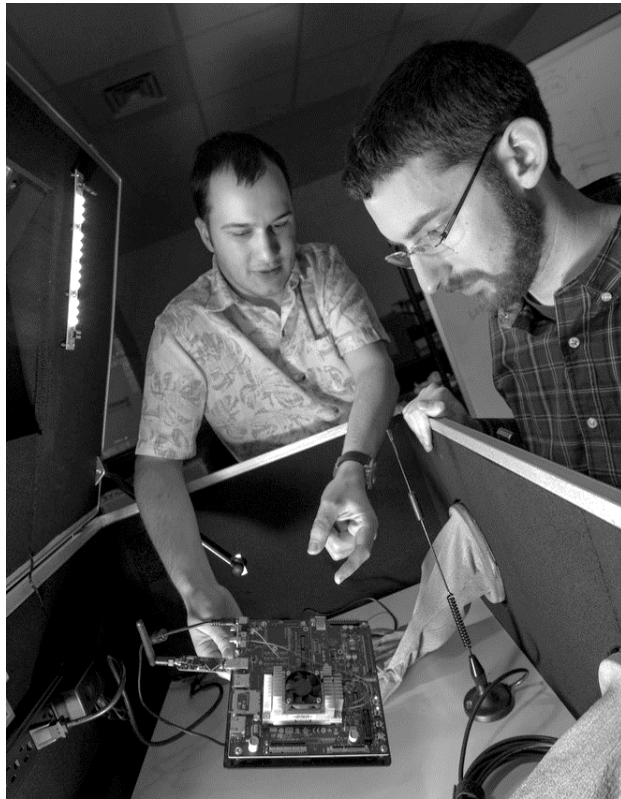




# *VIRGINIA TECH NATIONAL SECURITY INSTITUTE*



## Neural Networks: FNN, CNN, RNN, GNN

Tugba Erpek | Research Associate Professor

September 8, 2023



NATIONAL SECURITY INSTITUTE  
VIRGINIA TECH.

# Agenda

- Neural Function to Neural Networks
- Feedforward Neural Networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- Graph Neural Networks
- Hands-on Activity



# Neural Function

- Brain function (thought) occurs as the result of the firing of **neurons**.
- Neurons connect to each other through **synapses**, which propagate **action potential** (electrical impulses) by releasing **neurotransmitters**.
  - Synapses can be excitatory (potential-increasing) or inhibitory (potential-decreasing), and have varying activation thresholds.
  - Learning occurs as a result of the synapses' plasticity: They exhibit long-term changes in connection strength.
- There are about  $10^{11}$  neurons and about  $10^{14}$  synapses in the human brain!

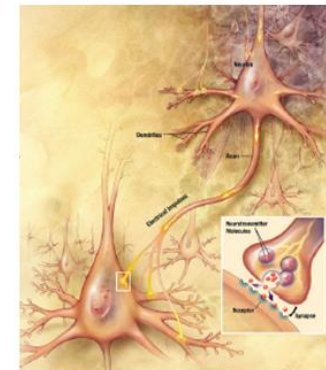
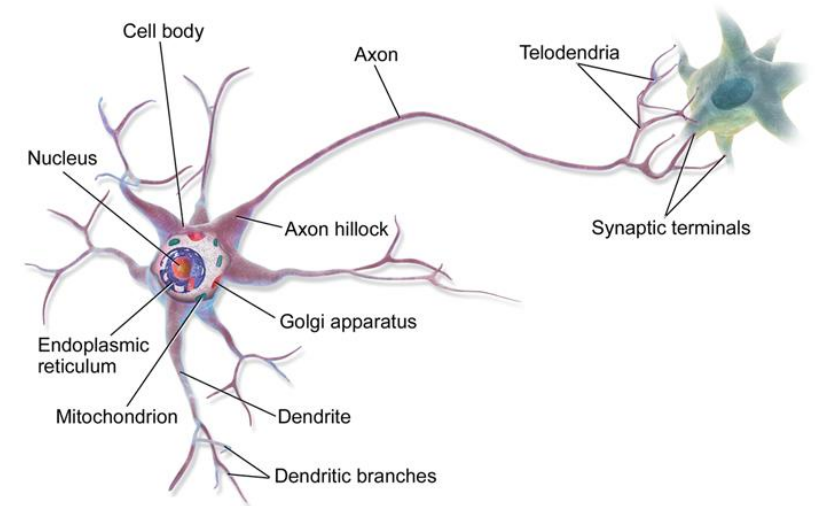
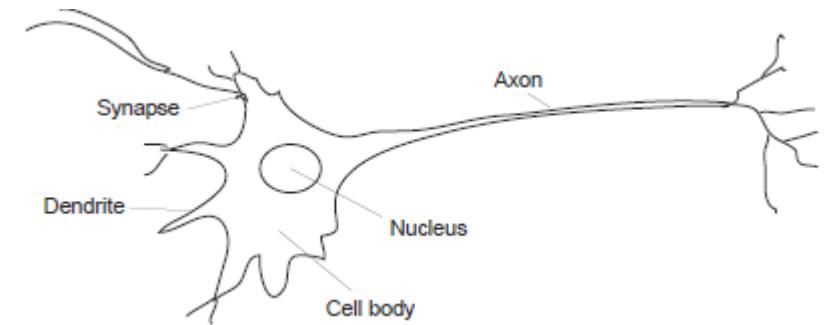


Image source: Bruce Blausen

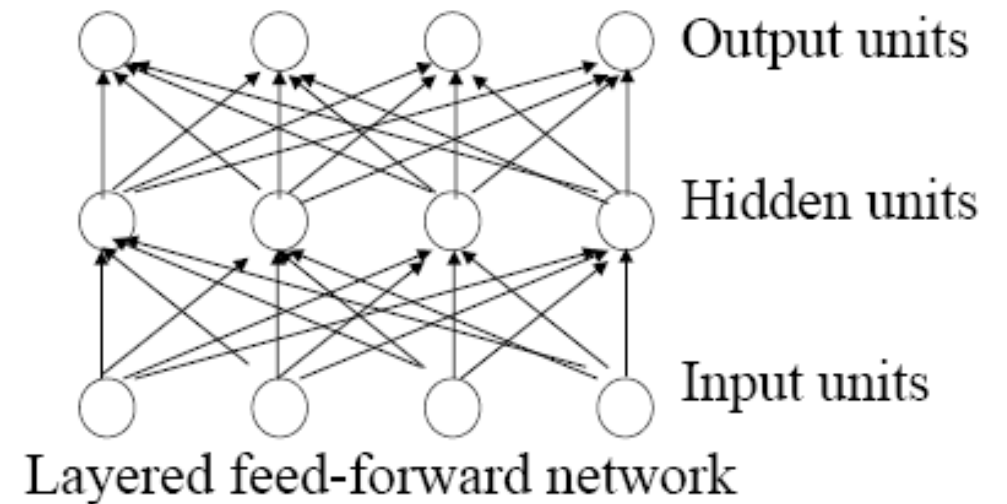
# Neural Networks - 1

- Origins: Algorithms that try to mimic the brain.
- Very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications.
- Artificial neural networks (ANNs) are not nearly as complex or intricate as the actual brain structure.



# Neural Networks - 2

- Neural networks are made up of **nodes** or **units**, connected by **links**.
- Each link has an associated **weight** and **activation level**.
- Each node has an **input function** (typically summing over weighted inputs), an **activation function**, and an **output**.





# Taxonomy of Neural Networks

- Three major categories:
  - Supervised learning
  - Unsupervised learning
  - Hybrid learning
- Supervised learning
  - Feedforward Neural Networks (FNNs)
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)
  - Graph Neural Networks (GNNs)

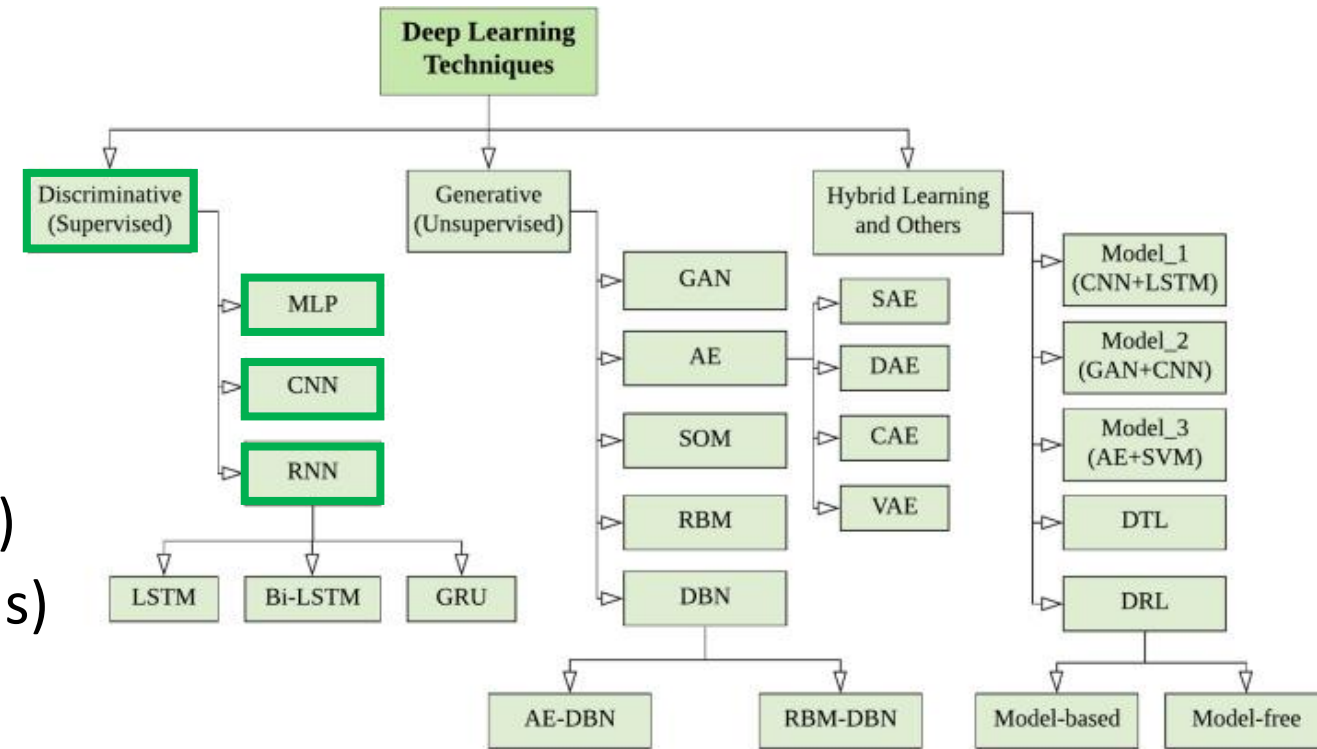


Figure from I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," SN Computer Science, 2021.

