

# An Intro to Deep Learning/ Artificial Neural Networks

Slides by Mark Llorente  
Notebook Borrowed Heavily  
from Jack Bennetto

``pip install keras`` or

<https://github.com/ignaciorlando/skinner/wiki/Keras-and-TensorFlow-installation>



## Morning Objectives:

- Build intuition with neural nets
- Describe a neural net
- Explain back propagation and use activation functions in NNs

## Afternoon Objectives:

- Build a NN on Keras
- Know how to tune structure *and* hyperparameters
  - Get convergence during training
  - Avoid under-learning and over-fitting
- Be familiar with different styles of NNs

# Let's Make Friends with Artificial Neural Networks!

A Great Series of Intro Articles:

<http://colah.github.io/>

<https://jalammr.github.io/visual-interactive-guide-basics-neural-networks/>

Interactive 2 Feature Neural Network with Cool Visuals :

<https://playground.tensorflow.org/>

Structure of a Neural Net: Big House/Empty Offices Metaphor

# Parts of a Neural Network

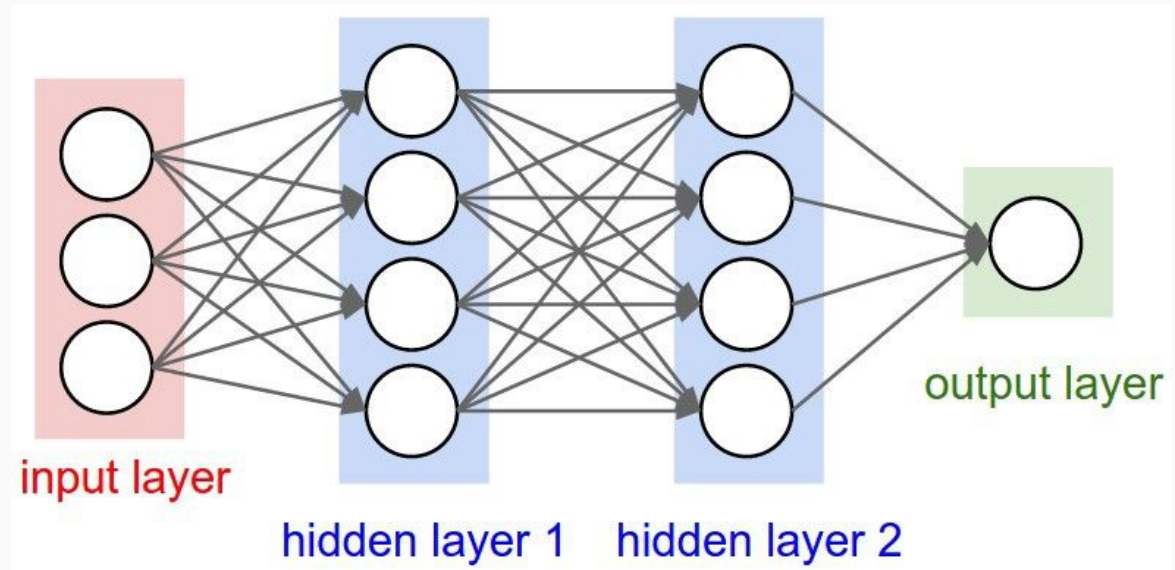
Nodes

Connections

Layers

Input Layer

Output Layer



# Description/Analogy (Graph Theory)

Nodes: Nodes, vertices

Connections: Edges (Directional)

Layers: *No Direct Analogy* but we often talk about order of layers similar to order of neighbors

Input Layer: Starting Terminal Nodes

Output Layer: Ending Terminal Nodes

# Description/Analogy (Ensemble of Regressors)

Nodes: Individual regressors

Connections: Inputs (connections from earlier nodes) and outputs (from each single node broadcast to all other nodes)

Layers: *No Direct Analogy*

Input Layer: Set of regressors that each take the full input data

Output Layer: Final regressor(s) to perform the prediction

# Description/Analogy (Latent Feature Detection)

Nodes: Latent feature detector

Connections: Input (values of latent features discovered from previous layer) and output (current latent feature detector's evaluation based on input)

