

McGill **Artificial Intelligence** Society



# Lecture 6: Convolutional Nets

Slides based off of Machine Learning at Berkeley

<https://github.com/mlberkeley/Machine-Learning-Decal-Fall-2018>




## Announcements

Assignment 4 should have been handed in by now

Assignment 5 will be out over the weekend (you'll have 2 weeks again)

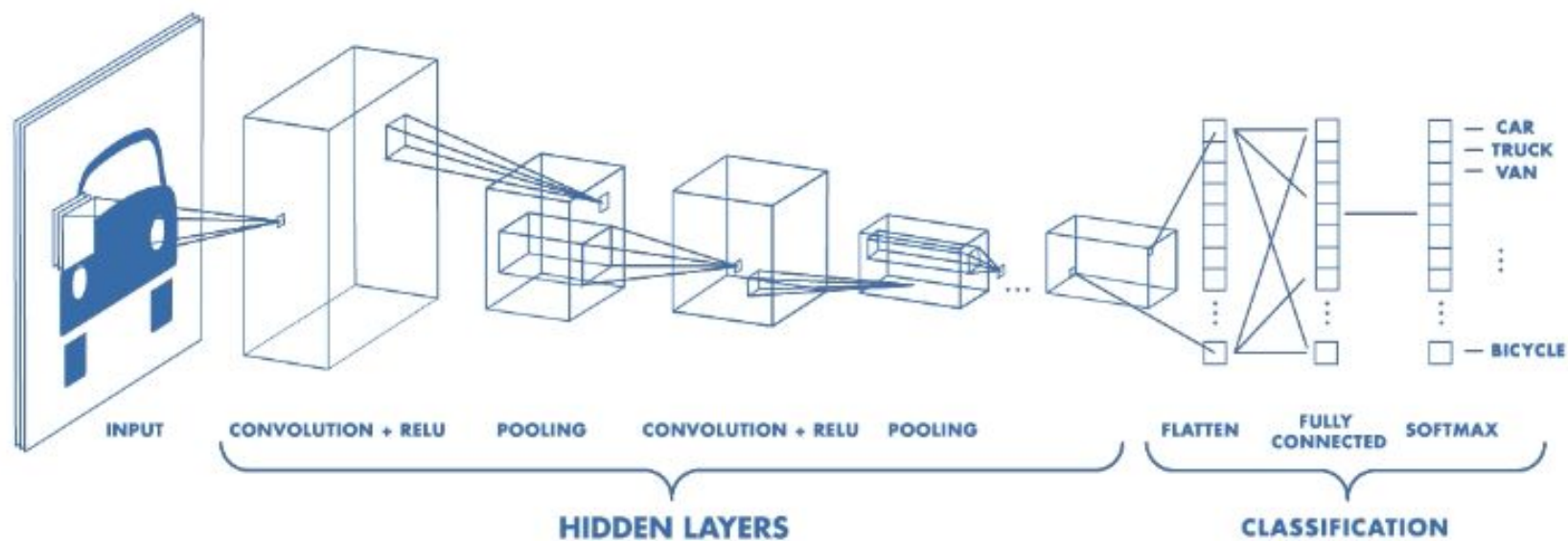
Work on your project over the break! Leave time for webapp integration or poster creation.

Who's down for forum (bowling, pool, food, etc)?



# Today's Lesson Plan

1. Convolution Operation
2. Motivation for using Convolutions
3. Pooling
4. Probabilistic Interpretation from a Bayesian Perspective
5. Fast algorithms
6. Neuroscientific Basis
7. Applications



# Sensor Problem

Say we have a noisy sensor, but want to attain accurate data

## **Solutions:**

1. Could calculate variance of data, and model a distribution
  - a. Assumes equal weights of all data in past  $t$  timesteps
2. Could average past  $t'$  data points
  - a. Same problem as above
3. Weighted average of data, giving more weight to recent data
  - a. How?

