IS "DEEP LEARNING" OVER MY HEAD?

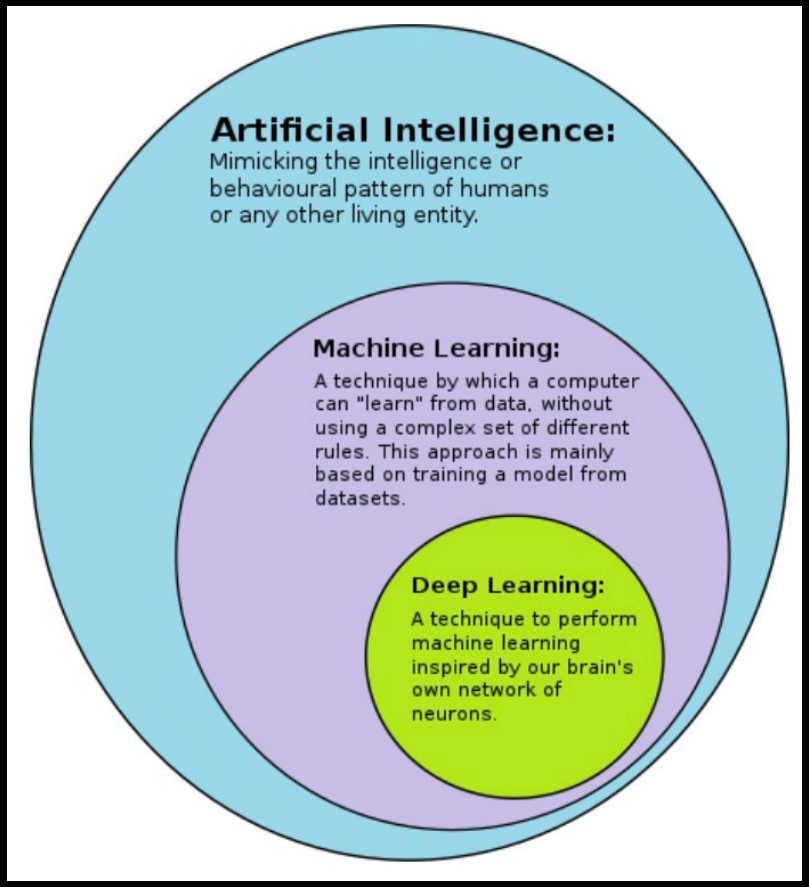
Erik Mayer – Electronics Engineering Technology

ABSTRACT

- In this talk, an overview of deep learning is given. Some of the different types of neural networks used in deep learning such as convolutional and recurrent neural networks will be discussed. Software libraries such as Keras and TensorFlow have been developed to make deep learning easier to use. An example application using Keras and TensorFlow will be discussed.

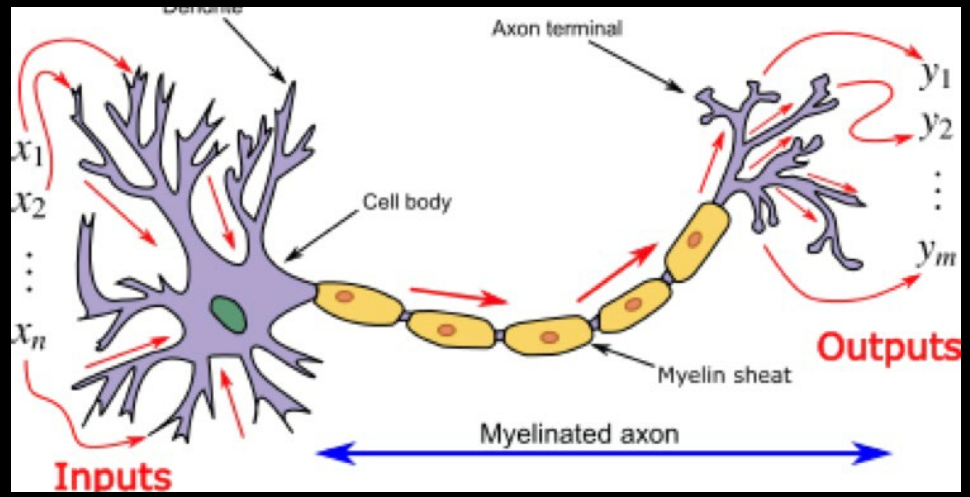
DEEP LEARNING

- Deep learning is a type of machine learning and artificial intelligence in which computers emulate the neurons in a human brain in a "neural network"
- Applications of deep learning
  - regression
  - pattern recognition
  - classification (example: dogs and cats)
  - image recognition
  - speech recognition
  - autonomous control (example: self-driving cars)



