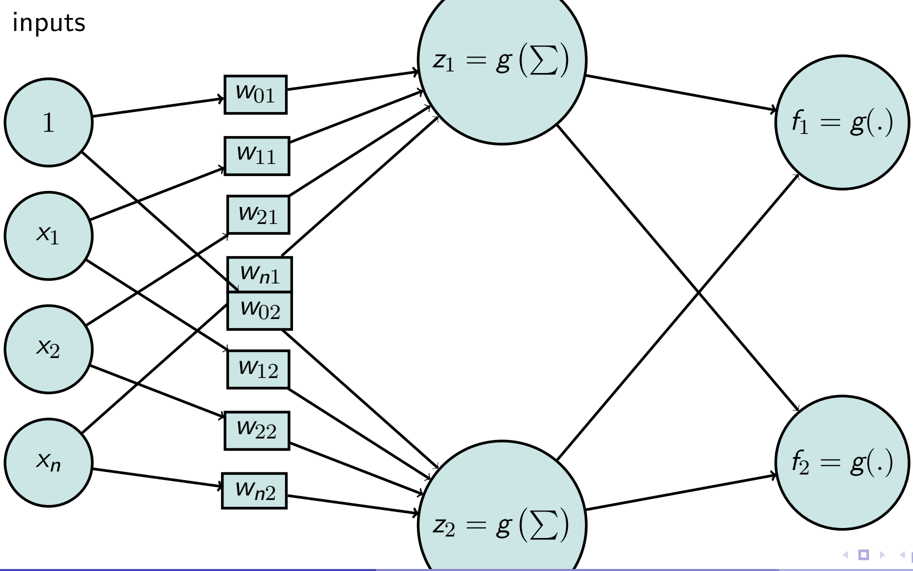
Lecture 21: Neural Network Training using Backpropagation, Convolutional Networks

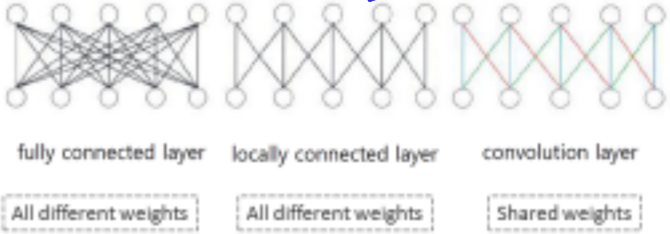
Instructor: Prof. Ganesh Ramakrishnan

Feed-forward Neural Nets



Convolutional Neural Network

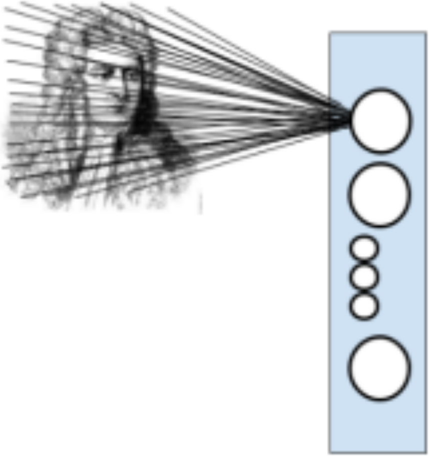
- Variation of multi layer feedforward neural network designed to use minimal preprocessing with wide application in image recognition and natural language processing
- Traditional multilayer perceptron(MLP) models do not take into account spatial structure of data and suffer from curse of dimensionality
- Convolution Neural network has smaller number of parameters due to **local connections** and **weight sharing**



MLP Issue: Parameter Explosion

200 X 200 image, 40k hidden units

around 2B parameters!



MLP Issue: Curse of Dimensionality

If dimension is large, number of samples may be too small for accurate parameter estimation.
Otherwise, we may end up in using a too complicated model for the data, *i.e.*, over-fitting,

