CS 412

APRIL 28TH – DEEP LEARNING

Convolutional NN

Recurrent NN

Course Grading

Avergr for Midtrm 85

Regrade requests

Blackboard

- "current percentage"
- Likely to be a very small curve
- At least 90,80,70

Promise

A- high 80s

B- high 70s

C- mid/high 60s

— just an estimate

Administrivia

HW5 Out

- Due Thursday – last chance to use late days

Midterm regrades

- If you get them in today, I'll finish them today

Final Exam

- Current plan per the general final schedule: 24 hour take-home exam on Wednesday May 6th
- Midnight-to-midnight CDT
- If this doesn't work scheduling-wise, let me know ASAP

Course evaluations

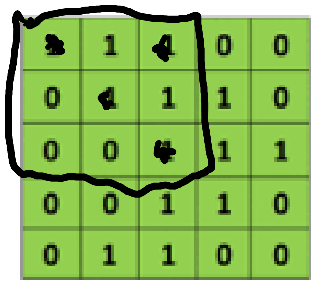
Convolutional Neural Networks (CNNs)

Deep learning
- automatic feature extraction

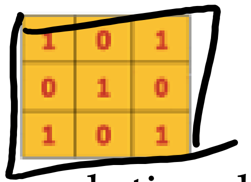
Main CNN idea for text:
Compute vectors for n-grams and group them afterwards

Example: “this takes too long” compute vectors for:
This takes, takes too, too long, this takes too, takes too long, this takes too long