https://en.m.wikipedia.org/wiki/Deep_learning#/media/File%3AAI-ML-DL.svg

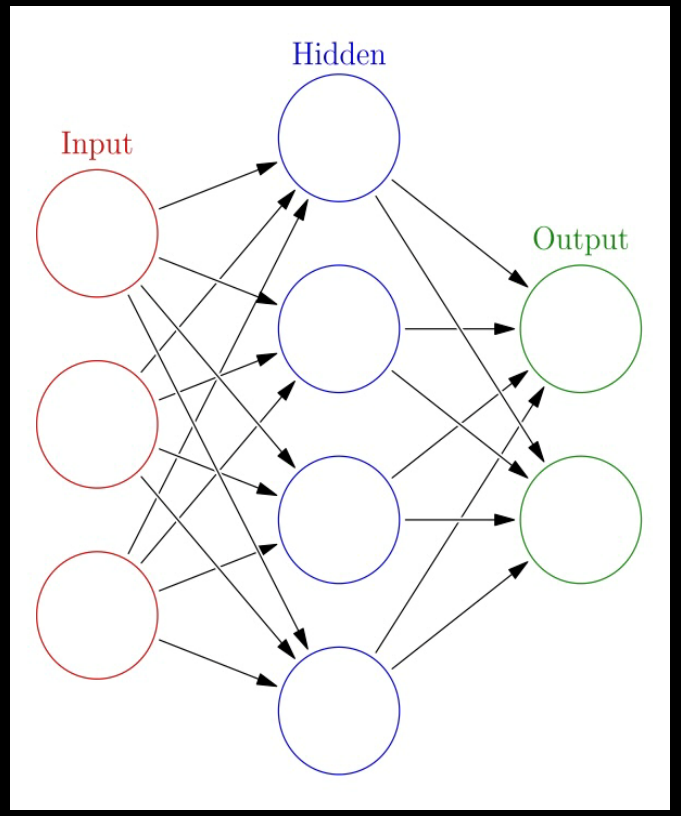
BIOLOGICAL NEURONS



https://en.m.wikipedia.org/wiki/Artificial_neural_network#/media/File%3ANeuron3.png

- Neural networks inspired by biological neurons in the brain
- Neurons receive electrochemical pulses through dendrites (inputs)
- Pulses from dendrites causes a voltage to be accumulated in soma
- Once the voltage exceeds a threshold, the soma will "fire" sending a pulse through the axon (output)
- Pulses from dendrites can be excitatory (help soma to fire) or inhibitory (prevent soma from firing)

ARTIFICIAL NEURAL NETWORKS



https://en.m.wikipedia.org/wiki/Artificial_neural_network#/media/File%3AColored_neural_network.svg

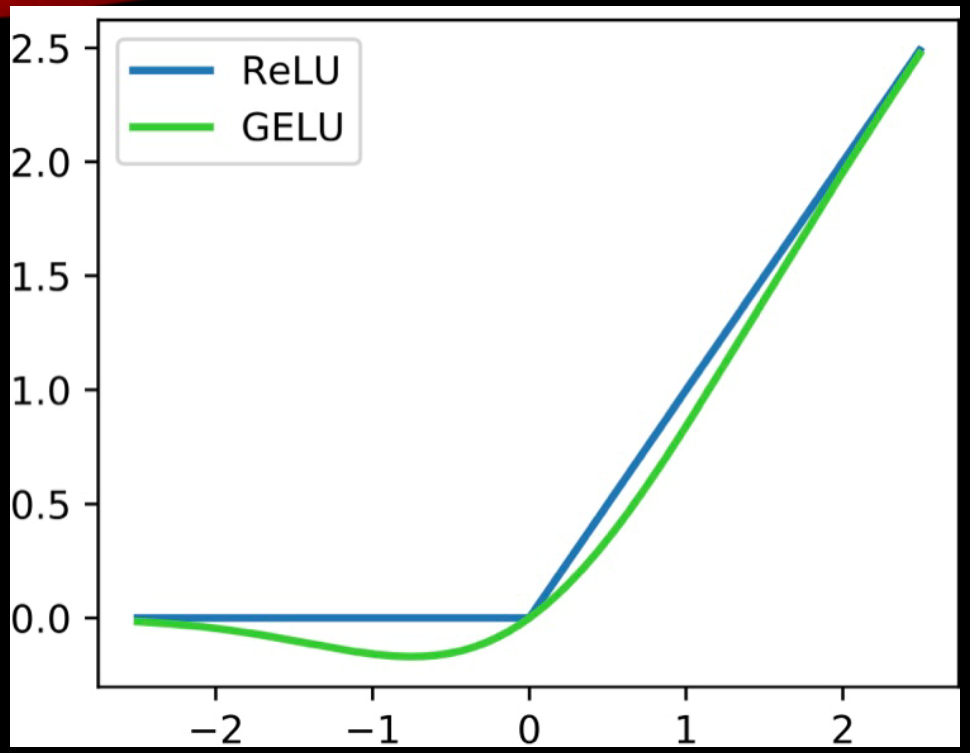
- Inputs and outputs of neurons represented by numbers
- The output of a neuron can be given by mathematical equation

