DEEP LEARNING

(PART 1)


Science
Platforms
Machine
Interview
Frameworks
Python
TensorFlow
Algorithms
Explained
Deep
Learning
Big
Analytics
Answers
Part
Must-Know
projects
Scientists
Questions
Top
Data
Spark

Contents

What is Deep Learning

Deep Learning

Deep learning is a form of artificial intelligence that uses a type of machine learning called an artificial neural network with multiple hidden layers that learns hierarchical representations of the underlying data in order to make predictions given new data.

Deep Learning was widely considered impossible in the 1990s when most researchers had abandoned the idea

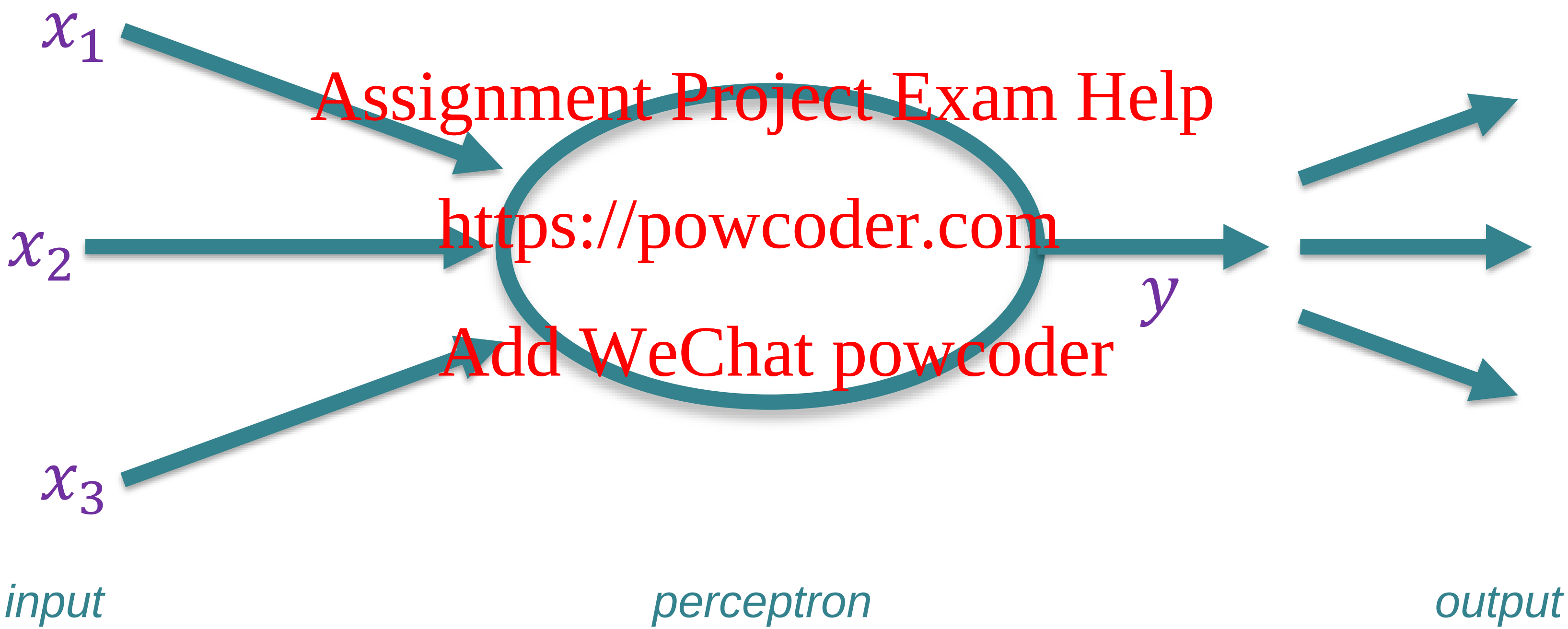
- In 2006, Geoffrey Hinton et al. published a paper showing how to train a deep neural network capable of recognizing handwritten digits with state-of-the-art precision (>98%)
- They branded this technique Deep Learning
- The paper revived the interest of the scientific community and before long many new papers demonstrated that DL was not only possible, but capable of mind-blowing achievements that no other ML technique could hope to match
- This enthusiasm soon extended to many other areas of ML
- Fast-forward 15 years and ML has conquered the industry: it is now at heart of much of the magic in today's high-tech products

