More on Deep Learning

CS 229

Overview

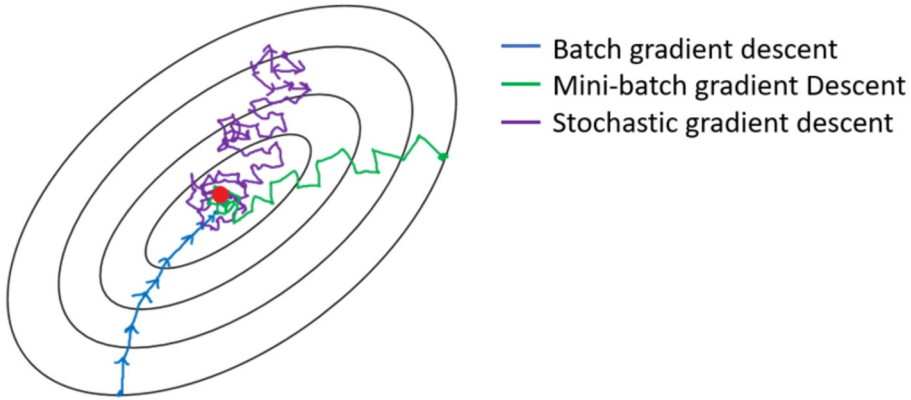
- Practicalities of deep learning
- Architectures
  - Convolutional neural networks
  - Recurrent neural networks
  - Transformers
- Deep learning tools (+ demo)

Mini-batch Gradient Descent

Gradient computation requires $O(n)$ compute time and memory size. Expensive for large datasets!

Mini-batch: randomly sample a subset of training data. Only requires $O(b)$ compute and memory ☺

Likely requires more steps. But, overall complexity still favorable.

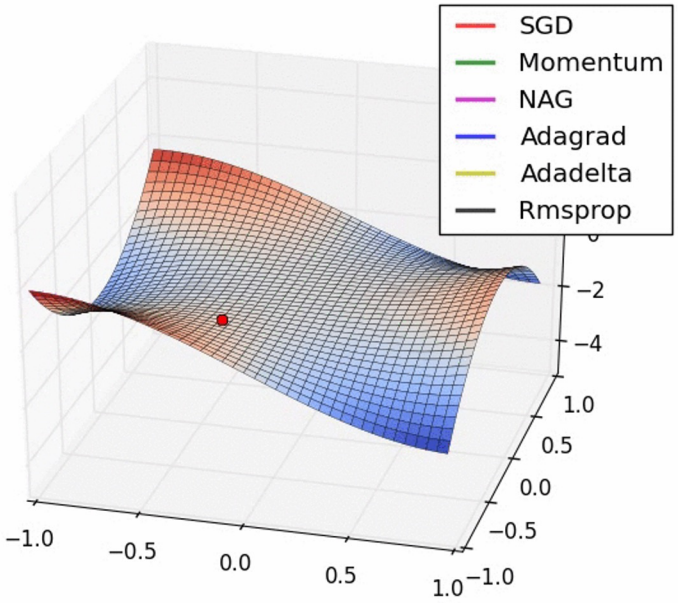


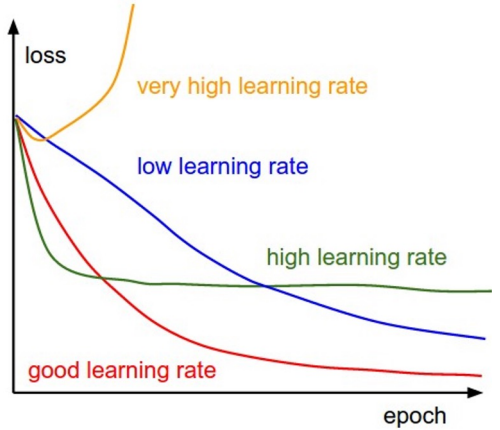
Fancy optimizers

- Momentum: $v_{t+1} = \mu v_t + \hat{\nabla}\ell(\theta_t)$

  $\theta_{t+1} = \theta_t - \alpha v_{t+1}$

- AdaGrad: Normalize grad with sqrt of sum of squared gradients
- RMSprop: Normalize grad with sqrt of moving avg of squared gradients
- Adam: RMSprop + momentum



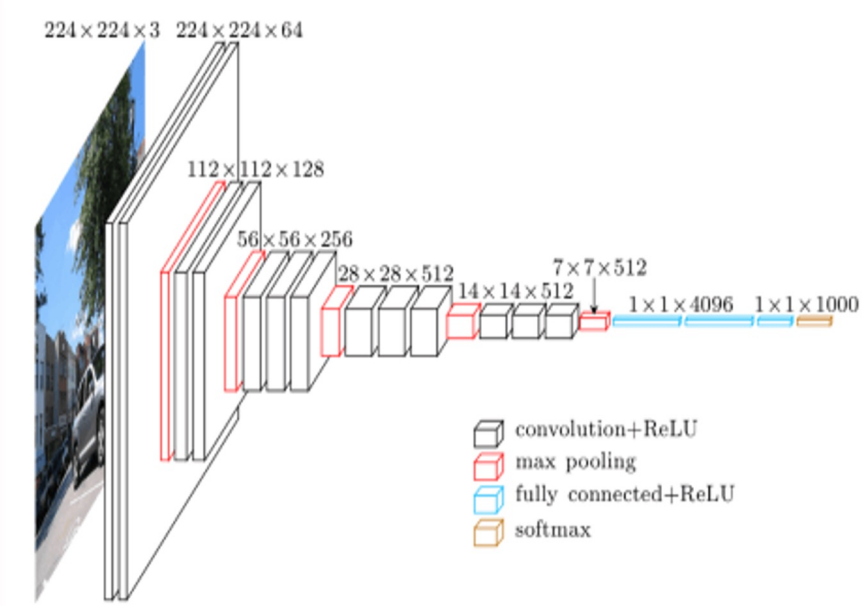


Is deeper better?

Deeper networks seem to be more powerful but harder to train.

