convolutional networks

- these convolution kernels (a layer of running a kernel thru the pixels of an image width-wise and then height wise) induce sparse interactions (only looking at a few pixels at a time), parameter sharing (same weights for the kernel over all pixels) and equivariant representations (weights that are invariant to changes in translation, etc)
- Stride is the step size as you move thru the pixels.
- CNNs also provide a way to work with variable size inputs.
- The result of the convolution layer is pooled with something like max pooling (just choose the output of the biggest feature map (kernel)). → we care more about if something is present rather than where exactly it is.
- convolution is an infinitely strong prior on parameter sharing and invariance to translation. if these assumptions aren't true to the domain, then this is a bad prior, just like any prior.
- convolution layers shrink the dimension of the input layer as it gets passed thru and this shrinking can happen pretty quickly. three kinds of convolutions:
  - valid convolution: shrinkage is greater than 0, so limits the number of convolutional layers that can be included.
  - same convolution: just enough zero padding around the image to keep the size of the output equal to size of the input. here the input pixels near the border influence fewer output pixels than the input pixels near the center. so the input pixels are somewhat underrepresented.
  - full convolution: enough zeros are added for every pixel to be visited k times.
- a locally connected layer (unshared convolution) → when we know that each feature should be a function of a small part of a space but there is no reason to believe that the same feature should occur across all of space.
- tiled convolution: learn a set of kernels that we rotate thru as we move thru space.
- CNNs for structured outputs, like labeling every pixel in an image. These emit a tensor instead of a single class label. one strategy is to produce an initial guess of image labels, then refine using the interactions between neighboring pixels. Repeating this refinement sever times corresponds to using the same convolutions at each stage, sharing weights between the last few layers of a deep net → a particular kind of recurrent neural net! the general idea is to have a prior that says "large groups of neighboring pixels usually belong to the same class label".
- we could also do specialized methods where we hand design kernels or learn them in an unsupervised fashion.
- CNNs have a lot of parallels with how the brain's visual cortex system works → they are an example of a successful application of brain principles to machine learning.