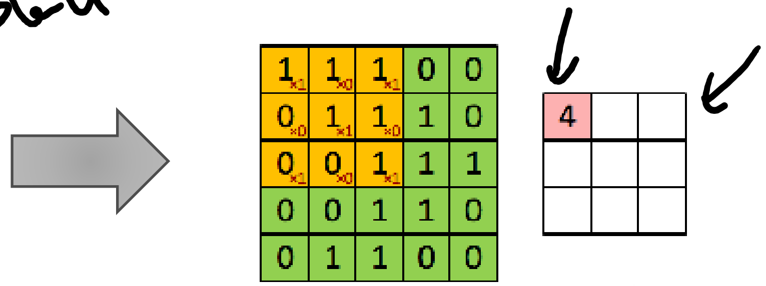
template



Input matrix

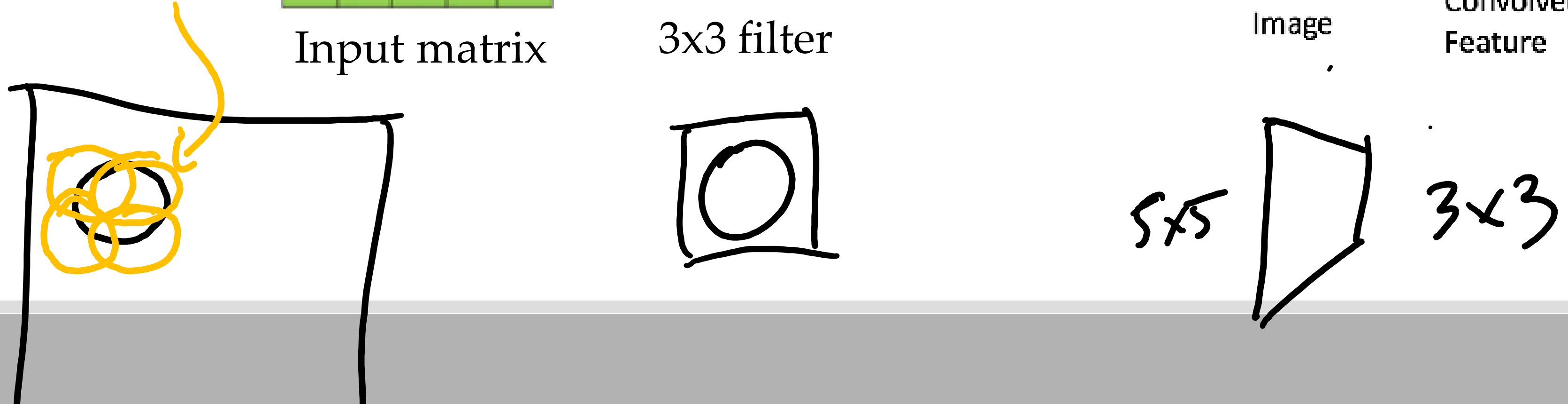


Convolutional 3x3 filter



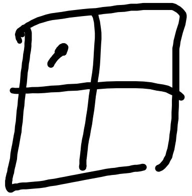
Image

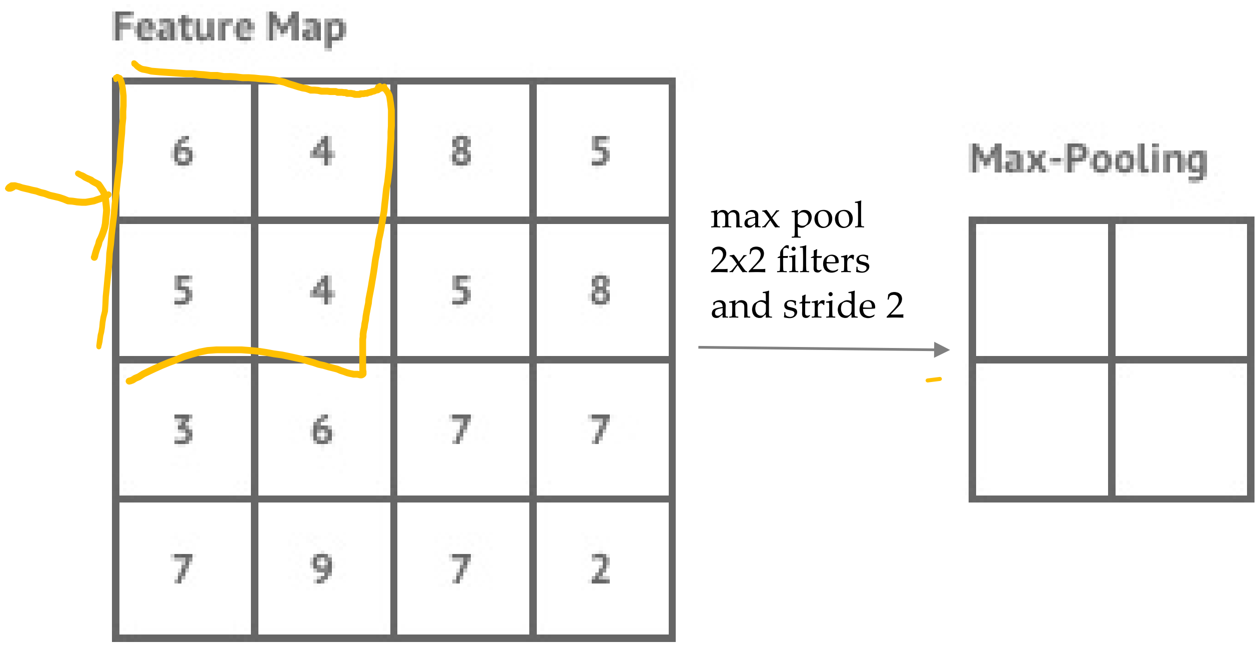
Convolved Feature



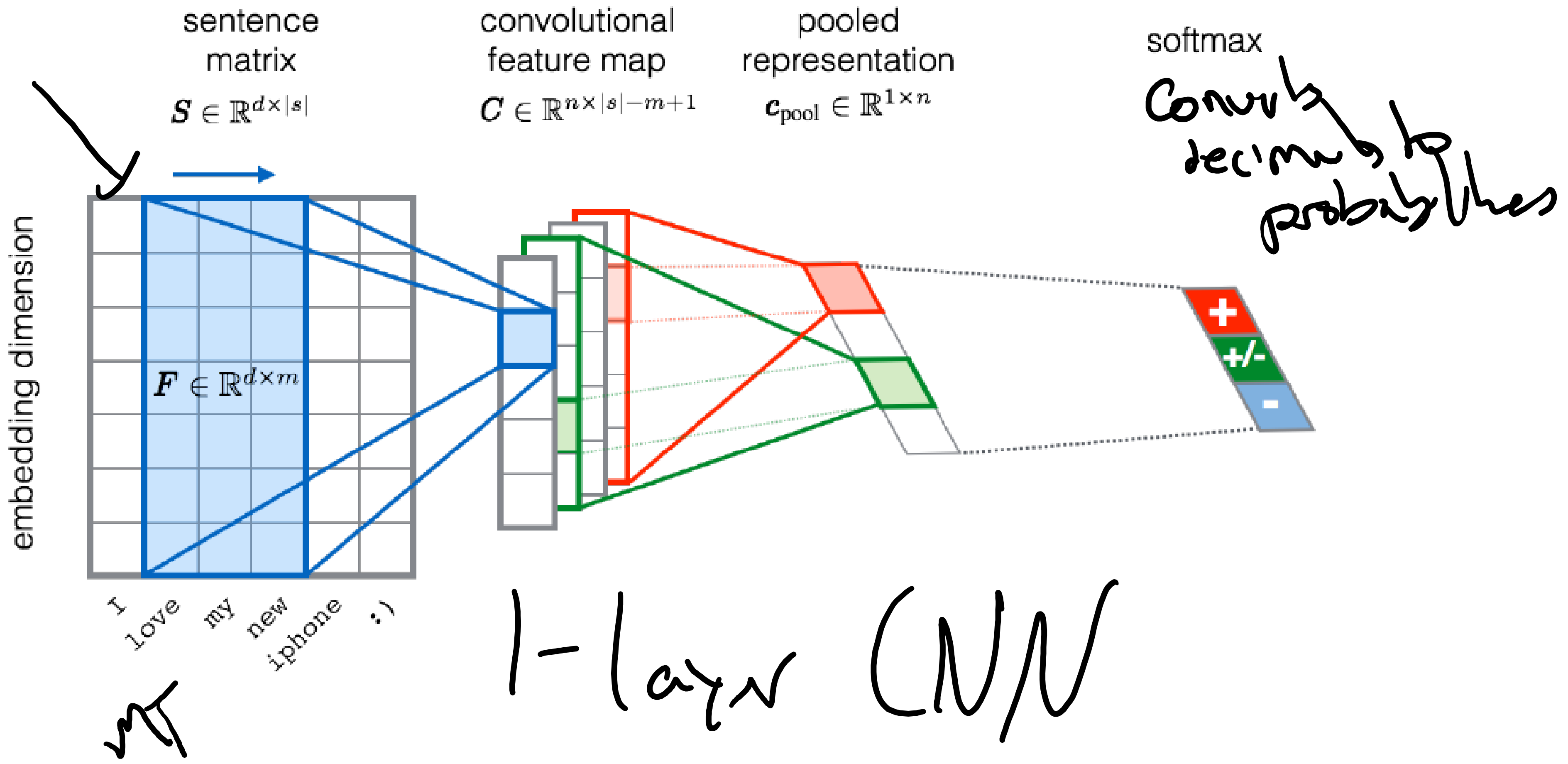
Convolutional Neural Networks (CNNs)

Main CNN idea for text:
Compute vectors for n-grams and group them afterwards



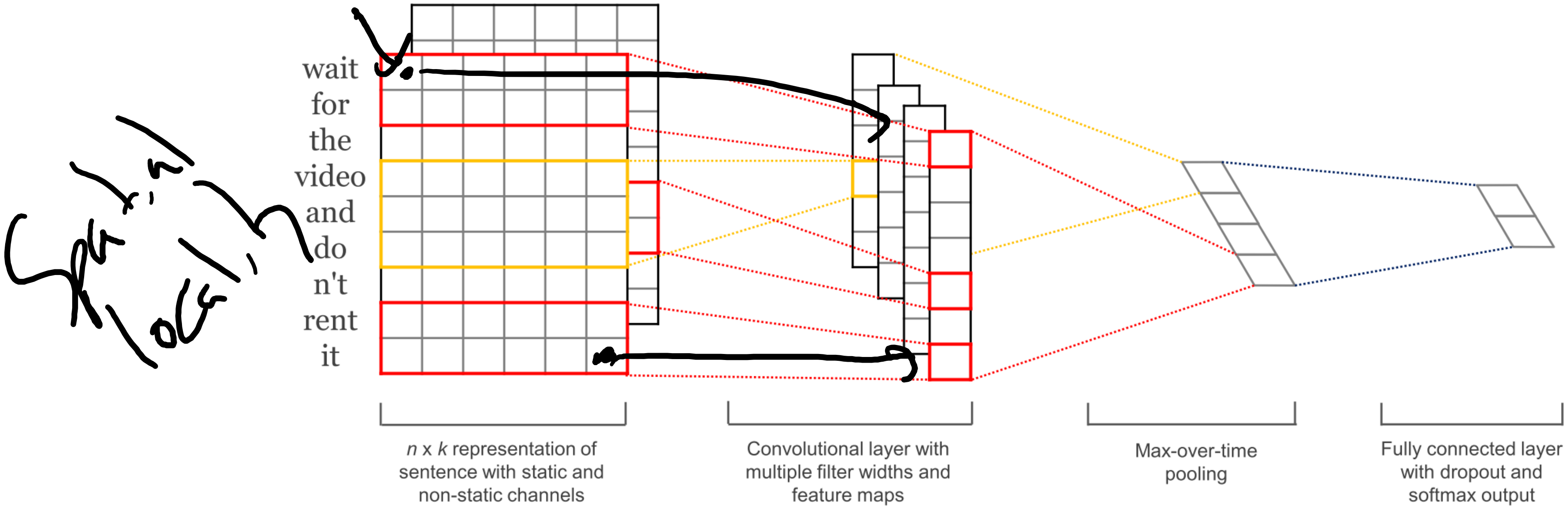


Convolutional Neural Networks (CNNs)



1-layer CNN

Convolutional Neural Networks (CNNs)



CNN Architecture

Separate from fully connected feed-forward

Intuition: Neural network with specialized connectivity structure,

- Stacking multiple layers of feature extractors
- Low-level layers extract local features.
- High-level layers extract learn global patterns.

A CNN is a list of layers that transform the input data into an output class/prediction.

