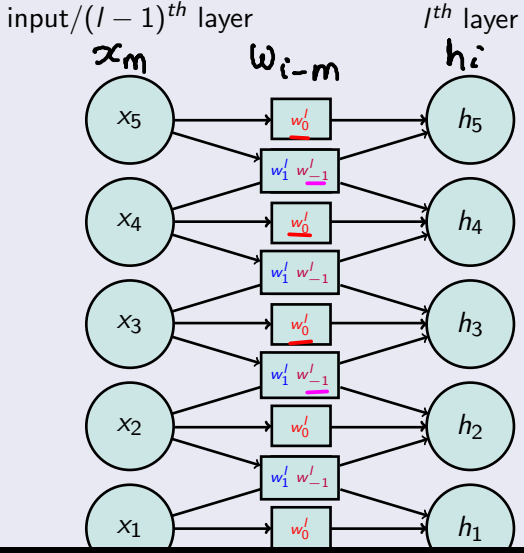
Lecture 24: Lego World of Deep Learning, Convolutional Neural Networks, etc

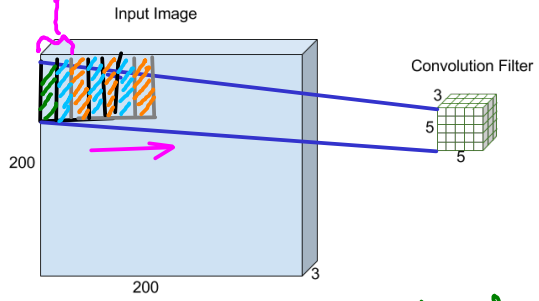
Instructor: Prof. Ganesh Ramakrishnan

Convolution: Strides and Padding (for Single Feature Map)



- 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 does **linear downsampling**
- 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

The Convolutional Filter



stride length 2 [move filter/kernel by 2 steps at a time]

} Similar analysis for vertical motion

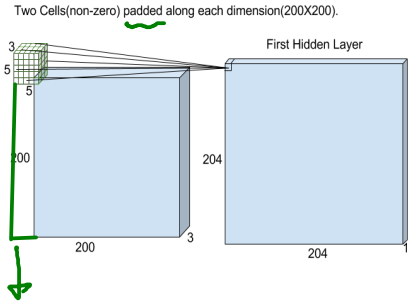
Down sampling

⇒ At edge considered once

⇒ Considered odd # of times (1, 3 ...)

⇒ Considered even # of times

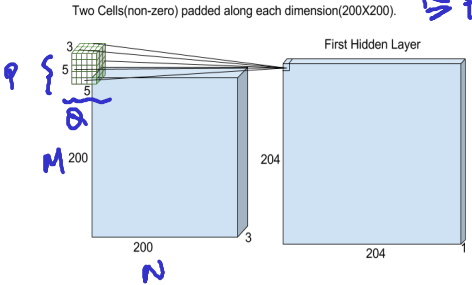
Padding lets you account for boundary/corner phenomena.

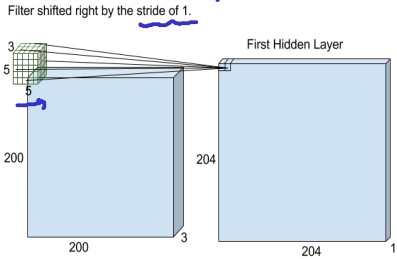


Corresponds to padding on left with 4 columns

The Convolutional Filter

Q: What will be size of output after applying the convolution filter? Is # of positions that filter can take?





Padding size = D

Image size = M × N.