The Lego Blocks in Modern Deep Learning

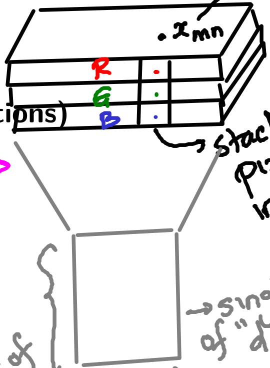
At input layer level: RGB

image coordinate

1. **Depth/Feature Map**
2. **Patches/Kernels (provide for spatial interpolations)**
3. **Strides (enable downsampling)**
4. **Padding (shrinking across layers)**
5. Pooling
6. Inception
7. Embeddings → part of unsupervised learning
8. Memory cell and Backpropagation through time

funneling

Recurrent Neural N/ws



stack for image pixel x_{mn} in color coordinate system of depth = 3

depth d = # of interim o/ps corresp to each pixel or set of pixels

→ single hidden layer of "d" feature maps

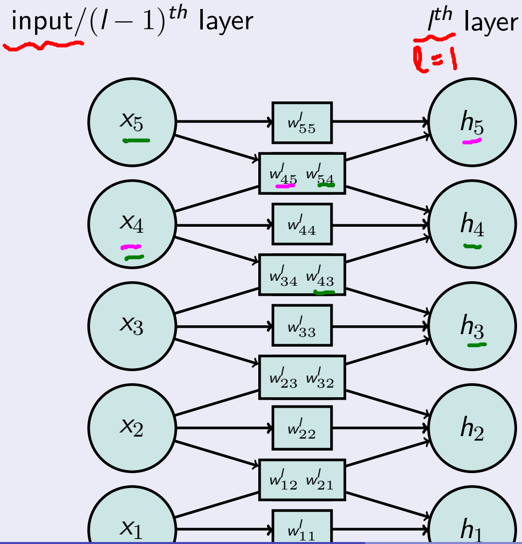
Eg: Whether this pixel is part of cat or dog object

Image Recognition: MLP Vs Blocks from the Lego

Fully connected

Input Image Size: 200 X 200 X 3 (RGB)
MLP: Hidden Layer with 40k neurons results in 4.8 billion parameters. $12\times10^4 \times 4\times10^4$
1 neuron corresponding to each pixel (o/p depth=1)
CNN: ??
Question: How many parameters?
Answer:
Question: How many neurons (location specific)?
Answer:

Convolution: Sparse Interactions through Kernels (for Single Feature Map)



1) Each input perceptron is connected to a local nbrhood in next layer

2) $h_i = \left(\sum_{m \in I_i} x_m w_{mi}\right)$ x_{mn}

$h_{ij} = \sum_m x_m w_{mi} K(i-m)$

$h = (xw) \cdot (K)$

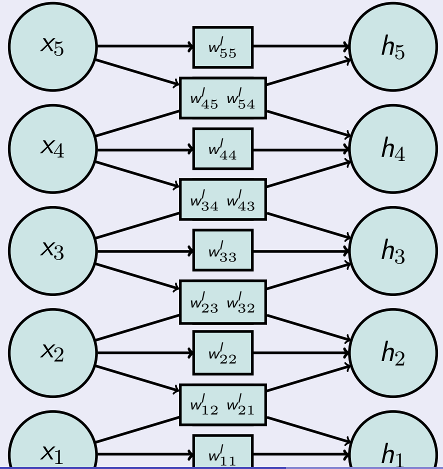
Convolution

Multiplication with toeplitz matrix

$K =$

Convolution: Sparse Interactions through Kernels (for Single Feature Map)

input/$(l-1)^{th}$ layer $\qquad$ l^{th} layer



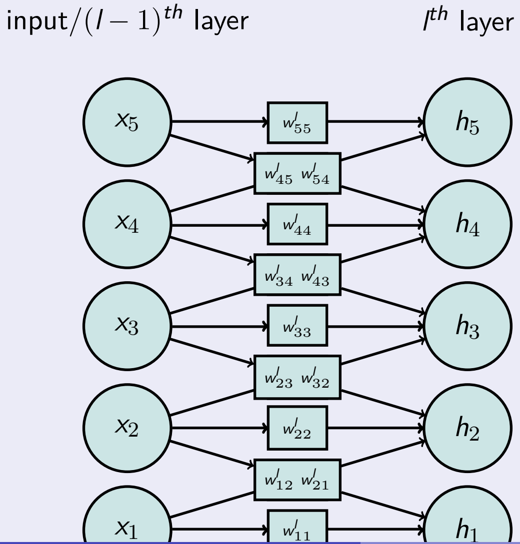
Recall nonparam kernels

- $h_i = \sum_m x_m w_{mi} K(i-m)$
- On LHS, $K(i-m) = 1$ *iff* $|m-i| \leq 1$
- For 2-D inputs (such as images):

$$h_{ij} = \sum_m \sum_n x_{mn} w_{mn,ij} * K(i-m, j-n)$$

Eg: h_{ij} could represent rate of change of intensity around (i,j) pixel .. Gives you gradient info

