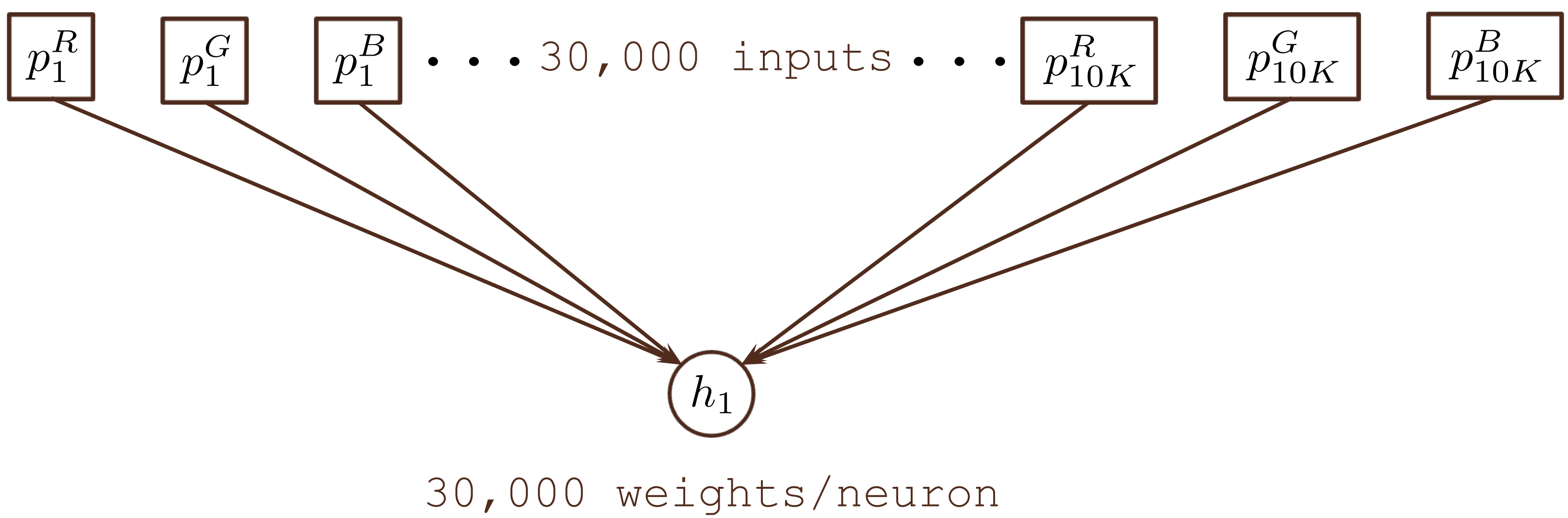
Class #14: Convolutional Neural Networks

Machine Learning (CS 135)

