DEEP LEARNING ARCHITECTURES

A QUICKSTART GUIDE TO THE NUTS & BOLTS OF DEEP LEARNING

- By: METIS

METIS

DEEP LEARNING OVERVIEW

- Deep Learning = Fancy ANNs (Artificial Neural Networks)
  - **Deep** means many hidden layers... think “patterns of patterns”
  - **Fancy** means complex network structures that capture more information from data
- Deep Learning neural networks are universal function approximators. But there is no free lunch! We pay for this power with having a **massive** number of parameters and hyperparameters to tune.
  - https://en.wikipedia.org/wiki/Universal_approximation_theorem

DEEP LEARNING APPLICATIONS

- Machine Translation
- Named Entity Recognition
- Image Processing (see CNNs lecture)
- Image + Video Captioning
- Chatbots
- Text Generation
- Question Answering

BIAS / VARIANCE TRADEOFF

- Bias = error from erroneous assumptions in the learning algorithm
- Variance = error from sensitivity to small fluctuations in training data
- Generalization = test performance vs. training performance
- As usual, larger training sets, dimensionality reduction, and feature selection can decrease variance by simplifying models.

- In neural networks, variance increases and bias decreases as the number of hidden units increases.
- Regularization can be applied via Dropout (we’ll see this soon).

FEED FORWARD / BACK PROP

The concepts in any Deep Learning model are all the same:

- Nodes feedforward to their successors and backpropagate to update weights to their predecessors
- Weights matrices represent transitions between nodes
- Depth can actually save us a lot of parameters
  - For dense layers, number of parameters is O(n^2) in nodes per layer, but O(n) in number of layers.

SIMPLE CONCEPTS, ENDLESS OPTIONS

- Take the input
- Run it thru the network
- Compute the Error/Cost
- Backpropagate to update weights via gradient of cost function
- **Everything is Gradient Descent.**
- That being said, Deep Learning will drown you in options!

CHOOSE THE ARCHITECTURE

Layers and Activation functions determine what kinds of patterns the network will detect in your data.

- Number of Layers
- Types of Layers
- Number of Neurons per Layer
- Activation Function for each Layer

CHOOSE AN OPTIMIZER

Optimizer algorithms are just variations of Gradient Descent

- SGD
- SGD w/ momentum
- Adam
- Adagrad
- RMSprop

CHOOSE A LOSS FUNCTION

- Mean Squared Error
- Hinge
- Binary Cross Entropy (Log loss)
- Categorical Cross Entropy
- Many more

GENERAL WORKFLOW

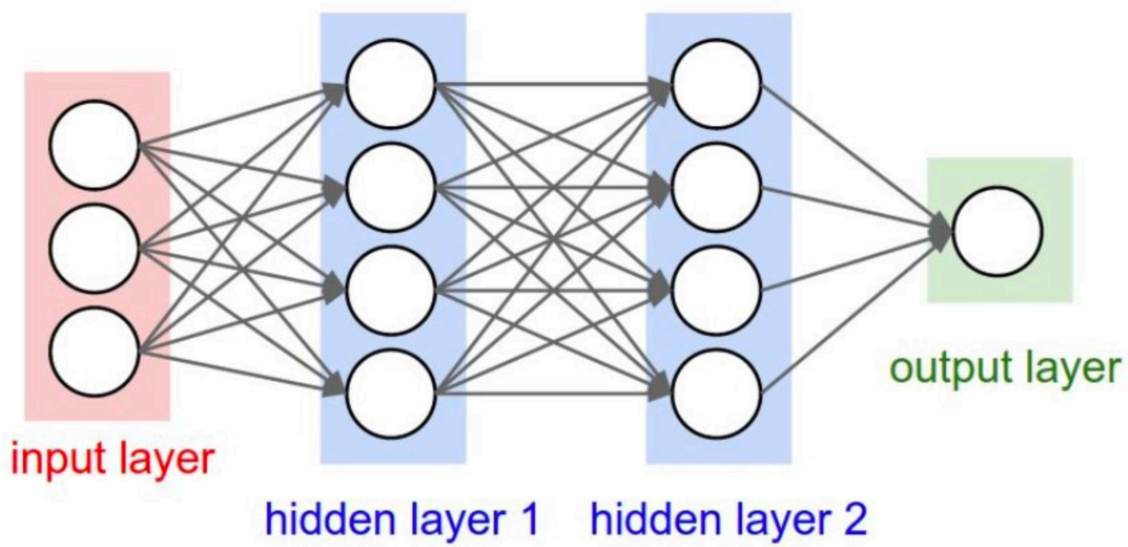
- Find an architecture that meets or exceeds your performance goals on the training data (potentially overfitting)
- Introduce regularization (dropout) to force generalization
- If training is too slow:
  - use GPUs to speed up calculations
  - try transfer learning from pretrained model or from autoencoder

