Convolutional Neural Networks

Deep Learning Decal

Hosted by Machine Learning at Berkeley

Overview

Agenda

Drawbacks of traditional nets

Neural Net drawbacks

- Neural Nets have no concept of proximity when it comes to features of an image. The neural net looks at the pixels individually rather than looking at them together.
- When we look at images, we don't just look at things as a collection of pixels, we take groups of those pixels and put them together, for example edges.

Convolutions explained

How to fix this? Convolutions!

- To understand what a convolution is lets look at an analogy with a spaceship where we want to track its position over time which we represent with the function $x(t)$.
- We have a noisy laser sensor which will give us the position, in order to make it less noisy we want to take a weighted average of the sensor measurements over time. We can update our estimation of the position by

$$s(t) = \int x(a)w(t-a)da$$

.

Example cont.

- In this example we need $w(a)$ to be a function which is non-negative across the domain.
- w also needs to be a valid probability density function, meaning that

$$\lim_{a \to \infty} \int_{-\infty}^{a} w(x)\, \mathrm{d}x. = 1$$

.

- The restrictions on w are only true for are particular example and in general any function w may be used as long as the integral

$$\int x(a)w(t-a)da$$

is defined. When talking about convolutions x is referred to as the input and w is referred to as the feature map or kernel.

How convolutions are actually done

- In practice we don't perform convolutions using the integral formulation because computer data is discretized. Our new formulation will be:

$$s(t) = \sum_{k=-\infty}^{\infty} x(k)w(t-k)$$

- When we run convolutional neural nets the goal is to learn what our kernel should be to properly capture feature interactions which in or case is $w(t)$.

Multi dimensional examples

- Suppose we are operating over a two dimensional image I and kernel K then we have

$$S(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

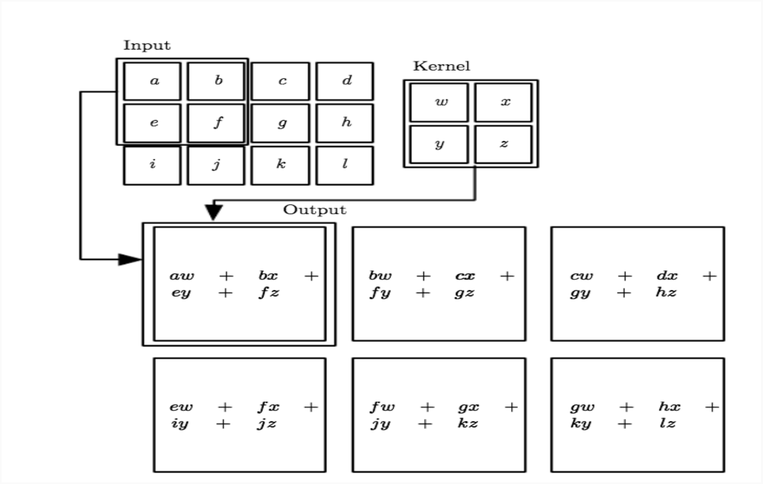
- If this was 3 dimensional instead we would have

$$S(i,j,h) = \sum_h \sum_m \sum_n I(m,n,l)K(i-m, j-n, h-n)$$

- Often we are convolving over 4D tensors (batchsize, channels, height, width)

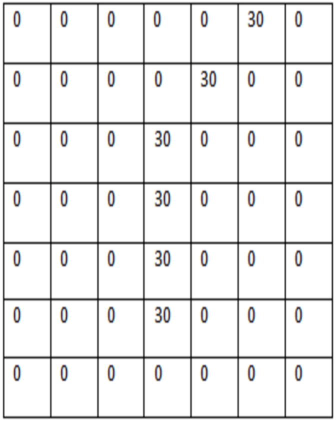
Convolution in pictures

- Lets look at an ex-
  ample of a convolution on an actual input with an actual kernel.



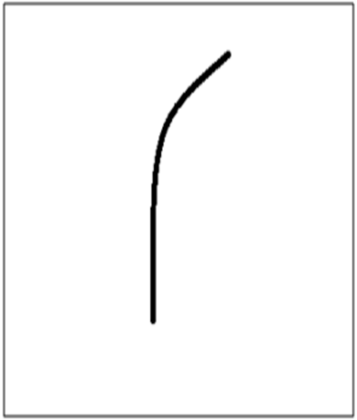
Why are kernels useful?