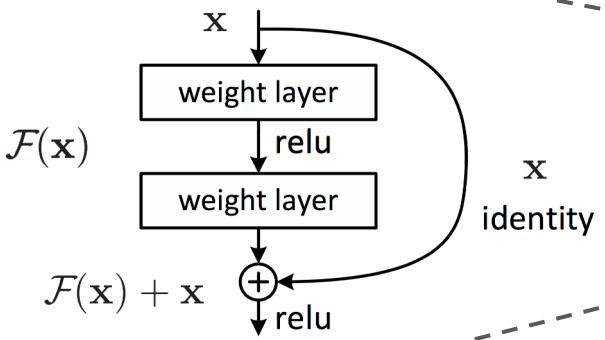
- Loss of information during forward propagation
- Loss of gradient info during back propagation

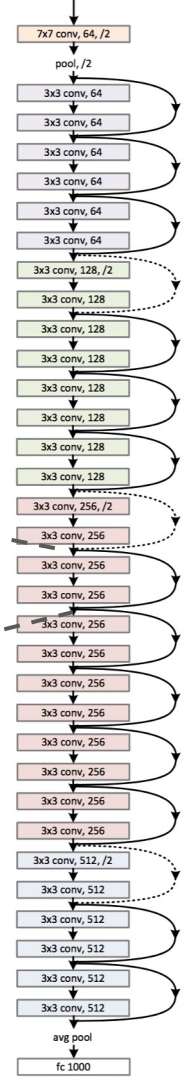
There are many ways to “keep the gradient going”



Solution: residual connections

Connect the layers, create a gradient highway or information highway.





ResNet (2015)

Image credit: He et al. (2015)

Initialization

Can we initialize all neurons to zero?

If all the weights are same we will not be able to break symmetry of the network and all filters will end up learning the same thing.

Large numbers might knock ReLU units out.

ReLU units once knocked out and their output is zero, their gradient flow also becomes zero.

We need small random numbers at initialization.

Mean: 0, Std: $O(1/\text{sqrt}(n))$

Popular initialization setups

Xavier/Glorot: n = fan-in + fan-out
Kaiming/He: n = fan-in

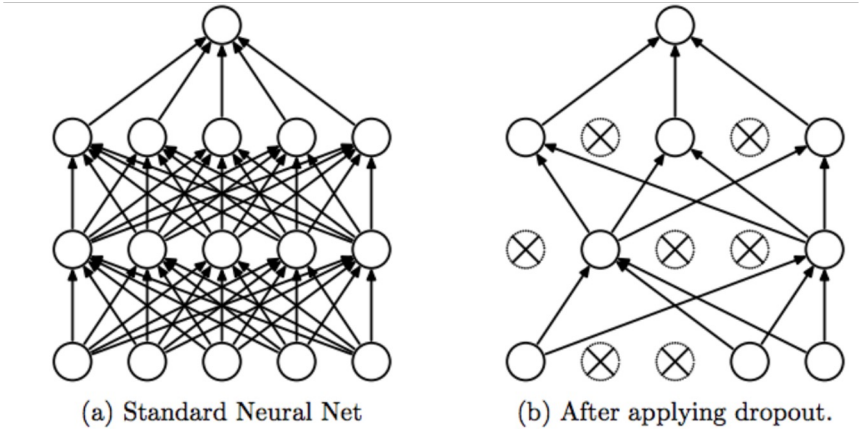
Dropout

What does cutting off some network connections do?

Trains many smaller networks in an ensemble.

Can drop entire layer too!

Regularizes training by preventing over-dependence on any particular neuron



(a) Standard Neural Net

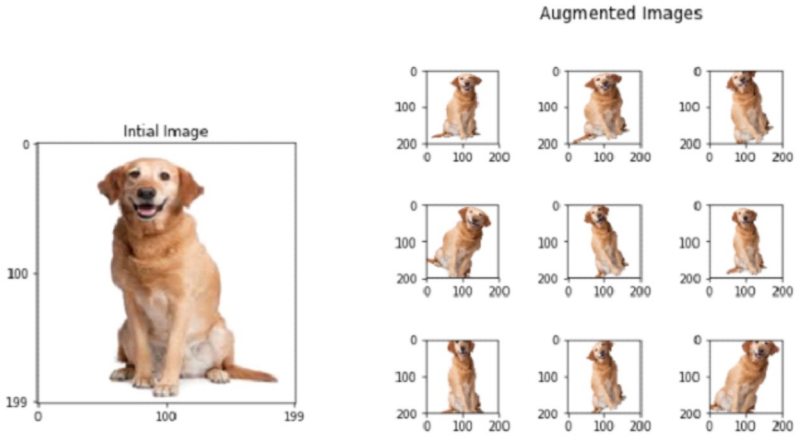
(b) After applying dropout.

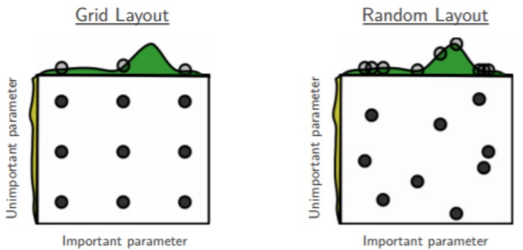
Other tricks for training

Data augmentation helps the network generalize more.

Early stopping if training loss goes above validation loss.