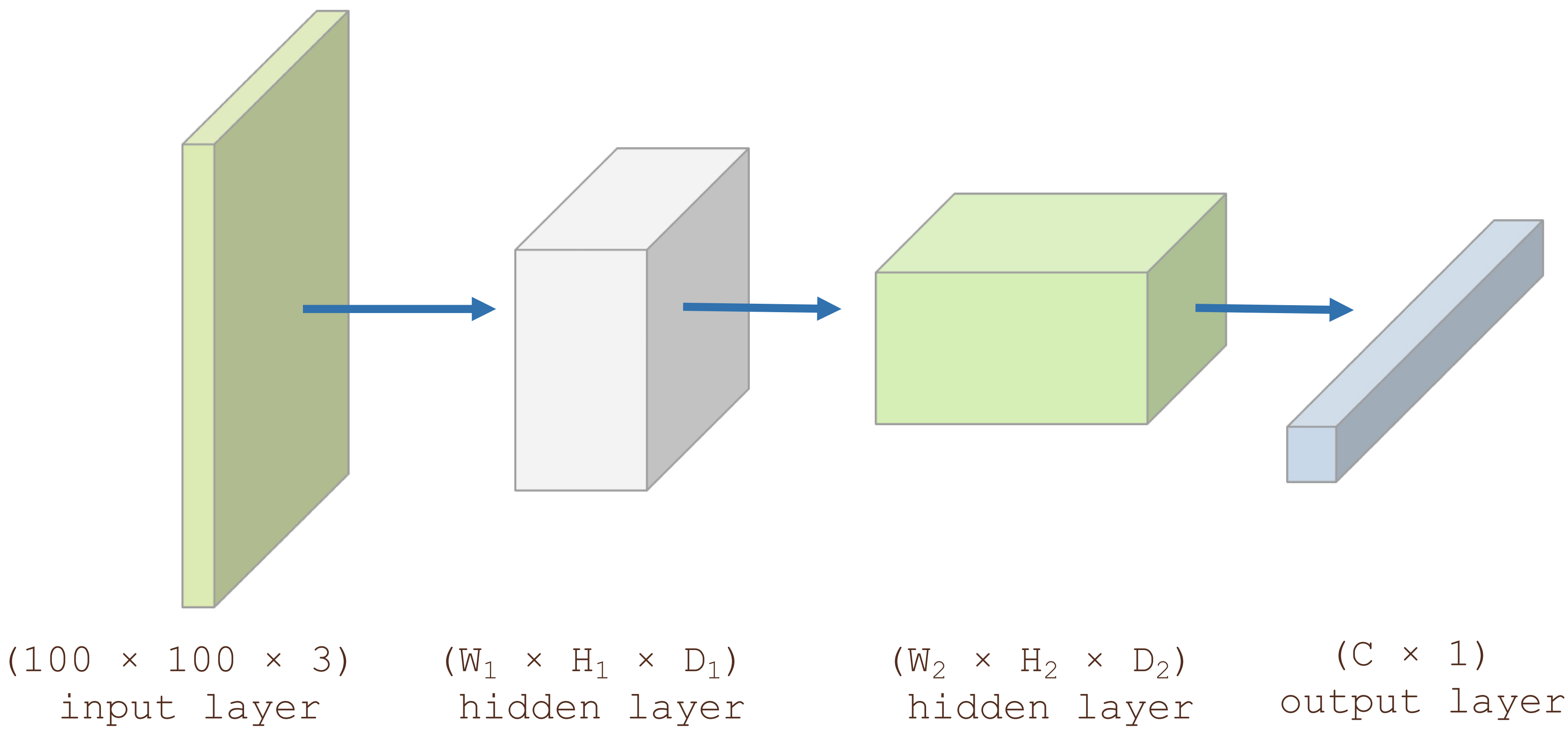
Neural Networks for Images

- A regular feed forward network can sometimes prove problematic for image-processing tasks
  - Given a (100×100) pixel color image, each with 3 color-channel (e.g. RGB) values, we end up with many, many weights to be learned
  - In addition, a 1-D weight-vector doesn't carry any real information about spatial relationships between image features (edges, blocks of color, …)



Convolutional Neural Networks (CNNs)

- **To capture image dynamics, and expand what the networks can do, we organize neurons into stacks of 3-dimensional volumes**
  - Each is connected to later volumes, filtering and flattening down to the usual final ($C \times 1$) classification-output layer (where C is the number of classes)

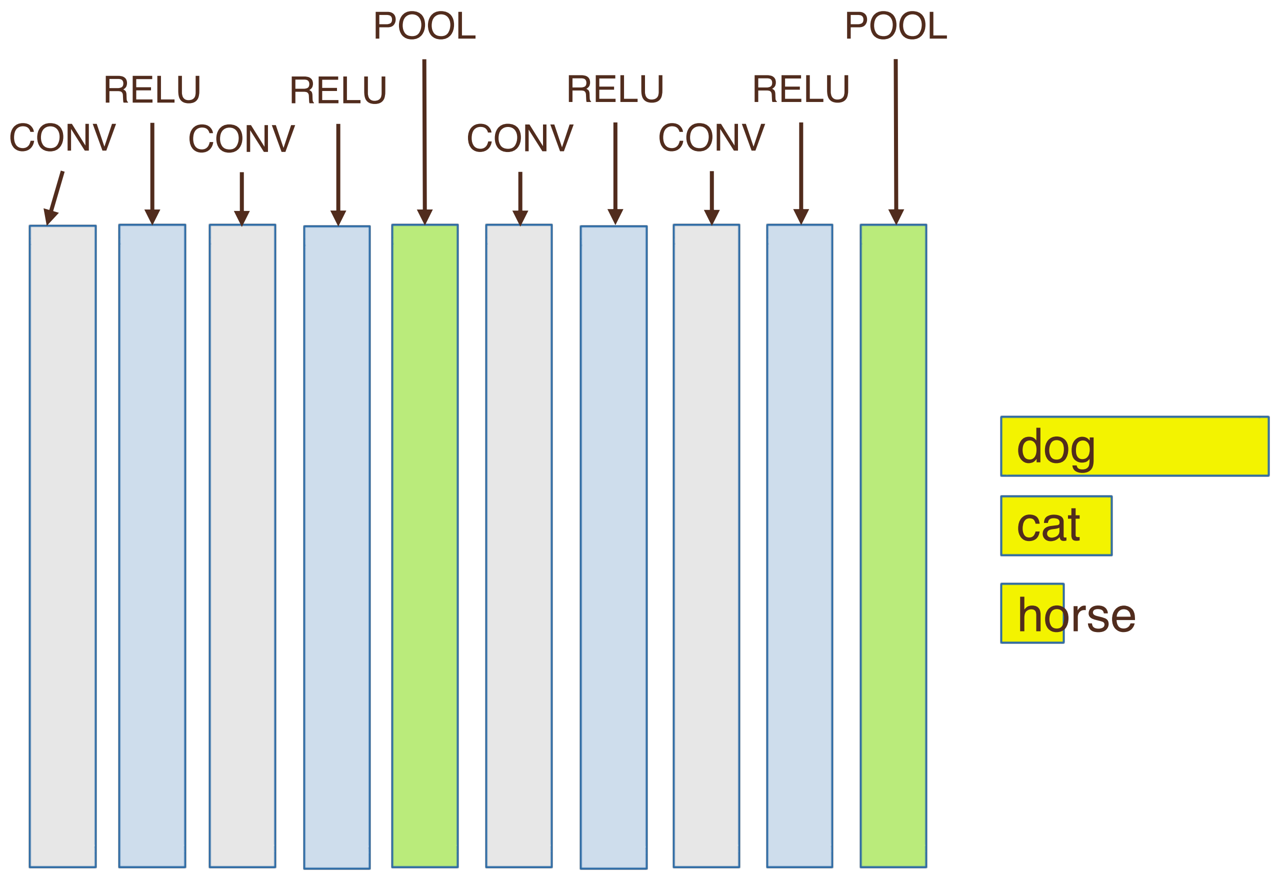


Types of Layers in CNNs

- INPUT: as in a typical NN, each neuron corresponds to a single input feature-value
  - Only the 3-D arrangement is different
- OUTPUT: again, as in a typical NN, these are fully-connected layers
  - Each neuron is connected to all of those in the volume above
  - Each computes a function, like the sigmoid (*softmax*), typically giving probabilities for each of the possible output classes
- OTHER: layers between can play different possible roles
  1. CONVOLUTION: transformations on sub-regions
  2. RELU: application of the $max(0, x)$ function
  3. POOLING: down-sampling to reduce volume size