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

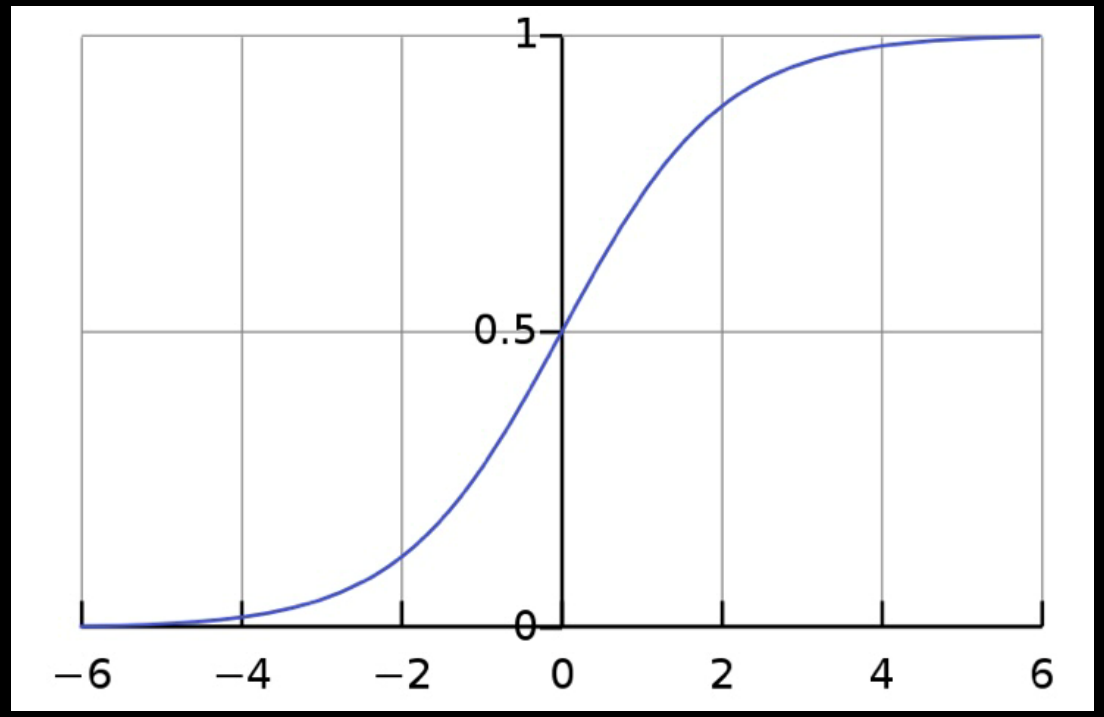
  - Xj is an input to the neuron
  - Wkj is the weight of the input
  - Usually, X0 = 1 and is the bias of the neuron
  - Phi is the activation function that is typically a threshold function