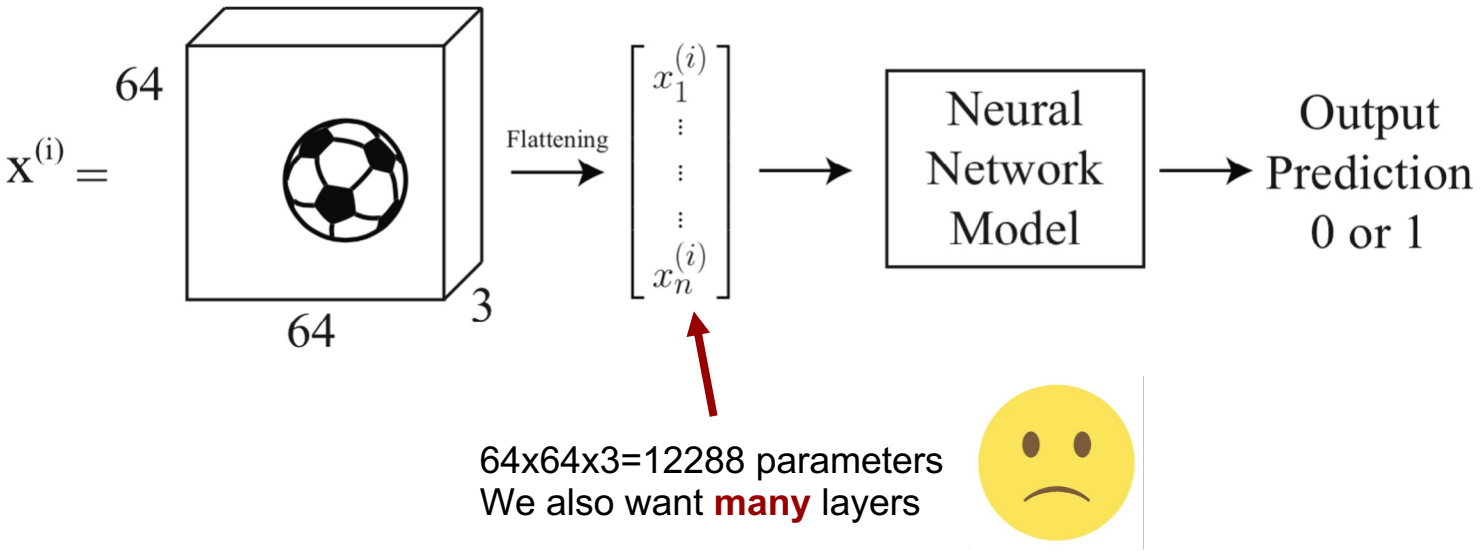
Random hyperparameter search or grid search?





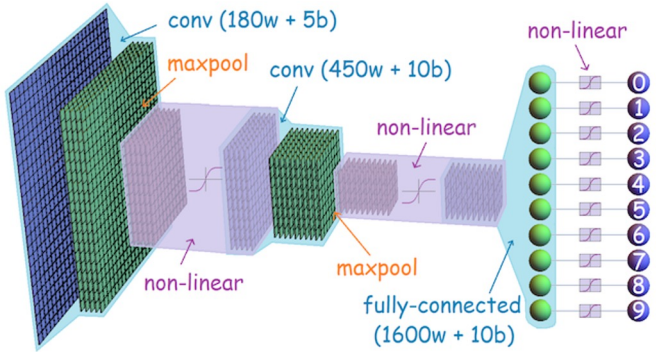
Why not just use MLPs for everything?

MLPs may not generalize well since they do not exploit the order and local relations in the data!

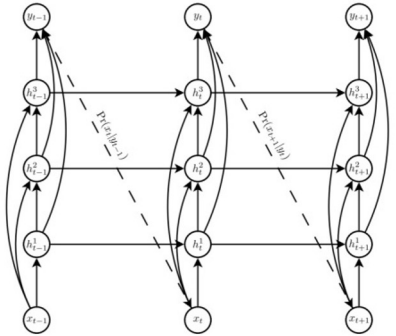


What are areas of deep learning?