Deep Convolutional Networks

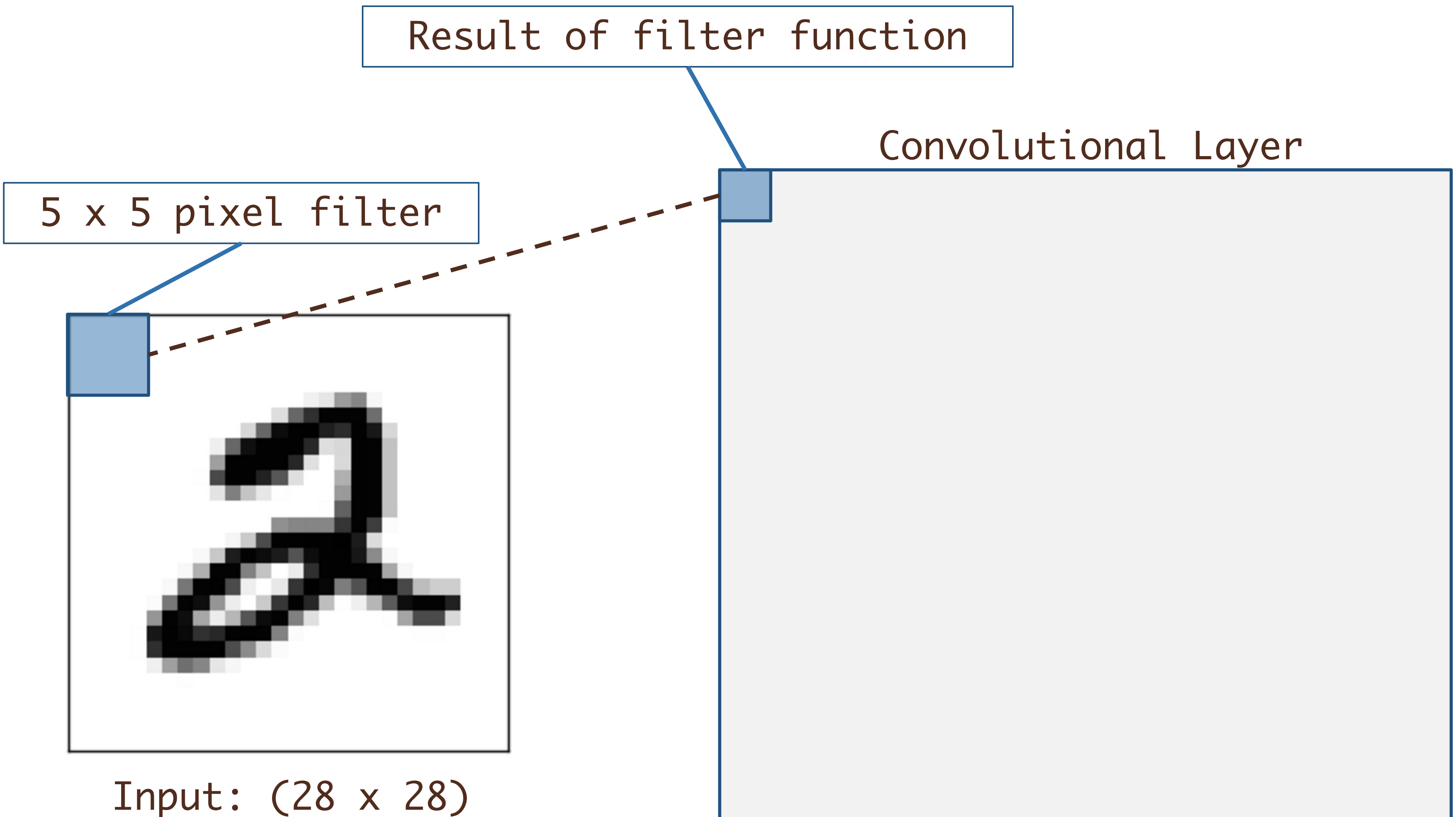
- For complex image-classification tasks, we may use many layers, combining the types in varying orders