# Question

What is a supervised learning algorithm?



# Agenda

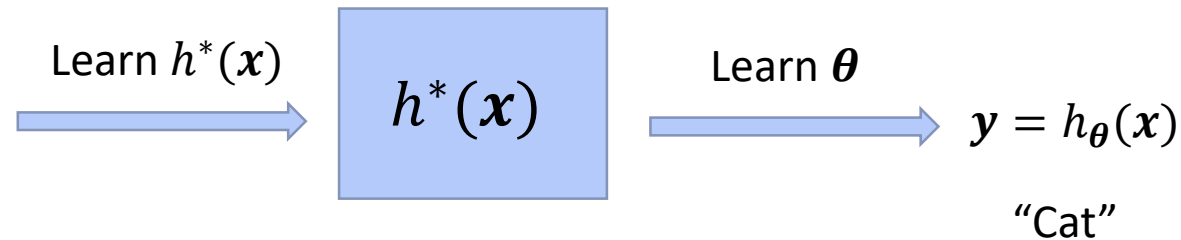
- Neural Function to Neural Networks
- **Feedforward Neural Networks**
- Convolutional Neural Networks
- Recurrent Neural Networks
- Graph Neural Networks
- Hands-on Activity





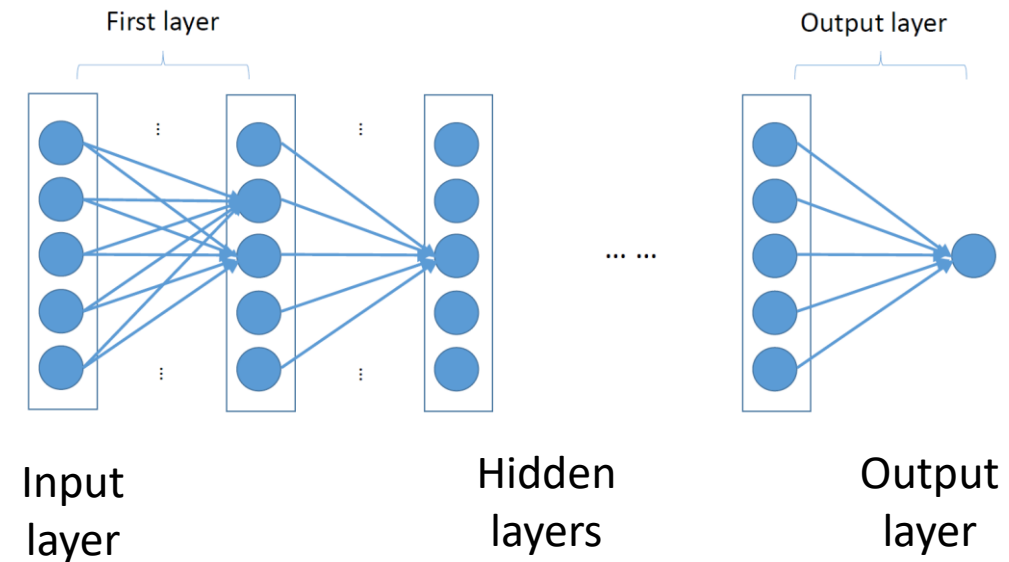
# Motivation

- Approximate some function  $h^*$  that maps an input  $x$  to an output  $y$ .
  - An FNN defines a mapping  $y = h_{\theta}(x)$  and learns the value of the parameters  $\theta$  that result in the best function approximation.



# Feedforward Neural Networks (FNNs)

- FNN architecture
  - Number of layers
  - Number of units/neurons
  - Activation functions
- Universal Approximator Theorem: One hidden layer is enough to represent (not learn) an approximation of any function to an arbitrary degree of accuracy
- So why deeper?
  - Shallow net may need (exponentially) more width
  - Shallow net may overfit more



# Input

- Represented as a vector
- May require some pre-processing
  - Subtract mean
  - Normalize to  $[-1,1]$



Image represented  
as a vector



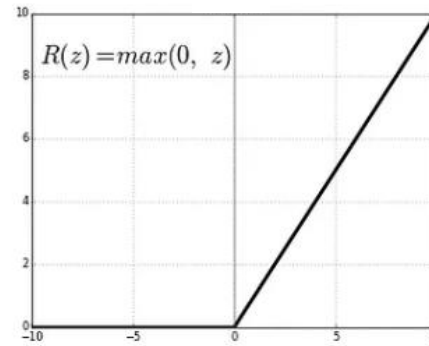
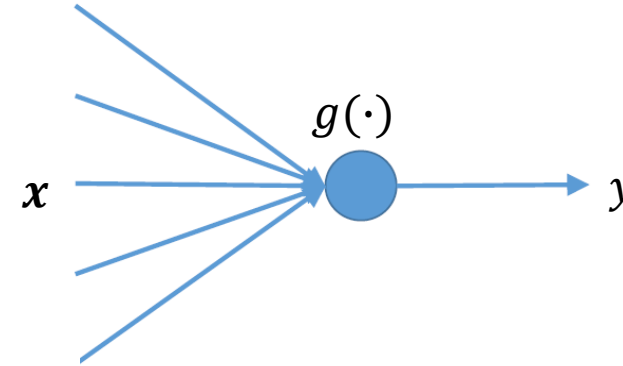
# Question

- What is the input size of an FNN?
  - 1D vector
  - 2D matrix
  - 3D matrix

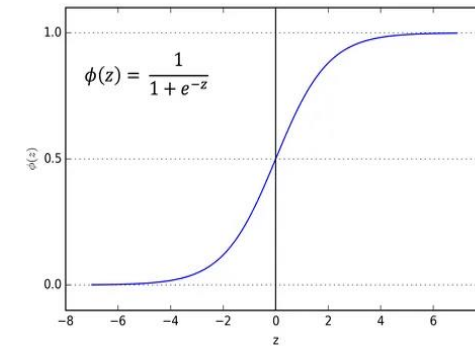


# Hidden Layers

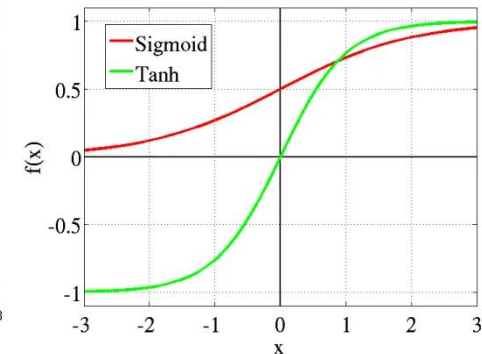
- $y = g(\theta^T x)$
- Activation function,  $g(\cdot)$ , can introduce nonlinearity.
- Typical activation functions:
  - Step
  - ReLU
  - Sigmoid
  - Softmax
  - Tanh
- Sigmoid and tanh functions saturate.
- Use ReLU activation 90% of the time.
- Use softmax in the last layer for a classification problem.