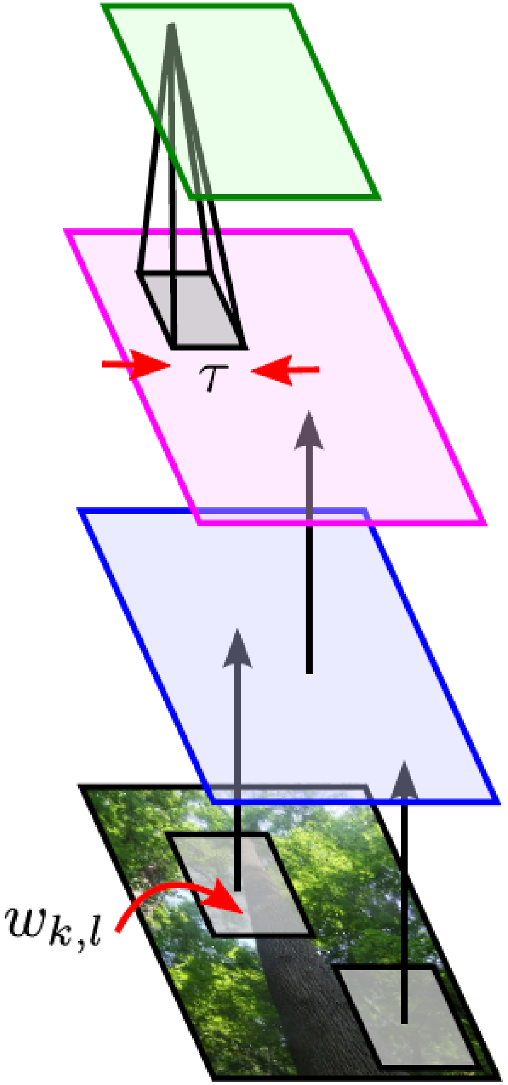
There are a few distinct types of layers:

- Convolutional layer
- Non-linear layer
- Pooling layer

training

Deep learning

layers



$$x_{i,j} = \max_{|k|<\tau, |l|<\tau} y_{i-k,j-l}$$

mean or subsample also used

pooling stage

Feature maps of a larger region are combined.

$$y_{i,j} = f(a_{i,j})$$

e.g. $f(a) = [a]_+$

$f(a) = \text{sigmoid}(a)$

non-linear stage

must go between

Feature maps are trained with neurons.

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

only parameters

convolutional stage

Each sub-region yields a feature map, representing its feature.

Shared weights

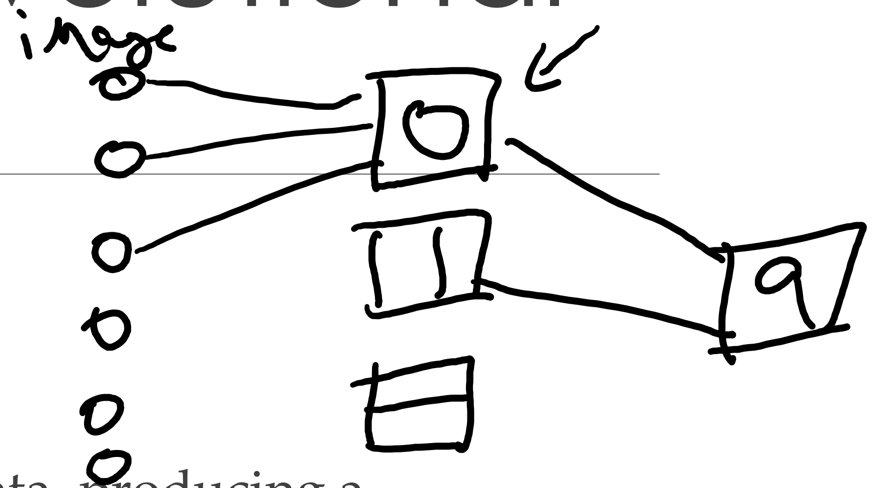
$z_{i,j}$

each CNN "level" is the same weights

input image

Images are segmented into sub-regions.

CNN Architecture: Convolutional Layer



The core layer of CNNs

The convolutional layer consists of a set of filters.
- Each filter covers a spatially small portion of the input data.

Each filter is convolved across the dimensions of the input data, producing a multidimensional feature map.
- As we convolve the filter, we are computing the dot product between the parameters of the filter and the input.

Intuition: the network will learn filters that activate when they see some specific type of feature at some spatial position in the input.

The key architectural characteristics of the convolutional layer is local connectivity and shared weights.

2D

CNN Convolutional Layer: Local Connectivity

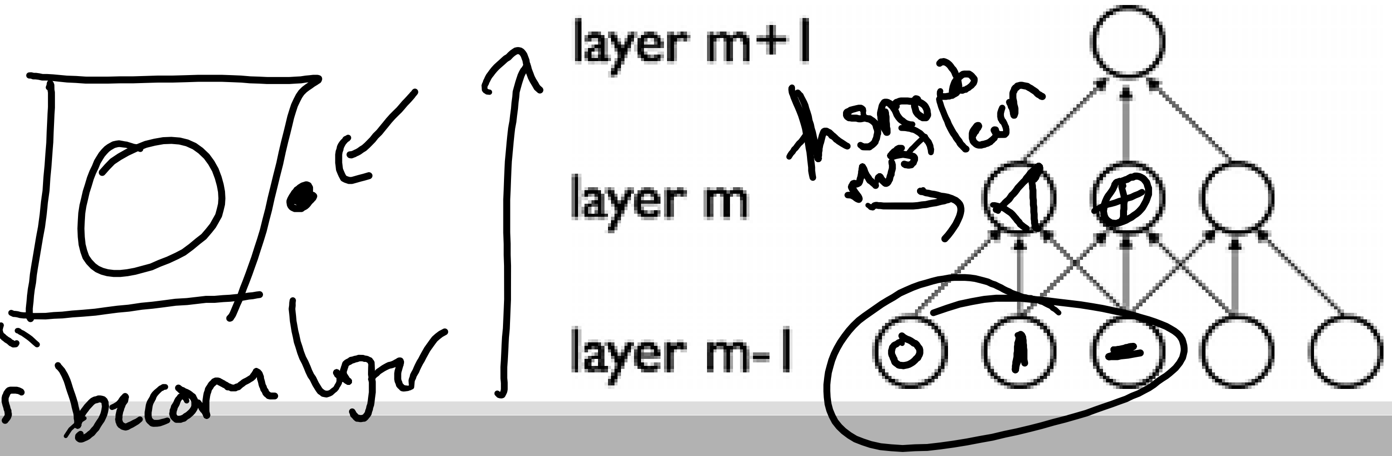
Neurons in layer m are only connected to 3 adjacent neurons in the m-1 layer.

Neurons in layer m+1 have a similar connectivity with the layer below.

Each neuron is unresponsive to variations outside of its receptive field with respect to the input.

- Receptive field: small neuron collections which process portions of the input data

The architecture thus ensures that the learnt feature extractors produce the strongest response to a spatially local input pattern.



our "fields" becom lgr

CNN Convolutional Layer: Shared Weights

We show 3 hidden neurons belonging to the same feature map (the layer right above the input layer).

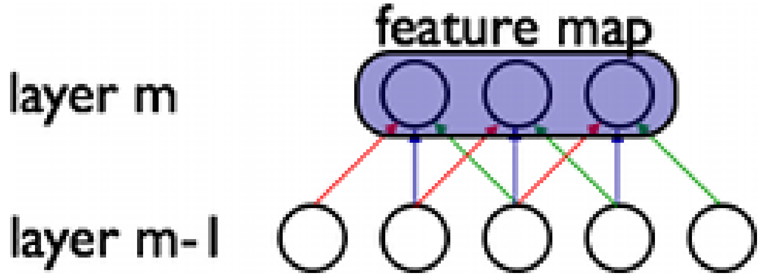
Weights of the same color are shared—constrained to be identical.

Gradient descent can still be used to learn such shared parameters, with only a small change to the original algorithm.

The gradient of a shared weight is simply the sum of the gradients of the parameters being shared.

Replicating neurons in this way allows for features to be detected regardless of their position in the input.