# Convolution Operation

This is where the convolution operation comes in

Definition:

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

or

$$s(t) = (x * w)(t)$$

# Convolution Operation

X - Input

W - Kernel (we'll talk more about this in a moment)

As continuous time measurements isn't realistic, we also have a discrete version (replace integral with sum)

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

# Convolution Operation in Images

Images are 2D, thankfully, it is a trivial extension

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

Flip it, and we get

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

Which is easier to implement, and is correct due to commutativity



# What's wrong with FeedForward?

Assume you have a 256x256 image, that's 65536 parameters

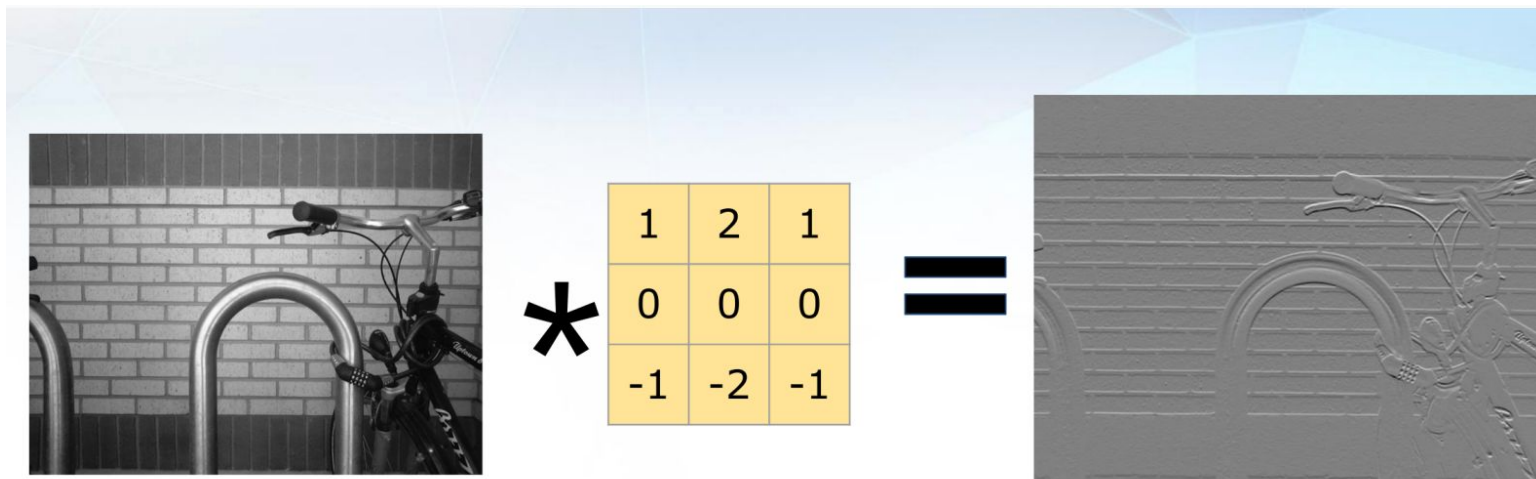
Take a hidden layer with 1000 neurons -> over 65 million parameters