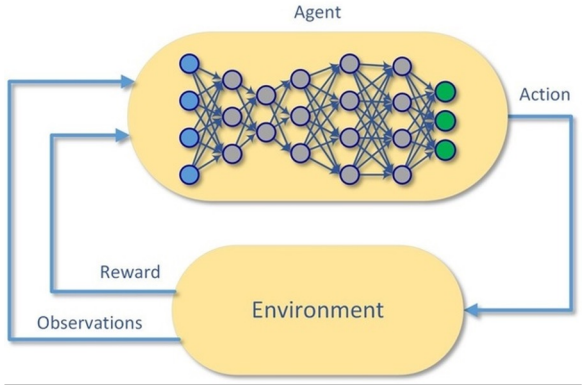
Images:
Convolutional NN



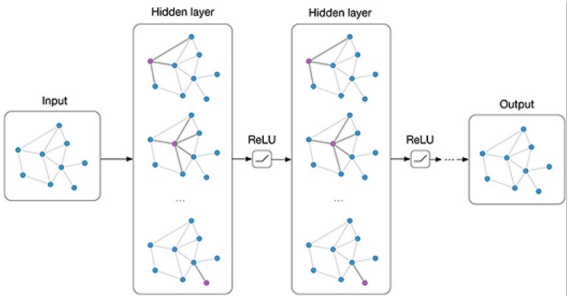
Sequences:
Recurrent NN
or
Transformer



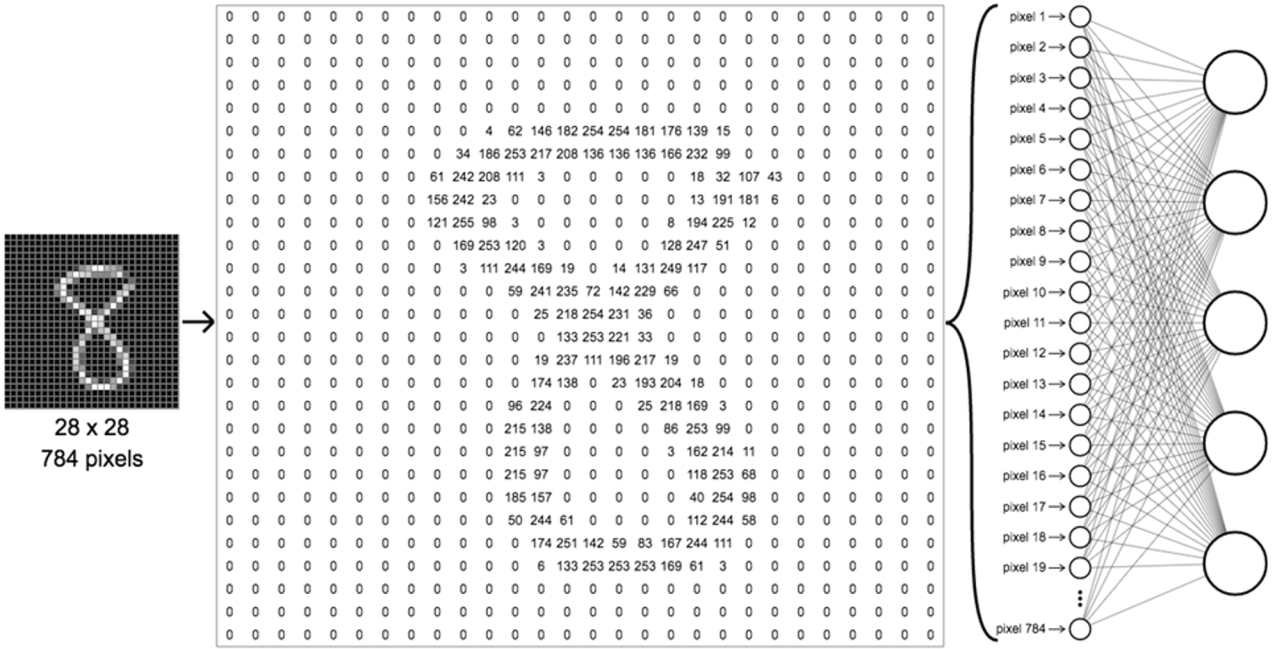
Control:
Deep RL



Relational:
Graph NN



Let us look at images in detail



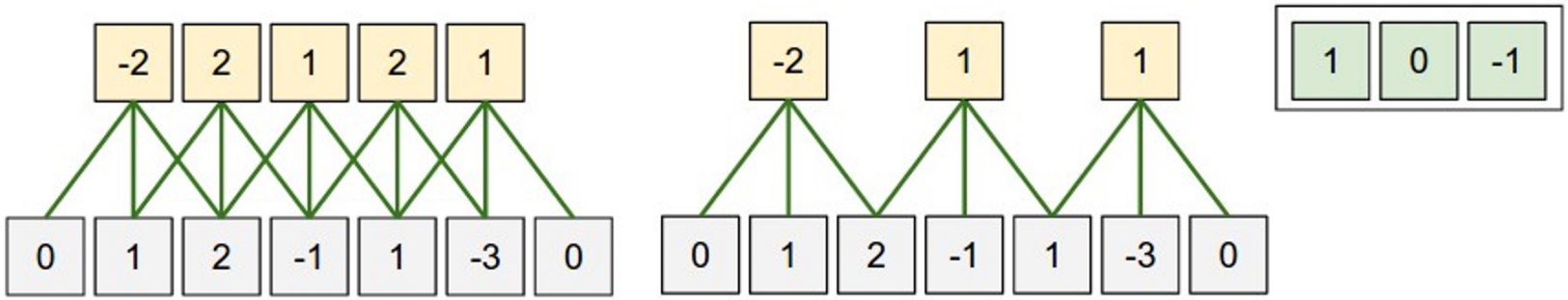
Note: grayscale images like this have only a single channel, but typical color images will have 3 (RGB)

Motivating convolutions

A cat may be appear anywhere in the image.

We should use the same features to detect cats regardless of where they are!

Parameter Sharing



Reduces the amount of memory required

May improve generalization

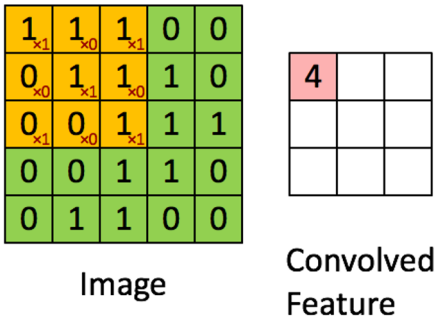
Convolutions

Convolutions apply a *filter*, sliding it over the image to compute features

Example filter: edge detector

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Can we do better than hand-designed filters? => Learn them!



Convolution is a linear operation

Convolution can be expressed as a giant matrix multiplication.

We can expand the 2 dimensional image into a vector and the conv operation into a matrix.

$$\begin{pmatrix} x1 & x2 & x3 \\ x4 & x5 & x6 \\ x7 & x8 & x9 \end{pmatrix} * \begin{pmatrix} k1 & k2 \\ k3 & k4 \end{pmatrix}$$

$$= \begin{pmatrix} k1\,x1 + k2\,x2 + k3\,x4 + k4\,x5 \\ k1\,x2 + k2\,x3 + k3\,x5 + k4\,x6 \\ k1\,x4 + k2\,x5 + k3\,x7 + k4\,x8 \\ k1\,x5 + k2\,x6 + k3\,x8 + k4\,x9 \end{pmatrix}$$

$$= \begin{pmatrix} k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 & 0 \\ 0 & k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 & 0 \\ 0 & 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 \end{pmatrix} \cdot \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \end{pmatrix}$$