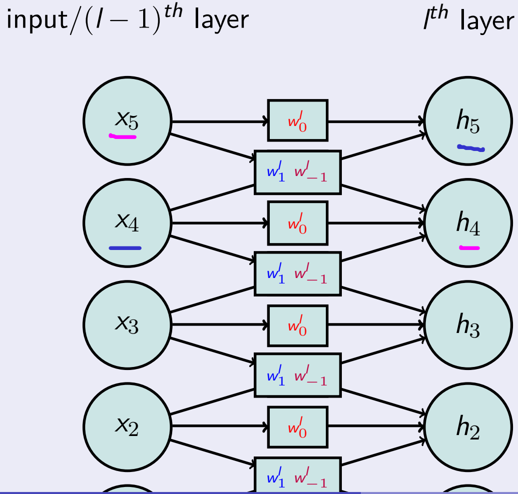
Convolution: Sparse Interactions through Kernels (for Single Feature Map)



- $h_i = \sum_m x_m w_{mi} K(i-m)$
- On LHS, $K(i-m) = 1$ *iff* $|m-i| \leq 1$
- For 2-D inputs (such as images):
  $h_{ij} = \sum_m \sum_n x_{mn} w_{ij,mn} K(i-m, j-n)$
- Intuition: Neighboring signals x_m (or pixels x_{mn}) more relevant than one's further away, reduces prediction time
- Can be viewed as multiplication with a Toeplitz[a] matrix K
- Further, K is often sparse (eg: $K(i-m) = 1$ *iff* $|m-i| \leq \theta$)

Here $\theta = 1$

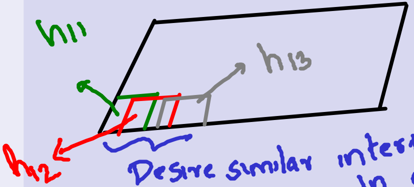
Convolution: Shared parameters and Patches (for Single Feature Map)



$x_{mn} \to \boxed{w_{i-m,j-n}} \to h_{ij}$

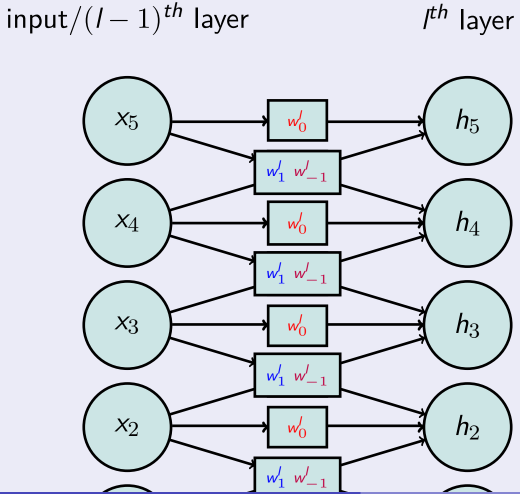
$x_m \to \boxed{w_{i-m}} \to h_i$

w_{i-m} provides a recurring neighbourhood template



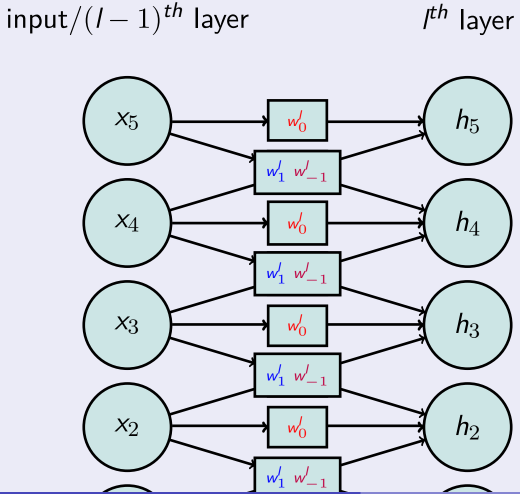
Desire similar interpolation in each patch

Convolution: Shared parameters and Patches (for Single Feature Map)