Artificial Neuron: Perceptron

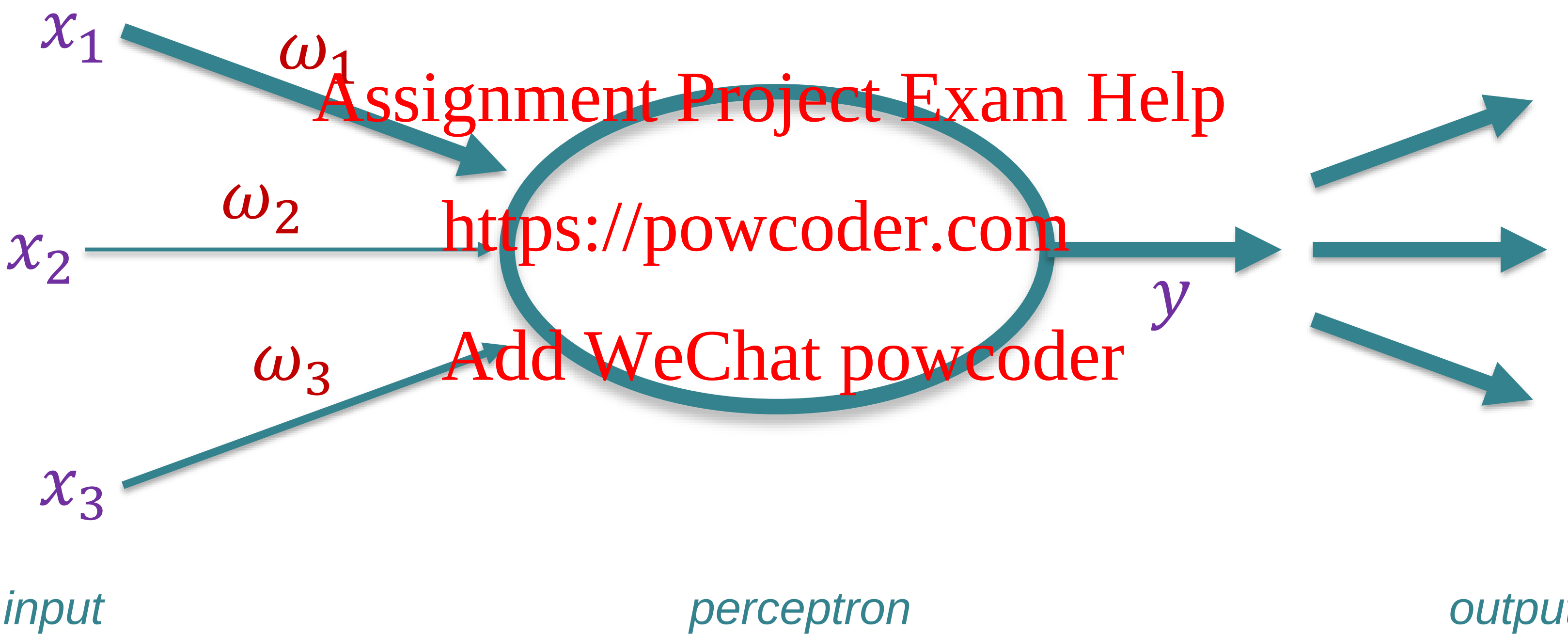
An artificial neural network is an ML algorithm based on a very crude approximation of a biological neural network in a brain

Artificial neural networks work quite differently than real biological neural networks; however, they were inspired by their biological counterpart