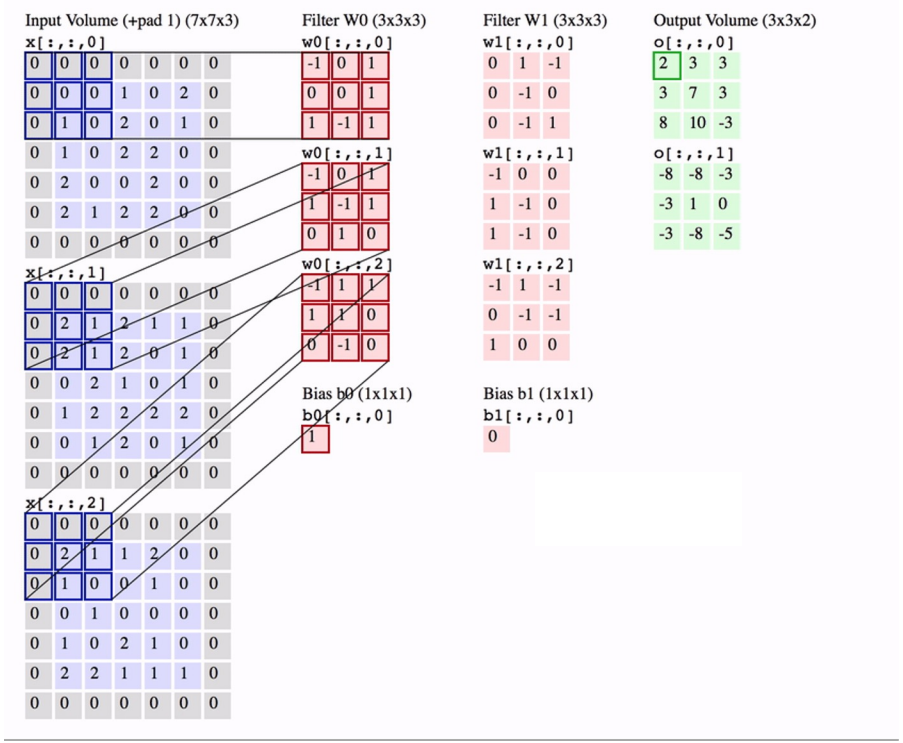
Convolution on multiple channels

Images are generally RGB

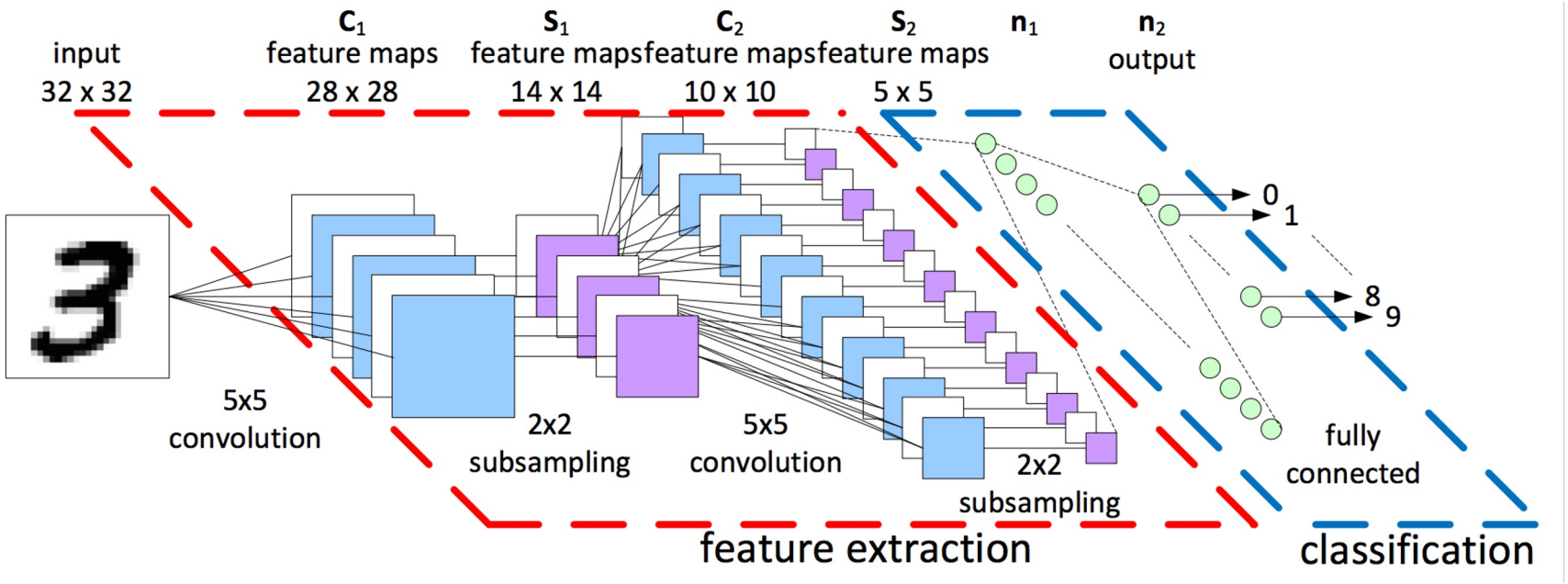
How would a filter work on an image with RGB channels?

The filter should also have 3 channels.

Now the output has a channel for every filter we have used.



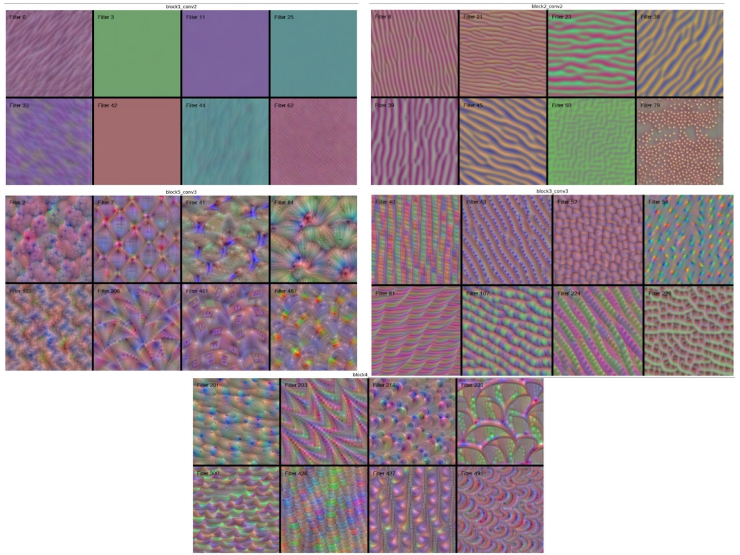
Convolutional neural networks



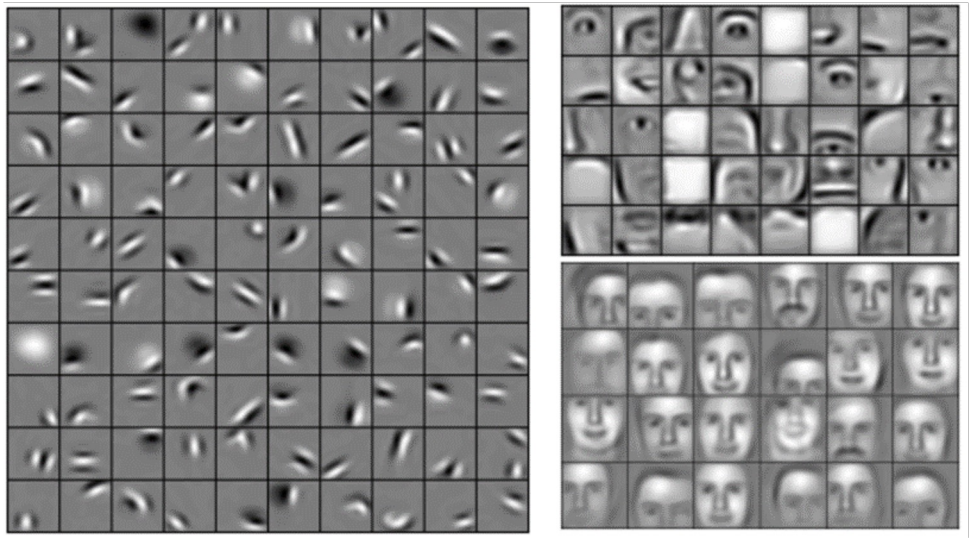
Learned conv filters extract features, final linear layer produces prediction

Image credit: LeCun et al. (1998)

Filters? Layers of filters?