https://en.m.wikipedia.org/wiki/Artificial_neuron

ACTIVATION FUNCTIONS



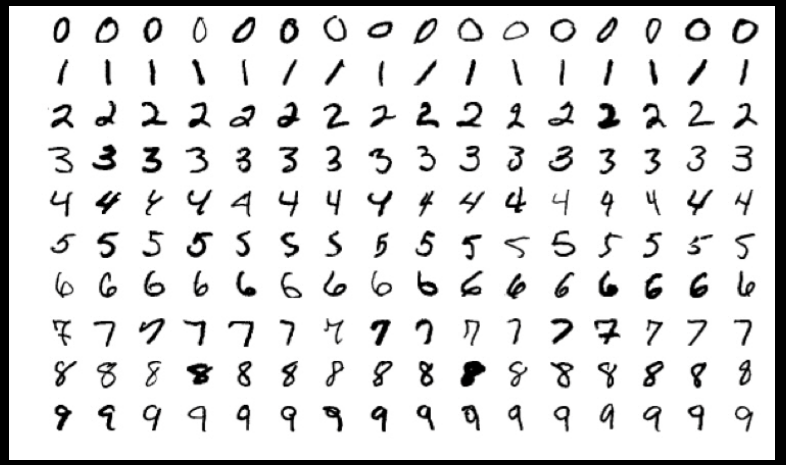
https://en.m.wikipedia.org/wiki/Rectifier_(neural_networks)#/media/File%3AReLU_and_GELU.svg



Sigmoid function
https://en.m.wikipedia.org/wiki/Sigmoid_function

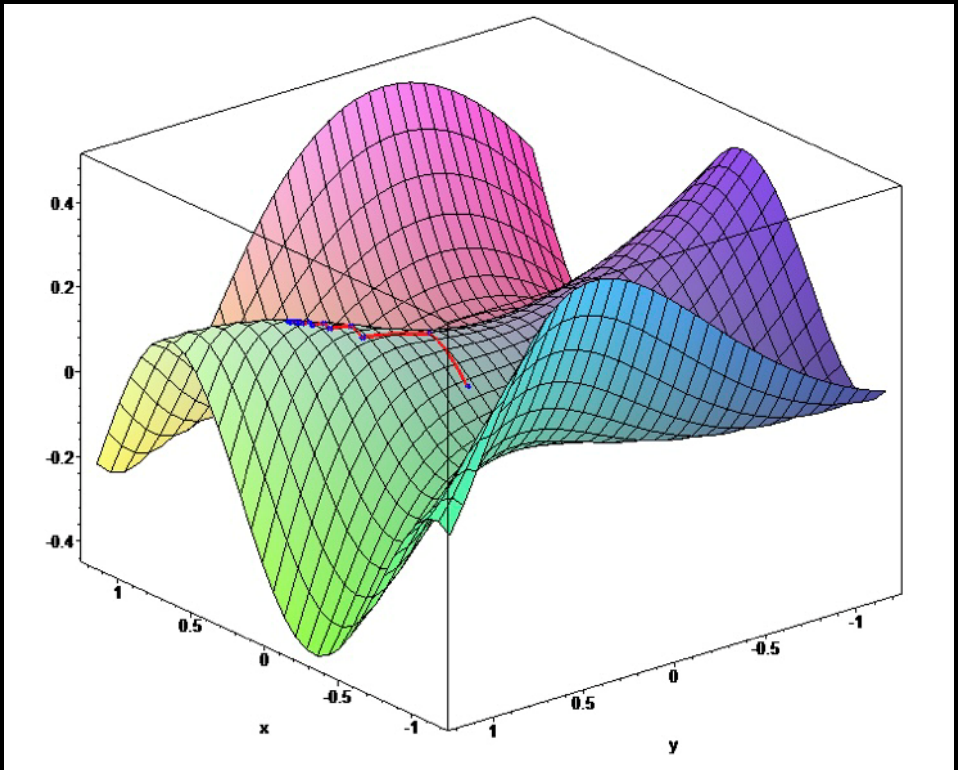
HOW TO TRAIN YOUR NEURAL NETWORK

- Neural networks can be trained by example similar to people
  - Example: A neural network can be shown examples of the letter "s" and then will be able to recognize a "s" it hasn't seen before
  - The weights in a neural network are usually randomized at the beginning. The error present will be represented by a loss function (also called a cost or error function
    - Example: sum of squared errors (SSE)
  - An algorithm called an optimizer adjusts the weights during training to minimize the loss function
    - Example: Gradient descent



Sample images from MNIST test dataset https://en.m.wikipedia.org/wiki/MNIST_database#/media/File%3AMnistExamples.png