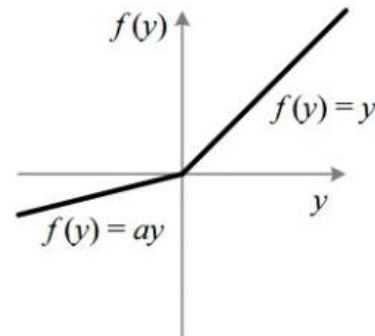
ReLU



Sigmoid

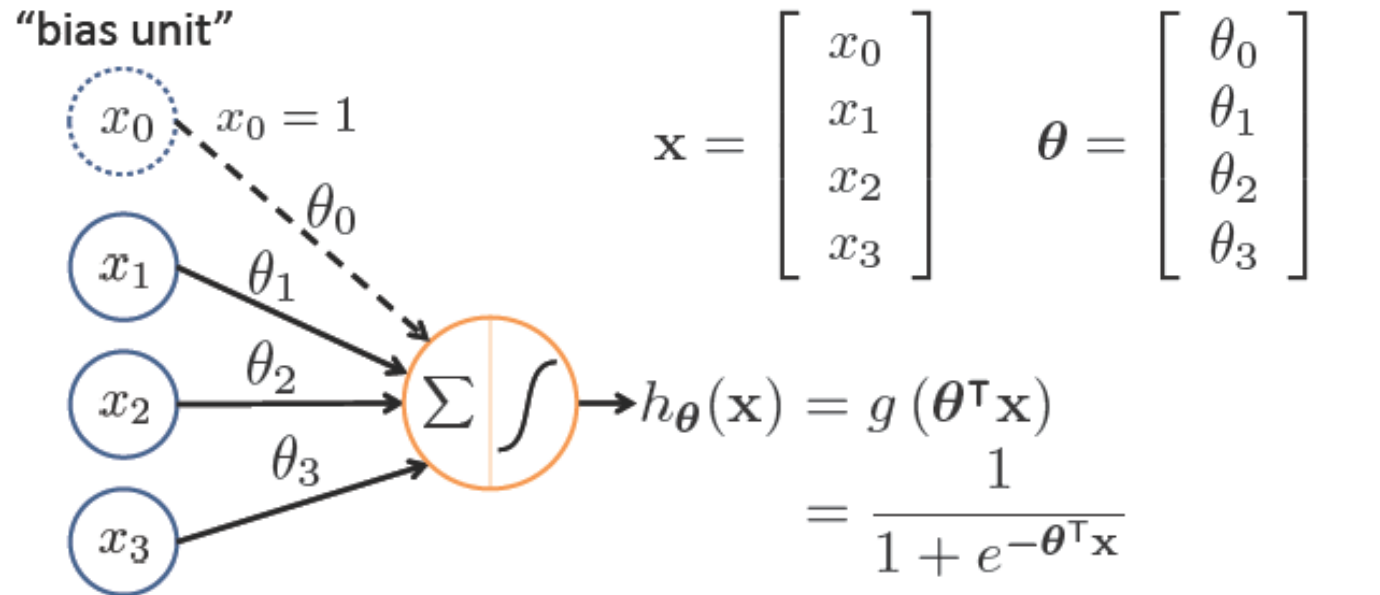


Tanh

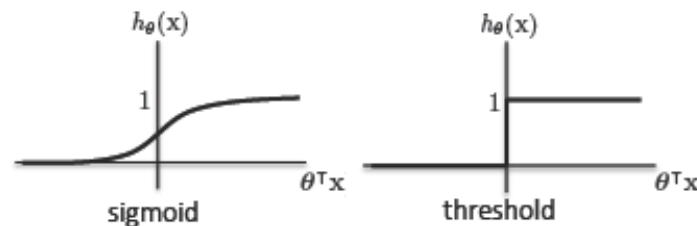


Leaky ReLU

# Sigmoid Activation Function



Sigmoid (logistic) activation function:  $g(z) = \frac{1}{1 + e^{-z}}$



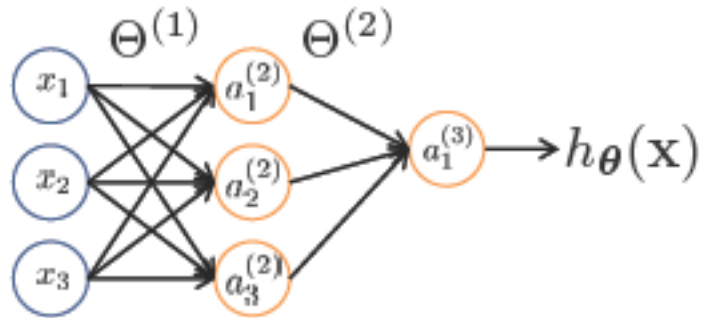
# Question

- What is the most suitable activation for the output of a classification problem?
  - Softmax
  - ReLU
  - Tanh





# Feed-Forward Process



$a_i^{(j)}$  = “activation” of unit  $i$  in layer  $j$

$\Theta^{(j)}$  = weight matrix controlling function mapping from layer  $j$  to layer  $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

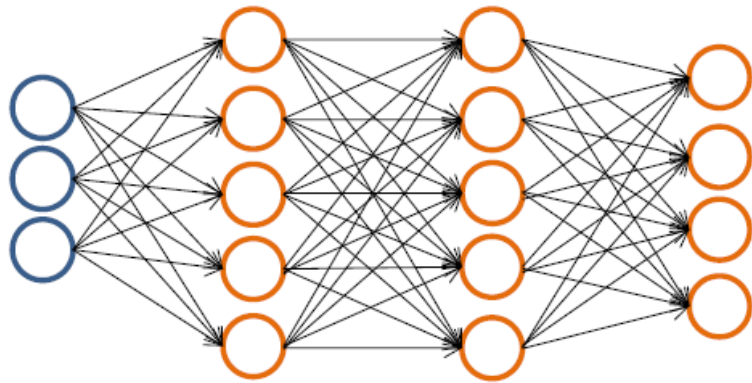
$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has  $s_j$  units in layer  $j$  and  $s_{j+1}$  units in layer  $j+1$ ,  
then  $\Theta^{(j)}$  has dimension  $s_{j+1} \times (s_j + 1)$ .

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4} \quad \Theta^{(2)} \in \mathbb{R}^{1 \times 4}$$

# Classification at the Output



## Binary classification

$y = 0$  or  $1$

1 output unit

## **Given:**

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\}$$

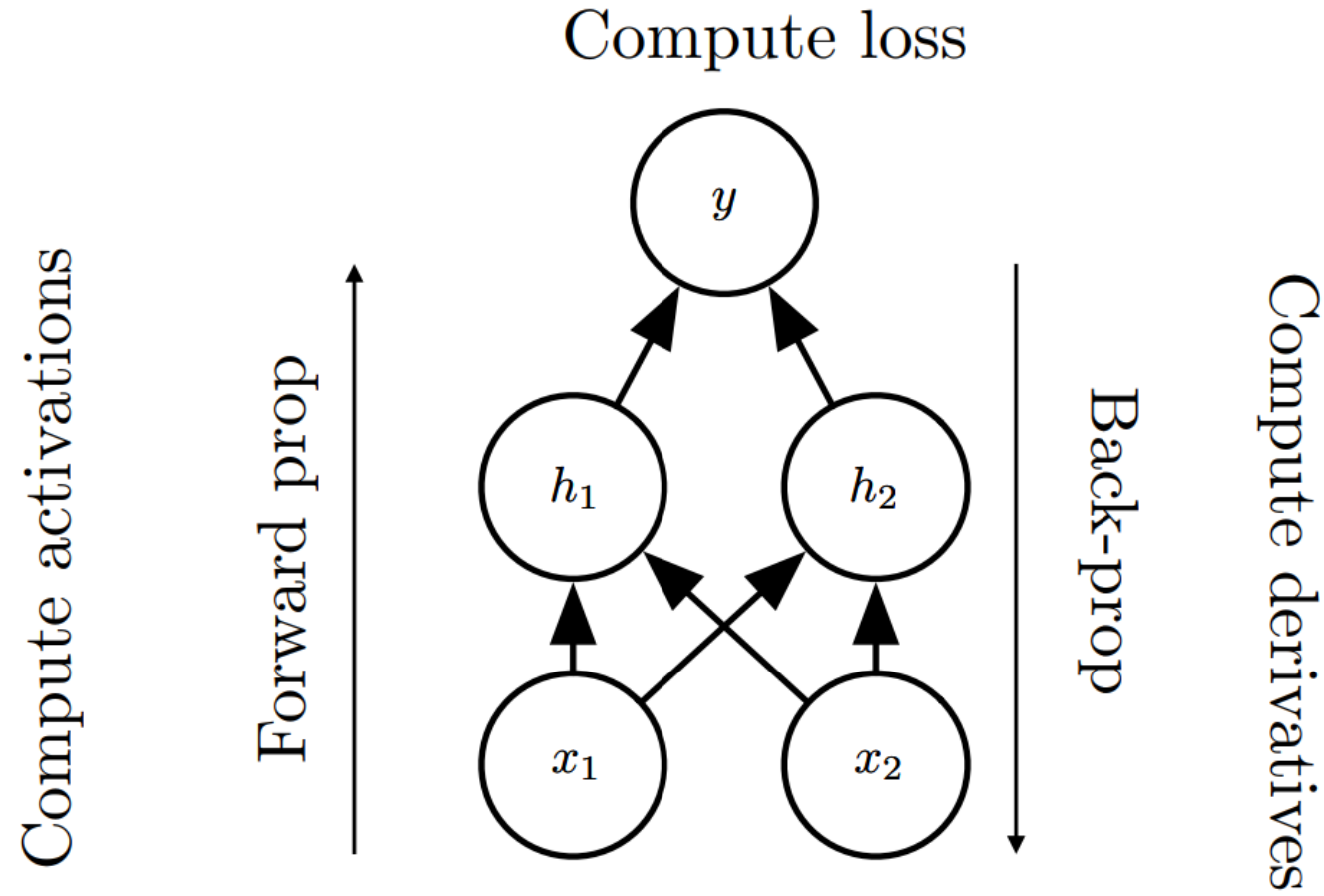
## Multi-class classification ( $K$ classes)

$\mathbf{y} \in \mathbb{R}^K$  e.g.  $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ ,  $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$   
pedestrian car motorcycle truck

$K$  output units



# Training Process



# Loss Functions

- **Loss Function:** The function used to evaluate a candidate solution.
- With neural networks, we seek to minimize the error between the estimated output and the true output.
- Regression Problem: Predict a real-value quantity
  - **Output Layer Configuration:** One node with a linear activation unit.
  - **Loss function:** Mean Squared Error.
- Binary Classification Problem: Classify an example as belonging to one of two classes
  - **Output Layer Configuration:** One node with a sigmoid activation unit.
  - **Loss Function:** Cross-Entropy, also referred to as Logarithmic loss.
- Multi-Class Classification Problem: Classify an example as belonging to one of more than two classes.
  - **Output Layer Configuration:** One node for each class using the softmax activation function.
  - **Loss Function:** Cross-Entropy, also referred to as Logarithmic loss.



