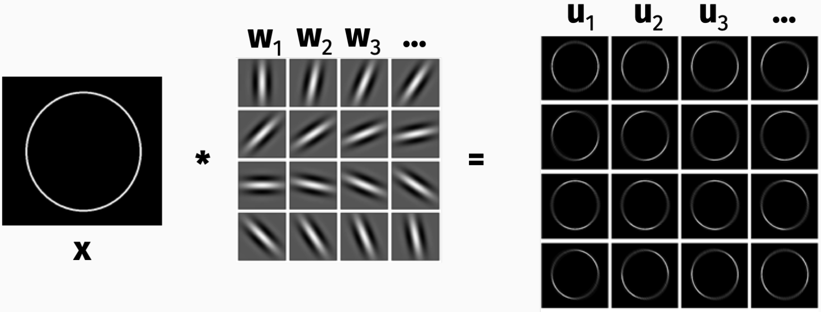
$$W_1 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad W_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



x * ?

x * ?

Can define edge detection filters for any orientation.



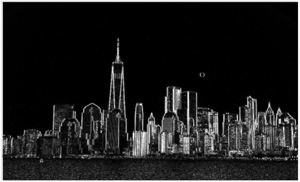
How would edge detection as a feature extractor help you classify images of city-scapes vs. images of landscapes?








$$* \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \longrightarrow$$



$\mathbf{I}_C$ $\mathbf{E}_C$

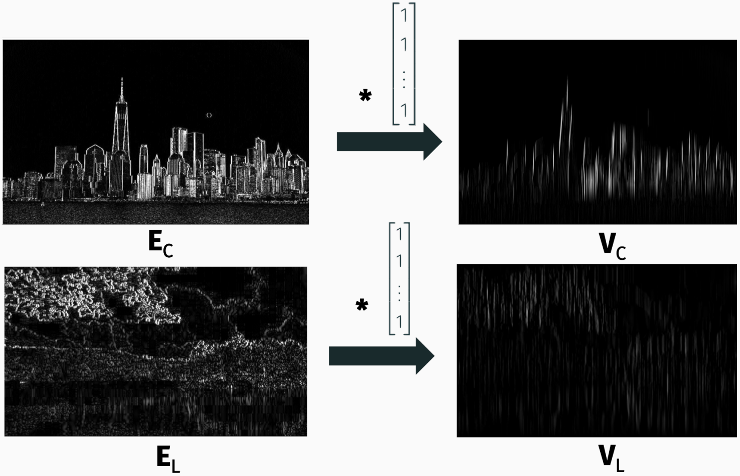
$$* \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \longrightarrow$$

$\mathbf{I}_L$ $\mathbf{E}_L$

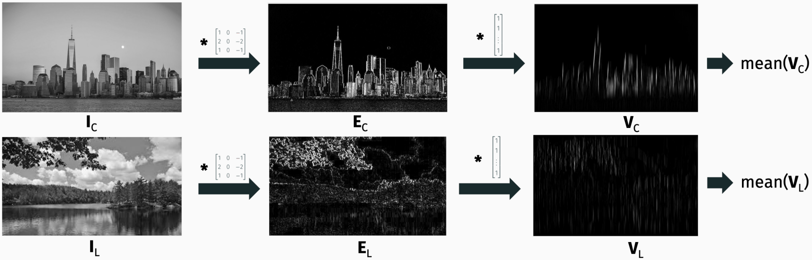
$$\texttt{mean}(\mathbf{I}_C) = .108 \quad \text{vs.} \quad \texttt{mean}(\mathbf{I}_L) = .123$$

The image with highest vertical edge response isn't the city-scape.

Feed edge detection result into pattern matcher that looks for long vertical lines.



HIERARCHICAL CONVOLUTIONAL FEATURES



$$\texttt{mean}(\mathsf{V}_C) = .062 \quad \text{vs. } \texttt{mean}(\mathsf{V}_L) = .054$$

$$\texttt{mean}(\mathsf{V}_C \cdot \mathsf{V}_C) = .042 \quad \text{vs. } \texttt{mean}(\mathsf{V}_L \cdot \mathsf{V}_L) = .018$$

The image with highest average response to (edge detector) + (vertical pattern) is the city scape.

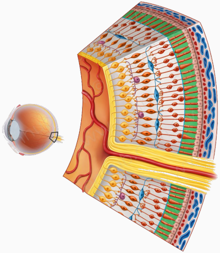
mean(V) is an <u>extracted scalar feature</u> which could be used for classifying cityscapes from landscapes using a linear classifier.