Additionally, weight sharing increases learning efficiency by greatly reducing the number of free parameters being learnt.



weight
re) input
same. shifty
input

CNN Architecture: Non-linear Layer

Intuition: Increase the nonlinearity of the entire architecture without affecting the receptive fields of the convolution layer

A layer of neurons that applies the non-linear activation function, such as,

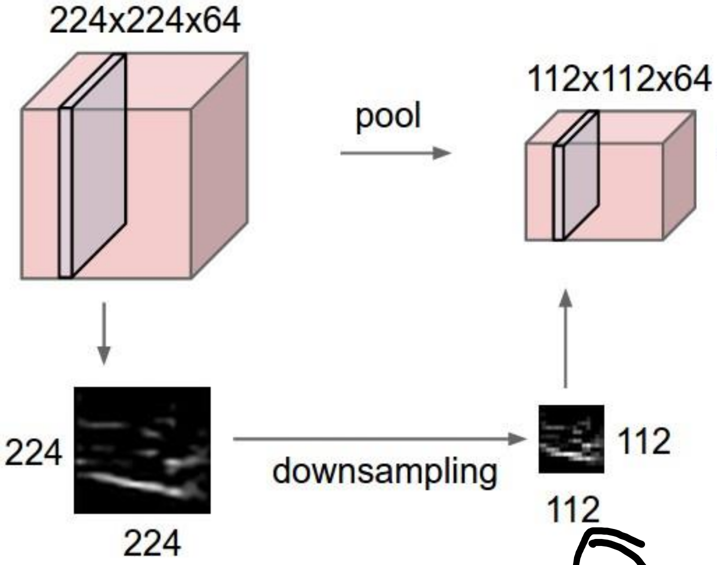
- $f(x) = \max(0, x)$
- $f(x) = \tanh x$
- $f(x) = |\tanh x|$
- $f(x) = (1 + e^{-x})^{-1}$

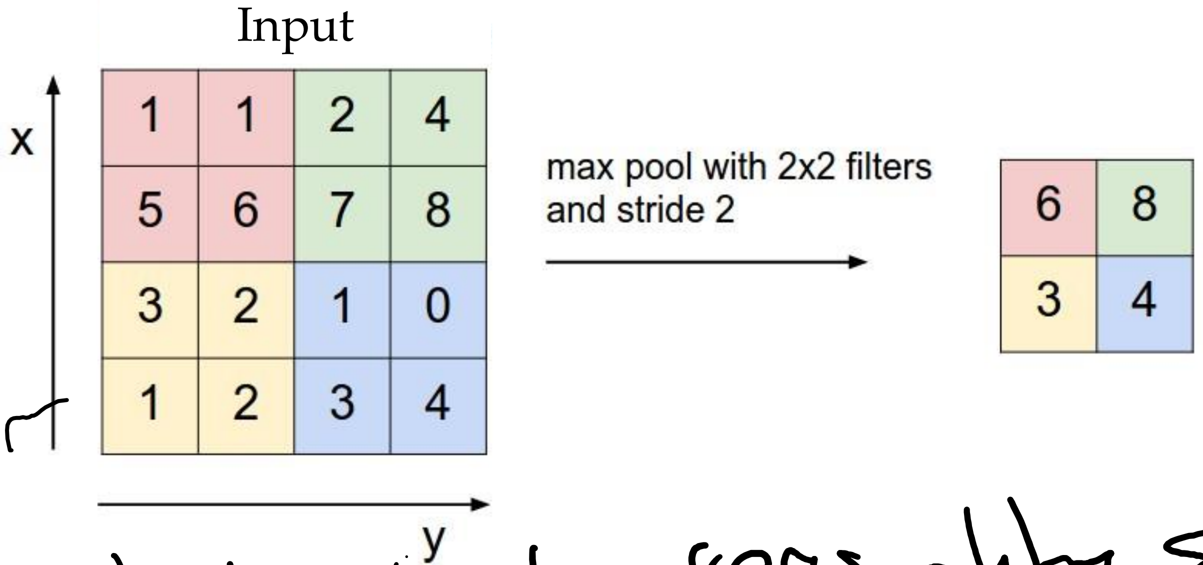
ReLU is popular

CNN Architecture: Pooling Layer

Intuition: to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting

Pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value of the features in that region.