# Back-Propagation Algorithm

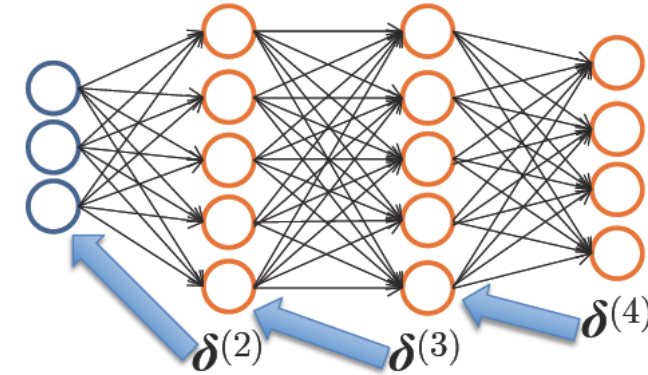
- Back-propagation is “just the chain rule” of calculus

$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx} \qquad \nabla_{\mathbf{x}} z = \left( \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)^{\top} \nabla_{\mathbf{y}} z$$

- But it’s a particular implementation of the chain rule
  - Uses dynamic programming
  - Avoids recomputing repeated subexpressions
  - Speed vs. memory tradeoff
- Calculate the error using the loss function
  - If the output of the network is correct, no changes are made
  - If there is an error, weights are adjusted using the back-propagation algorithm to reduce the error

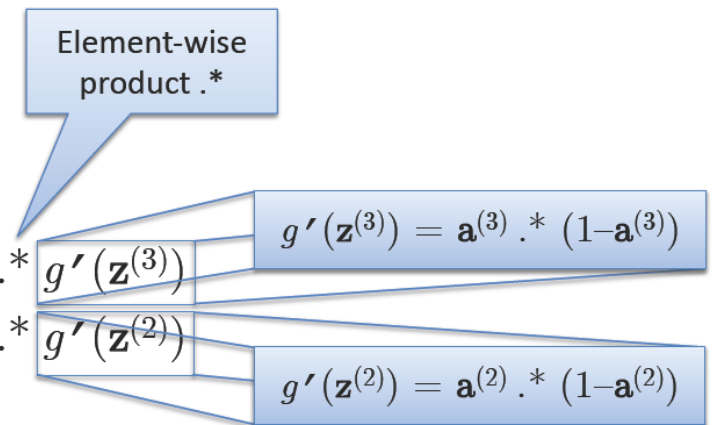
# Back-propagation for Neural Networks

- Each hidden node  $j$  is responsible for some fraction of the error  $\delta_j^{(l)}$  in each of the output nodes to which it connects
- Neural network parameters are updated with back-propagation to minimize the error.
- Let  $\delta_j^{(l)}$  = “error” of node  $j$  in layer  $l$  (#layers  $L = 4$ )



## Backpropagation

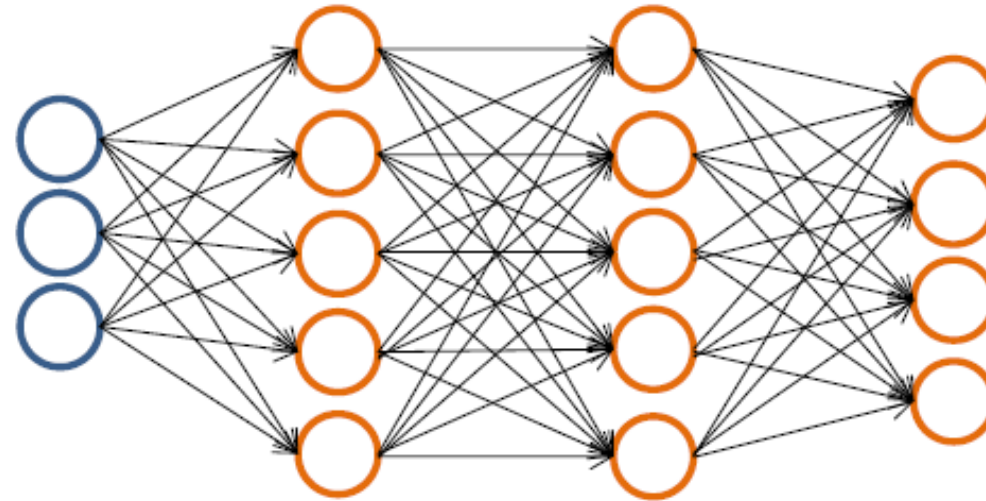
- $\delta^{(4)} = \mathbf{a}^{(4)} - \mathbf{y}$
- $\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(\mathbf{z}^{(3)})$
- $\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(\mathbf{z}^{(2)})$
- (No  $\delta^{(1)}$ )



$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)} \quad (\text{ignoring } \lambda; \text{ if } \lambda = 0)$$

# Neural Network Parameters

- Pick a network architecture
- Parameters to consider
  - Input vector size
  - Number of output classes
  - Number of layers
  - Number of neurons
  - Activation functions





# Training Steps

1. Initialize weights
  - Different methods available
2. Implement forward propagation to get  $h^{\Theta}(x_i)$  for any instance  $x_i$
3. Implement code to compute cost function  $J(\Theta)$
4. Implement backprop to compute partial derivatives
5. Use gradient descent with backprop to fit the network



# Demonstration

- Iris Dataset Classification

- Sepal Length
- Sepal Width
- Petal Length
- Petal Width



- Setosa [1 0 0]
- Versicolor [0 1 0]
- Virginica [0 0 1]

The demo files can be found in the 'FNN Example' folder on Teams

# Agenda

- Neural Function to Neural Networks
- Feedforward Neural Networks
- **Convolutional Neural Networks**
- Recurrent Neural Networks
- Graph Neural Networks
- Hands-on Activity



# Motivation

- Scale up neural networks to process very large images / video.
  - A 1000x1000 image will represent 3 million feature/input to the full connected neural network.
  - If the following hidden layer contains 1000, then we will want to learn weights of the shape [1000, 3 million] which is 3 billion parameter only in the first layer!
  - One of the solutions is to build this using convolution layers instead of the fully connected layers.
    - Parameter sharing
    - Sparsity of connections
- Automatically generalize across spatial translations of inputs.
- Applicable to any input laid out on a grid (1D, 2D, 3D, ...)



# Convolutional Neural Networks

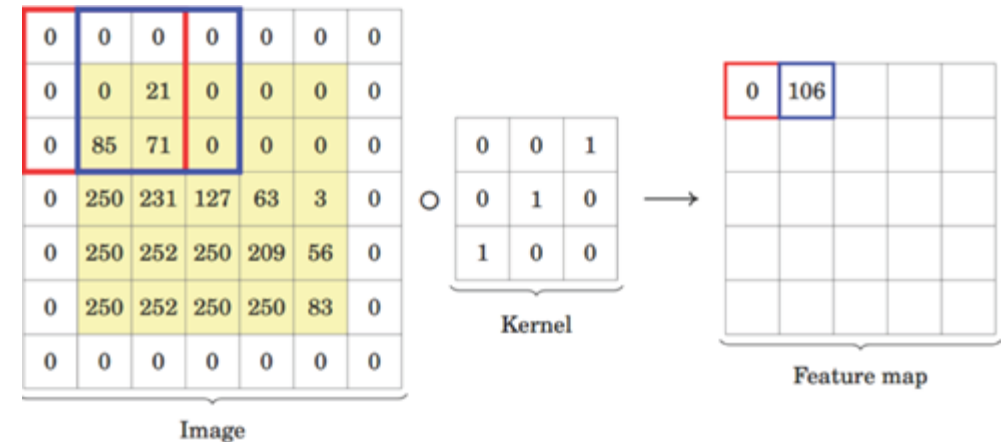
- Neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- Classification, retrieval, detection, segmentation,...
- Layers structured as matrices instead of vectors.
- Convolution of matrices by small (e.g. 3 x 3) matrices called **kernels**.
  - The kernels are the learned weights.
  - Its hyperparameters to be tuned: the filter size and stride.
- Intuitively, convolution layers learn how to relate neighboring pixels.
- The resulting output called **feature map** or **activation map**.



*A white teddy bear sitting in the grass*



*A man riding a wave on top of a surfboard*



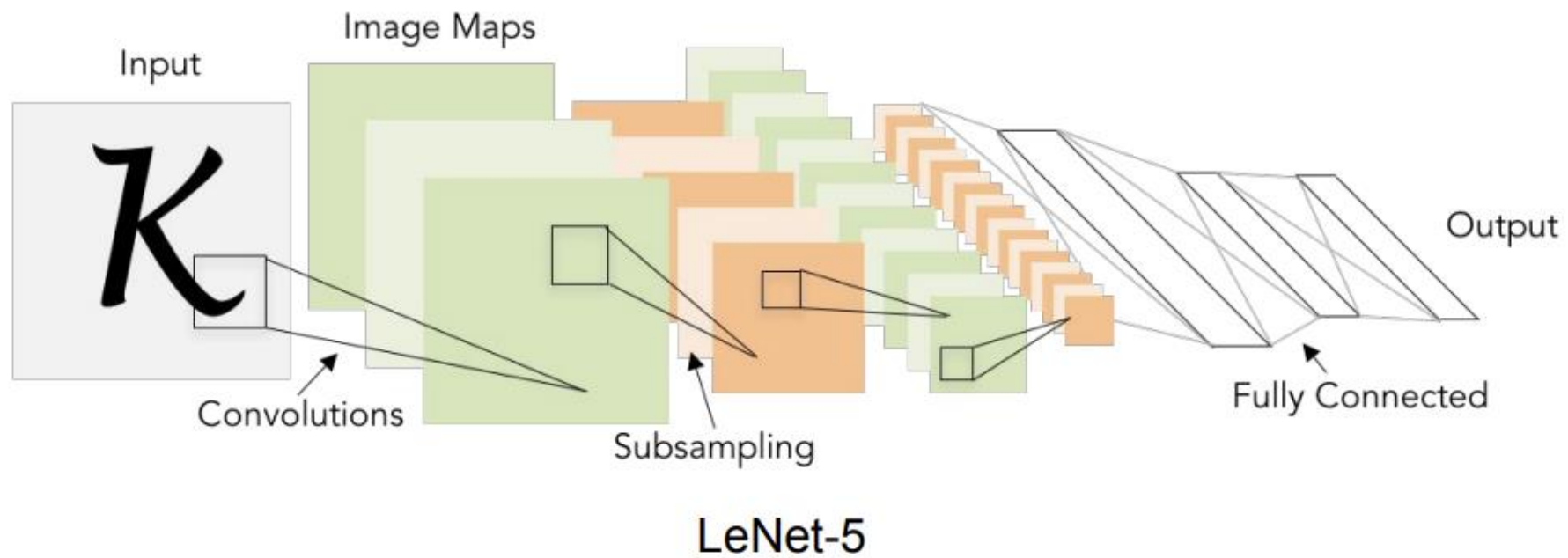
# Question

- CNNs are mostly used for
  - Image recognition and classification
  - Object detection
  - Face recognition
  - All of them