- We have spent time talking about kernels without really explaining why they are useful.



0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

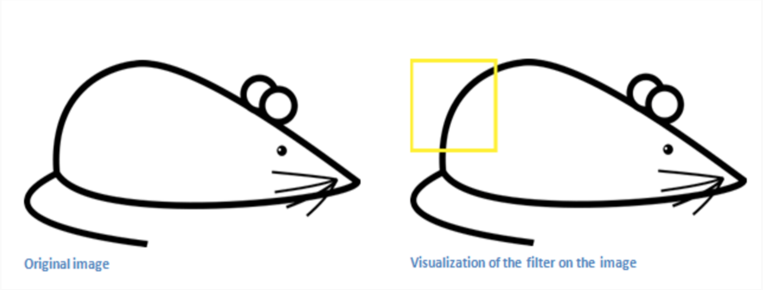
Pixel representation of filter



Visualization of a curve detector filter

Why are kernels useful? cont

- Above was an example of a kernel representing an edge. Lets see how that fits on an actual image.



- Here we can see that the kernel above is a part of an image. Lets see what happens when we convolve the filter with the relevant part of the image.

Why are kernels useful? cont

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

$*$

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

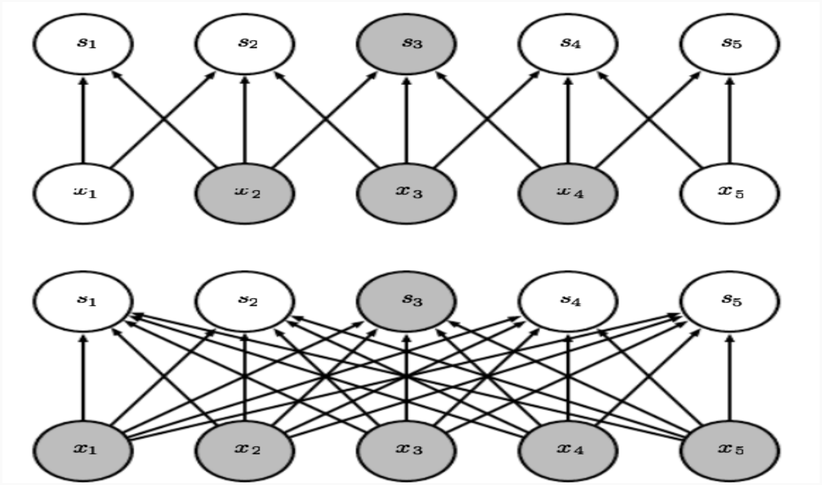
- Above we see that when our filter is similar to the part of the image the result of the convolution is a large number.

Advantages of Convolutions

Sparse interactions

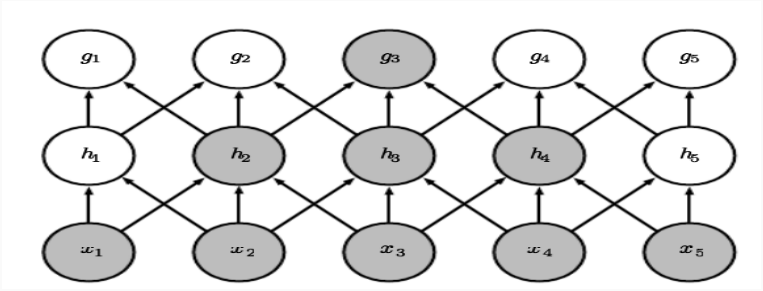
- Suppose we are processing an image with tens of millions of pixels, with traditional neural networks that will mean we have correspondingly many weights.
- If we take a convolutional approach we can detect meaningful features with square kernels with only tens or hundreds of features.
- This can vastly speed up algorithms while maintaining good accuracy.
- Also with fewer parameters that makes models less likely to overfit and make them more generalizable .

Sparse Interaction in Pictures



Sparse Interaction Continued

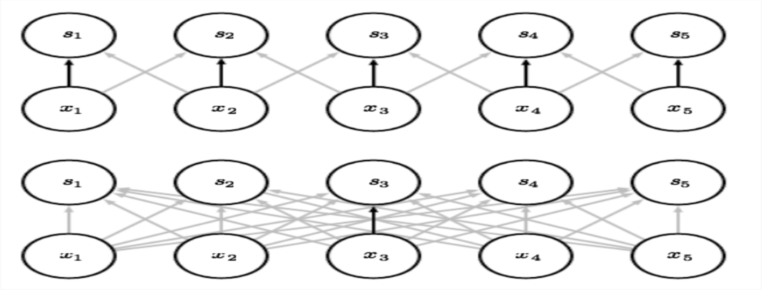
- As the example shows for s_3 above it is only connected to x_2, x_3 and x_4 whereas below it is connected to x_1 through x_5. The units which affect a particular neuron are called the receptive field.