Deep Learning Layers

WE'RE JUST PLAYING WITH LEGOS FROM HERE ON OUT

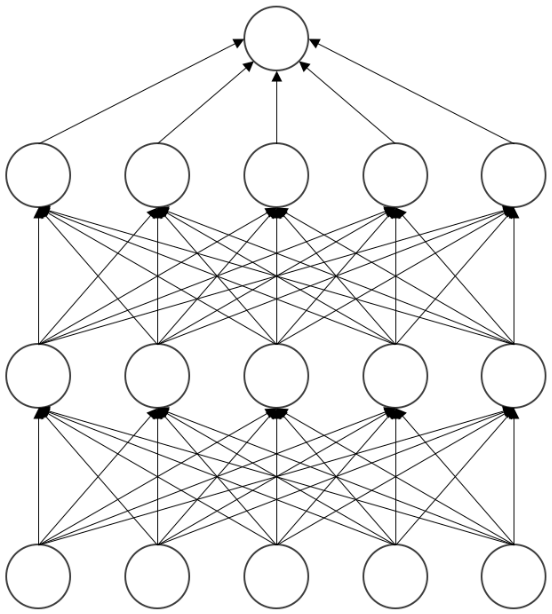
DENSE LAYERS

- Simplest type of neural network layer to understand.
- Also known as “fully connected” layers.
- Every node in the output layer is a linear combination of each node in the input layer, passed through an activation function.
- Basically just a layer full of logistic regressions with different weights.

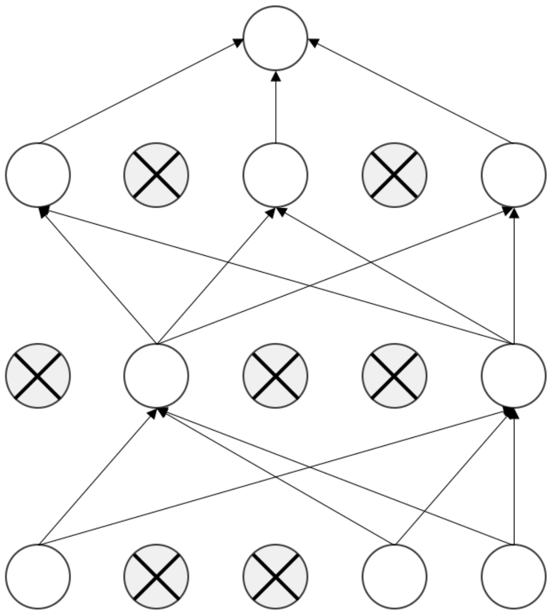


DROPOUT LAYERS

- Dropout is Deep Learning's take on Regularization.
- Weights are randomly ignored during training so that the remaining nodes are forced to learn useful information.