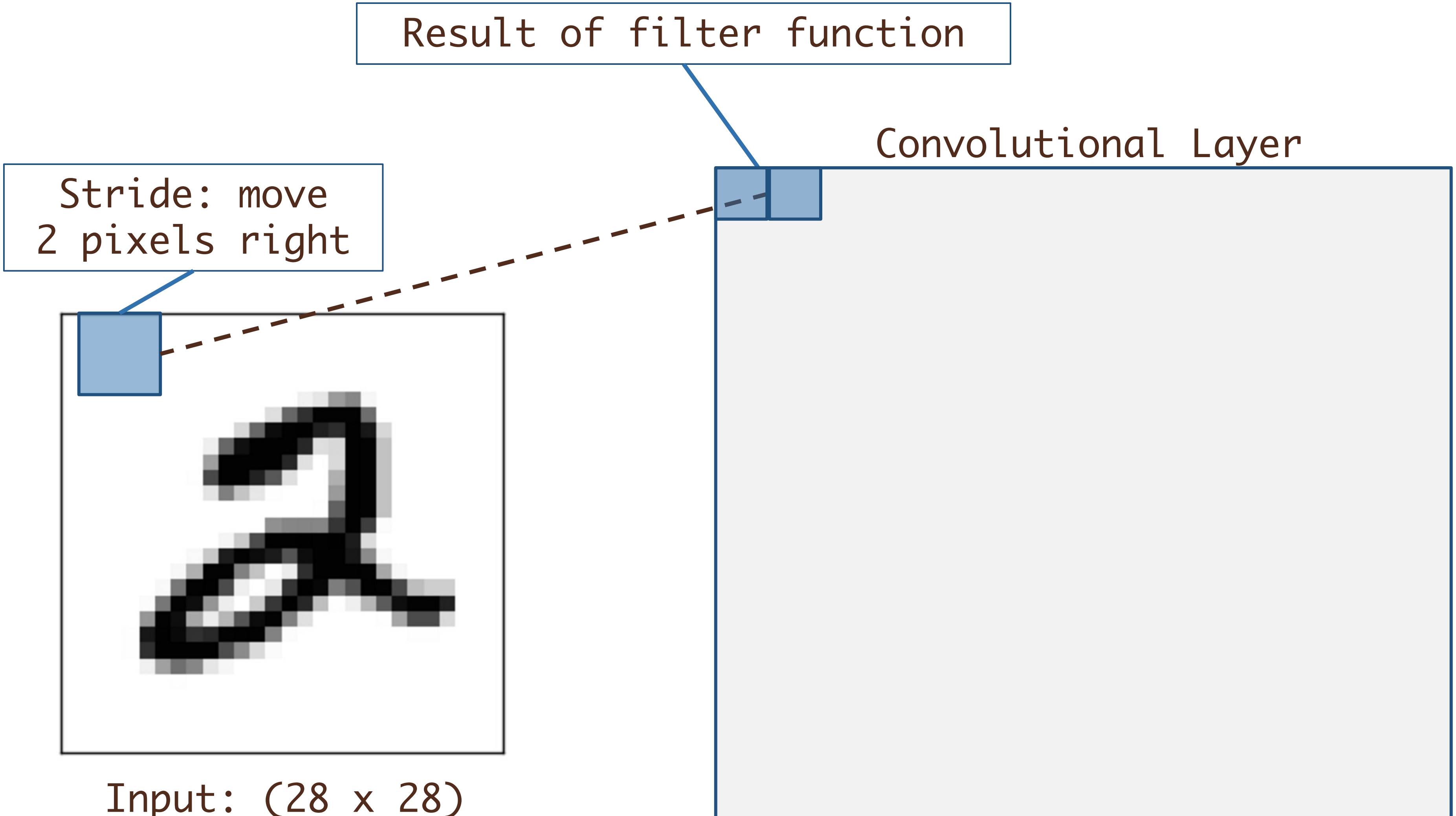
Convolutional (CONV) Layers

- The core innovation in a CNN is the idea of a spatial filter, which is a 3-D volume where:
  1. Each neuron in one layer computes a function on a proper ***sub-region*** of the layer above
  2. We form the CONV layer by "tiling" the prior layer, in (possibly) overlapping sub-regions
  3. Every neuron in one layer shares a ***single set*** of weights, and so computes the same function

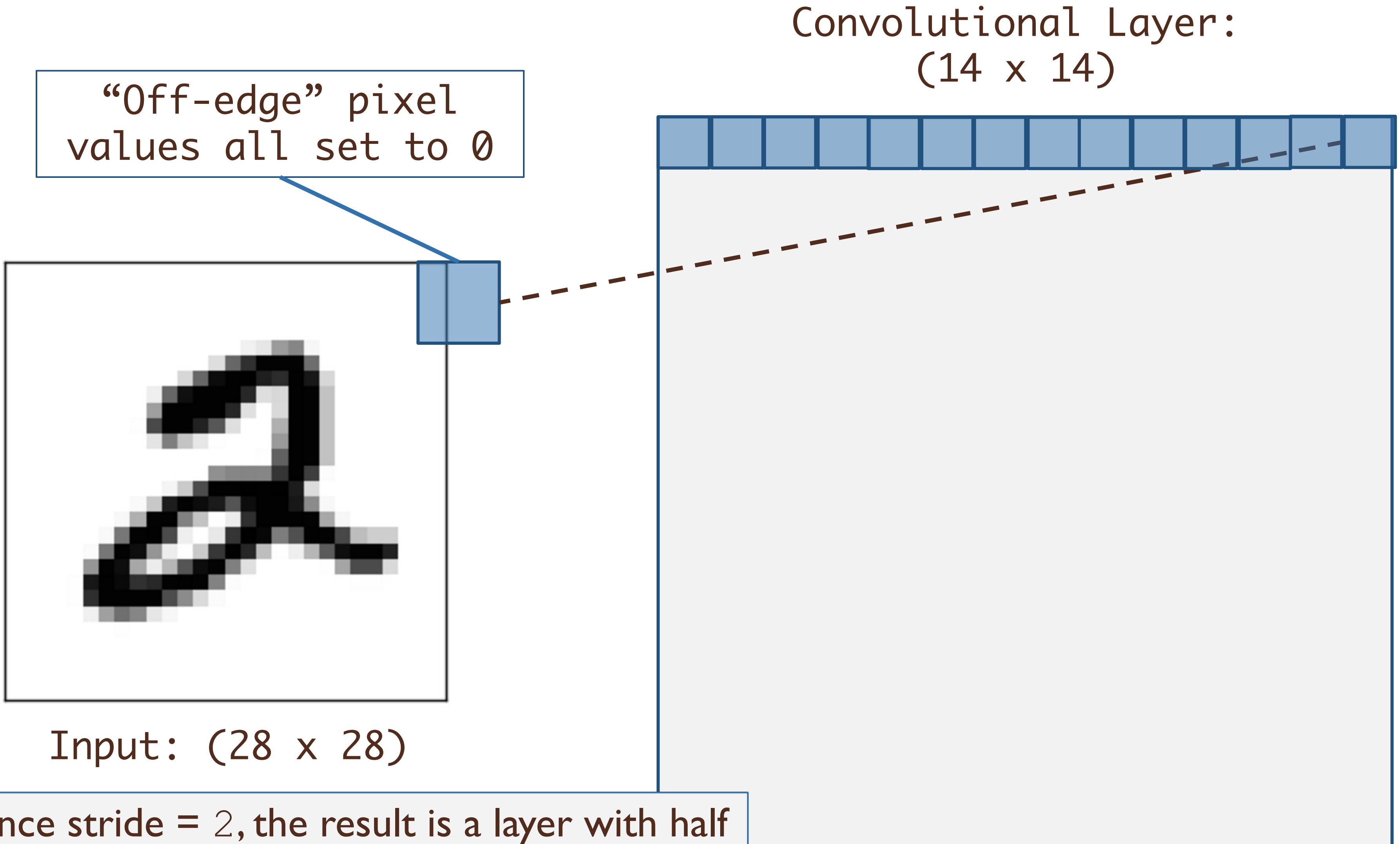
- Two main decisions in building such a layer:
  1. What ***size*** of sub-region should we use?
  2. What is our stride; i.e., ***how far*** do we move over each time we connect our next sub-region?