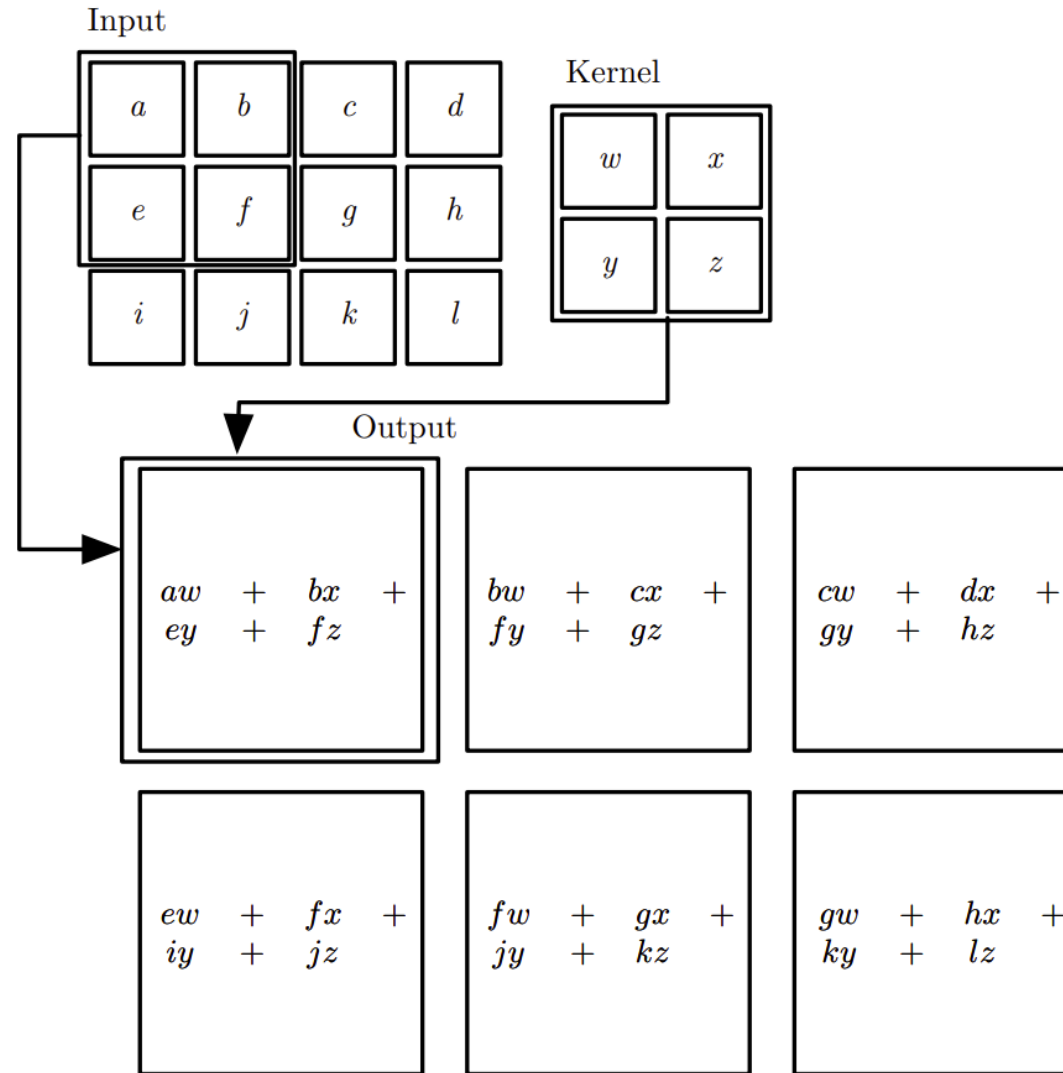
# Convolutional Network Architecture

- Typically three layers: (i) a convolutional layer, (ii) a pooling layer, (iii) a fully connected layer.



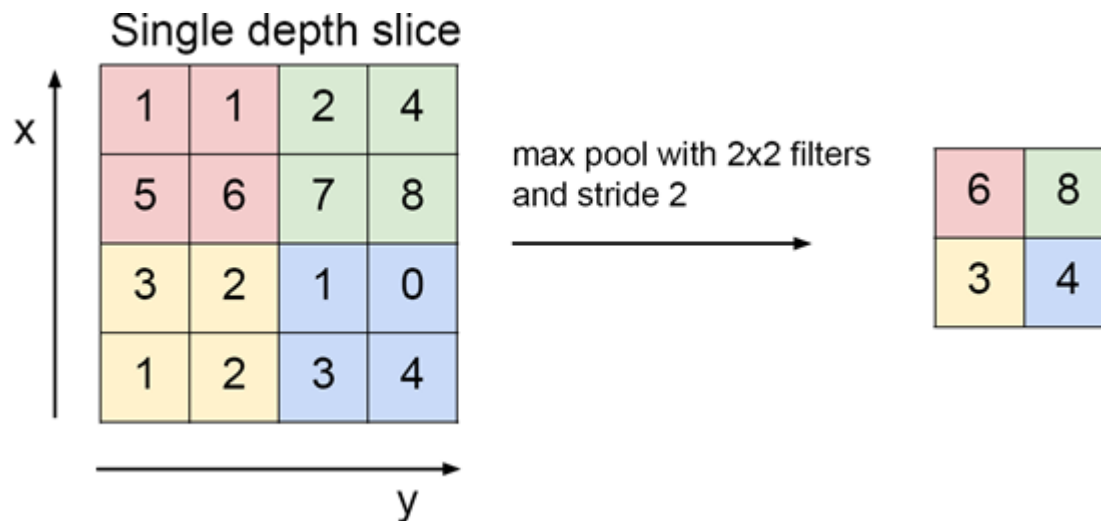


# 2D Convolution



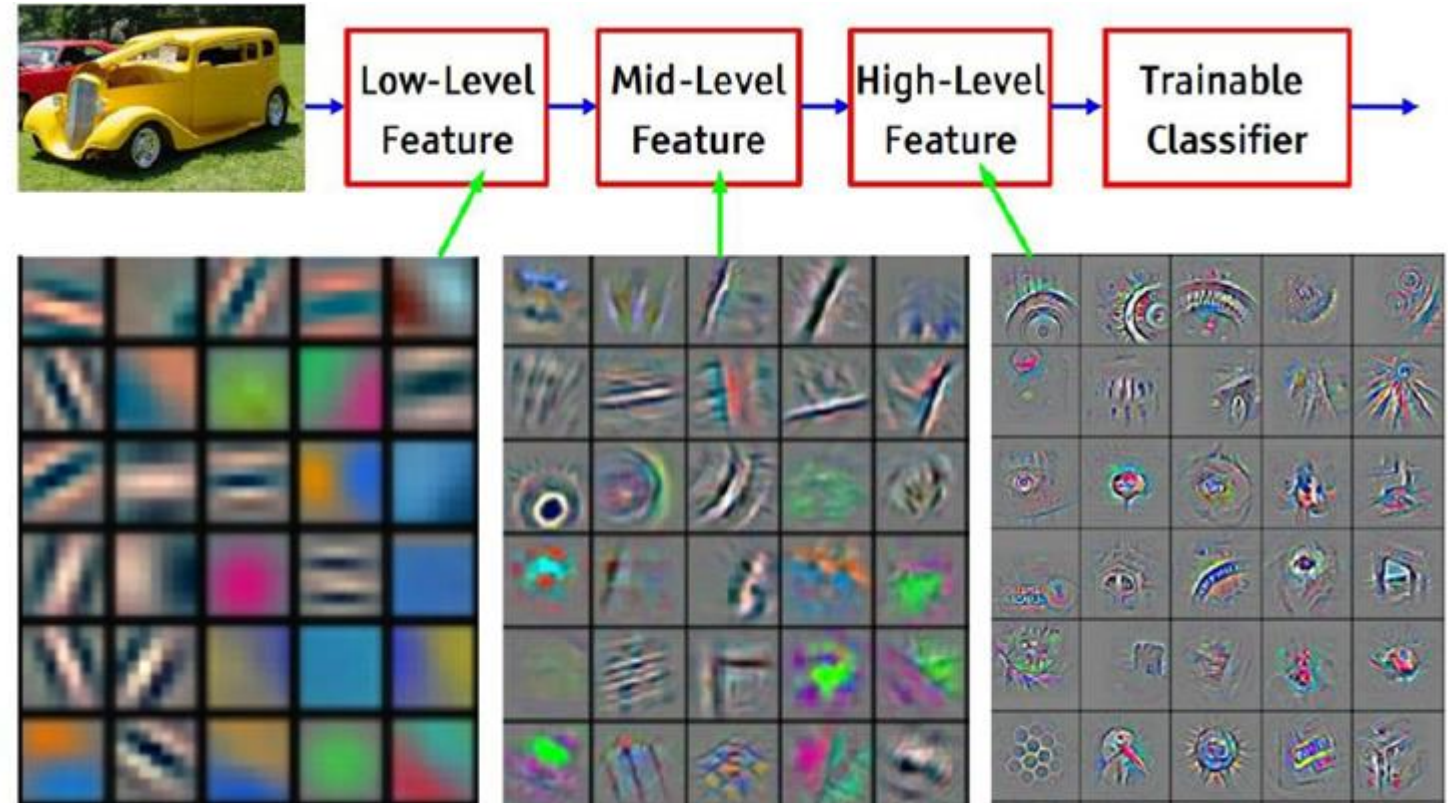
# Pooling

- Downsampling operation, typically applied after a convolution layer, which does some spatial invariance.
  - Max and average pooling are special kinds of pooling where the maximum and average value is taken, respectively.
  - Intuitively, Max pooling is used because nearby pixels will give very similar output.