GRADIENT DESCENT

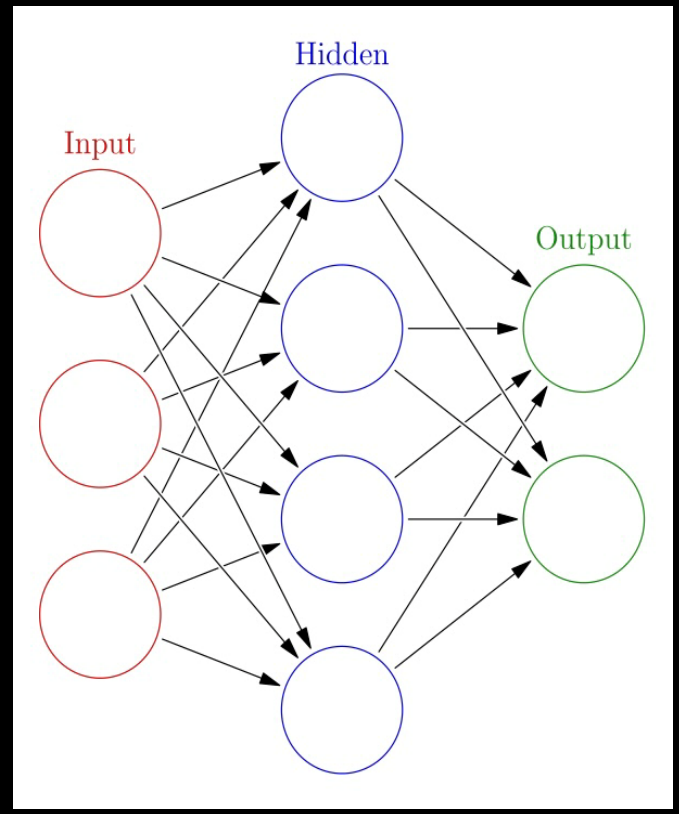


https://en.m.wikipedia.org/wiki/Gradient_descent#/media/File%3AGradient_ascent_(surface).png

- Calculates the slope of the loss function
- Adjusts weights so the value of loss function is decreased (moves down slope)
- Adjustments may be needed such that a global minimum is found instead of becoming stuck in a local minimum
  - Example: momentum

BACKPROPAGATION

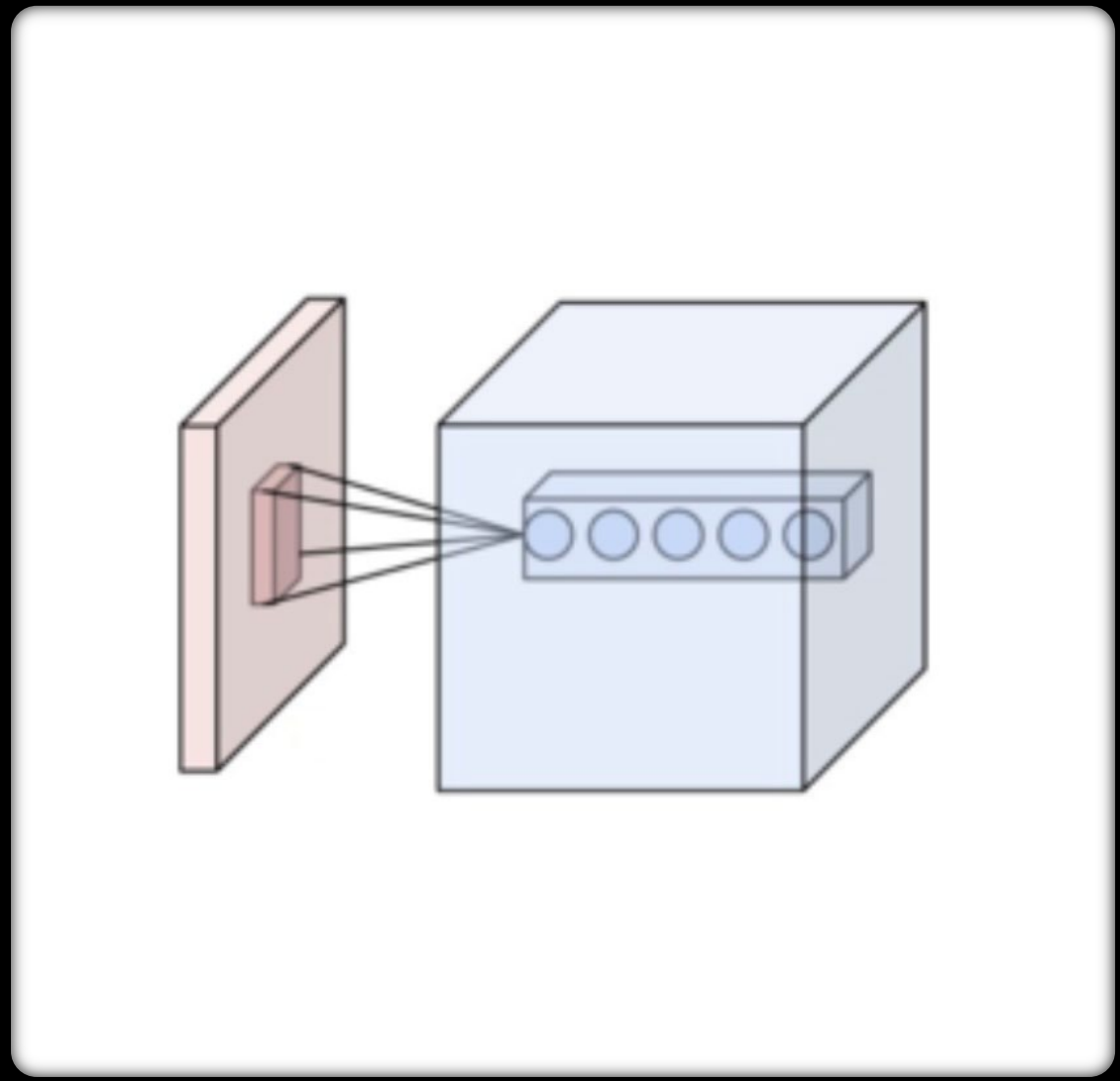
- Begins by adjusting weights in output layer to decrease the loss function
- Repeats with last hidden layer
- Repeat for each hidden layer working backwards to the input