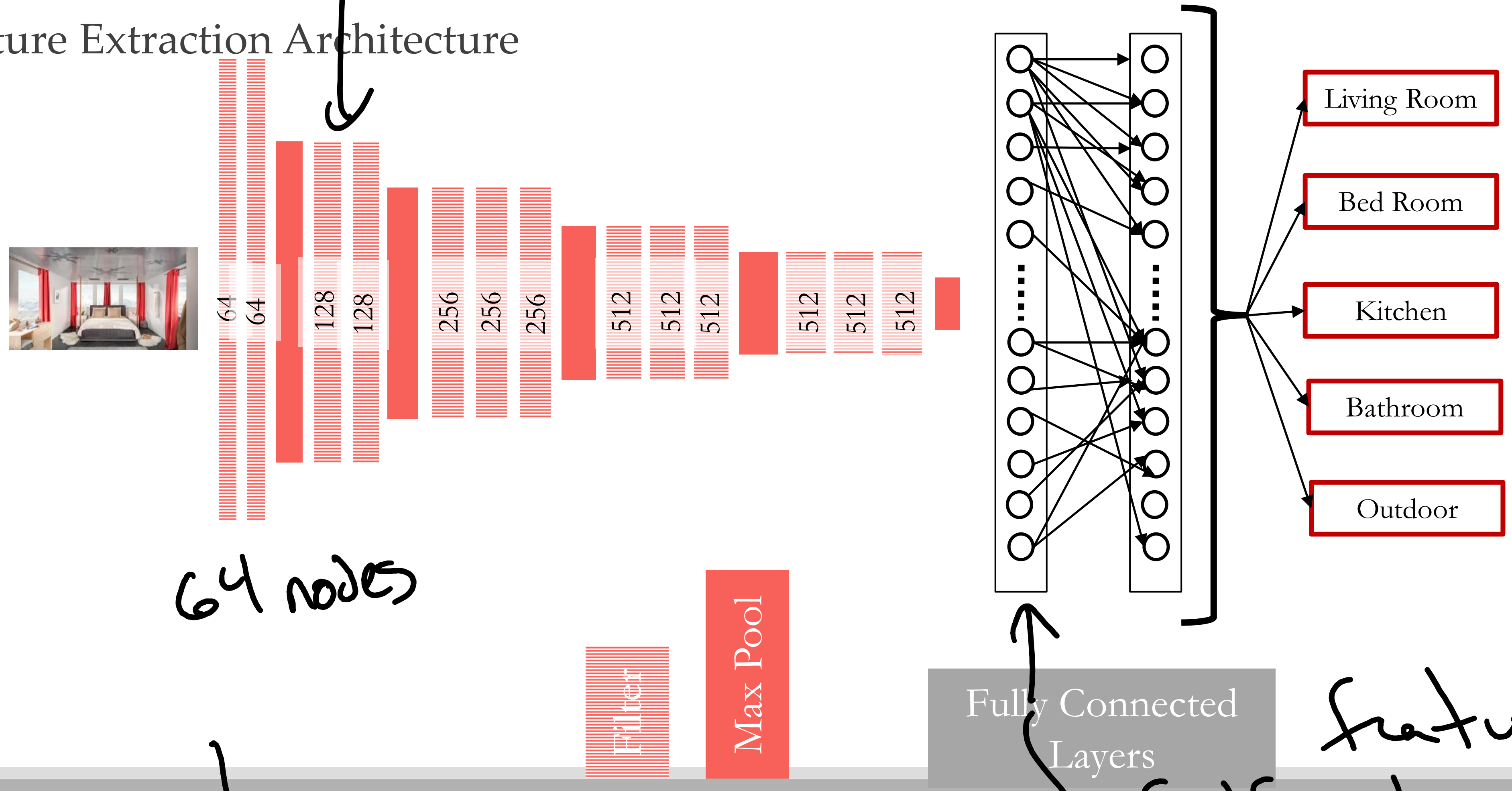


want this to be a representative sample

Notice:
more nodes. Why? More complex features to learn

Convolutional Neural Network

Feature Extraction Architecture



64 nodes

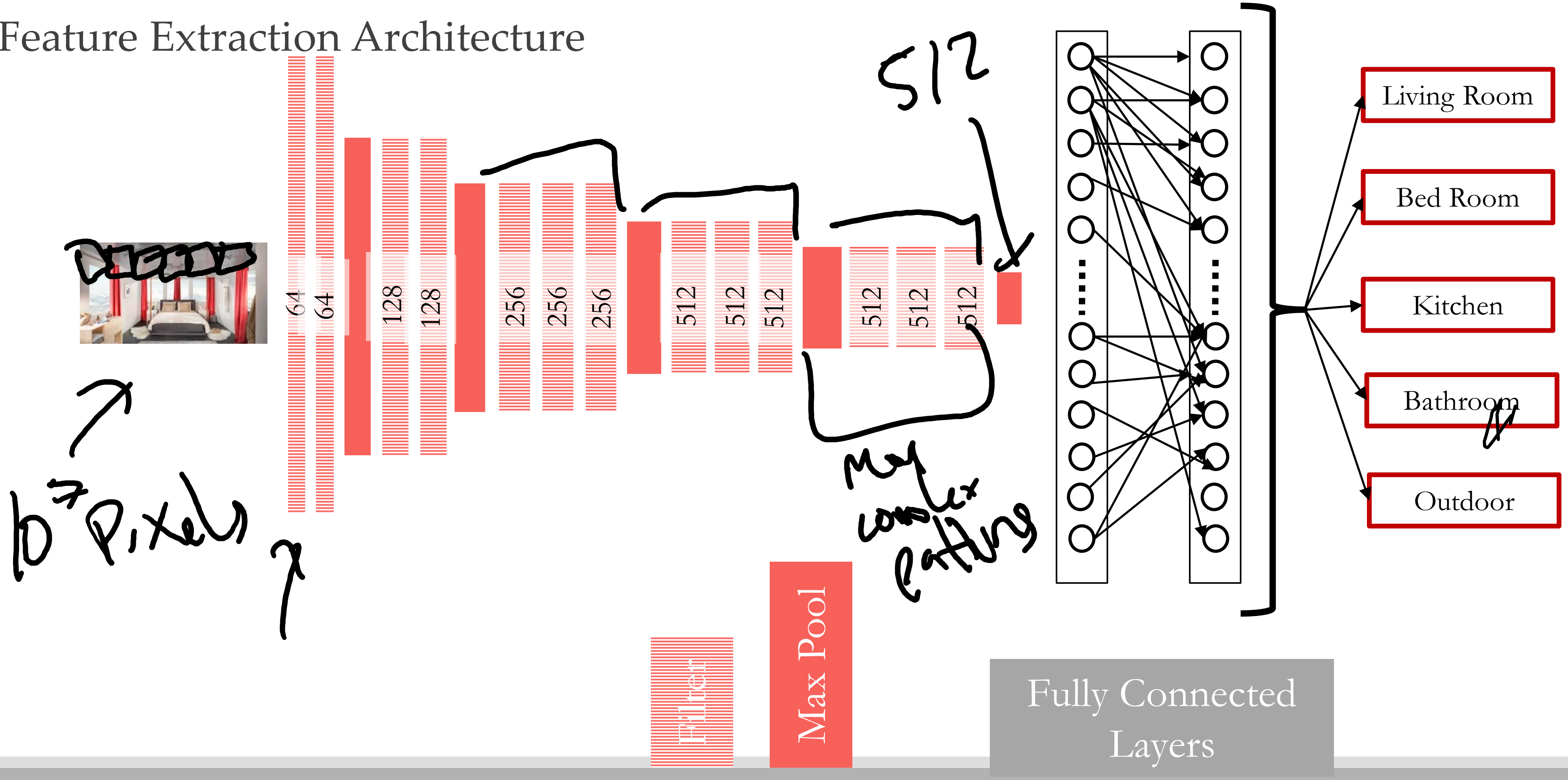
features

feedforward

Or

Convolutional Neural Network

Feature Extraction Architecture



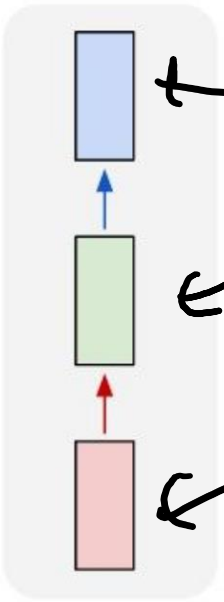
Recurrent Neural Networks

Convolutional neural networks
- Preserve spatial locality
- Good for images + speech

Recurrent neural networks
- Preserve temporal locality ← tm series
- Good for sequence to sequence translation