Two hidden layers with 15000 neurons -> over 1 billion parameters

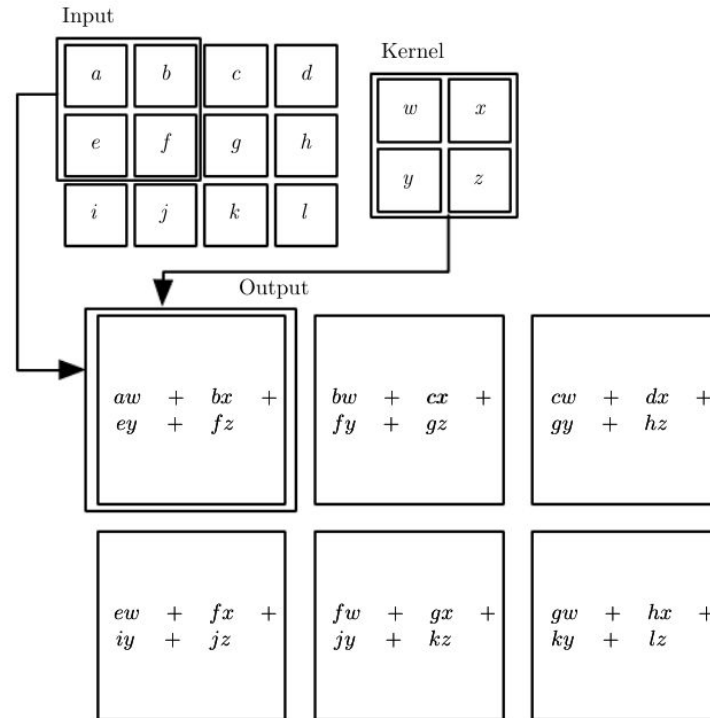
Call this the curse of dimensionality

# Solution?

## Convolutions



# Physical Example of Convolution Operation



# Convolution Operation

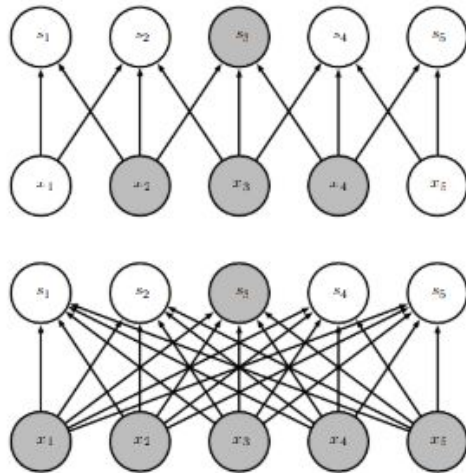
## Three Advantages:

- Sparse Interactions
  - Not every weight has to interact with every other weight, hence reducing computational load
- Parameter Sharing
  - The kernel is shared across pixels, once again, reducing computational load
- Equivariant representation
  - Due to the property of the convolution, it doesn't matter where the features the filter seeks are.

# Sparse Interactions

## Advantages:

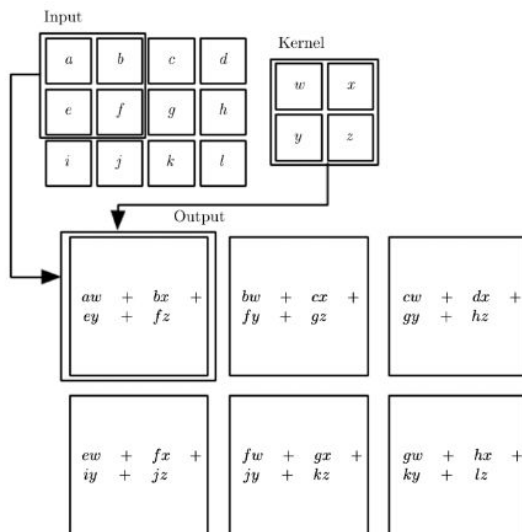
- In Feedforward nets, where you have  $m$  inputs and  $n$  outputs, the standard matrix multiplication will have a  $O(m*n)$  computation time
- Now, take a much smaller kernel with dimension  $k*k$ , then the matrix multiplication runtime will be  $O(k*n)$

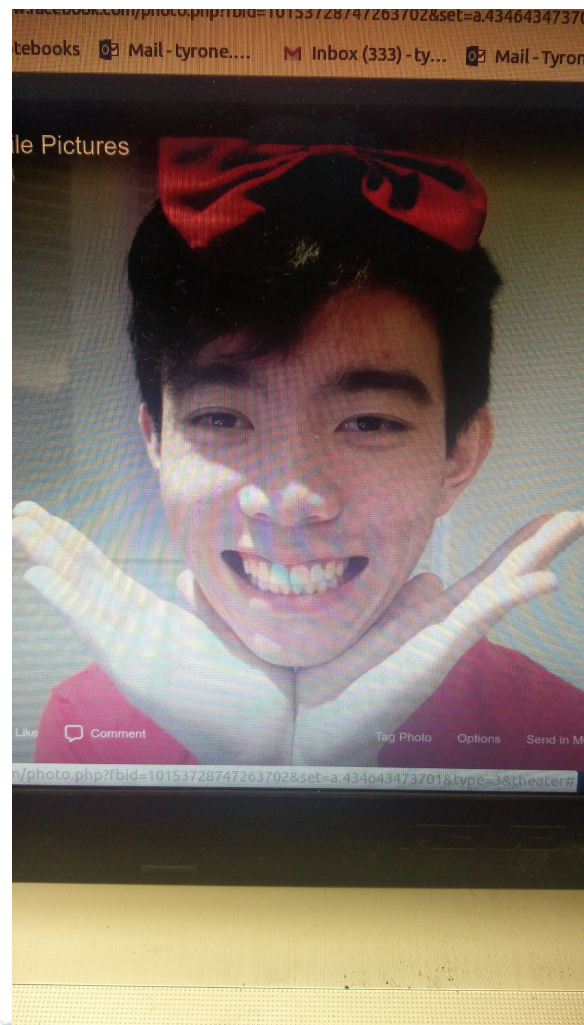


# Parameter Sharing

## Advantages:

- In a standard feedforward neural net, the kernel is passed through the entire image, so instead of having each feature (read: weight) be associated with a trainable weight, the parameters in the kernel are passed through the entire image