Images that maximize filter outputs at certain layers. We observe that the images get more complex as filters are situated deeper



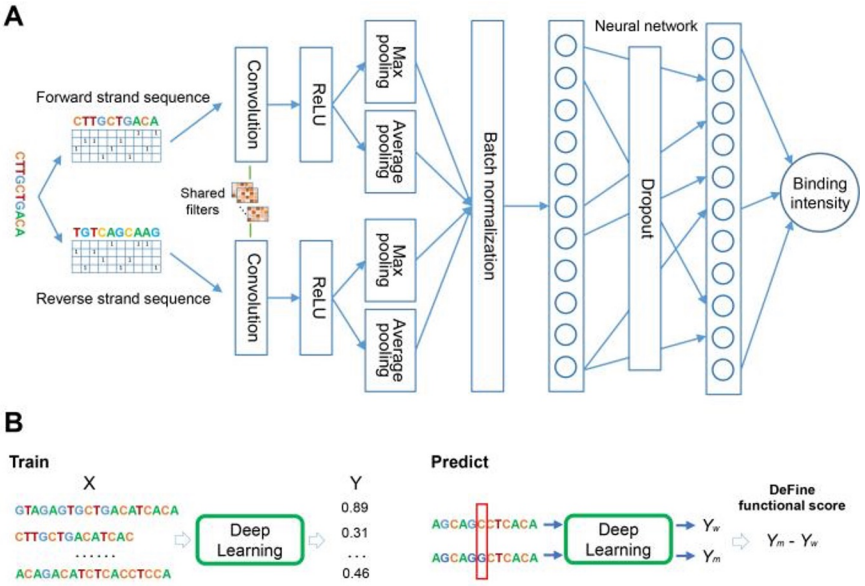
How deeper layers can learn deeper embeddings. How an eye is made up of multiple curves and a face is made up of two eyes.

1-D Convolutions

CNN works on any data where there is a local pattern

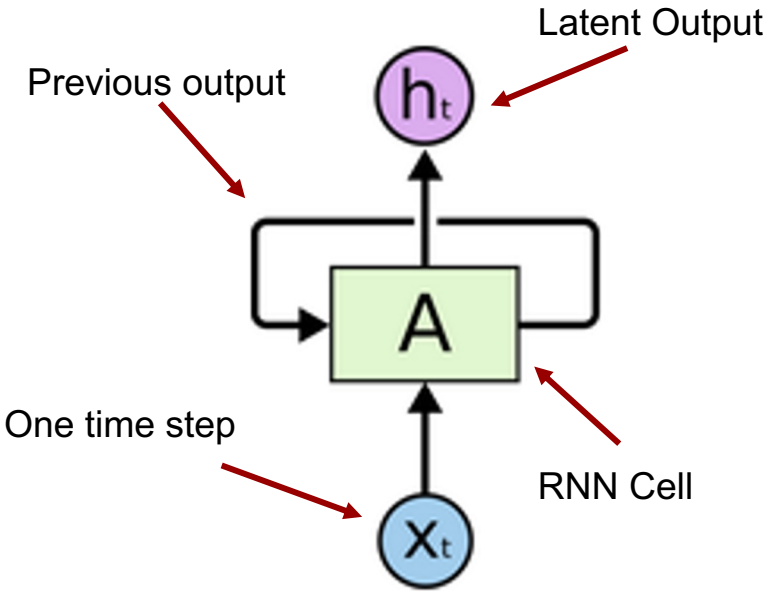
We use 1-D convolutions on DNA sequences, text sequences and music notes

But what if time series has **causal dependency** or any kind of **sequential dependency**?

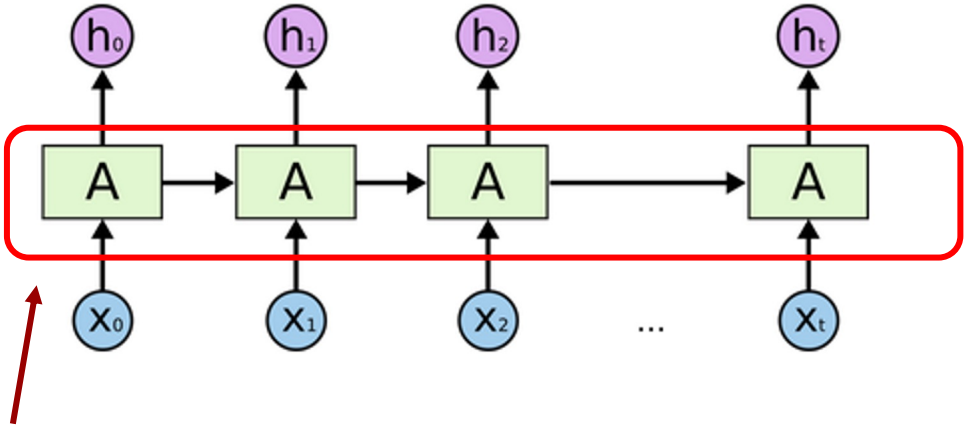


To address sequential dependency?