Suppose we choose a sub-region size of (5 x 5) pixels



Suppose we also choose a
stride-value = 2

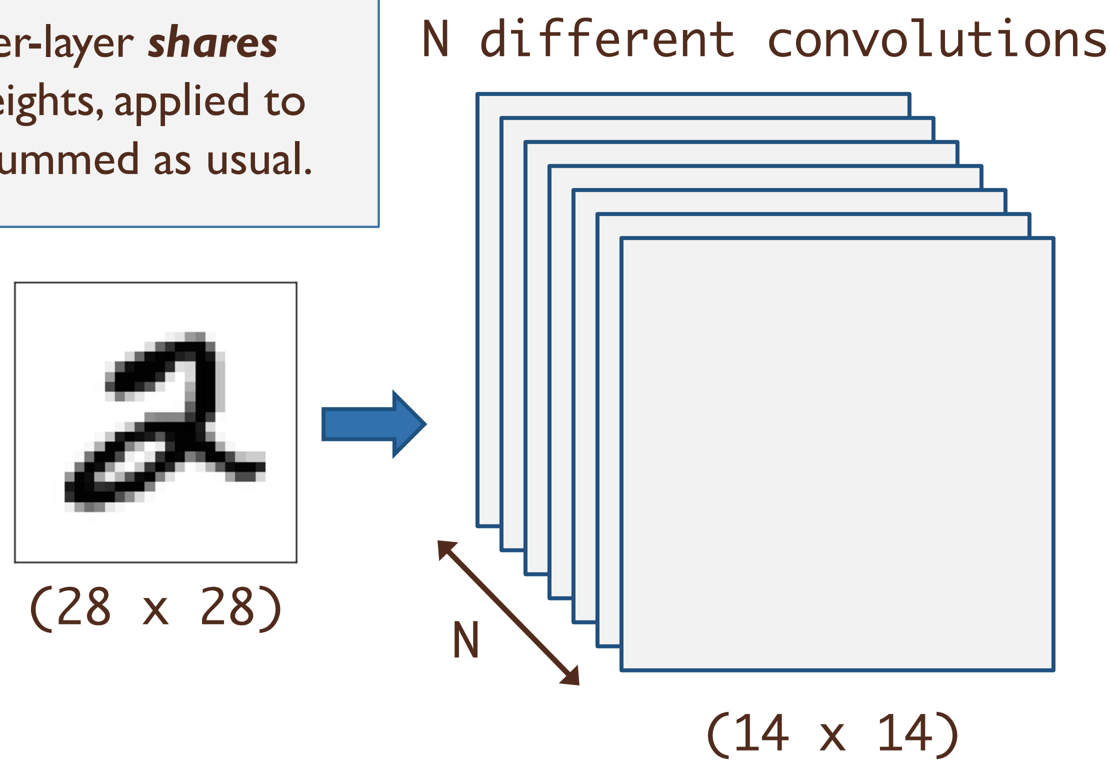


Since stride = 2, the result is a layer with half as many neurons in each dimension