Standard Neural Net



After applying dropout

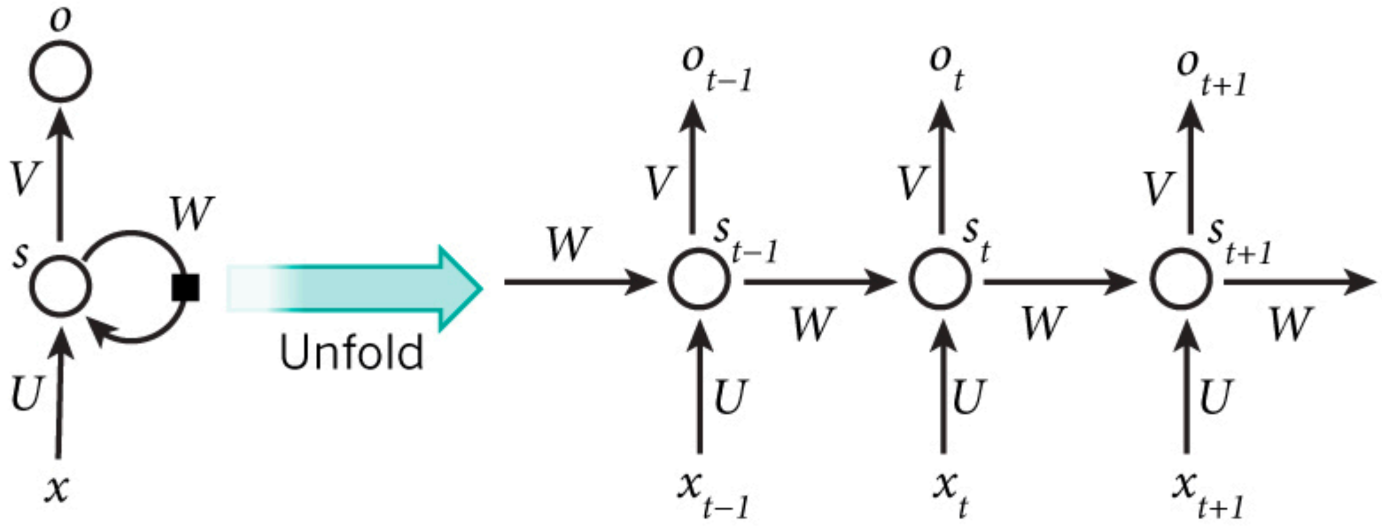
source: [Deep Learning for Computer Vision by Rajalingappaa Shanmugamani]

RECURRENT NEURAL NETWORKS

- So far, we've stuck to simple ANNs (fully connected)
- Recurrent Neural Networks (RNNs) change the game
- Connections between units form a **directed cycle**
- Allows network to retain internal state from past units → **memory**
- Allows dynamic temporal behavior
- Can use memory to process arbitrary input sequences!
- Terrific success in various NLP tasks

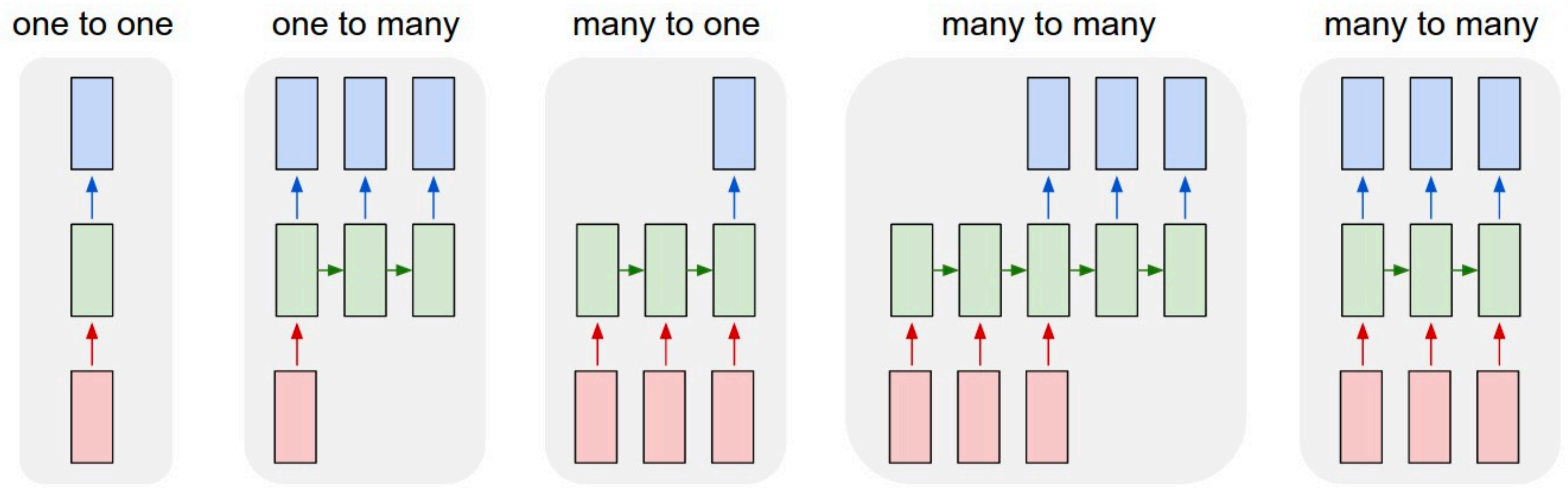
RECURRENT LAYERS

- How do they do it?
- Hidden Units at a time step t are dependent on:
  - The previous hidden unit
  - The input at time step t