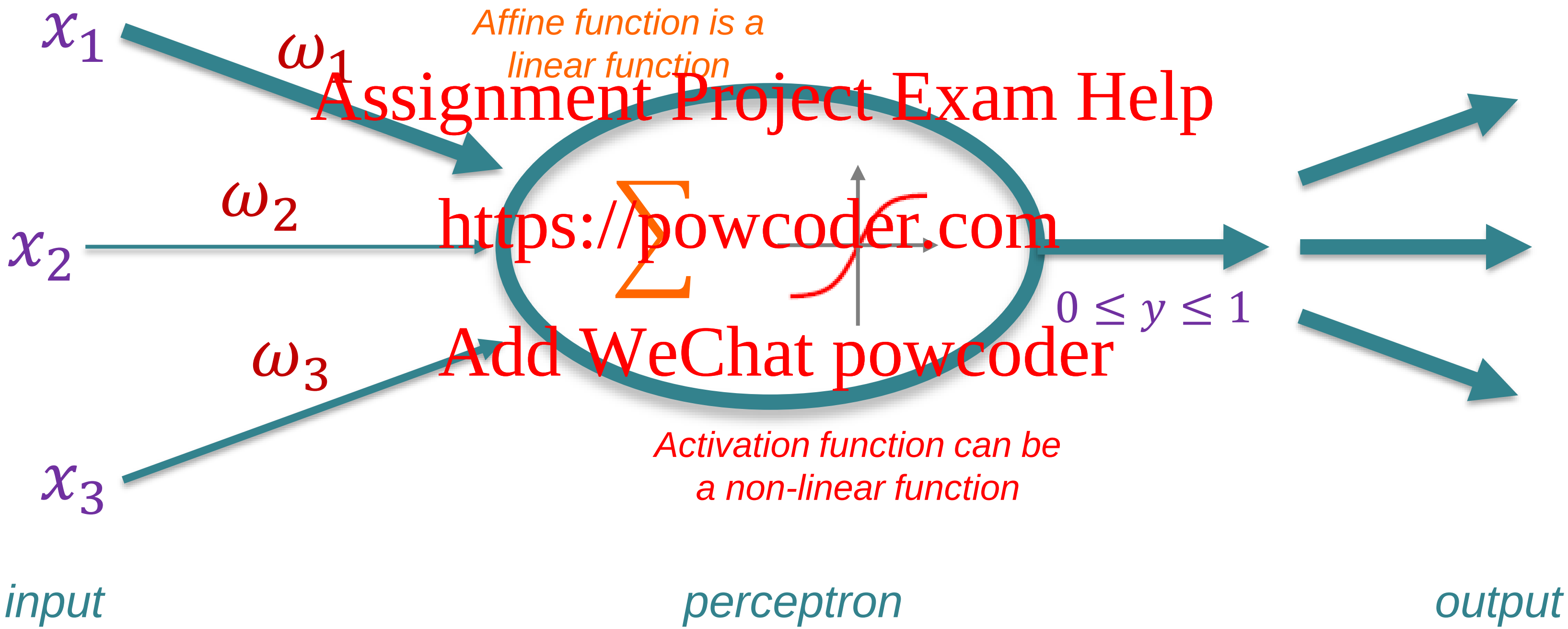
A perceptron takes a collection of inputs that can carry different weights and produces outputs to other perceptrons