# Equivariance

## Advantages:

- Direct result of parameter sharing in convolution operation
- Define function  $f$  to be equivariant to function  $g$  if  $f(g(x)) = g(f(x))$
- Theorem: Let  $g$  be a function that translates (shifts) input, then convolution function is equivariant to  $g$
- Example
  - Let  $I$ : image brightness  $\rightarrow \mathbb{Z}^2$  (integer coordinates)
  - Let  $g: \{ I \rightarrow I' \mid I'(x,y) = I(x-1, y) \}$
  - Then  $\text{conv}(g(I)) = \text{conv}(I)$
- Not equivariant to changes in scale of image or rotations in the image, so we do those via data augmentation.

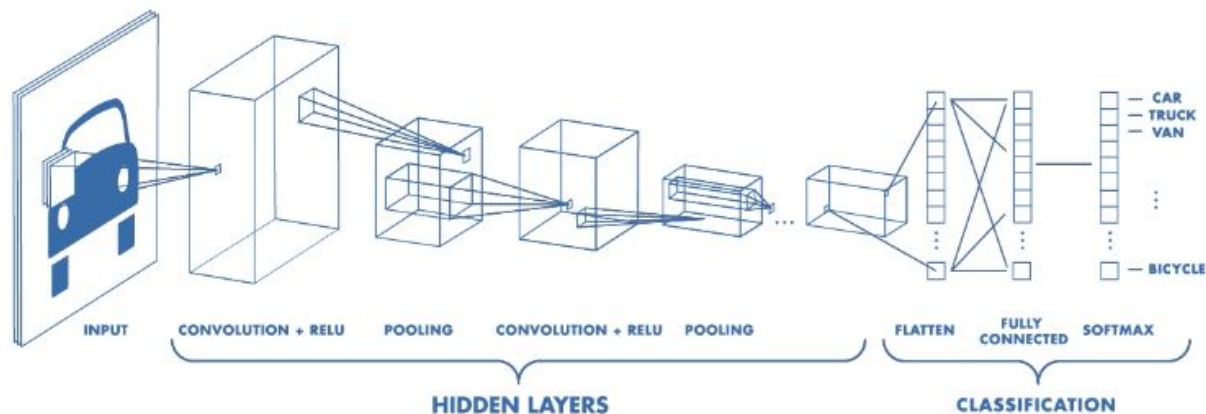


# Big Picture

Let's step back for a second

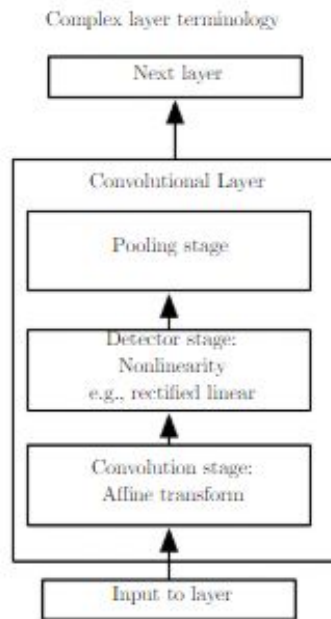
Convolutional Neural Nets (CNNs) have 3 stages

1. Network performs several convolutions in parallel in order to produce a set of linear activations
2. Each linear activation is run through a non-linear activation (e.g. ReLU)
3. Pooling