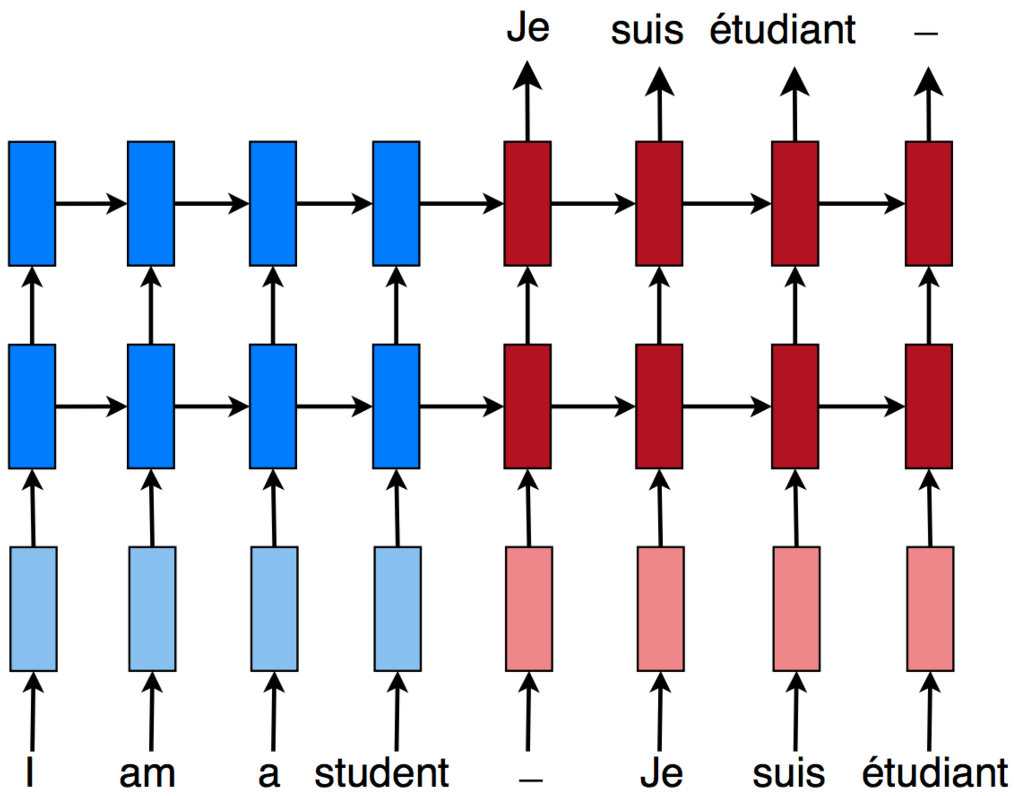
RECURRENT LAYERS

- Drawback of ANNs: Fixed # of inputs and outputs
- RNNs can map arbitrary length sequences of each.
- Don't stress! Just a different architecture with some nice features. Same solving concepts apply. (Keras takes care of this for you anyway!)