A Full Convolutional Layer

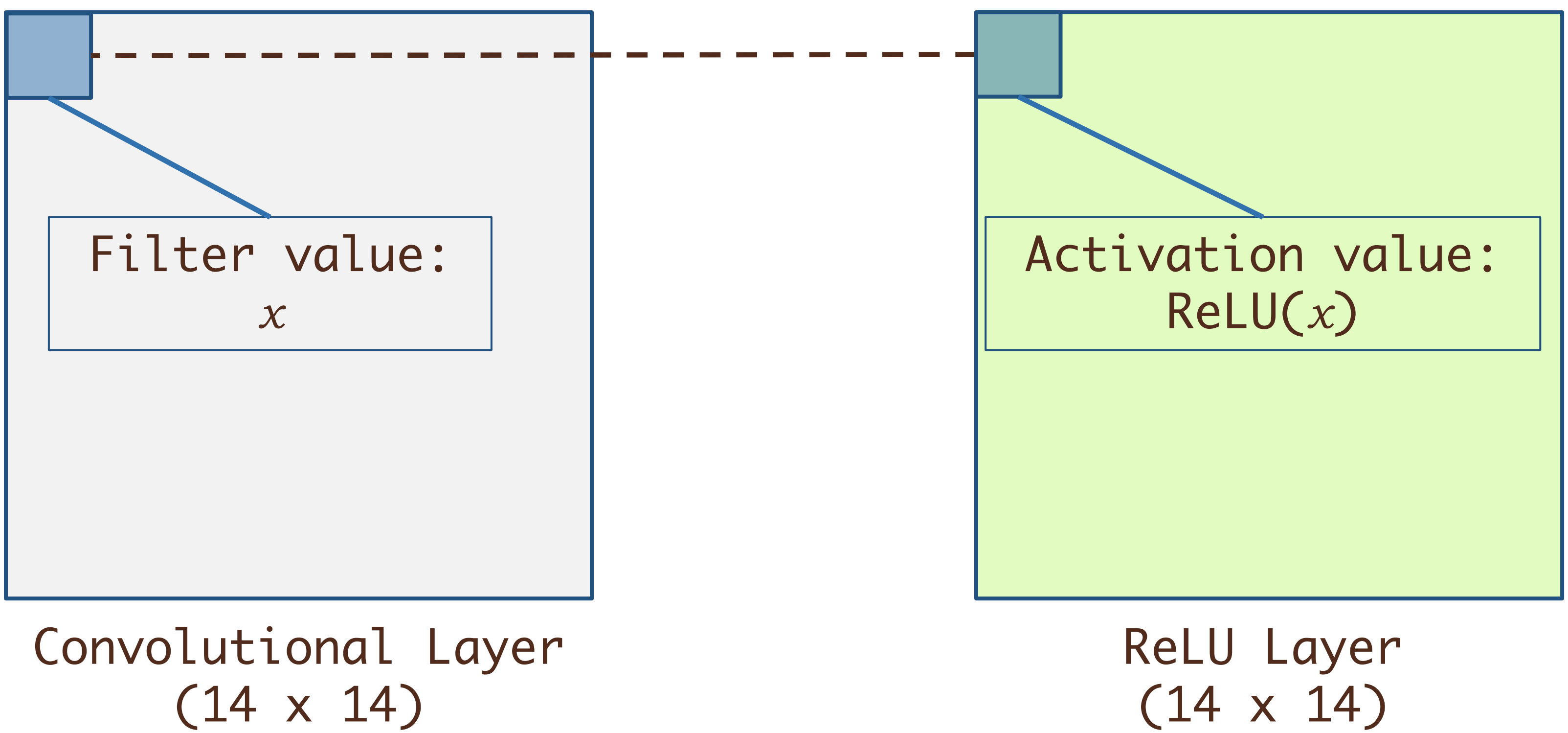
The 3-dimensional CONV layer consists of a stack of N such filters, of dimensionality:
`(14 x 14 x N)`

Every neuron in each filter-layer ***shares*** a single set of common weights, applied to inputs, with the products summed as usual.



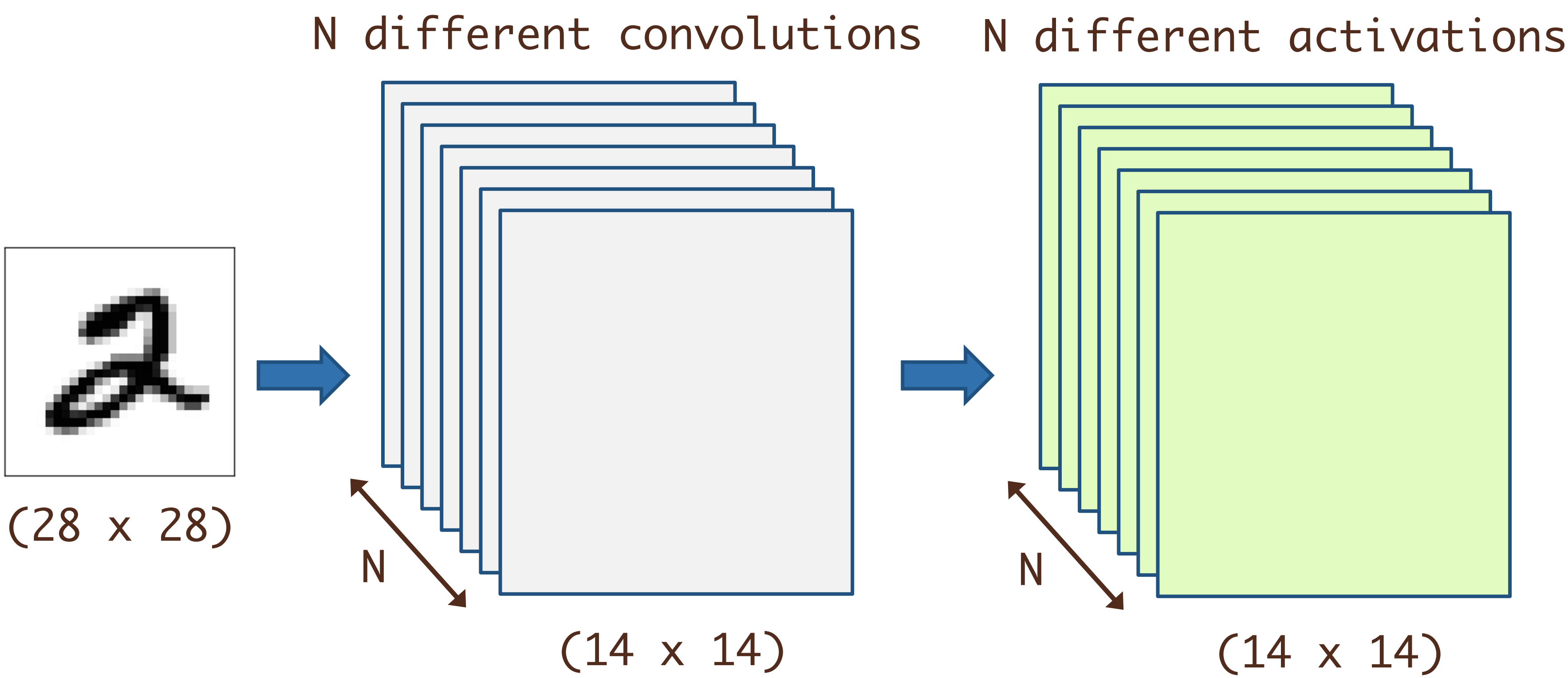
ReLU (Activation) Layers

- CONV layer may or may not change input size (depends upon stride)
- ReLU layer keeps size the same, simply applying its function to neurons
  - ReLU is very popular, but other activation function layers are allowed