Recurrent Neural Networks

one to one

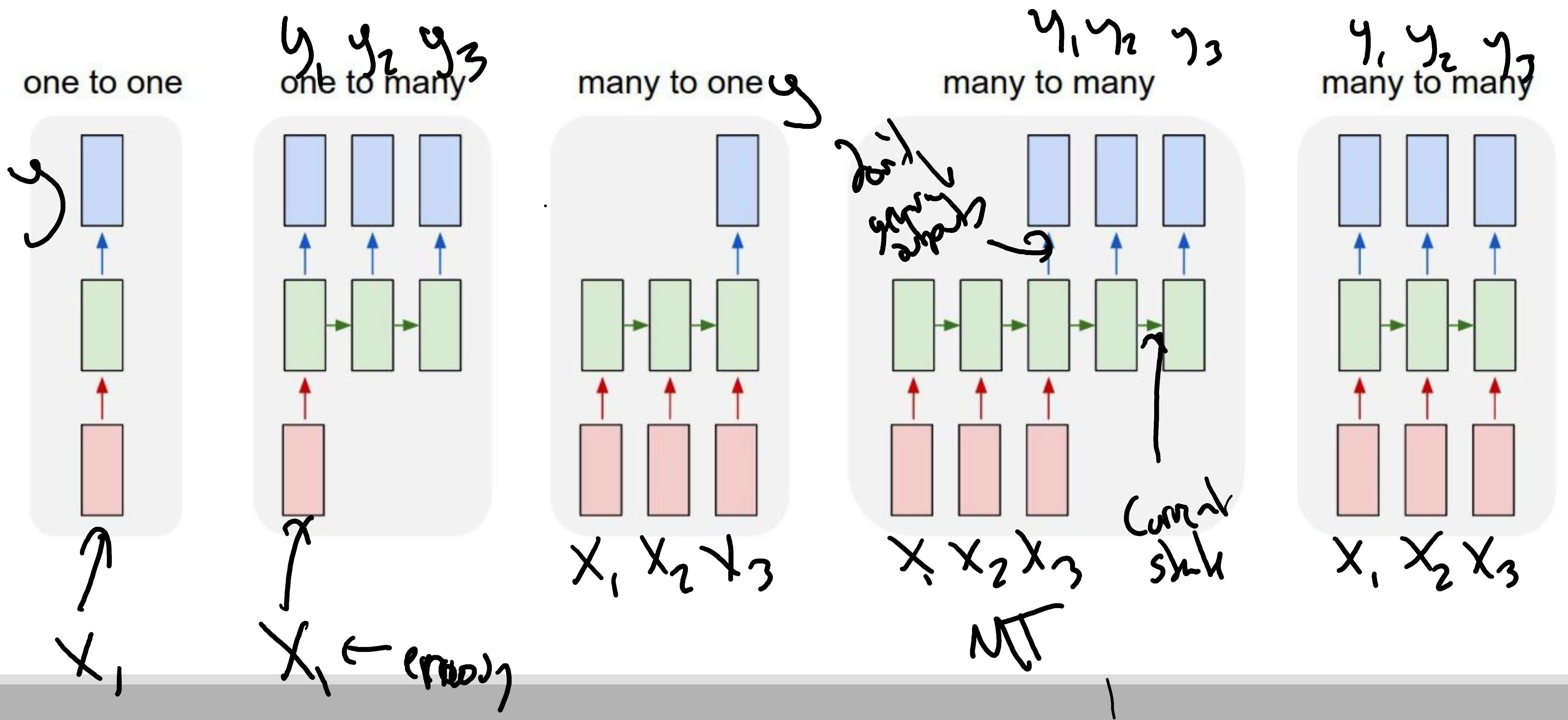


one output

feedforward NN

one input

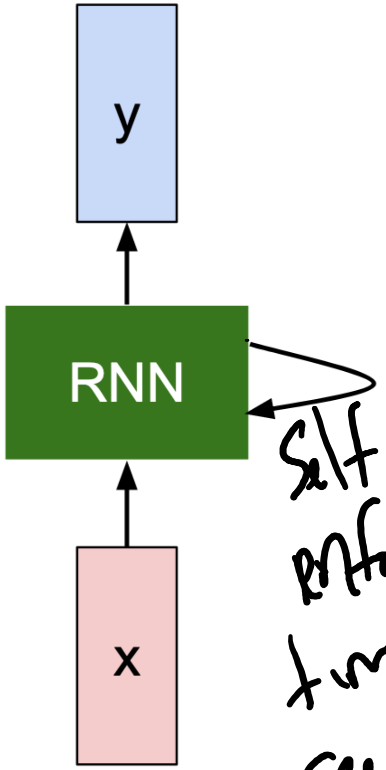
Recurrent Neural Networks



Recurrent Neural Networks

Recurrent Neural Networks (RNNs) learn temporal patterns

- $h_t = f_W(h_{t-1}, x_t)$
- The RNN maintains some sort of "memory" of its current state
- The new state h_t is some function of the previous state and the input x



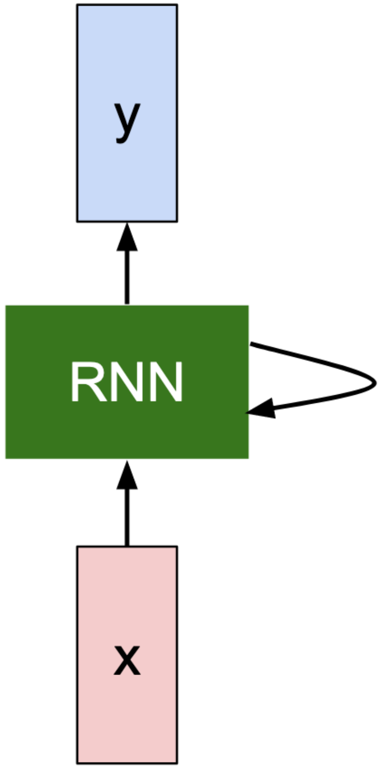
Self information from sequence

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) learn temporal patterns

- $h_t = f_W(h_{t-1}, x_t)$
- The RNN maintains some sort of "memory" of its current state
- The new state h_t is some function of the previous state and the input x

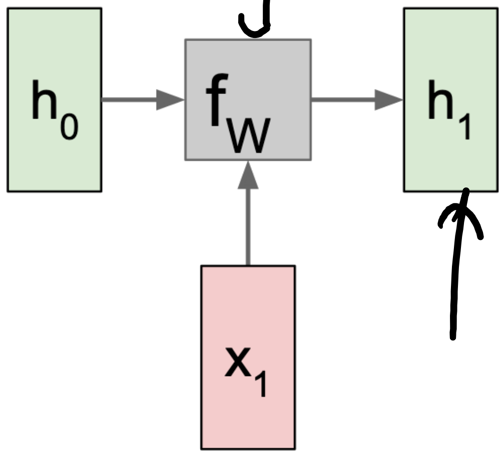
- Commonly, we use the following
  - $h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$ ← Produces a vector "state" of the RNN at time
  - $y_t = W_{hy}h_t$
  - Where all W are learned matrices