Layers: Family of latent feature detectors that are trained to use the previous layer

Input Layer: Latent feature detectors that use original data

Output Layer: Final latent feature detector(s) to perform the prediction

To the Notebook!

galvanize

# Additional Info for Back Propagation (the algorithm that makes fitting possible!)

<https://www.youtube.com/watch?v=llg3gGewQ5U> Heuristic Explanation

<https://www.youtube.com/watch?v=tleHLnjs5U8> Calculus Explanation

<https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e> A clear story in one article



# Activation Layer to Change Layer Behavior

Real organic neurons have on/off behavior. A simple node has weights corresponding to the input that comes in and acts like a linear regressor with a single output per node.

Instead of each node acting as a linear regressor, we have options for what the node can do with that output. For example, logistic regression has a linear regressor looking function built into it! The standard output of a node is fed into the activation layer and turned into a logistic regression or other on-off style function.

# Types of Activation Layers (to Change Layer Output Behavior)

Linear (no special activation). In practice, not used.

“Logistic” (pseudo-binary on or off similar to fire/no-fire of a real organic neuron)

Sigmoid and Tanh functions

Rectified Linear Unit (ReLU)

Other variations of ReLU: Leaky ReLU (negatives not forced to output 0) or Noisy ReLU which adds noise

# Using Keras and Specialized NNs

(with Tensorflow under the hood)



## Afternoon Objectives:

- Build a NN on Keras
- Know how to tune structure *and* hyperparameters
  - To get convergence during training
  - Avoid under-learning and over-fitting
- Be familiar with different styles of NNs

To the Notebook!

galvanize

# Handy Keras Options

Validation Set Selection, Batch Size for Stochastic Gradient Descent (SGD) during training, and Batch Normalization “Layers” between node layers instead of L1 or L2 Regularization

<https://keras.io/models/sequential/>

Flow Images for CNNs from Labeled Directories

<https://keras.io/preprocessing/image/#imagedatagenerator-methods>

# Tips for Building a (Sequential) NN

How many hidden layers? Start with 1 or 2 to see if we get convergence and only add layers as needed. **For all neural nets, the fewer necessary hidden layers, the better to avoid overfitting and getting stuck in local minima.** I personally call NNs with too many hidden layers “conspiracy theorists”.

How many nodes per layer? The number of hidden nodes in each layer should be somewhere between the size of the input and output layer, potentially the mean. The number of hidden nodes shouldn't need to exceed twice the number of input nodes, as you are probably grossly overfitting at this point.

# Tuning a Neural Net

Regularization (functions to add as kwargs to layers)

<https://keras.io/regularizers/>

```
from keras import regularizers
model.add(Dense(64, input_dim=64,
                  kernel_regularizer=regularizers.l2(0.01),
                  activity_regularizer=regularizers.l1(0.01)))
```

Batch Normalization Layers (reduce overfitting to previous layer's strongest latent features)

<https://keras.io/layers/normalization/>

```
model.add(Dense(64, input_dim=14, init='uniform'))
model.add(BatchNormalization())
model.add(Activation('tanh'))
model.add(Dropout(0.5))
```