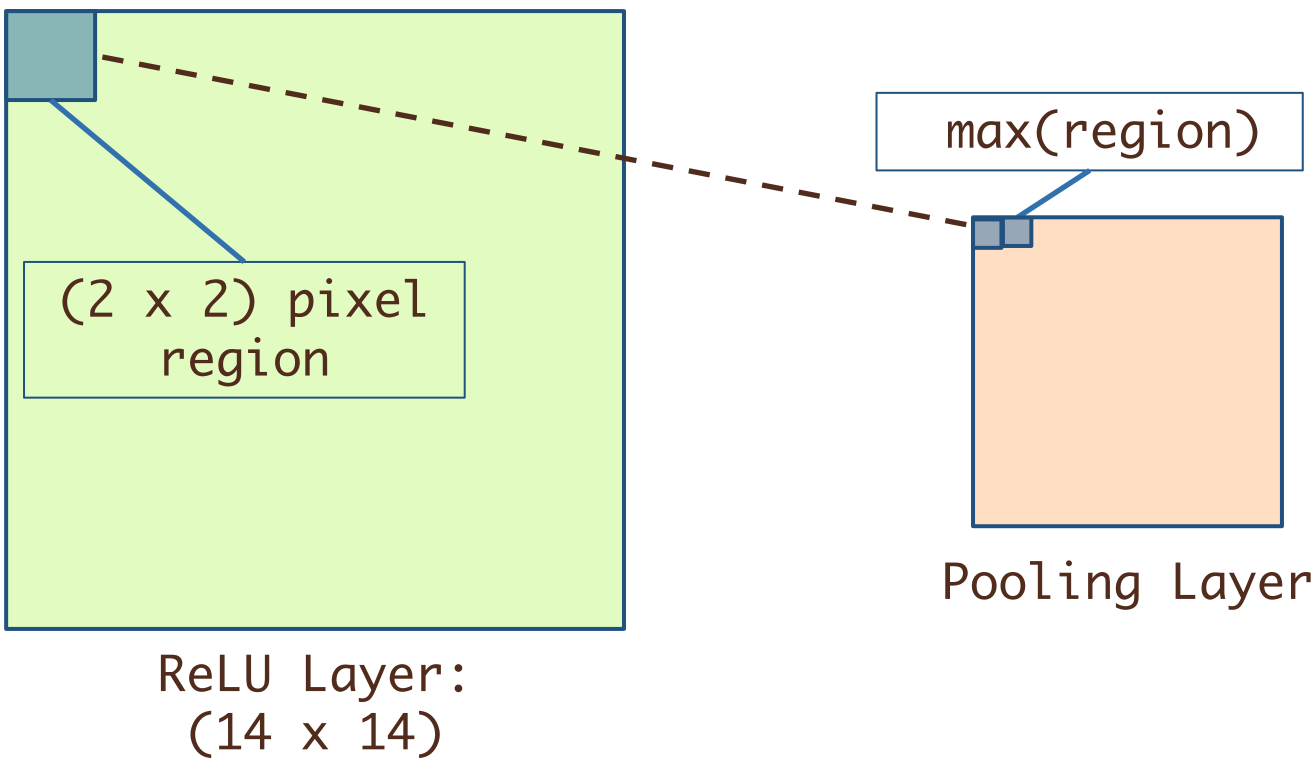
Combining Layers

Using a 3-dimensional convolutional layer of multiple filters means that we will have a matching number of activation layers.