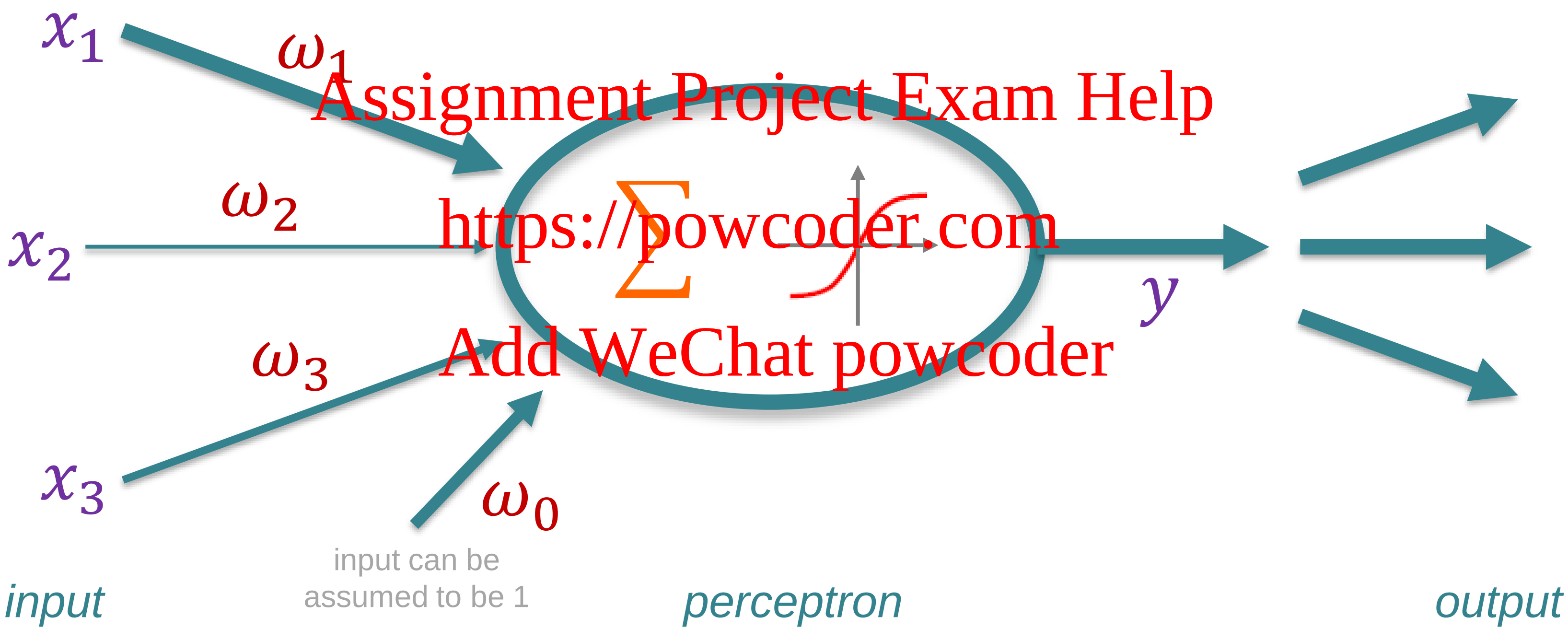
The weights for input can be either increased or decreased



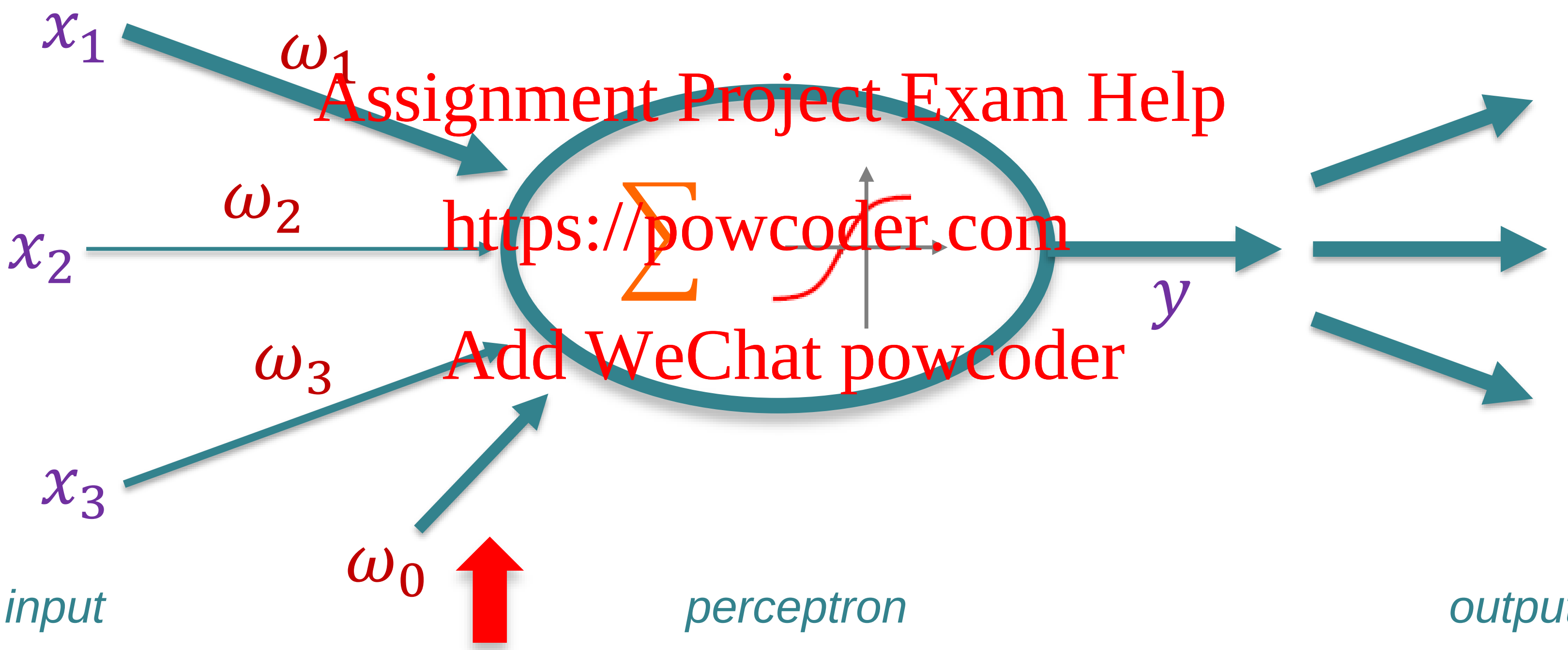
The activation function determines how much output it will produce given the summed input values