Filter size = P × Q

Stride length = S

Ans: $\left(\frac{M+2D-P}{S}+1\right) \times \left(\frac{N+2D-Q}{S}+1\right)$ (+1 for start pt; +1 for starting pt)

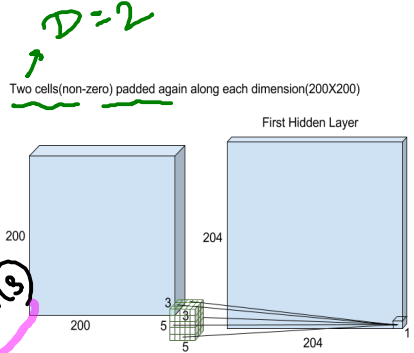
Hint: Try out

M = 200

D = 0

S = 2, P = 5

The Convolutional Filter



Question: MLP Vs CNN

Convolution leverages three important ideas that can help improve a machine learning system: (a) sparse interactions, (b) parameter sharing and (c) equivariant representations: $f(g(x)) = g(f(x))$ when f is convolution and g is shift function.

Let us see these in action:

Question: MLP Vs CNN

20 feature maps = 20 filters = 20 higher level image characteristics

Convolution leverages three important ideas that can help improve a machine learning system: (a) sparse interactions, (b) parameter sharing and (c) equivariant representations: $f(g(x)) = g(f(x))$ when f is convolution and g is shift function.

Let us see these in action:

Input Image Size: $200 \times 200 \times 3$ → $M=200, N=200$

MLP: Hidden Layer has 40k neurons, resulting in **4.8 billion** parameters. $P=5, Q=5$

CNN: Hidden layer has 20 feature-maps each of size 5 X 5 X 3 with stride $= 1$ and zero padding of 4 on each side, *i.e.*, maximum overlapping of convolution windows. $S=1$

Consider single hidden layer

A feature map corresponds to one set of weights w_{ij}^{l}. F feature maps $\Rightarrow$ F times the number of weight parameters

$D=4$

Question: How many parameters?

Answer: Wt w_{ij} is associated between j^{th} convolution o/p 4 i^{th} neuron of i/p layer

Question: How many neurons (location specific)?

Answer:

Answer: MLP Vs CNN

MLP: Hidden Layer has 40k neurons, so it has 4800000 parameters.
CNN: Hidden layer has 20 feature-maps each of size 5 X 5 X 3 with stride $= 1$, and zero padding of 4 on each side, *i.e.*, maximum overlapping of convolution windows.
Question: How many parameters?
Answer: Just 1500
Question: How many neurons (location specific)?

Let $M \times N \times 3$ be dimension of image and $P \times Q \times 3$ be dimension of patch for kernel convolution. Let D be number of zero paddings and s be stride length.
Answer:**Output size** $= \left(\frac{M+2D-P}{s}+1\right) \times \left(\frac{N+2D-Q}{s}+1\right)$

Answer: MLP Vs CNN

40000 < 832320

MLP: Hidden Layer has 40k neurons, so it has 4800000 parameters. >> 1500

CNN: Hidden layer has 20 feature-maps each of size 5 X 5 X 3 with stride = 1, and zero padding of 4 on each side, *i.e.*, maximum overlapping of convolution windows.

Question: How many parameters?

Answer: Just 1500

Question: How many neurons (location specific)?

} wts = 20 × 5 × 5 × 3

Let $M \times N \times 3$ be dimension of image and $P \times Q \times 3$ be dimension of patch for kernel convolution. Let D be number of zero paddings and s be stride length.

Answer:**Output size =** $\left(\frac{M-P+2D}{s}+1\right) \times \left(\frac{N-Q+2D}{s}+1\right)$.

In current case, $D = P - 1 \Rightarrow$ Output size $= \left(\frac{M+P}{s}-1\right) \times \left(\frac{N+Q}{s}-1\right)$.

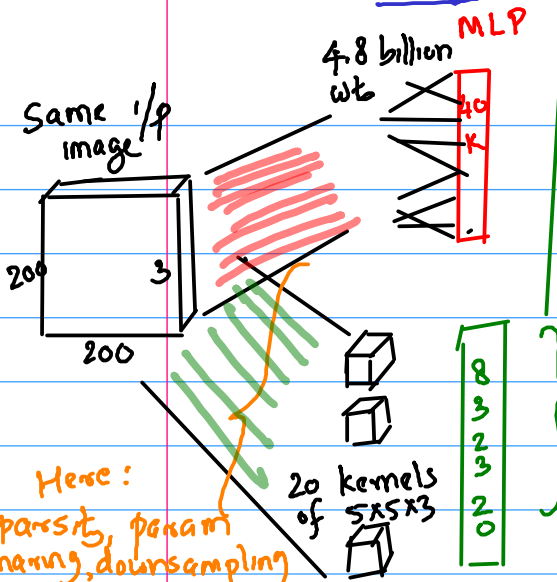
$20 \times ((200+5)/s) - 1) \times ((200+5)/s) - 1)$

$= 832320$ (around 830 thousand which can increase with max-pooling).