Optimizers as alternatives for standard SGD used in fitting. ADAM is common

<https://keras.io/optimizers/>

```
from keras import optimizers

model = Sequential()
model.add(Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(Activation('softmax'))

sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

Dropout Layers to increase redundancy/stability of NNs (as shown in notebook)

# Some often used types of Non-Basic Neural Networks

- Convolutional Neural Nets
- Recursive Neural Nets
- Word2vec
- (there are many others)

These do not follow the same “best practice” rules from two slides ago

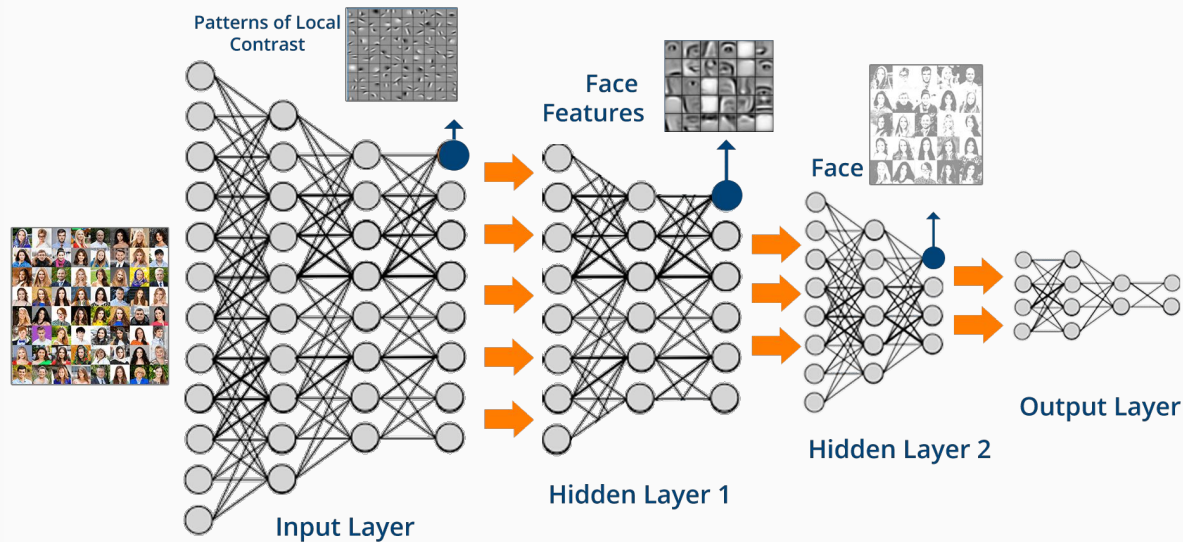


# Some other types of neural nets

Convolutional Neural Nets

Recurrent Neural Nets

word2vec



<https://cdn.edureka.co/blog/wp-content/uploads/2017/05/Deep-Neural-Network-What-is-Deep-Learning-Edureka.png>

galvanize

# Some other types of neural nets

Convolutional Neural Nets

Recurrent Neural Nets

word2vec

Goal: Scan over entire image of a specific size one small window at a time to find patterns (pixel dimensions and dimensions of color data set at the beginning)

Common Terms:

Filters - The nodes that are specifically fitted at the input layer to *detect* signal carrying patterns in the pixels within a square/rectangle of a set size of our choice.

Max Pooling - Our goal is to continually reduce down the signal from all the pixels to a set of single numerical values that represent the signal. We have to do this step by step creating smaller representations of the original image but with the relevant signal still showing! Pooling is choosing some aggregate measurement of the values from a set of pixels and feeding that result forward. "9 pixels" -> "1 pixel" Max pooling uses the maximum value of the 9 pixels.