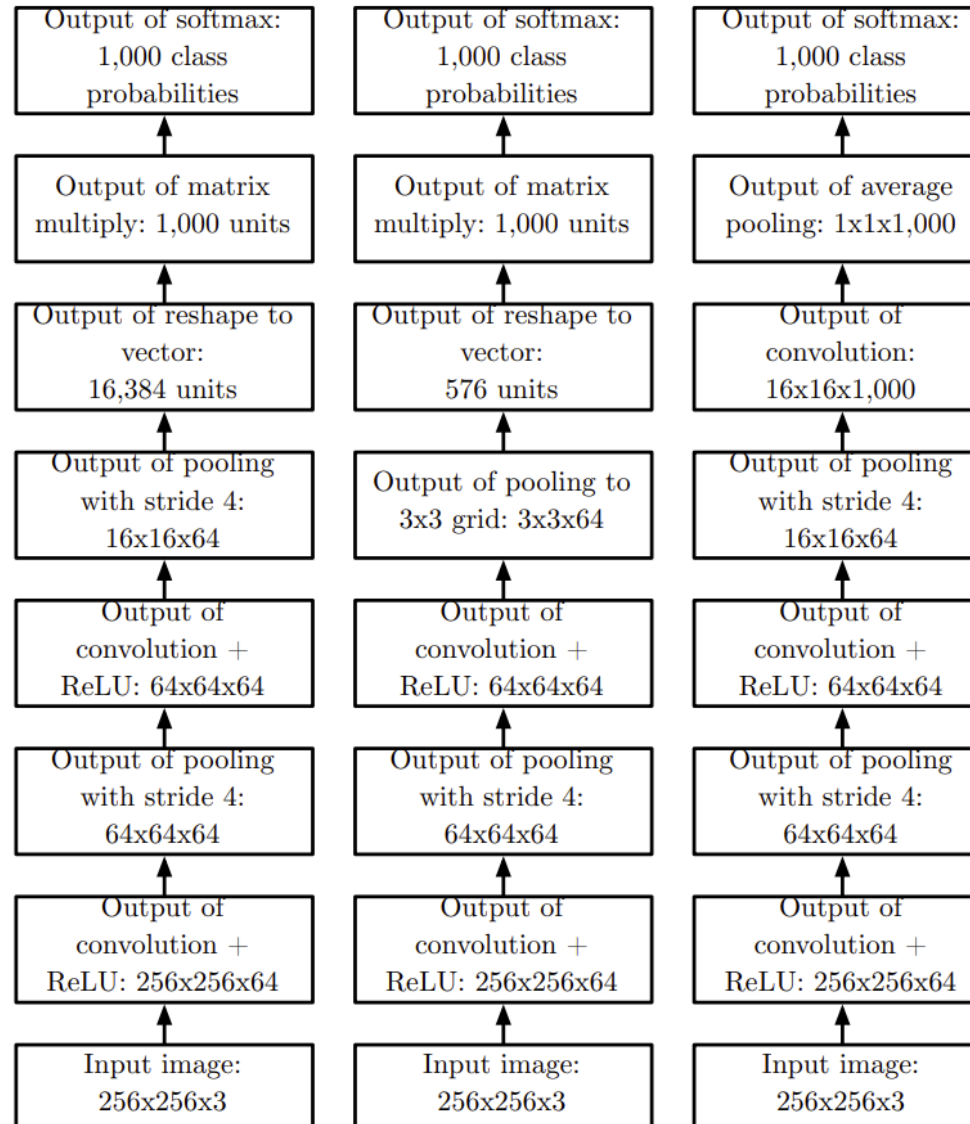
# What CNNs Learn

- Early layers of CNN might detect edges then the middle layers will detect parts of objects and the later layers will put the these parts together to produce an output.

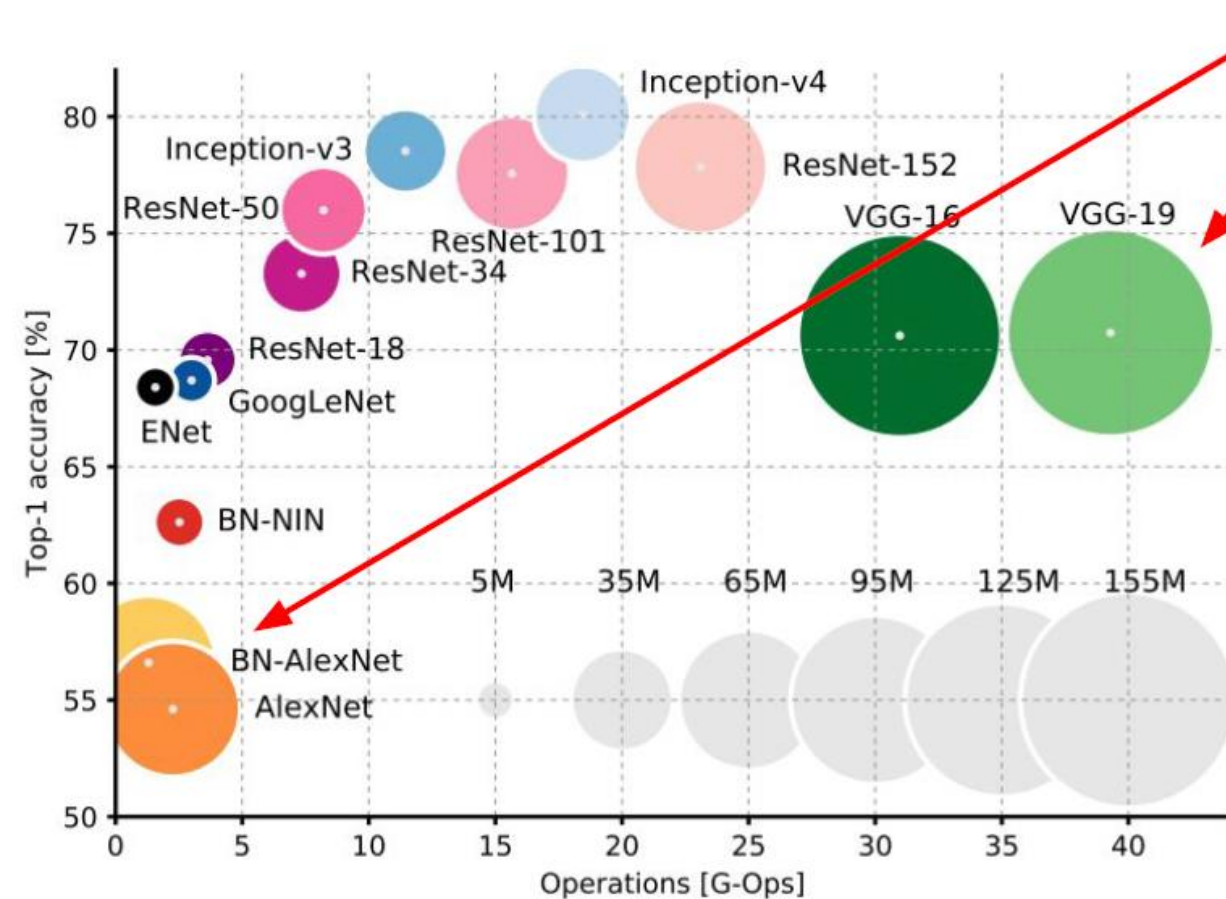


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# Example Classification Architectures



# CNN Architectures



AlexNet and VGG have tons of parameters in the fully connected layers

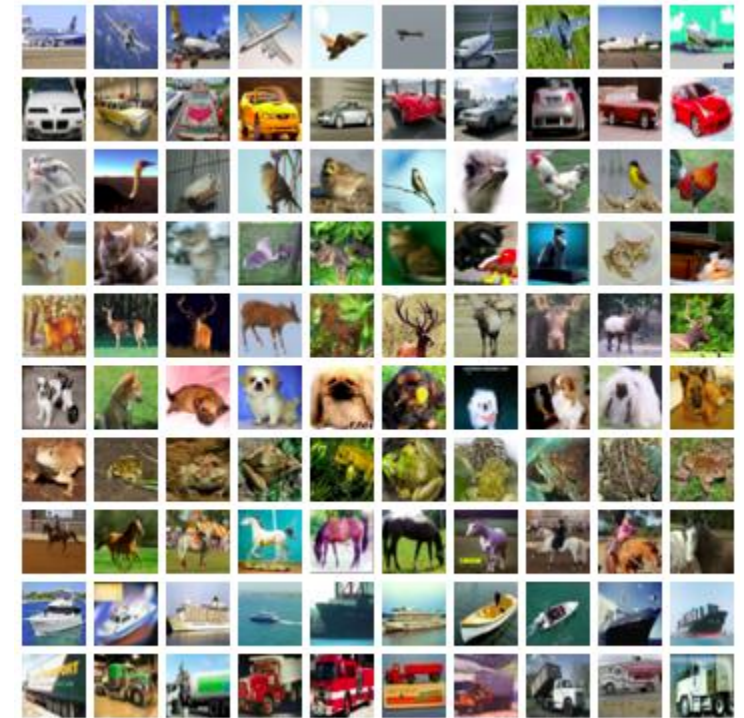
**AlexNet: ~62M parameters**

FC6: 256x6x6 -> 4096: 38M params  
FC7: 4096 -> 4096: 17M params  
FC8: 4096 -> 1000: 4M params  
~59M params in FC layers!