If $D = (P-1)/2$ **and** $S = 1$,

} In the video = # of neurons in hidden layer

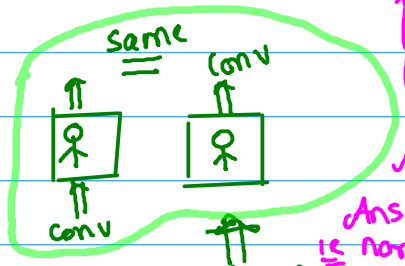
Tradeoff



Here: sparsity, param sharing, downsampling

Convolution = equivariant

$f(g(x)) = g(f(x))$



Invariance? Is shift or no shift, o/p is same...

Ans: (max) pooling is nonlinear sampling

Larger # hidden layer perceptrons, much smaller #wt. Q: Can large # of perceptrons be reduced?

Answer: MLP Vs CNN

MLP: Hidden Layer has 40k neurons, so it has 4800000 parameters.
CNN: Hidden layer has 20 feature-maps each of size 5 X 5 X 3 with stride $= 1$, and zero padding of 4 on each side, *i.e.*, maximum overlapping of convolution windows.
Question: How many parameters?
Answer: Just 1500
Question: How many neurons (location specific)?

Let $M \times N \times 3$ be dimension of image and $P \times Q \times 3$ be dimension of patch for kernel convolution. Let D be number of zero paddings and s be stride length.
Answer:**Output size** $= \left(\frac{M-P+2D}{s} + 1\right) \times \left(\frac{N-Q+2D}{s} + 1\right)$.
In current case, $D = P - 1 \Rightarrow$ Output size $= \left(\frac{M+P}{s} - 1\right) \times \left(\frac{N+Q}{s} - 1\right)$.
$20 \times ((200 + 5)/s) - 1) \times ((200 + 5)/s) - 1)$
$= 832320$ (around 830 thousand which can increase with max-pooling).
If $D = (P - 1)/2$ and $S = 1$, output will be of same size as input!

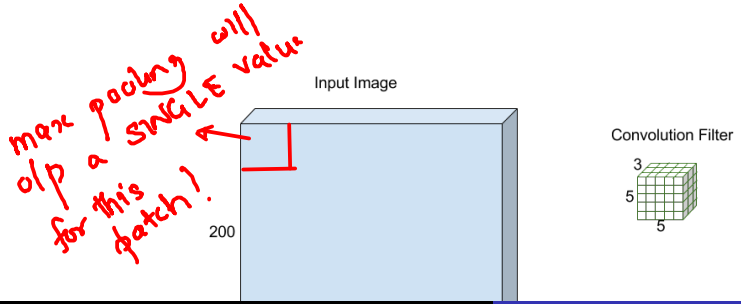
The Lego Blocks in Modern Deep Learning

1. Depth/Feature Map
2. Patches/Kernels (provide for spatial interpolations) - **Filter**
3. Strides (Linear downsampling)
4. Padding (shrinking across layers)
5. **Pooling** (Non-linear downsampling) - **Filter**
6. **RNN and LSTM** (Backpropagation through time and Memory cell) (??)
7. Embeddings (After discussing unsupervised learning)

The Max Pooling Filter

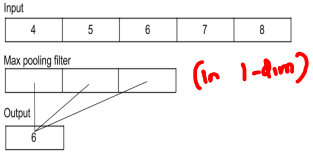
A non-linear downsampling filter/kernel that selects the maximum value from its patch.