Hierarchical combinations of simple convolution filters are very powerful for understanding images.

In particular, edge detection seems like a critical first step.

Lots of evidence from biology.

Light comes into the eye through the lens and is detected by an array of photosensitive cells in the **retina**.



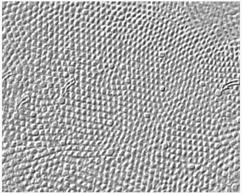
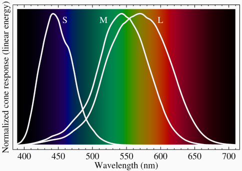


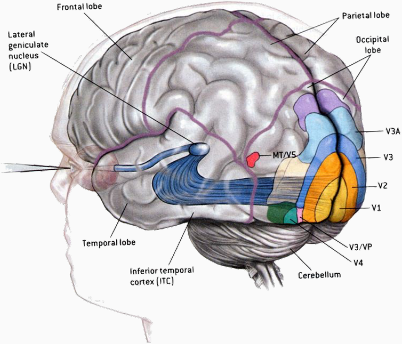
Fig. 13. Tangential section through the human fovea. Larger cones (arrows) are blue cones. From Ahnelt et al. 1987.

Rod cells are sensitive to all light, larger **cone** cells are sensitive to specific colors. We have three types of cones:



VISUAL SYSTEM

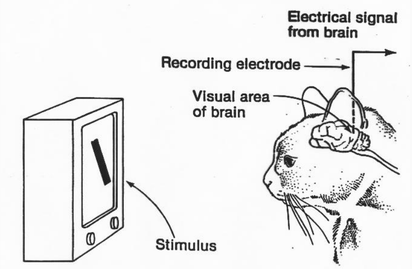
Signal passes from the retina to the primary (V1) visual cortex, which has neurons that connect to higher level parts of the brain.



What sort of processing happens in the primary cortex?

Lots of edge detection!

Huber + Wiesel, 1959: “Receptive fields of single neurones in the cat’s striate cortex.” Won Nobel prize in 1981.

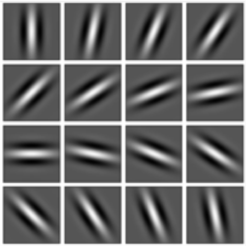


Different neurons fire when the cat is presented with stimuli at different angles. Cool video at `https://www.youtube.com/watch?v=OGxVfKJqX5E`.

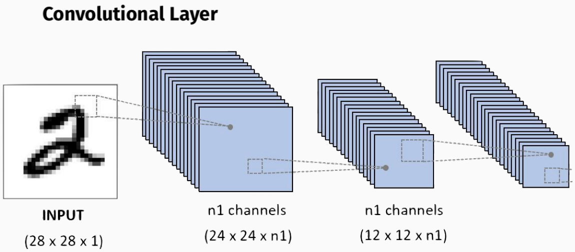
”What the Frog’s Eye Tells the Frog’s Brain”, Lettvin et al. 1959. Found explicit edge detection circuits in a frogs visual cortex.

State of the art until $\sim$ 10 years ago:

- Convolve image with edge detection filters at many different angles.
- Hand engineer features based on the responses.
- SIFT and HOG features were especially popular.



Neural network approach: Learn the parameters of the convolution filters based on training data.

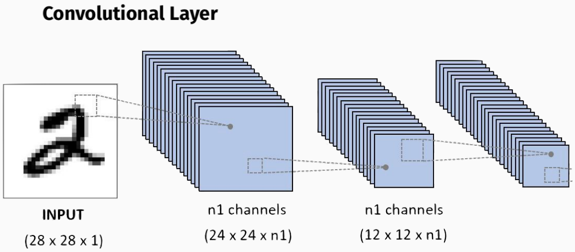


First convolutional layer involves $n1$ convolution filters $\mathbf{W}_1, \ldots, \mathbf{W}_{n1}$. Each is small, e.g. 5×5. Every entry in $\mathbf{W}_i$ is a free parameter: $\sim 25 \cdot n1$ parameters to learn.

Produces $n1$ matrices of hidden variables: i.e. a tensor with depth $n1$.

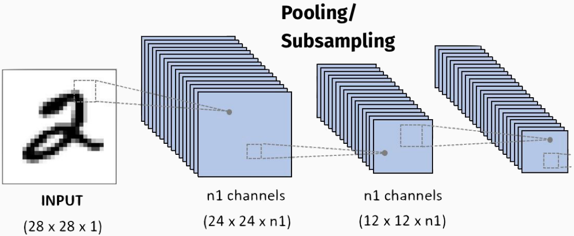
A fully connected layer that extracts the same feature would require $(28 \cdot 28 \cdot 24 \cdot 24) \cdot n1 = 451,584 \cdot n1$ parameters.