# Demonstration

- CIFAR-10 Classification
  - Established computer-vision dataset used for object recognition.
  - Subset of the 80 million tiny images dataset.
  - Consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class.
- ConvNetJS CIFAR-10 demo (stanford.edu)



The demo files can be found in the 'CNN Example' folder on Teams

# Agenda

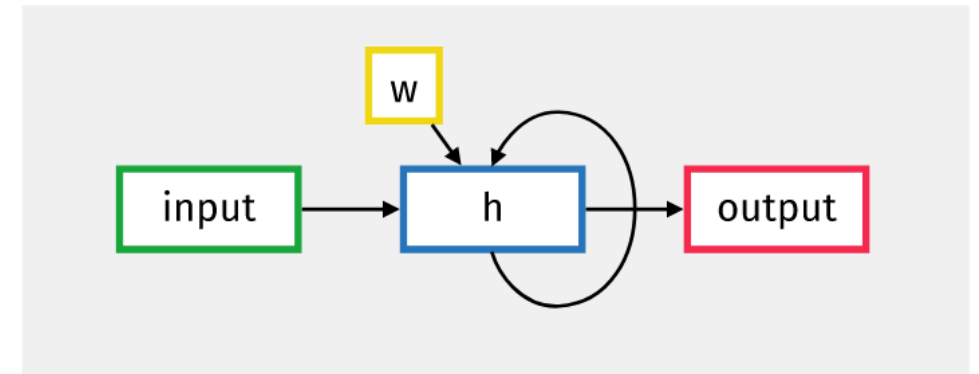
- Neural Function to Neural Networks
- Feedforward Neural Networks
- Convolutional Neural Networks
- **Recurrent Neural Networks**
- Graph Neural Networks
- Hands-on Activity





# Recurrent Neural Networks

- Dates back to (Rumelhart et al., 1986)
- A family of neural networks for handling sequential data, which involves variable length inputs or outputs
  - Audio, video or text
- Sequential data can be found in any time series such as audio signal, stock market prices, vehicle trajectory and natural language processing.



# Sequential Data

- Each data point: A sequence of vectors  $x(t)$ , for  $1 \leq t \leq \tau$
- Batch data: many sequences with different lengths  $\tau$
- Label: can be a scalar, a vector, or even a sequence



# Question

- Which deep learning algorithm is used for sequential problems?
  - FNN
  - CNN
  - RNN



# Sequence Applications: One-to-Many

- **Input:** fixed-size
- **Output:** sequence
- E.g., image captioning



Features:	
Feature Name	Value
Description	{ "type": 0, "captions": [ { "text": "a man swimming in a pool of water", "confidence": 0.7850108693093019 } ] }
Tags	[ { "name": "water", "confidence": 0.9996442794799805 }, { "name": "sport", "confidence": 0.9504992365837097 }, { "name": "swimming", "confidence": 0.9062816288803101, "hint": "sport" }, { "name": "pool", "confidence": 0.8787588477134705 }, { "name": "water sport", "confidence": 0.631849467754364, "hint": "sport" } ]
Image Format	jpeg
Image Dimensions	1500 x 1155
Clip Art Type	0 Non-clipart
Line Drawing Type	0 Non-LineDrawing
Black & White Image	False

Captions: <https://www.microsoft.com/cognitive-services/en-us/computer-vision-api>

# Sequence Applications: Many-to-One


- **Input:** Sequence
- **Output:** Fixed-Size
- E.g., sentiment analysis (hate? Love?, etc.)

**CRITIC REVIEWS FOR *STAR WARS: THE LAST JEDI***

All Critics (371) | Top Critics (51) | Fresh (336) | Rotten (35)

 What's most interesting to me about The Last Jedi is Luke's return as the mentor rather than the student, grappling with his failure in this new role, and later aspiring to be the wise and patient teacher.

December 26, 2017 | Rating: 3/4 | [Full Review...](#)

 **Leah Pickett**  
Chicago Reader  
★ Top Critic

 Fanatics will love it; for the rest of us, it's a tolerably good time.

December 15, 2017 | Rating: B | [Full Review...](#)

 **Peter Rainer**  
Christian Science Monitor  
★ Top Critic

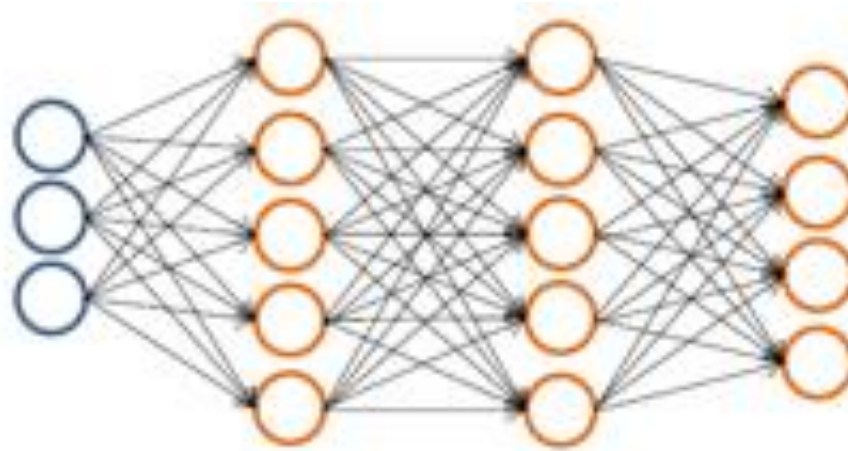
[https://www.rottentomatoes.com/m/star\\_wars\\_the\\_last\\_jedi](https://www.rottentomatoes.com/m/star_wars_the_last_jedi)

# Sequence Applications: Many-to-Many

- **Input:** Sequence
- **Output:** Sequence
- E.g., language translation



# Recall: Feedforward Neural Networks



- **Problem:** many model parameters
- **Problem:** No memory of past since weights are learned independently

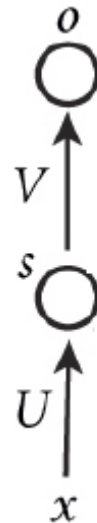
Each layer serves as input to the next layer with no loops

# Recurrent Neural Networks

- Main idea: Use hidden state to capture information about the past.

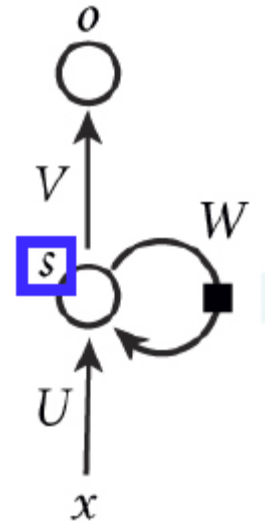
## Feedforward Network

Each layer receives input from the previous layer with no loops



## Recurrent Network

Each layer receives input from the previous layer and the output from the previous time step



- Use **the same** computational function and parameters across different time steps of the sequence
- Each time step: takes the input entry and **the previous hidden state** to compute the output entry
- Loss: typically computed at every time step



# Advantages

- Hidden state: a lossy summary of the past
- Shared functions and parameters: greatly reduce the capacity and good for **generalization** in learning
- Explicitly use the **prior knowledge** that the sequential data can be processed by in the same way at different time step (e.g., NLP)
- Yet still powerful (actually **universal**): any function computable by a Turing machine can be computed by such a recurrent network of a finite size (see, e.g., Siegelmann and Sontag [1995])



# RNN Variants

- RNNs allow a lot of flexibility in architecture design.
- Vanilla RNNs are simple but don't work very well.
- Common to use Long Short Term Memory (LSTM) or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish.
- Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM).
- Better/simpler architectures are a hot topic of current research.



# Demonstration

- Train an RNN model for time series prediction
  - Sunspots data per month
  - Train on 80% of the data
  - Test on 20% of the data

The demo files can be found in the 'RNN Example' folder on Teams



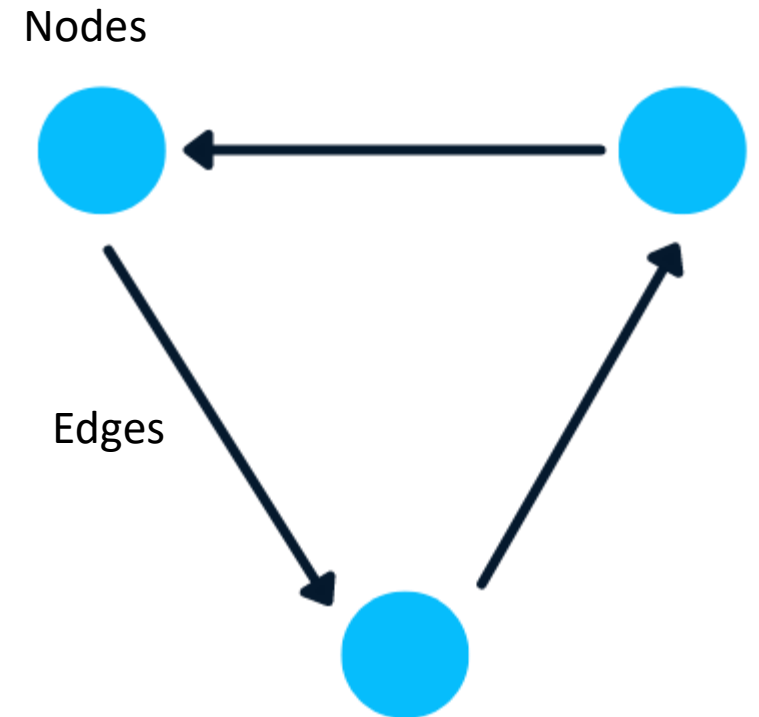
# Agenda

- Neural Function to Neural Networks
- Feedforward Neural Networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- **Graph Neural Networks**
- Hands-on Activity