- It is a sample-based discretization process.
- Objective is dimensionality reduction through down-sampling of input representation (eg: image),
- Allows for translation invariance to the internal representation.
- Helps avoid overfitting and reduces the number of parameters to learn.



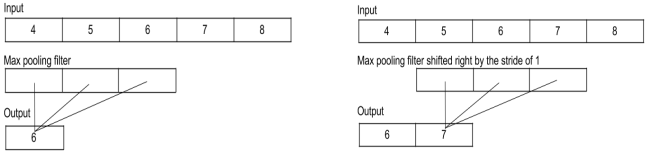
Max pooling (with downsampling) for a Single Feature Map

1-d example



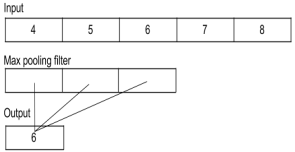
Max pooling (with downsampling) for a Single Feature Map

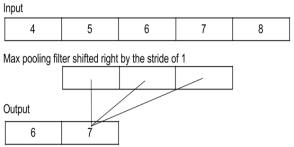
1-d example

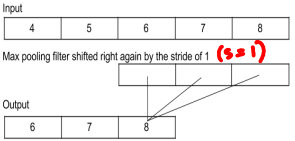


Max pooling (with downsampling) for a Single Feature Map

1-d example







M=5

(s=1)

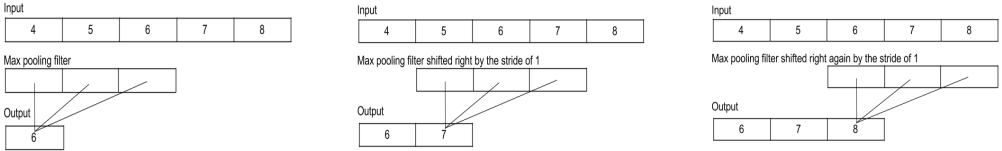
P=3

$\frac{M-P}{S}+1 = 2+1 = 3$

D=0 always (padding disallowed)

Max pooling (with downsampling) for a Single Feature Map

1-d example

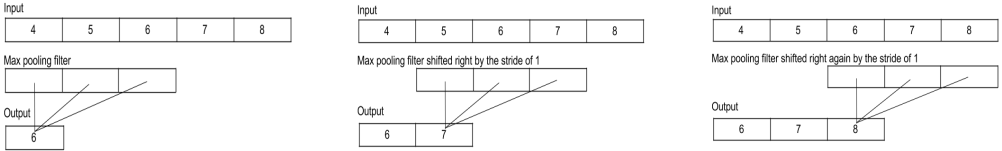


What will be the output if input and max pooling filter remains same but stride changes to 2?

$$\frac{M-P}{S}+1 = \frac{2}{2}+1 = 2$$

Max pooling (with downsampling) for a Single Feature Map

1-d example



What will be the output if input and max pooling filter remains same but stride changes to 2?

[6,8]

Max pooling in 2-D for a Single Feature Map

- Let $M \times N \times 3$ be dimension of image and $P \times Q \times 3$ be dimension of patch for kernel convolution. Let s be stride length
- Max pooling takes every $M \times N \times 3$ patch from the input and set the output to the maximum value in that patch
- **Output size =** $\left(\frac{M-P}{s}+1\right) \times \left(\frac{N-Q}{s}+1\right)$. For Eg:
  - Input: A 3D image of size with $M = N = 5$, $P = Q = 3$ and with (default) stride of 1.
  - Output size will be $3 \times 3 \times 1$

(You generally apply (max) pooling after convolution)

Tutorial 8, Problem 5

ConvNetJS (`http://cs.stanford.edu/people/karpathy/convnetjs/`) is a Javascript library for training Deep Learning models (Neural Networks) entirely in your browser. Try different choices of network configurations which include the choice of the stack of convolution, pooling, activation units, number of parallel networks, position of fully connected layers and so on. You can also save some network snapshots as JSON objects. What does the network visualization of the different layers reveal?
Also try out the demo at `http://places.csail.mit.edu/demo.html` to understand the heat maps and their correlations with the structure of the neural network.