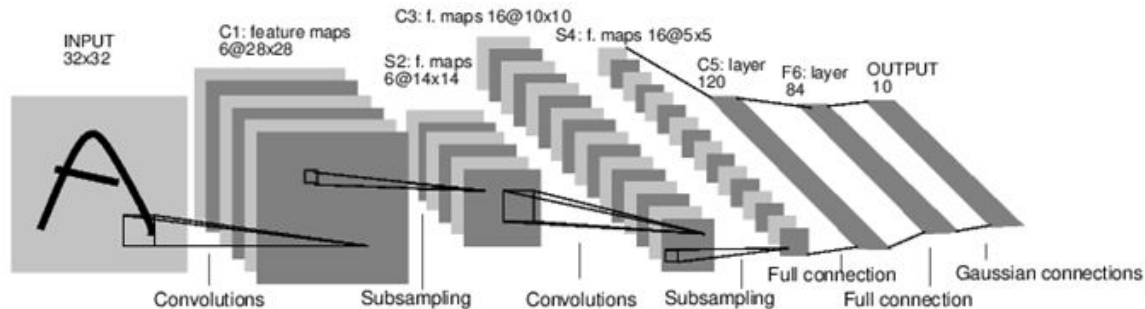
# Some other types of neural nets

## Convolutional Neural Nets

### Recurrent Neural Nets

### word2vec

Fully Connected Layer - A final flattening of all the signal captured from reducing down to the smallest representations of it.



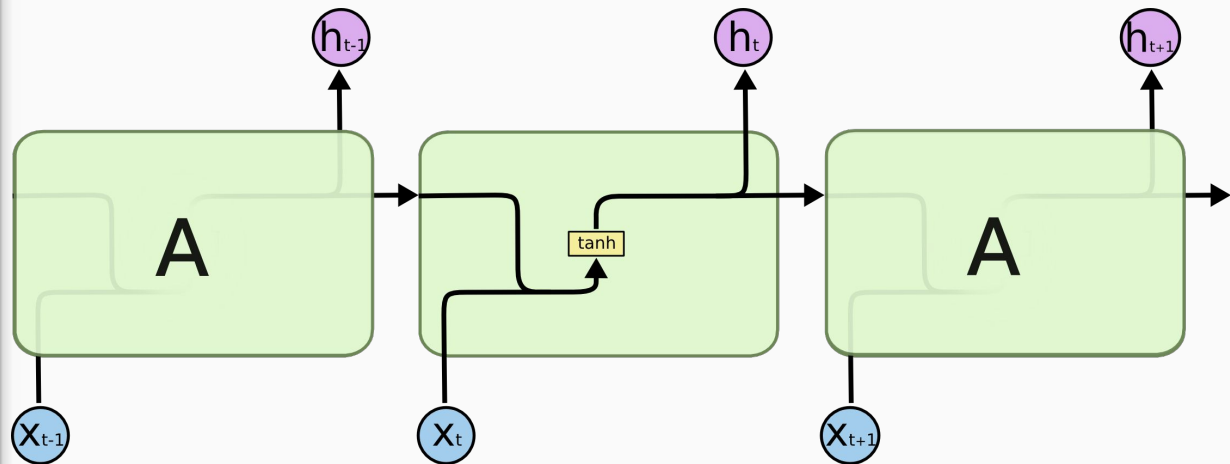
A Full Convolutional Neural Network (LeNet)

# Some other types of neural nets

Convolutional Neural Nets

**Recurrent Neural Nets**

word2vec



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

galvanize

# Some other types of neural nets

Convolutional Neural Nets

Recursive Neural Nets

word2vec

Source Text	Training Samples			
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)
The	quick	brown		
The <table><tr><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)
quick	brown	fox		
The quick <table><tr><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
brown	fox	jumps		
The quick brown <table><tr><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
fox	jumps	over		

Uses one hidden layer to process word neighbor associations. The number of nodes = dimensions the words can take up, somewhere between 100-1000 is useful.

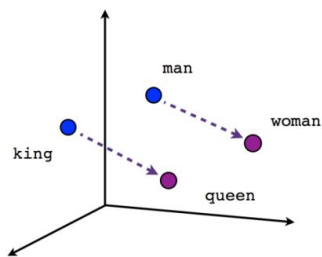


# Some other types of neural nets

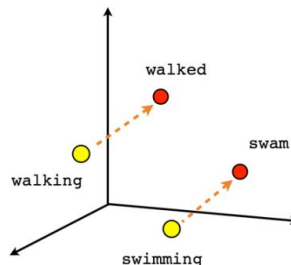
Convolutional Neural Nets

Recursive Neural Nets

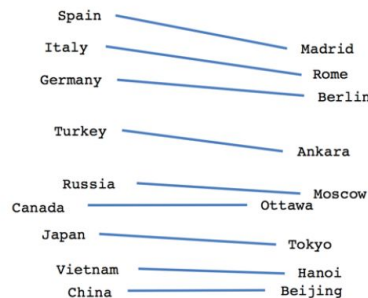
**word2vec**



Male-Female



Verb tense



Country-Capital

Uses one hidden layer to process word neighbor associations. The number of nodes = dimensions the words can take up, somewhere between 100-1000 is useful.

galvanize

Thanks for your  
Time!