Use recurrent neural network (RNN)

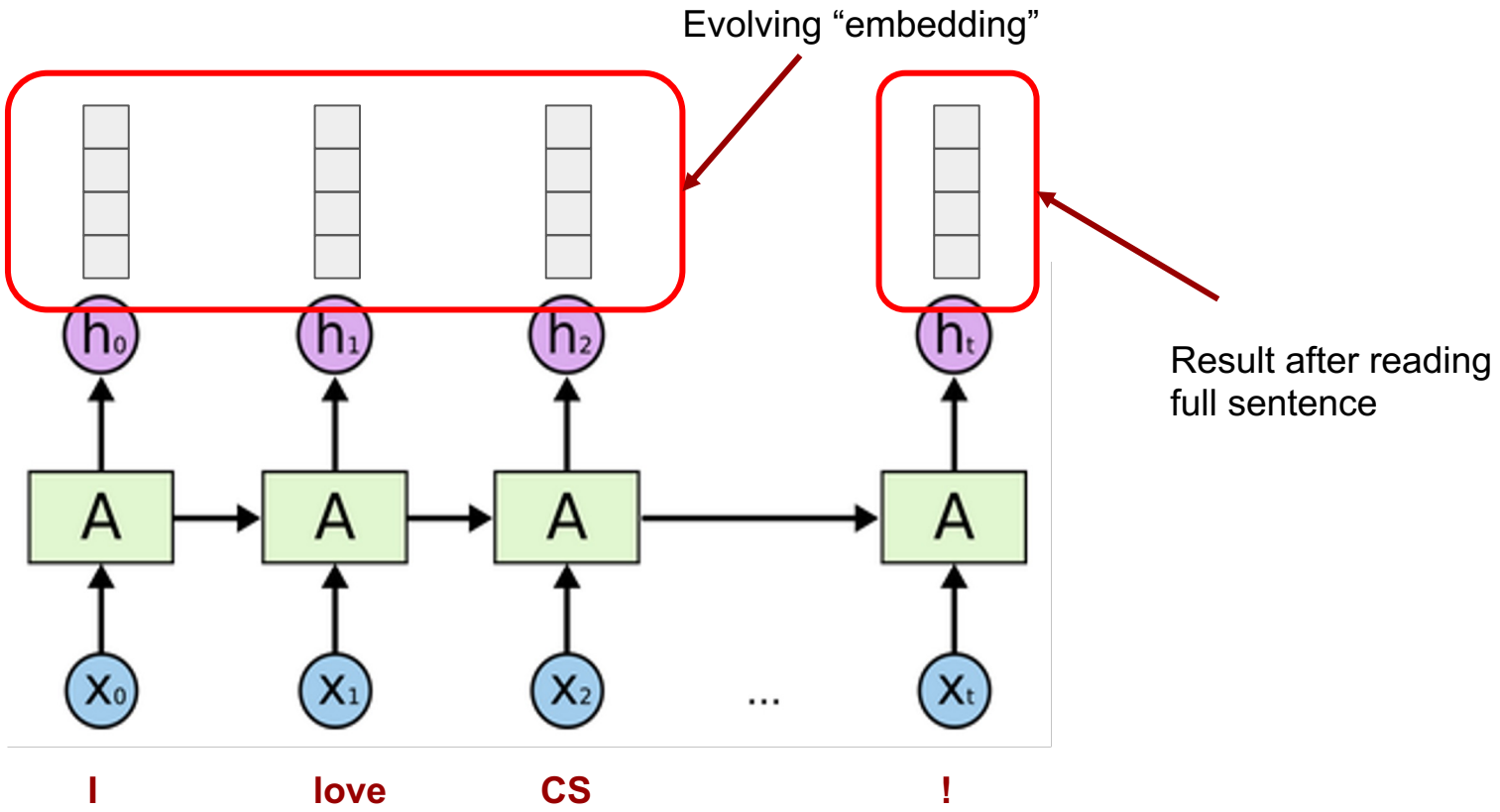


Unrolling an RNN



They are really the same cell,
NOT many different cells like kernels of CNN

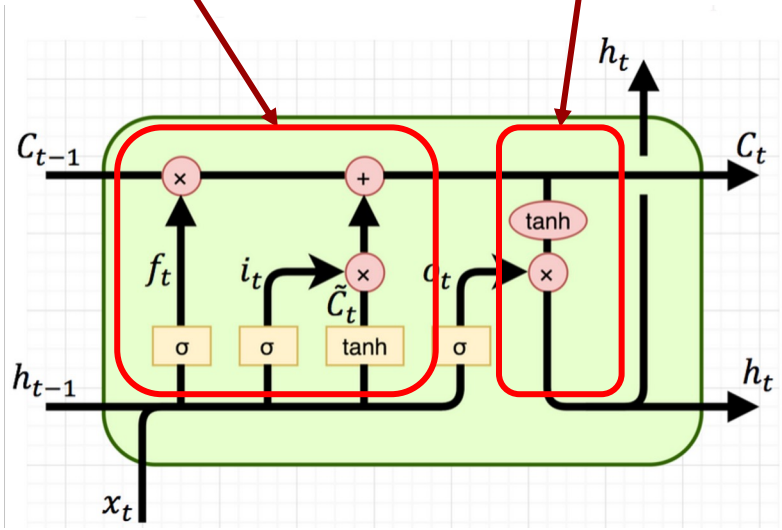
How does RNN produce result?



The two most common types of RNN cells

Store in “long term memory”

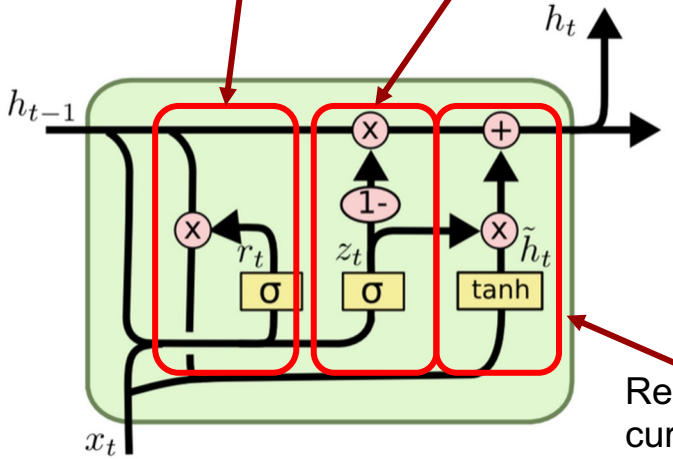
Response to current input



Long Short Term Memory (LSTM)

Reset gate

Update gate



Response to current input

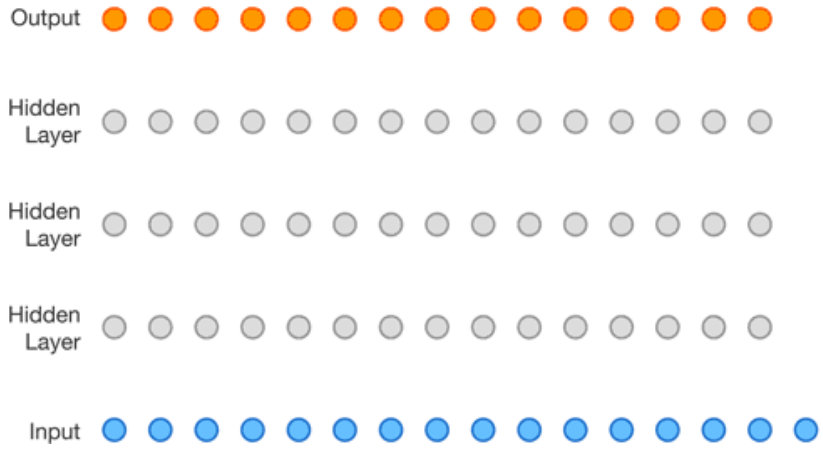
Gated Recurrent Unit (GRU)

“Recurrent” AND convolutional?