Tutorial 8, Problem 6

Discuss the advantages and disadvantages of different activation functions: tanh, sigmoid, ReLU, softmax. Explain and illustrate when you would choose one activation function in lieu of another in a Neural Network. You can also include any experiences from Problem 5 in your answer.

(Try out these choices in convjs)

Alex-net [NIPS 2012]

Stack of two types of **parallel** networks

- First 5 convolution layers
  - First convolution layer takes input of size $224 \times 224 \times 3$, 48 ($\times 2$) features each with filter/kernel of size $11 \times 11 \times 3$ with stride of 4
    - Thus, $((224 + 11)/4 - 1) \times ((224 + 11)/4 - 1) = 57 \times 57$.
  - Max-pooling ($3 \times 3 \times 1$ with stride of 1) in the end reduces size to 55×55 for each filter
- Fully connected last 3 layers

Created two parallel pipelines for redundancy for same task

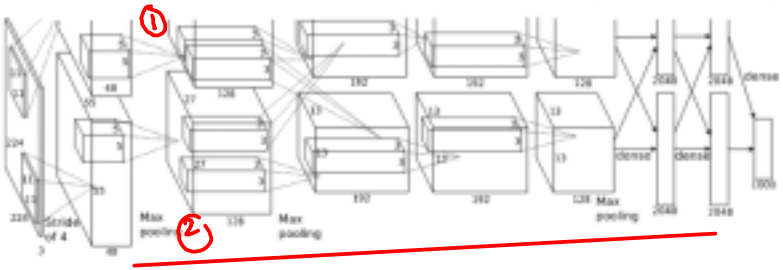
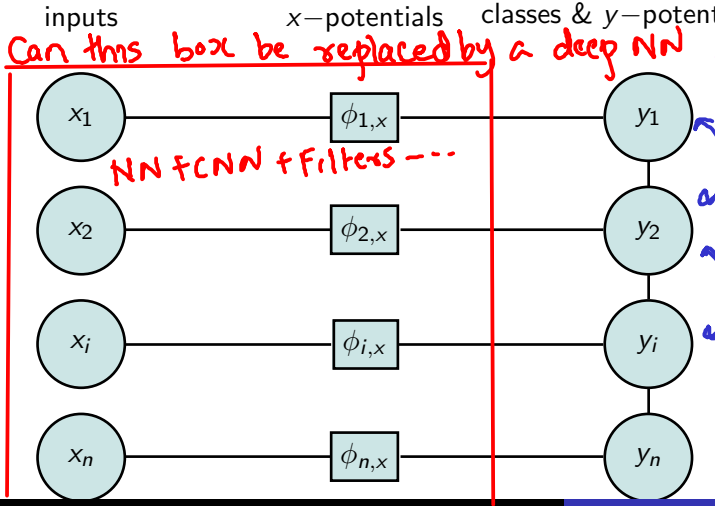


Image reference: "Imagenet Classification with Deep Convolution Neural Networks", NIPS 2012.

Recap: Linear Conditional Random Fields (CRF)

Just as CRF was an extension of Logistic Regression (LR) can Neural Networks (cascade of LRs) be extended to sequential output?