- We see that the receptive field for g_3 includes every neuron in the first layer despite using convolutions instead of fully connected layers.

Parameter sharing

- In fully connected layers all parameters are separate. This is not true for convolutional layers.



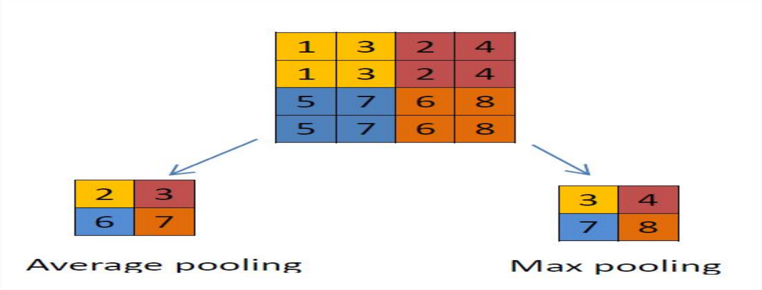
Varied image sizes

- With fully connected nets all images that are fed into the net have to be the same size.
- With convolutional nets the kernel can be applied a different number of times and the output can be scaled appropriately.

Pooling

Pooling

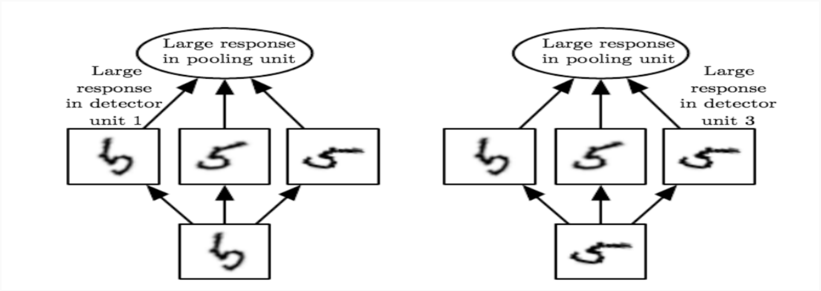
- Another operation which frequently occurs after convolutions is pooling. Pooling replaces the output with a consolidation of nearby inputs.



Advantages of pooling

- Reduces the dimension size of our input.
- Makes our input become more invariant to small translations such as horizontally or vertically translating it.
- Provides consolidation mentioned so small translations will be ignored.
- Helps reduce dimensionality to output a fixed number of outputs regardless of the initial size.

Invariance illustrated


Large response in pooling unit
Large response in detector unit 1
Large response in pooling unit
Large response in detector unit 3

- Digit 5 is rotated in different ways and different filters are able to pick it up depending on the orientation.

Convolution variants

Padding

- In some cases we want to skip over certain kernel positions in our convolution, we can shift it over by s where s is our stride.
- To make sure that we don't limit ourselves to sub regions of the image where the kernel is entirely contained we can zero pad the image independently controlling kernel width and the output.
- If we limit ourselves as mentioned above that is called a "valid" convolution.
- If we specify the convolution to have the same output size as the input that is called a "same" convolution.

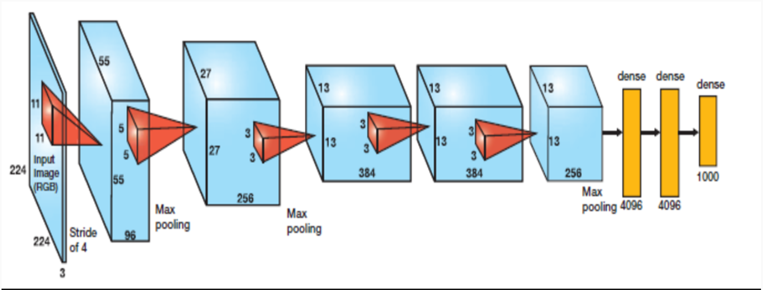
Designing Convolution Kernels

- When it comes to initializing convolutional kernels there are a few different approaches we can take:
- Start with random kernels and have the network learn good ones over time.
- Set kernels to detect specific features such as edges and orientations.
- Learn kernels in an unsupervised manner. One example of this is applying k-means clustering to small image patches and using each learned centroid as a kernel.