RECURRENT LAYERS

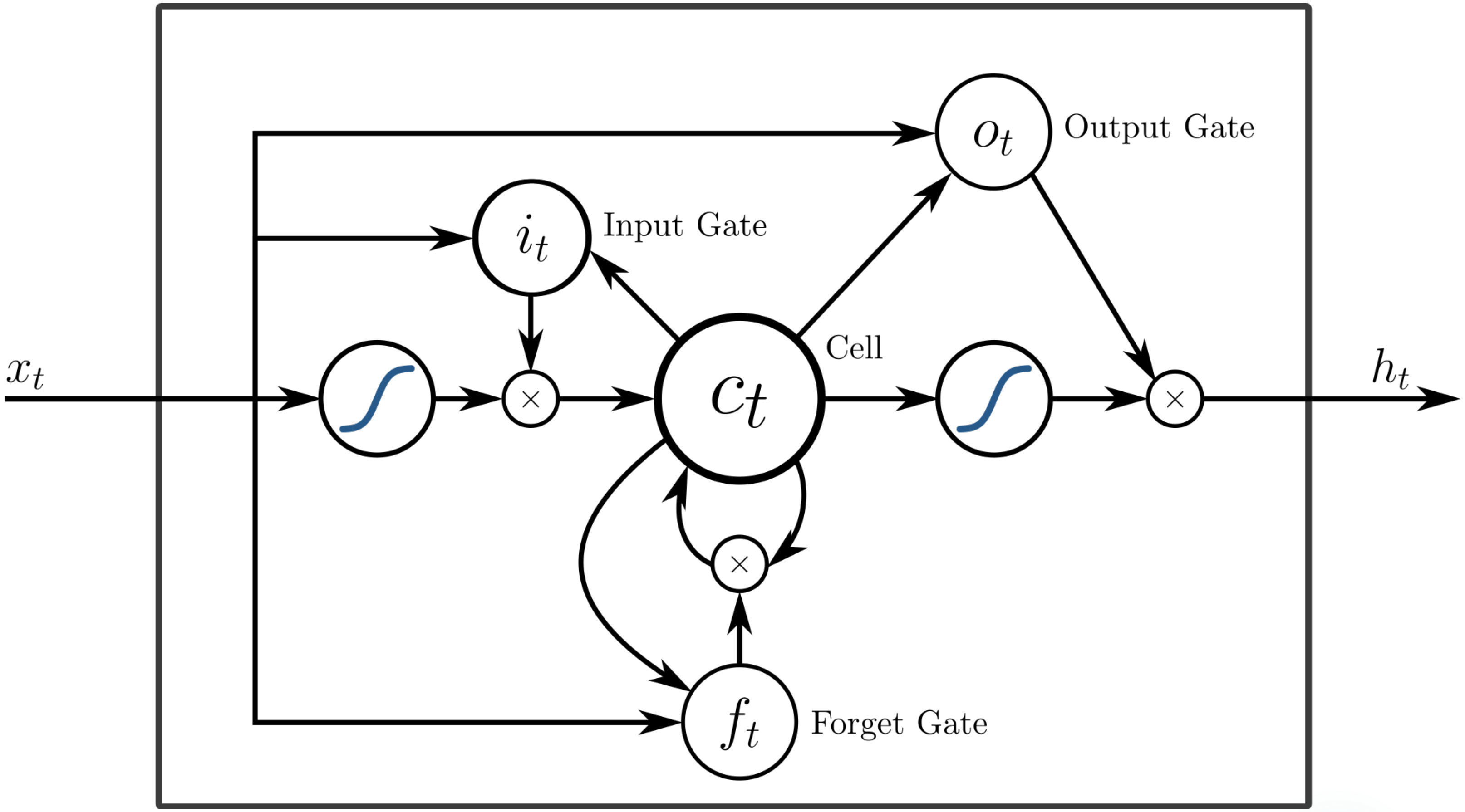
- Sequence-to-sequence mapping



LONG SHORT-TERM MEMORY NETWORKS

- LSTMs are a special kind of RNN (invented in 1997)
- “State of the art” (the idea is old but the available computing power is new) for many sequence to sequence mapping and text generation tasks
- Adds an explicit "memory" unit
- Augment RNNs with a few additional **Gate Units**
  - Gate Units control how long/if events will stay in memory
  - **Input Gate**: If its value is such, it causes items to be stored in memory
  - **Forget Gate**: If its value is such, it causes items to be removed from memory
  - **Output Gate**: If its value is such, it causes the hidden unit to feed forward (output) in the network