Recurrent Neural Networks

Matrix Mult

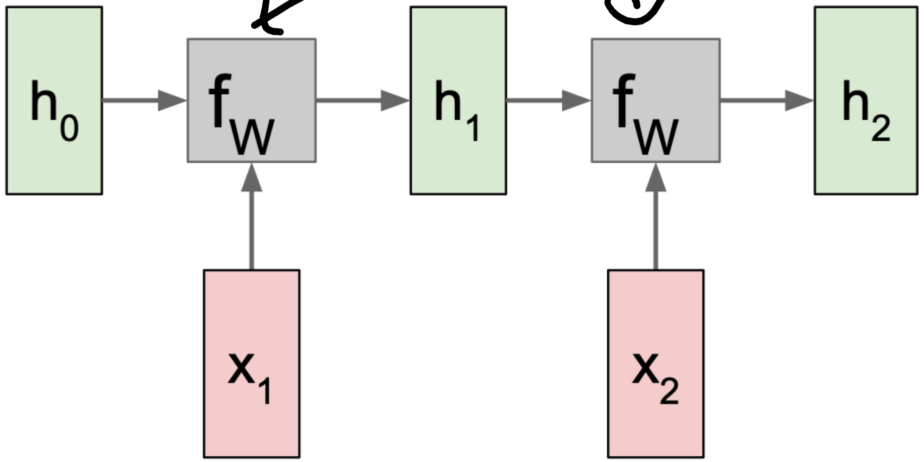
Start state



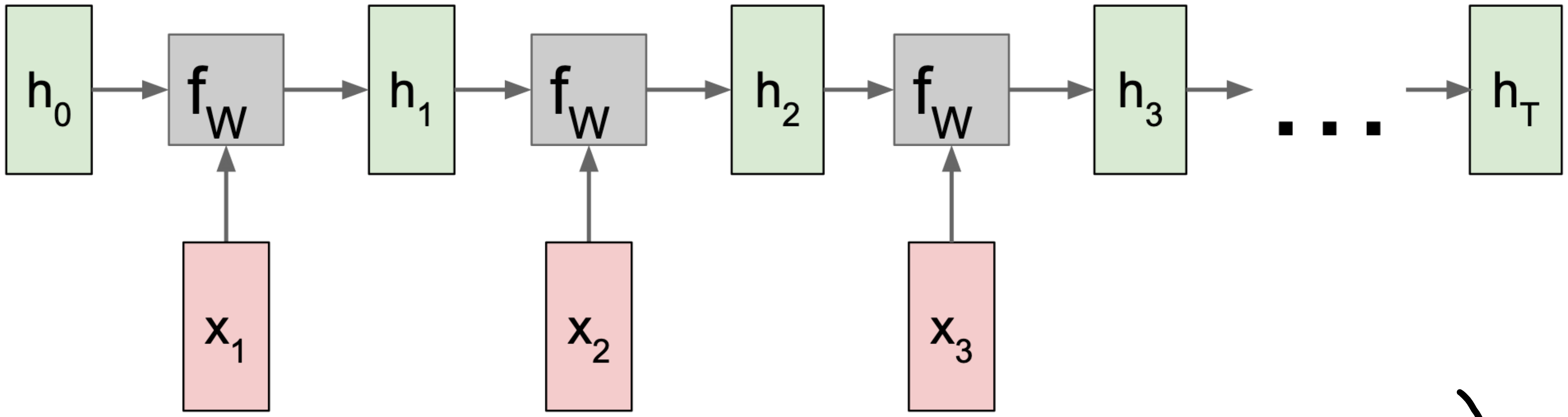
usually

Recurrent Neural Networks

Same neuron! Same W_{hh} W_{hx} W_{hy}



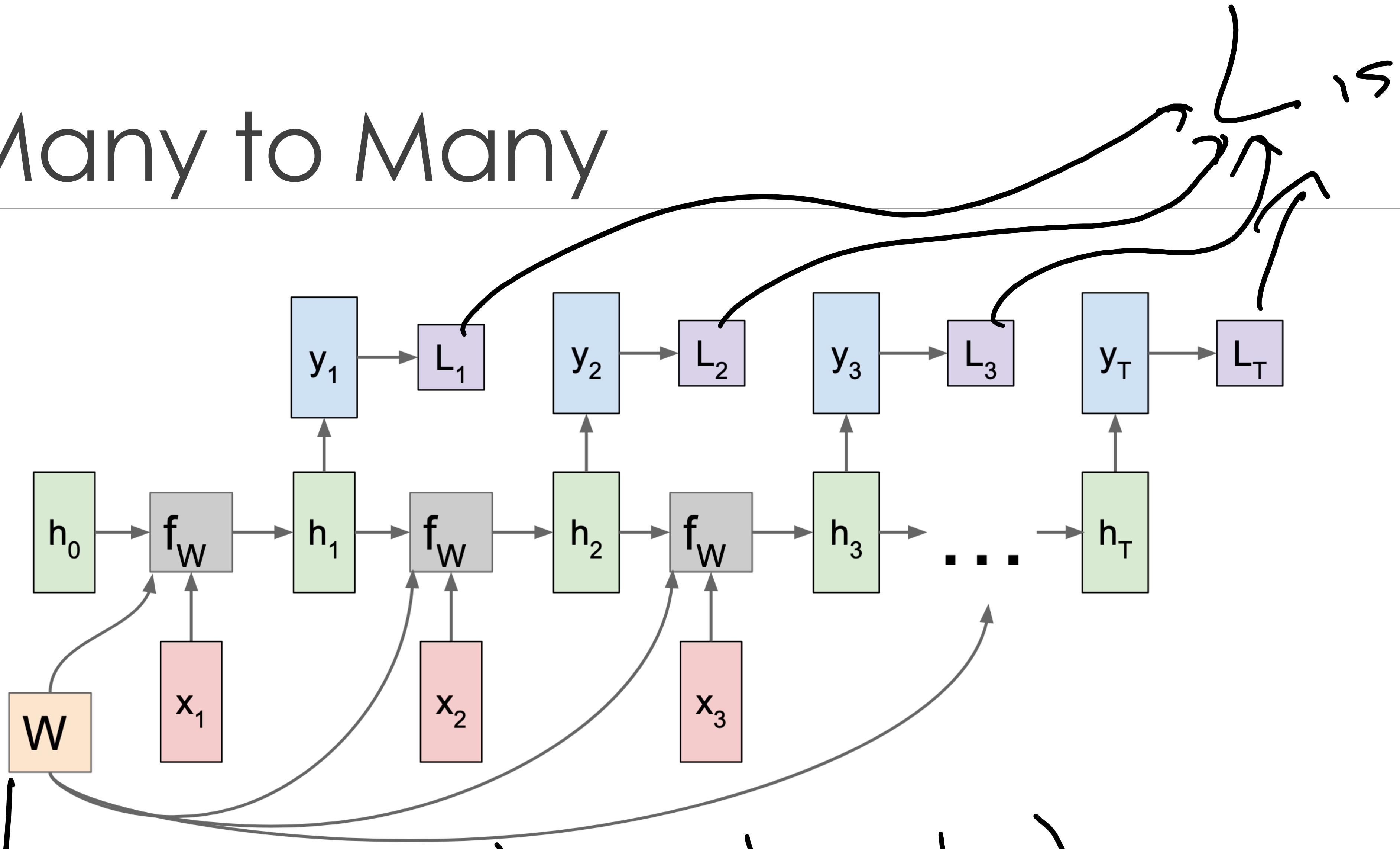
Recurrent Neural Networks



encodes states

No outputs here

Many to Many



↳ Same weights at each time

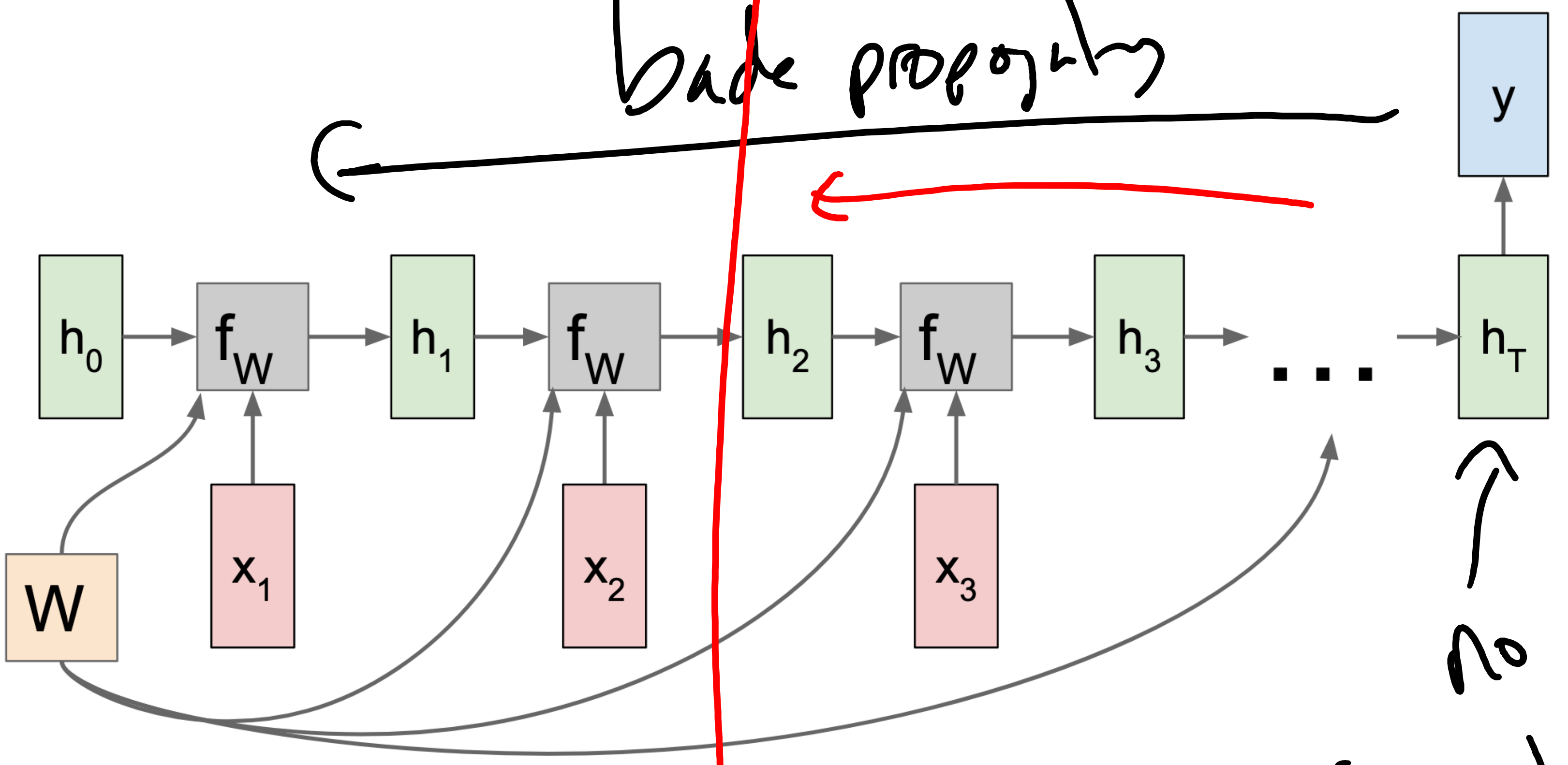
Many to One

truncate

only bp a few layers

get loss

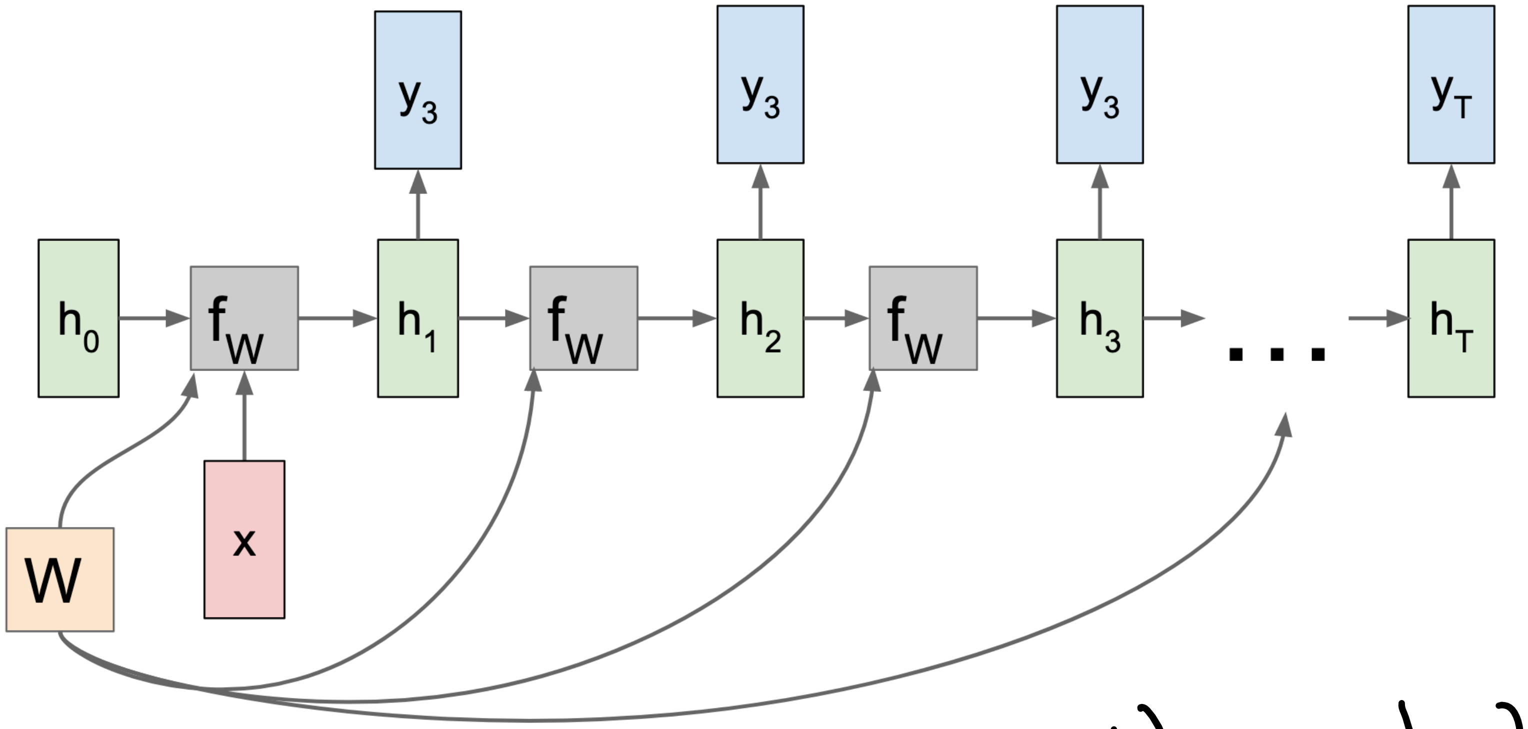
back propagating



no input here

"feed forward"

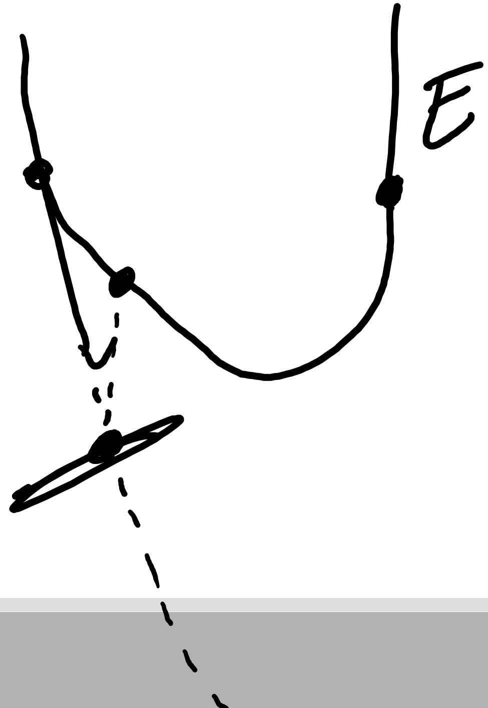
One to Many



go through all points to get loss
and back prop

Usage

- Good for time series data ← data in a sequence
- Sequenced data ← different sizes for inputs and outputs
- Gradient training
  - Exploding gradient — if gradient multiple > 1, gradient clipping
  - Vanishing gradient — if each layer < 1, different architecture, truncated backprop
    - LSTM
    - GRU



Conclusion

Deep learning algorithms

- Structure → why
- Purpose
- Training

↳ lots of weights → optimize

How do we extract relevant features?

- Time locality — RNN
- Space locality — CNN
- ???

autoencoder
CNN
RNN
} feature extraction

post slides
MAP
MLE

Next Class

Ethical considerations

Exam discussion

Package Resources PyTorch

Name	Language	Link	Note
Pylearn2	Python	http://deeplearning.net/software/pylearn2/	A machine learning library built on Theano
Theano	Python	http://deeplearning.net/software/theano/	A python deep learning library
Caffe	C++	http://caffe.berkeleyvision.org/	A deep learning framework by Berkeley
Torch	Lua	http://torch.ch/	An open source machine learning framework
Overfeat	Lua	http://cilvr.nyu.edu/doku.php?id=code:start	A convolutional network image processor
Deeplearning 4j	Java	http://deeplearning4j.org/	A commercial grade deep learning library
Word2vec	**C**	**https://code.google.com/p/word2vec/**	**Word embedding framework**
GloVe	C	http://nlp.stanford.edu/projects/glove/	Word embedding framework
Doc2vec	C	https://radimrehurek.com/gensim/models/doc2vec.html	Language model for paragraphs and documents
StanfordNLP	**Java**	**http://nlp.stanford.edu/**	**A deep learning-based NLP package**
TensorFlow	**Python**	**http://www.tensorflow.org**	**A deep learning based python library**