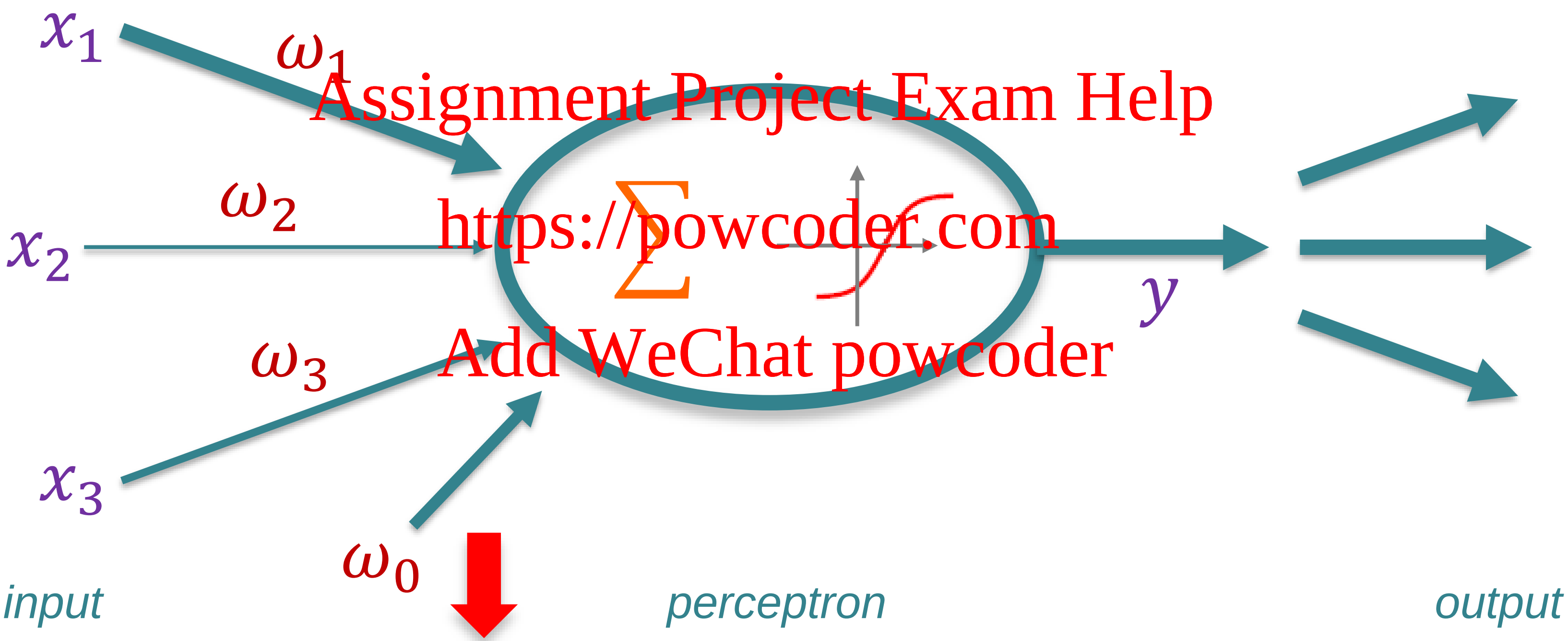
Bias is a weighted input that can be used to control the output value