Convolution Architectures

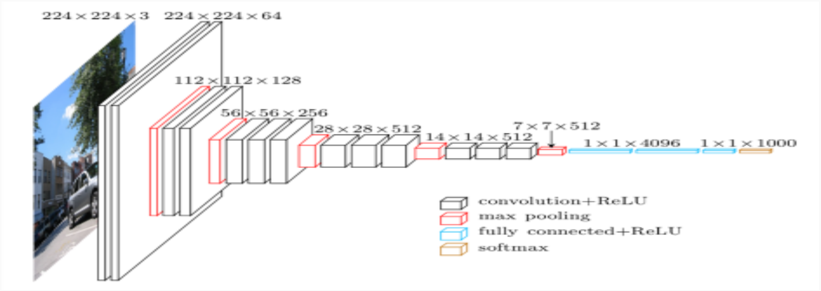
AlexNet

- AlexNet has 5 convolutional layers followed by 3 fully connected layers.
- To combat overfitting a dropout layer is used between each fully connected layer.
- After each fully connected layer, ReLU activation function is used.



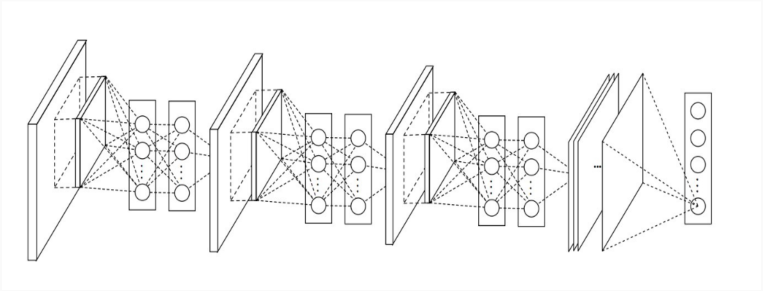
VGG Net

- VGG net improves upon AlexNet by replacing the large kernels of size and 11 and 5 with smaller 3x3 kernels.
- Larger receptive field due to stacking kernels



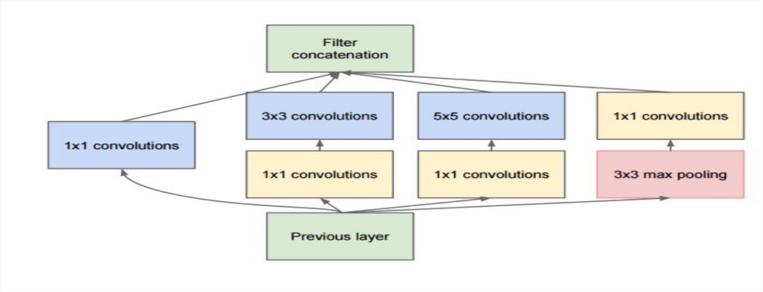
NiN

- This is a special layer which uses 1x1 convolutions before fully connected layers.
- Spatially combine features across different feature maps.



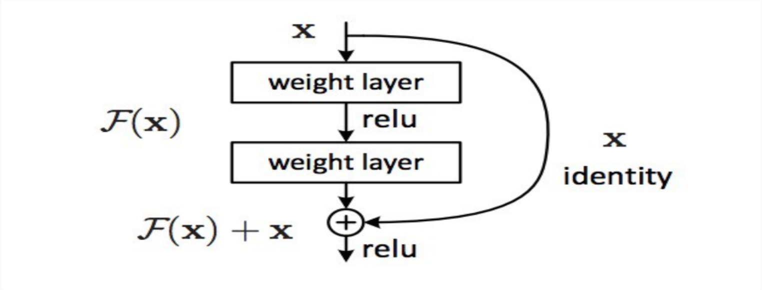
Inception

- Builds upon NiN and does multiple convolutions in parallel and combines them together
- Used 1x1 convolutions to reduce features before expensive parallel blocks



ResNet

- This architecture takes the input to a convolution and concatenates it with the input two convolutions later.
- Bypassing by two layers makes a small classifier.
- First time network of $>$ 100 layers was trained.



Questions

Questions?