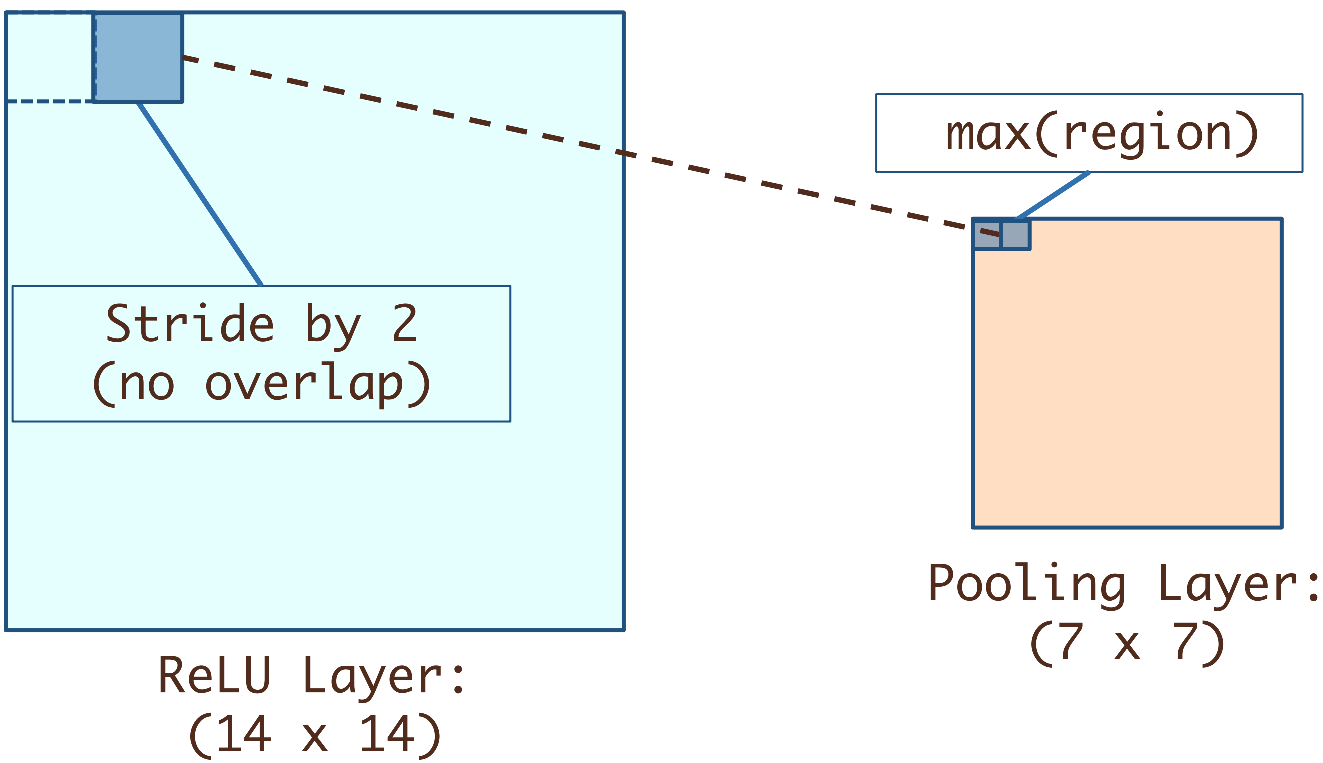
Pooling Layers

- While CONV and ReLU layers can compute more complex functions, POOL layers down-sample a region, reducing it to something simpler (usually its MAX value)



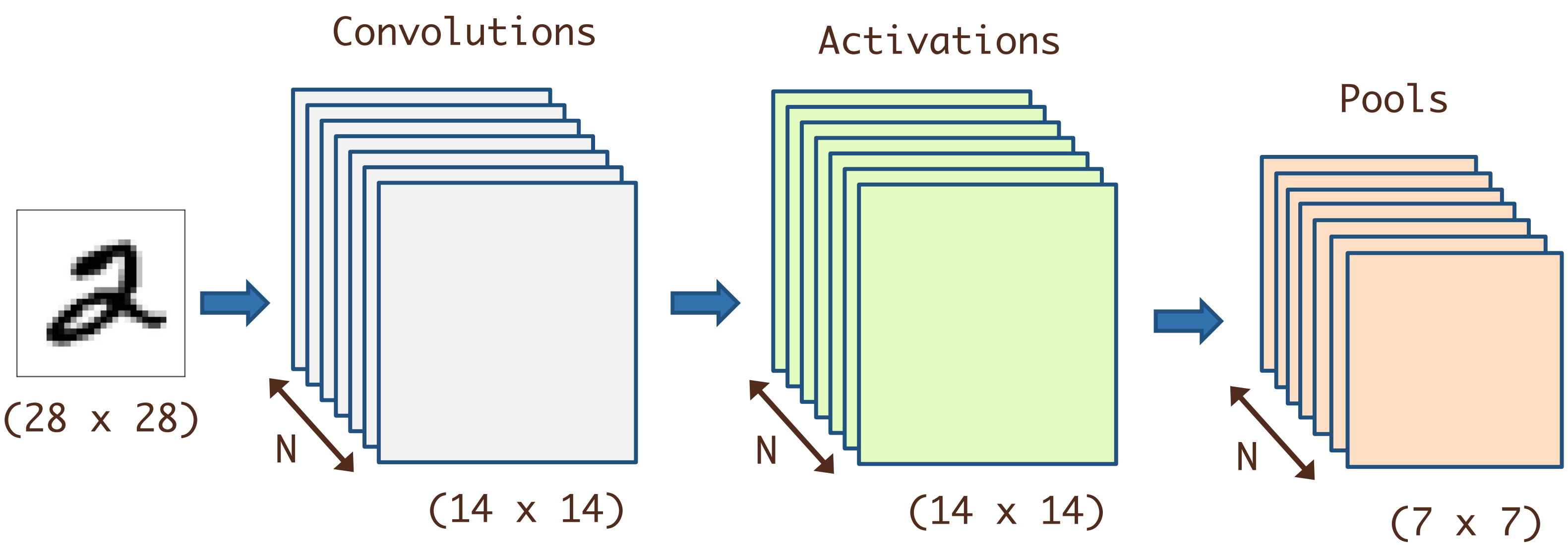
Pooling Layers

- Again, we stride across the layer, reducing the overall size by avoiding overlap
  - Most common approach: (2×2) region, with stride = 2



Combining Layers

Again, each layer is 3-dimensional (until the final output layer).



Uses of CNNs and Other Deep Networks

- Convolutional networks have become increasingly popular for image and other spatial data
- Browser-based demos:

  https://cs.stanford.edu/people/karpathy/convnetjs/
- A variety of applications of neural network models to a number of research problems

  https://youtu.be/Bui3DWs02h4

  https://youtu.be/hPKJBXkyTKM

  https://youtu.be/aKSILzbAqJs
- Cat drawings!

  https://affinelayer.com/pixsrv/