Increasing the bias will shift the activation function to the left

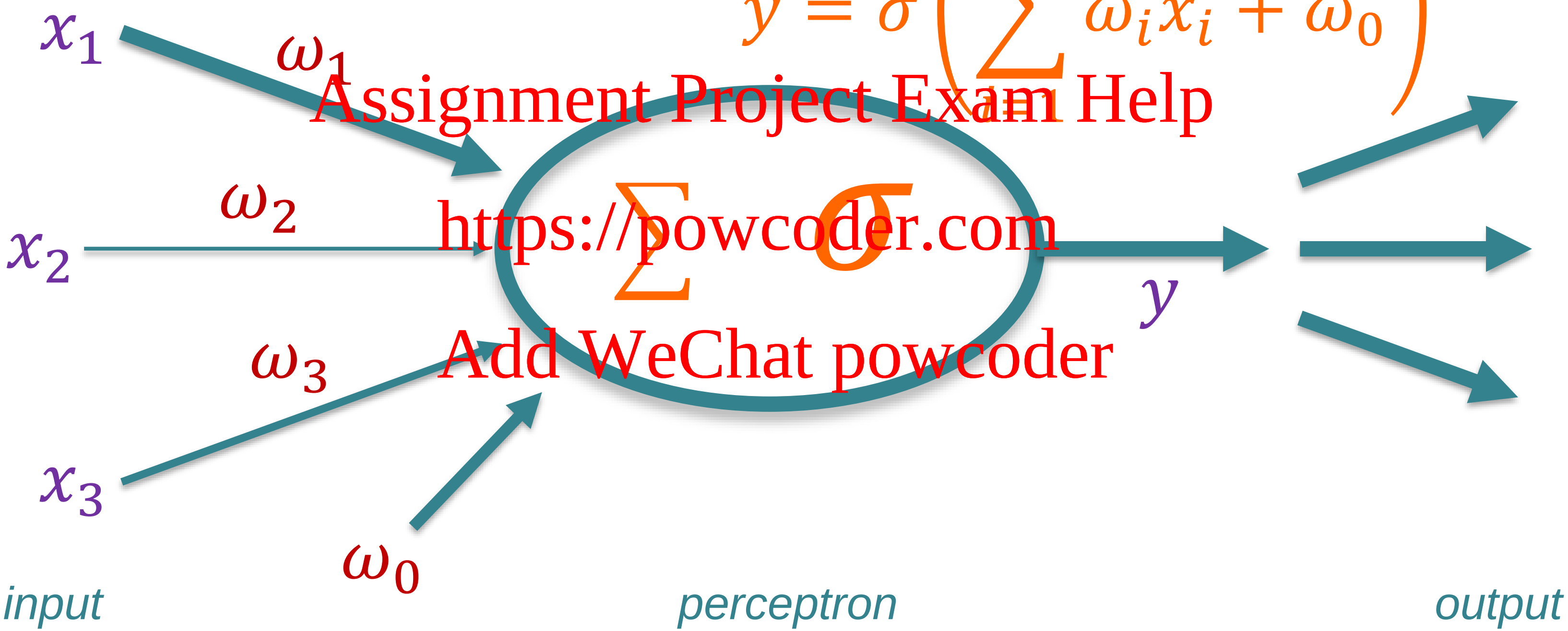


Decreasing the bias will shift the activation function to the right



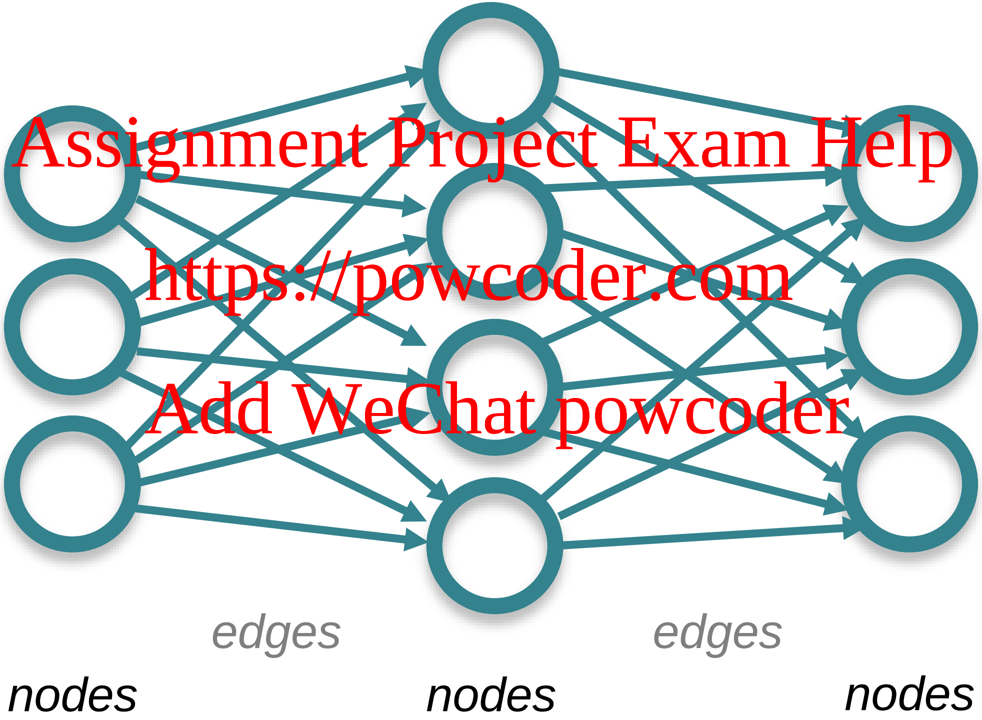