- $h_i = \sum_m x_m w_{i-m} K(i-m)$
- On LHS, $K(i-m) = 1$ *iff* $|m-i| \leq 1$
- For 2-D inputs (such as images):

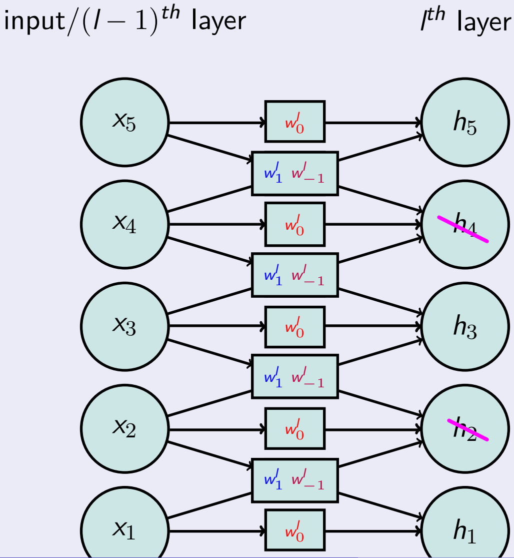
Convolution: Shared parameters and Patches (for Single Feature Map)



- $h_i = \sum_m x_m w_{i-m} K(i-m)$
- On LHS, $K(i-m) = 1$ *iff* $|m-i| \leq 1$
- For 2-D inputs (such as images):
  $h_{ij} = \sum_m \sum_n x_{mn} w_{i-m,j-n} K(i-m, j-n)$
- **Intuition:** Neighboring signals x_m (or pixels x_{mn}) affect in similar way irrespective of location (*i.e.*, value of m or n) ↳ invariance to location
- **More Intuition:** Corresponds to moving **patches around the image**
- Further reduces *storage* requirement;

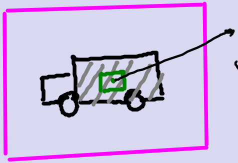
To detect the apple invariant to its location, we desire that the params are also similar

Convolution: Strides and Padding (for Single Feature Map)



Strides: Use only select h_s's & don't even compute the rest

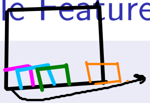
$\equiv$ Downsampling



Using kernels we already have representative features for each such green patch

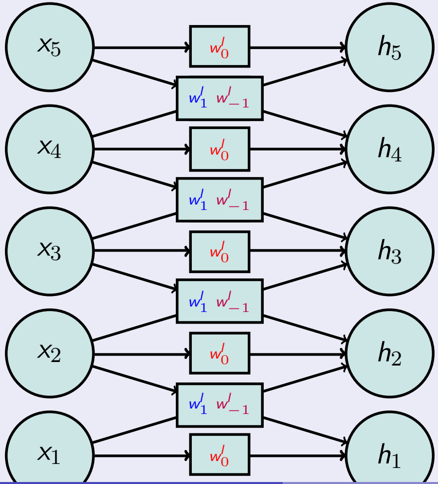
So why not use only some of them

Convolution: Strides and Padding (for Single Feature Map)



shifting by "s" strides at a time

input/$(l-1)^{th}$ layer | l^{th} layer



- Consider only h_i's where i is a multiple of s.
- **Intuition:** Stride of s corresponds to moving the patch by s steps at a time
- **More Intuition:** Stride of s corresponds to downsampling by s
- What to do at the ends/corners: Ans: **Pad** with either 0's (**same padding**) or let the next layer have fewer nodes (**valid padding**)
- Reduces *storage* requirement as well as prediction time

Homework: Image Example MLP Vs CNN

Input Image Size: $200 \times 200 \times 3$

MLP: Hidden Layer has 40k neurons, resulting in **4.8 billion** parameters.

CNN: Hidden layer has 20 feature-maps each of size 5 X 5 X 3 with stride = 1, i.e. maximum overlapping of convolution windows.

A feature map corresponds to one set of weights w^l_{ij}. M feature maps $\Rightarrow$ M times the number of weight parameters

Depth "d" of the hidden layer

Question: How many parameters?

Answer:

Question: How many neurons (location specific)?

Answer:

Each feature map can help you discover "aspects" of sections in image such as "contains cat", "is in shadow", "is moving",