By "baking in" knowledge about what type of features matter, we greatly simply the network.



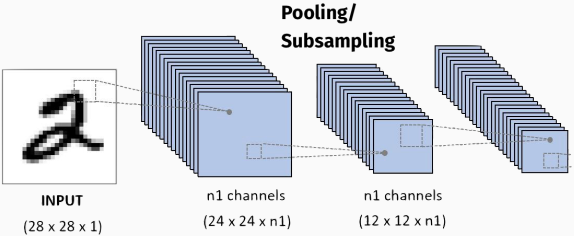
Each of the $n1$ ouputs is typically processed with a **non-linearity**. Most commonly a Rectified Linear Unity (ReLU): $x = \max(\bar{x}, 0)$.

Convolution + non-linearity are typically followed by a layer which performs **pooling + down-sampling**.



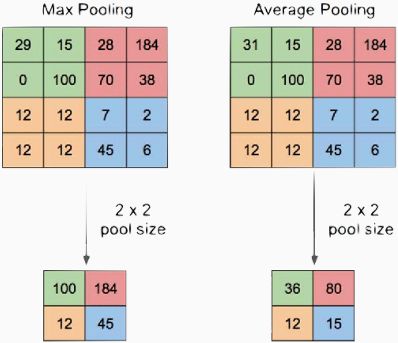
Most common approach is max-pooling.

Convolution + non-linearity are typically followed by a layer which performs **pooling + down-sampling**.

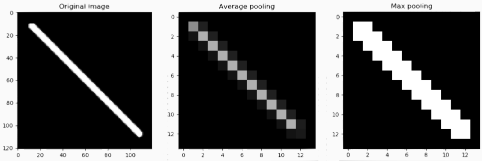


Most common approach is max-pooling.

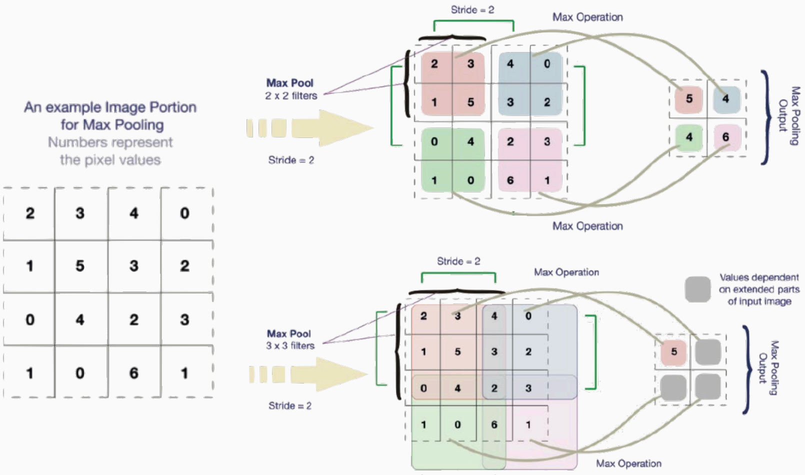
POOLING AND DOWNSAMPLING



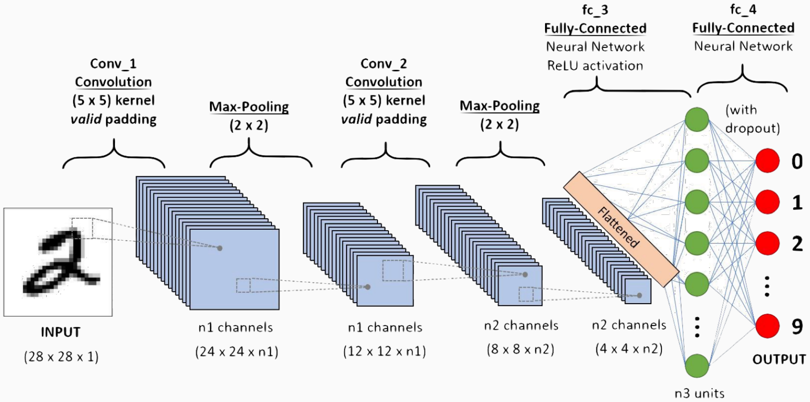
- Reduces number of variables, helps prevent over-fitting and speed up training.
- Helps “smooth” result of convolutional filters.
- Improves shift-invariance.