https://en.m.wikipedia.org/wiki/Artificial_neural_network#/media/File%3AColored_neural_network.svg

CONVOLUTIONAL NEURAL NETWORKS

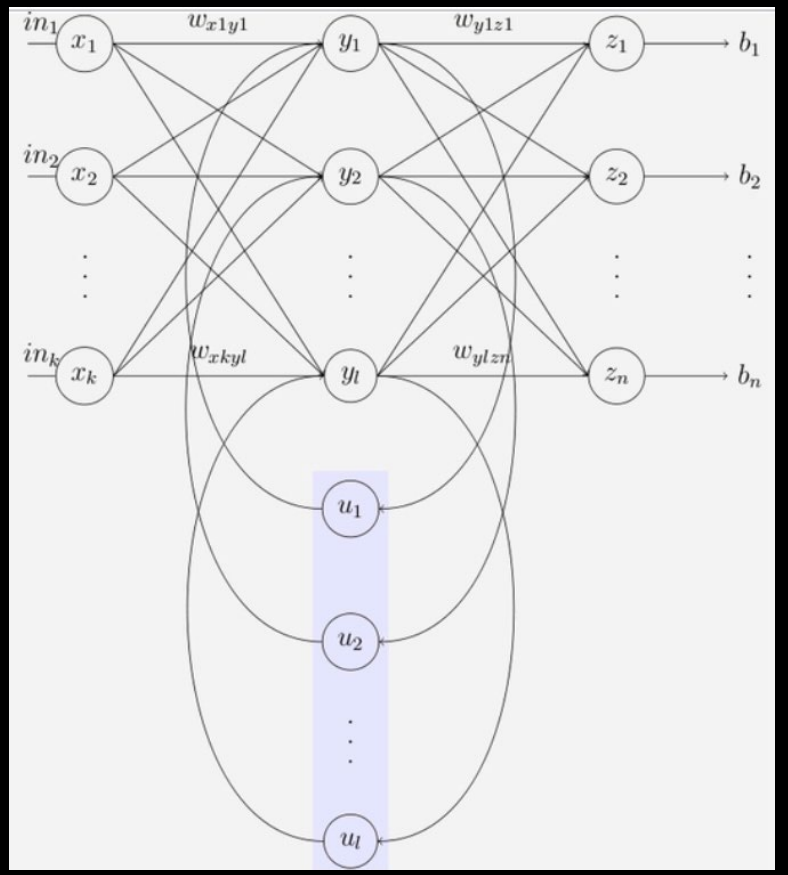
- Used in image and video recognition and image classification
- Uses convolution kernels or filters



https://en.m.wikipedia.org/wiki/Convolutional_neural_network#/media/File%3AConv_layer.png