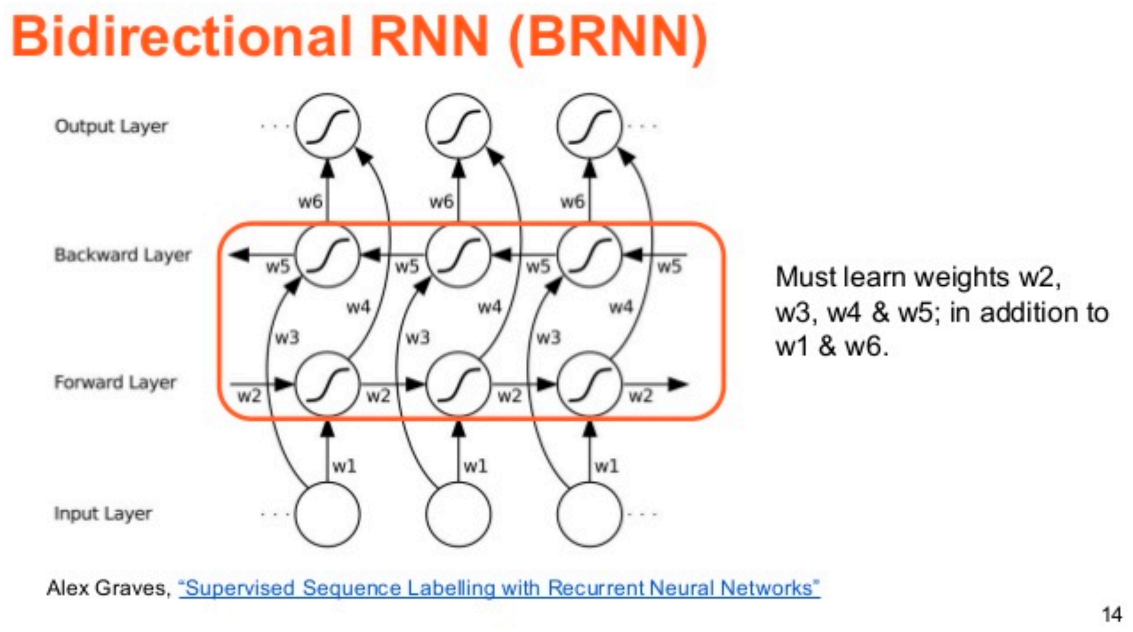
LONG SHORT-TERM MEMORY NETWORKS



https://en.wikipedia.org/wiki/Long_short-term_memory

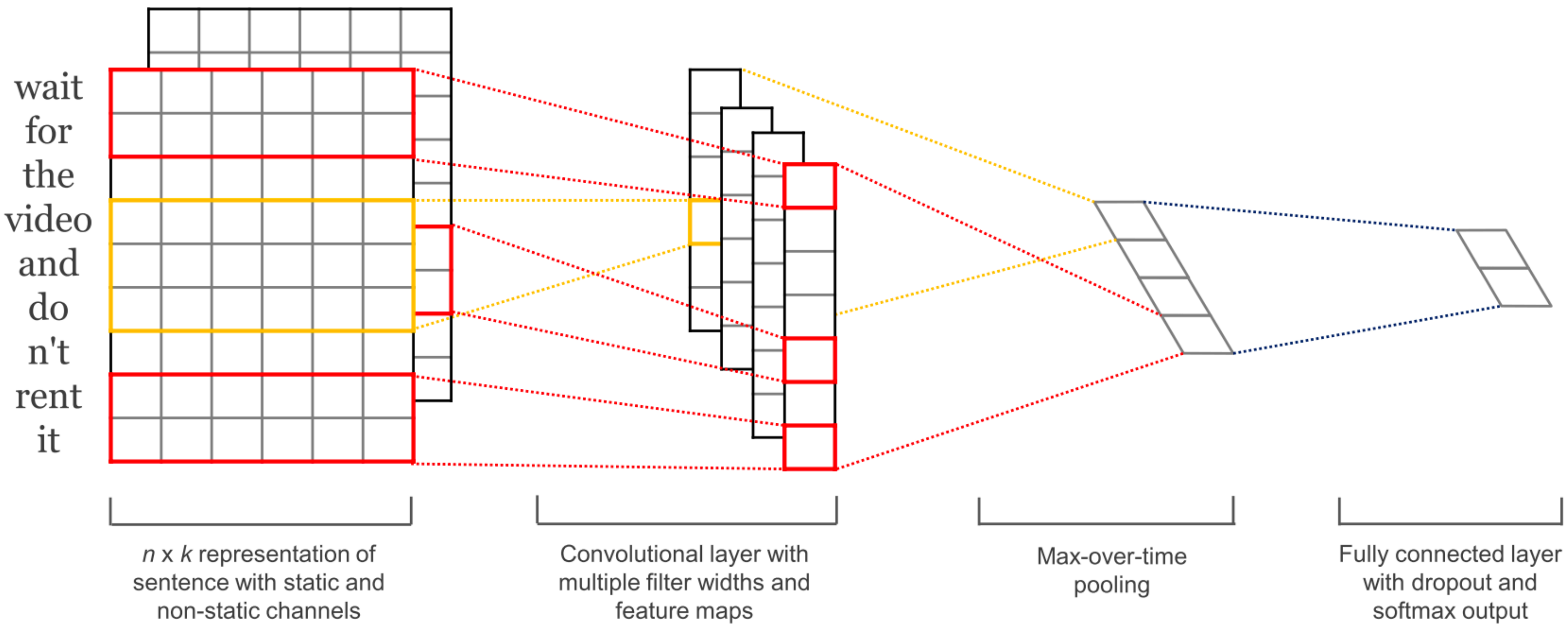
BIDIRECTIONAL RECURRENT NEURAL NETWORKS

- Bidirectional RNNs simply connect in both directions
- Thus, output can be dependent on both future and past inputs
- Good for context around a word, for instance
  - e.g. Named Entity Recognition, is this a "person" token?



CONVOLUTIONAL AND POOLING LAYERS