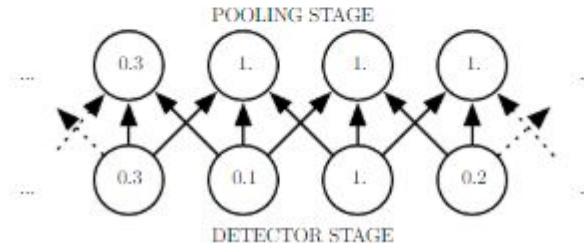
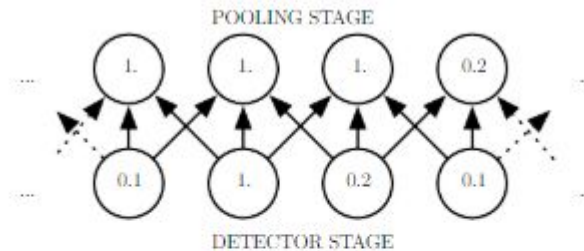
# Pooling

- Pooling essentially takes an area of the net and outputs a summary statistic of that area
- Examples of pools
  - Max Pooling (most popular): Takes the maximum value of a rectangular area
  - Average Pooling
  - L2 Norm
  - Weighted average based on distance from center



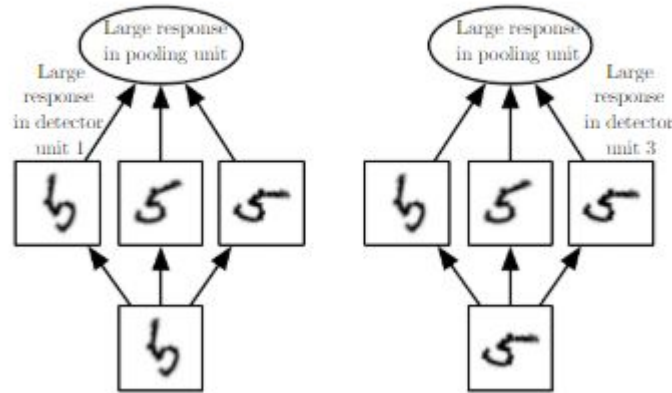
# Advantages of Pooling

- Approximate Local Invariance to Translation
  - But I thought you said the convolution operator already made us invariant to translation?
    - Yes, but local invariance to translation means that things of relative distance, such as distance between eyes of a face, would affect the net less



# Advantages of Pooling

- Pooling over spatial regions can involve invariance to translation
- Yet we pool over sequentially outputted convolutions, why?
  - Doing so allows our model to select which transformations to be invariant to



# A Probabilistic Perspective

- Convolution and Pooling can be seen as infinitely strong priors
  - I.e. certain parts of the probability density are completely abandoned
- We can say that CNNs and FFNNs are the same, except CNNs have an infinitely strong prior over its weights
  - Prior dictates that weight for one hidden unit must be identical to weight of its neighbor but shifted in space
  - Prior also dictates that the weight must be 0 except for the small spatial area dictated by its receptive field.
- Of course, it'd be manically insane to implement an FFNN with infinitely strong priors due to the computational constraint, but this gives us some insight

# Key Insights

1. This prior can cause underfitting. Priors are only useful if the assumptions they make are accurate.
  - a. For example, if a task requires preserving the spatial information, then using pooling on all features can increase training error
    - i. There are networks that exist that use pooling on some channels and not others, in order to get both translation invariance AND translation variance when that assumption is incorrect
    - ii. <https://arxiv.org/pdf/1409.4842.pdf>
2. Another key insight is that we should compare only conv nets with other conv nets since they share the same priors

# Fast Algorithms

1. Fast Fourier Transforms
  - a. <https://arxiv.org/abs/1412.7580>
  - b. Theoretically awesome, but only applies to large filters (kernels)
2. Winograd
  - a. Newer algorithm and is the state of the art, works great for small kernels (specifically designed for 3x3 kernels)
  - b. <https://arxiv.org/abs/1509.09308>

# Neuroscientific Basis

- Despite the name, neural networks share very little similarities with the human brain
- However, convolutional neural nets are actually one of the closest algorithms to which our algorithms compare to humans
- Essentially, the filters in the eye do an operation quite similar to the convolution operation.



## Further Reading

- Alexnet: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- VGG: <https://arxiv.org/pdf/1409.1556.pdf>
- Resnet: <https://arxiv.org/pdf/1512.03385.pdf>
- Inception-Resnet v4: <https://arxiv.org/pdf/1602.07261.pdf>
- MobileNet: <https://arxiv.org/pdf/1704.04861.pdf>
- YOLOv3: <https://pjreddie.com/media/files/papers/YOLOv3.pdf>
- Original Style Transfer: <https://arxiv.org/pdf/1508.06576.pdf>