Many possible variations on standard 2x2 max-pooling.

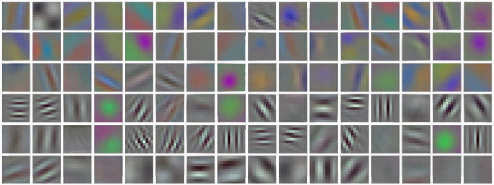


OVERALL NETWORK ARCHITECTURE

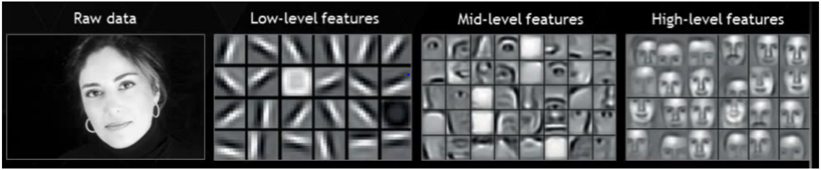


Each layer contains a 3D tensor of variables. Last few layers are standard fully connected layers.

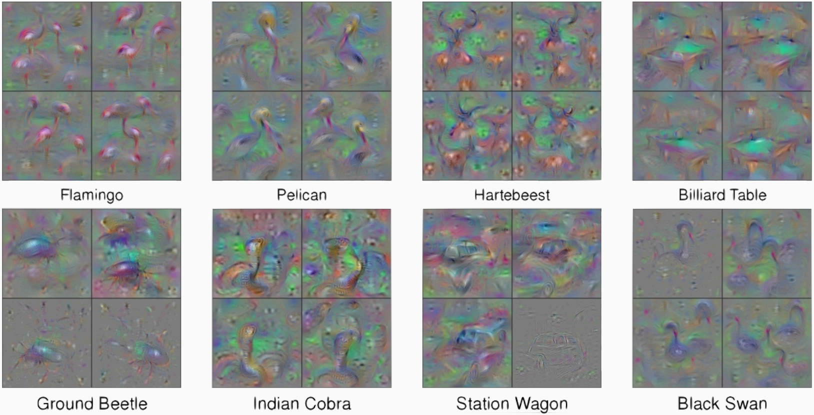
What type of convolutional filters do we learn from gradient descent? **Lots of edge detectors in the first layer!**



Other layers are harder to understand... but the hypothesis is that hidden variables later in the network encode for "higher level features":



Technique to probe later neurons: Use optimization to find images that most strongly "activate" a given neuron deep in the network.



"Understanding Neural Networks Through Deep Visualization", Yosinski et al.

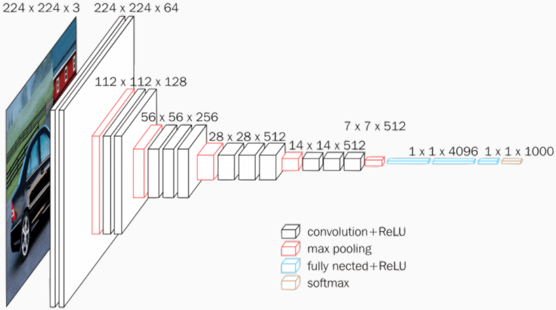
Beyond techinques discussed for general neural nets (back-prop, batch gradient descent, adaptive learning rates) training convolutional networks requires a lot of “tricks”.

- Batch normalization (accelerate training).
- Dropout (prevent over-fitting)
- Residual connections (accelerate training, allow for more depth – 100s of layers).

And convolutional networks require lots of training data.

What if you want to apply deep convolutional networks to a problem where you don't have a lot of data?

Idea behind transfer learning: features transformations learned when training a classifier on e.g. Imagenet are often useful in other problems, even with different inputs, classes, etc.

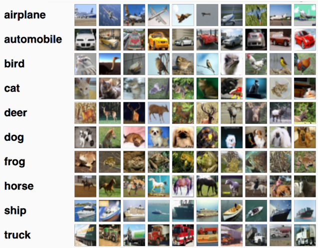


- Download state of the art pre-trained network (Alexnet, VGG, Inception, etc.)
- Chop off classification layer.
- Use first part of network as feature extractor.
- Solve classification problem using more scalable methods: kernel SVM, logistic regression, shallow fully connected net, etc.

Very easy to do in Tensorflow/Keras. Many pre-trained networks are made available.

Two demos to be released shortly:

- Classification of CIFAR-10 dataset using 2 layer neural nets in Keras. You will likely want to use Google Collab to access a GPU.



- Transfer learning in Keras.