RECURRENT NEURAL NETWORKS

- Used for handwriting and speech recognition
- Output of a neuron fed back as an input
  - Gives neural network memory
- Used for temporal sequences of data



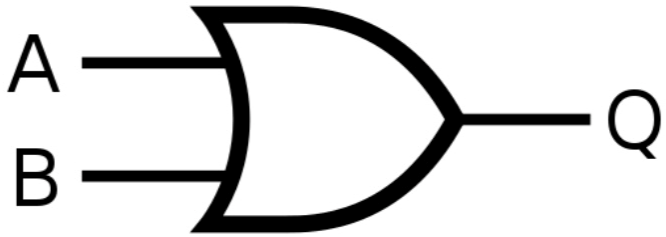
https://en.m.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Elman_srnn.png

KERAS

- https://keras.io
- API (application programming interface) for deep learning
- Built on top of TensorFlow 2, an open-source machine learning platform (https://www.tensorflow.org)
- Has a Python interface

https://keras.io

Input		Output
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



https://en.m.wikipedia.org/wiki/OR_gate

KERAS EXAMPLE: OR GATE

KERAS EXAMPLE: OR GATE

- Created in Google Colaboratory (https://colab.research.google.com)

```
# import required modules
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.optimizer import SGD
Import numpy as np
```

KERAS EXAMPLE: OR GATE

```
# Training data

# input data
X = np.array([[0.1,0.1],[0.1,0.9],[0.9,0.1],[0.9,0.9]])

# output data
y =  np.array([[0.1],[0.9],[0.9],[0.9]])
```

KERAS EXAMPLE: OR GATE

```
# Create neural network using functional API

# Create input layer with two inputs
inputs = keras.Input(shape=(2,))

# Create output layer with one neuron
outputs = layer.Dense(1, activation = “sigmoid”)(inputs)

model = keras.Model(inputs=inputs, outputs=outputs, name=“OR”)
```

KERAS EXAMPLE: OR GATE

```
# Didplay neural network
model.summary()
```

```
Model: "OR"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense (Dense)                (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

KERAS EXAMPLE: OR GATE

```
print("Weights and biases:")
for layer in model.layers:
  print(layer.name)
  print(layer.get_weights())

print("Predictions:")
predictions = model.predict(X)
print(predictions)
```

```
Weights and biases:
input_2
[]
dense_1
[array([[0.7737204 ],
       [0.17491889]], dtype=float32), array([0.], dtype=float32)]


Predictions:
[[0.5236982 ]
 [0.55843157]
 [0.67124915]
 [0.7013585 ]]
```

KERAS EXAMPLE: OR GATE

```
# Training

# Use mean squared error for loss function and use gradient descent
model.compile(loss='mean_squared_error',optimizer=SGD(learning_rate=1))

# Using all four training data, adjust weights 2000 times, and show progress
history = model.fit(X, y, batch_size=4, epochs=2000, verbose=1)
```

KERAS EXAMPLE: OR GATE

```
print("Weights and biases:")
for layer in model.layers:
  print(layer.name)
  print(layer.get_weights())

print("Predictions:")
predictions = model.predict(X)
print(predictions)
```

```
Predictions
[[0.12026423]
 [0.89011884]
 [0.890119  ]
 [0.9979211 ]]


Weights and biases:
input_2
[]
dense_1
[array([[5.1023602],
        [5.1023574]], dtype=float32), array([-3.0104022], dtype=float32
```

FURTHER READING

- YouTube: 3Blue1Brown, Neural Networks series
- Practical Deep Learning Online, https://www.doulos.com/training/deep-learning/practical-deep-learning-online/

ACKNOWLEDGEMENTS

- This research was supported in part by Kansas National Space Grant College and Fellowship Program – Opportunities in NASA STEM FY 2020-2024
  - “This material is based upon work supported by the National Aeronautics and Space Administration under Grant No. 80NSSC20M0109. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration nor of Wichita State University."
- Thanks for the support of Artificial Intelligence (AI) Curriculum Committee