Tracking object across image frames in a video y = rat

$w^T \phi(x, y_1, y_2)$ } RHS

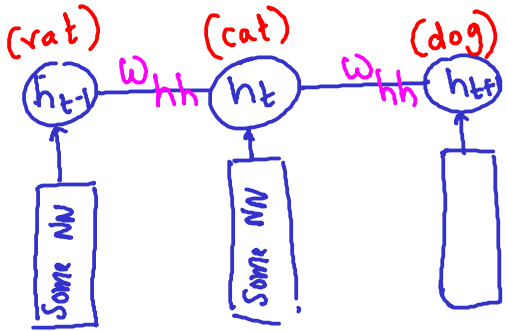
$w^T \phi(x, y_2, y_3)$

w was shared across dependencies between o/ps

Recurrent Neural Network (RNN) Intuition

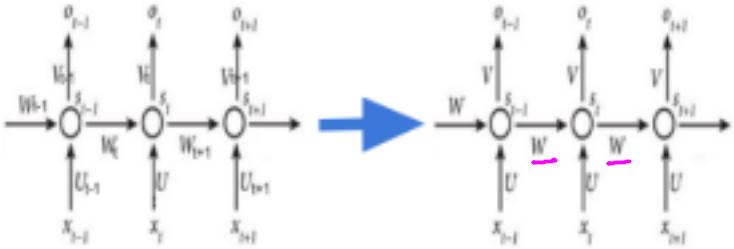
Captures RHS in conjunction with existing deep NN

- Recall: In CNN we used the trick of common parameters for many neurons
- RNN intuition 1: We want a neuron's output at time t to depend on its state s at time $t-1$
- RNN intuition 2: Share parameters across time steps
- Recurrent $\Rightarrow$ Performing the same task for every element of sequence.



Recurrent Neural Network (RNN) Intuition

- Recall: In CNN we used the trick of common parameters for many neurons
- RNN intuition 1: We want a neuron's output at time t to depend on its state s at time $t-1$
- RNN intuition 2: Share parameters across time steps
- Recurrent $\Rightarrow$ Performing the same task for every element of sequence.

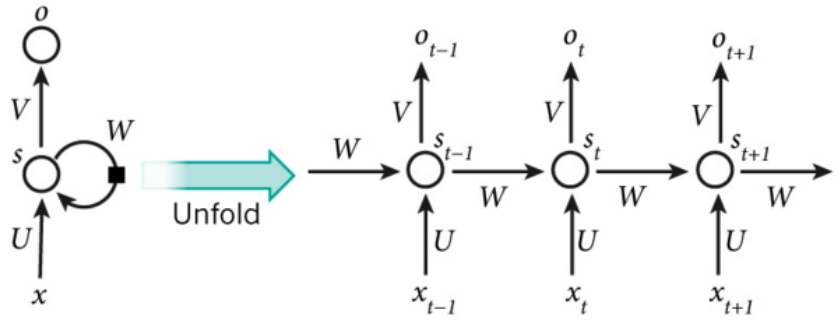


Tutorial 8: Problem 7

Try the text generation RNN (Recurrent Neural Network) demo at `http://www.cs.toronto.edu/~ilya/rnn.html`. State any interesting observations. How would you improve the performance of the RNN?

RNN: Compact representation

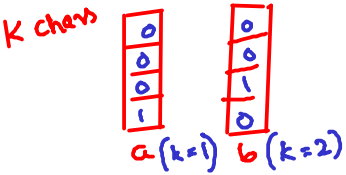
- Generalization of Neural networks to Sequential tasks such as *language modeling*, *word prediction*, *etc.*.
- Perform the same *task* for every element of the sequence, with the output being dependent on the previous computation



A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

RNN: One Hot Encoding for Language Model

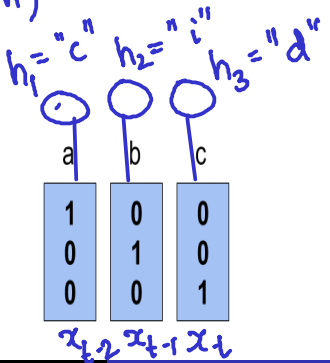
- With 3 characters in vocabulary, a,b and c, what would be the best encoding to inform each character occurrence to the network?
- One Hot Encoding: Give a unique key k to each character in alpha-numeric order, and encode each character with a vector of vocabulary size, with a 1 for the k^{th} element, and 0 for all other elements.



RNN: One Hot Encoding for Language Model

- With 3 characters in vocabulary, a,b and c, what would be the best encoding to inform each character occurrence to the network?
- One Hot Encoding: Give a unique key k to each character in alpha-numeric order, and encode each character with a vector of vocabulary size, with a 1 for the k^{th} element, and 0 for all other elements.

$h_t = g(h_{t-1} * w_{hh} + x_t * w_{xh})$



RNN: Language Model Example with one hidden layer of 3 neurons