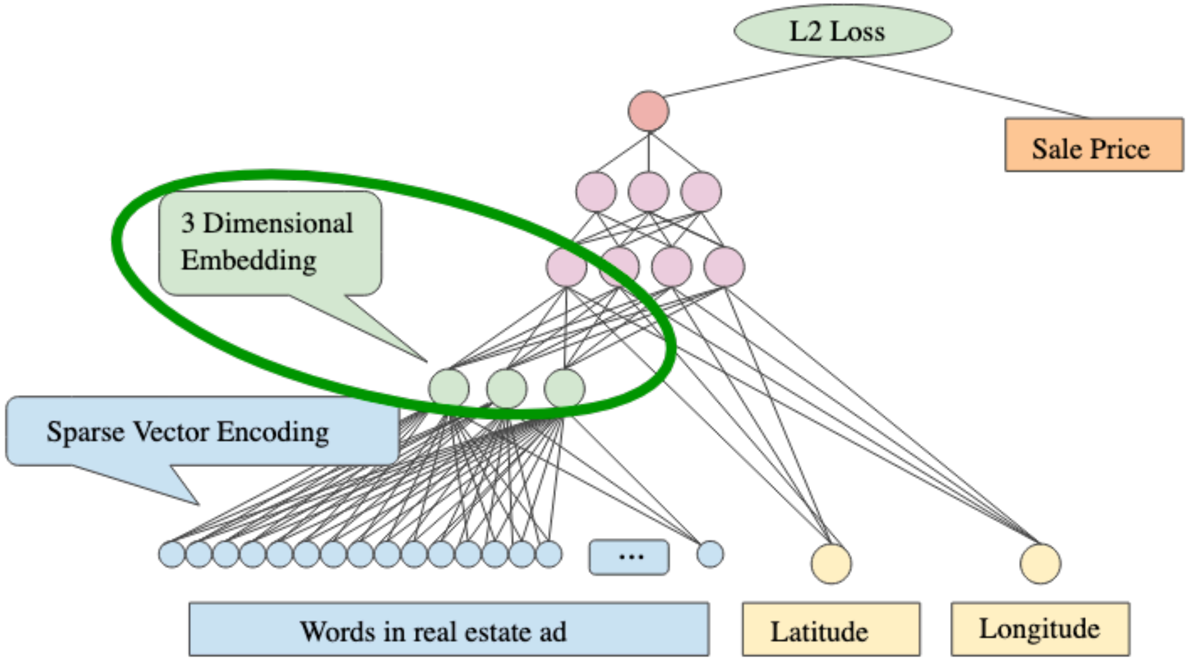
Commonly used for Image data, but they work for NLP too!



EMBEDDINGS, MERGE LAYERS

- Embeddings turn positive integers (indexes) into dense vectors of fixed size… perfect for mapping tokenized word vectors into a latent space.
- Merge layers combine outputs from multiple layers, allowing us to concatenate network branches.

(These are separate concepts but this image contains good examples of both layers.)



THANK YOU!