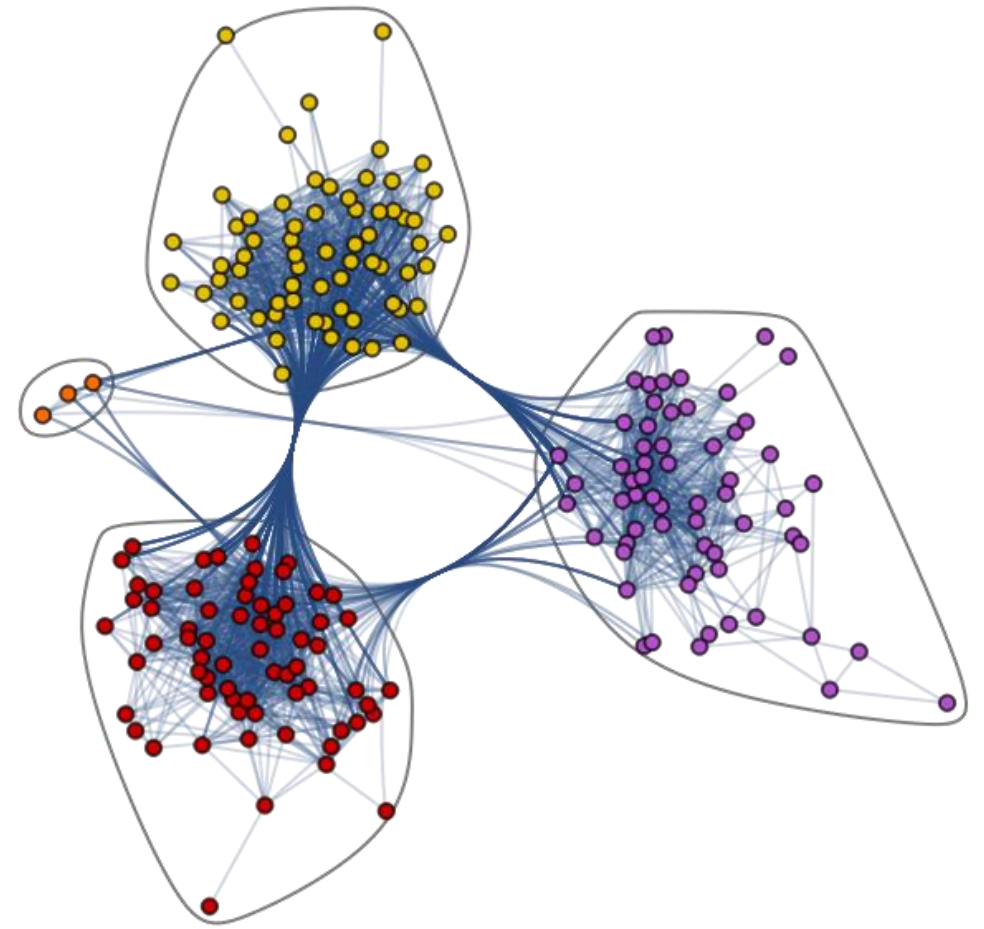
# What is a Graph?

- The type of data structure that contains nodes and edges.
  - A node can be a person, place, or thing.
  - The edges define the relationship between nodes.
- Graphs are used in
  - Pattern recognition
  - Social networks analysis
  - Recommendation systems
  - Semantic analysis



# Graph Example

- Community Graph Plot by Jazz Musicians Network
  - 198 nodes and 2742 edges.
  - Different colors of nodes represent various communities of Jazz musicians and the edges connecting them.
- Used to deal with complex problems with relationships and interactions.



# Graph Challenges

- A graph exists in non-euclidean space.
  - Does not exist in 2D or 3D space, which makes it harder to interpret the data.
  - To visualize in 2D space, must use various dimensionality reduction tools.
- Graphs are dynamic; they do not have a fixed form.
  - Two visually different graphs might have similar adjacency matrix representations.
  - Makes it difficult for us to analyze data using traditional statistical tools.
- Large size and dimensionality will increase the graph's complexity for human interpretations.
  - The dense structure with multiple nodes and thousands of edges is harder to understand and extract insights.



# Graph Neural Networks

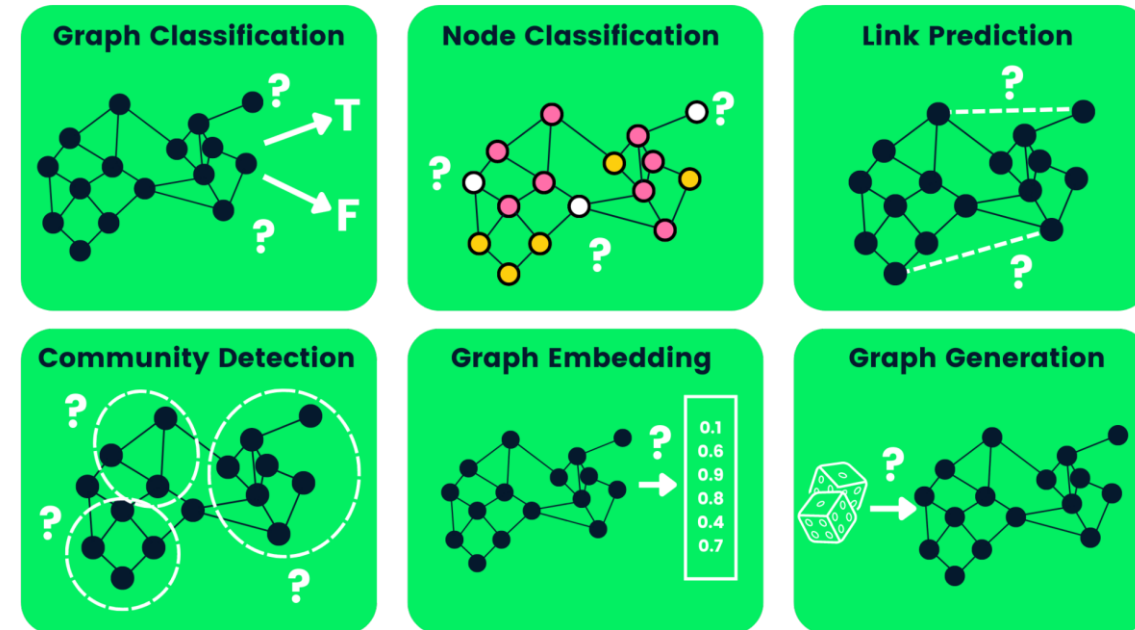
- GNNs are special types of neural networks capable of working with a graph data structure.
  - Highly influenced by CNNs and graph embedding.
  - GNNs are used in predicting nodes, edges, and graph-based tasks.
- CNNs are used for image classification. Similarly, GNNs are applied to graph structure (grid of pixels) to predict a class.
- RNNs are used in text classification. Similarly, GNNs are applied to graph structures where every word is a node in a sentence.
- GNNs were introduced when CNNs failed to achieve optimal results due to the arbitrary size of the graph and complex structure.





# Example GNN Tasks

- **Graph Classification:** Used to classify graphs into various categories.
  - Applications: social network analysis, text classification.
- **Node Classification:** Uses neighboring node labels to predict missing node labels in a graph.
- **Link Prediction:** Predicts the link between a pair of nodes in a graph with an incomplete adjacency matrix.
  - Commonly used for social networks.
- **Community Detection:** Divides nodes into various clusters based on edge structure.
  - Learns from edge weights, and distance and graph objects similarly.
- **Graph Embedding:** Maps graphs into vectors, preserving the relevant information on nodes, edges, and structure.
- **Graph Generation:** learns from sample graph distribution to generate a new but similar graph structure.



# Agenda

- Neural Function to Neural Networks
- Feedforward Neural Networks
- Convolutional Neural Networks
- Recurrent Neural Networks
- Graph Neural Networks
- Hands-on Activity



# Hands-On Activity

1. Use the MNIST dataset, database of handwritten digits (0-9) consisting of 70,000 data points (7,000 examples per digit).
  - Each data point is represented by a 784-d vector, corresponding to the (flattened) 28×28 images in the MNIST dataset.
  - **Goal:** Train an FNN (using Keras) to obtain > 90% accuracy on this dataset.

Start with MNIST\_FNN\_Low\_Acc.ipynb file

2. ConvNetJS CIFAR-10 demo (stanford.edu)

The demo files can be found in the 'Hands on Activity' folder on Teams



# References

- Deep learning book (<https://www.deeplearningbook.org/>)
- University of Maryland DATA/MSML 603: Principles of Machine Learning lecture slides, Kemal Davaslioglu, 2022
- Stanford University, CS 231 Lecture Slides, <http://cs231n.stanford.edu/slides/2017/> , 2017
- Princeton University, COS 495, Lecture Slides, Yingyu Liang, <https://www.cs.princeton.edu/courses/archive/spring16/cos495/> , 2016
- University of Texas at Austin, Recurrent Neural Network, Lecture Slides, Danna Gurari, 2021
- [Chapter 8 Recurrent Neural Networks | Deep Learning and its Applications \(frcs.github.io\)](https://frcs.github.io/Chapter8RecurrentNeuralNetworksDeepLearninganditsApplications/)
- Carnegie Mellon University, Introduction to Graph Neural Networks, Minji Yoon, [https://minjiyoon.xyz/Paper/Minji Yoon intro deep graph learning.pdf](https://minjiyoon.xyz/Paper/Minji%20Yoon%20intro%20deep%20graph%20learning.pdf) , 2022
- <https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial>