Temporal convolutional network

Temporal dependency achieved through one-sided, dilated convolutions

More efficient because deep learning packages are optimized for matrix multiplication = convolution



Transformers

Uses *attention* between pairs in the sequence

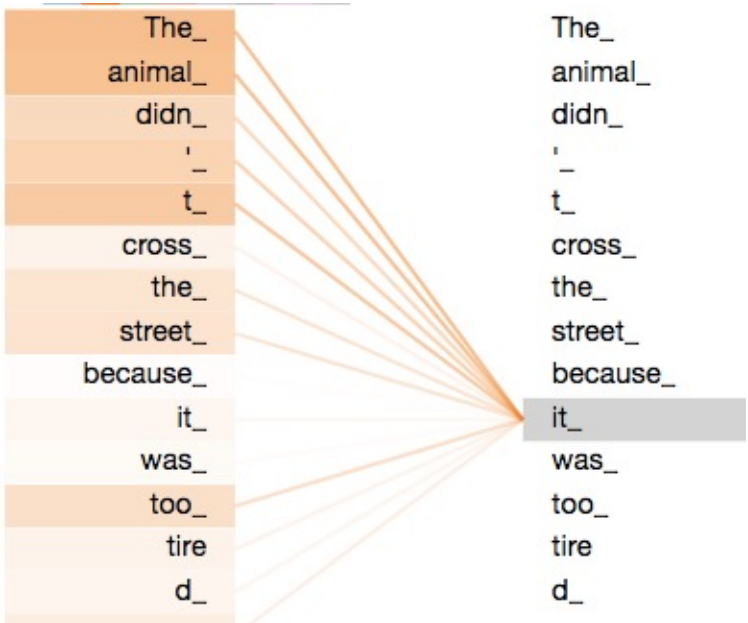
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$Q = XW^Q,\ K = XW^K,\ V = XW^V$$

Masking is used to condition only on past

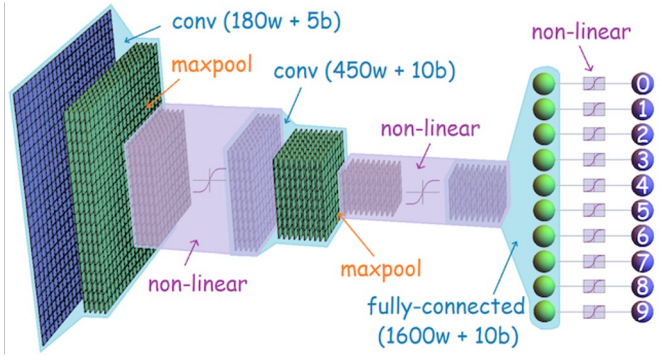
More scalable than RNNs:

* more parallelizable
* excellent performance as model size grows

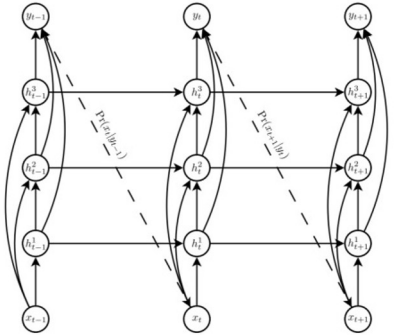


More?

CS 231N: Deep Learning for Computer Vision
Covers CNNs

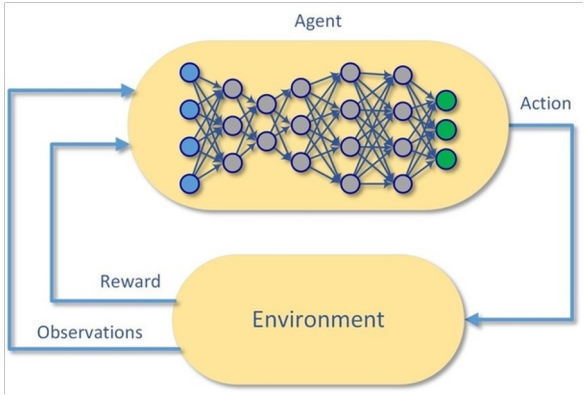


CS 224N: NLP with Deep Learning
Covers sequence models

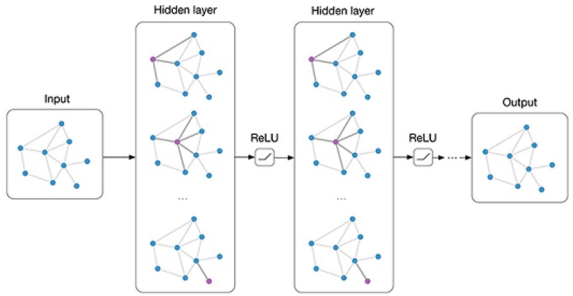


Not today, but take CS234 and CS224W

CS 224R: Deep Reinforcement Learning



CS 224W: Machine Learning with Graphs
Covers Graph NN



Tools for deep learning

Keras

TensorFlow

theano

PYTORCH

Popular Tools

Specialized Groups

Caffe2

mxnet

Microsoft CNTK

$50 not enough! Where can I get free stuff?

Google Colab

- Free (limited-ish) GPU access
- Works nicely with Tensorflow
- Links to Google Drive

Register a new Google Cloud account

- => Instant $300??
- => AWS free tier (limited compute)
- => Azure education account, $200?

To **SAVE** money:

CLOSE your GPU instance

~$1 an hour

Good luck!
Well, have fun too :D