Modifying the weight parameters will change the behaviour of the perceptron

$$y = \sigma\left(\sum_{i=1}^{m} \omega_i x_i + \omega_0\right)$$

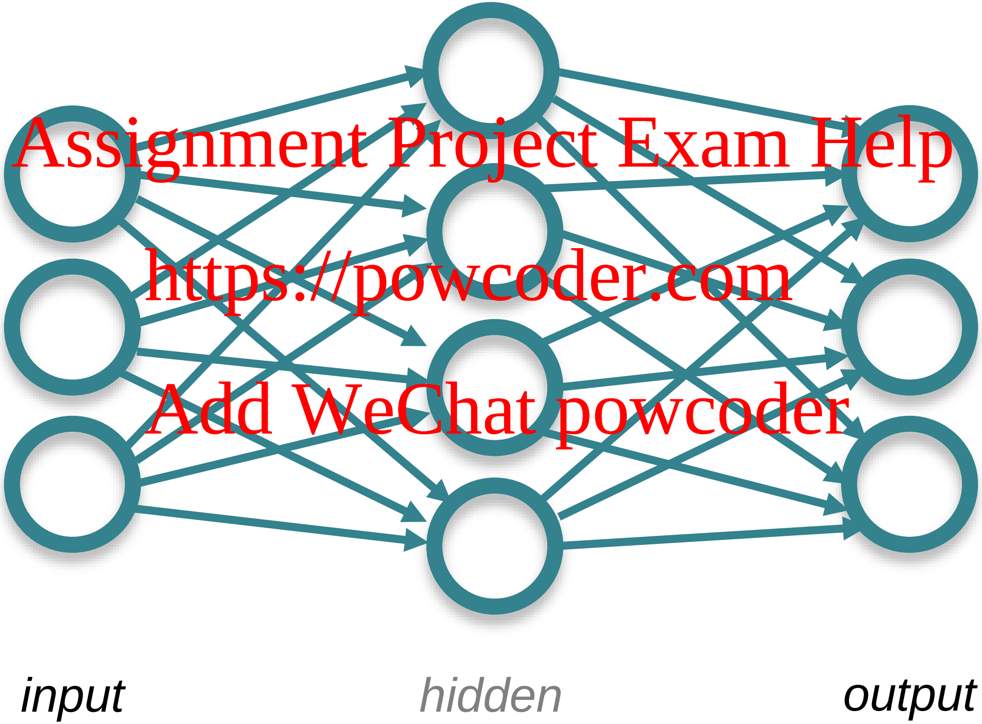


Artificial Neural Network

An artificial neural network is obtained by connecting the inputs and outputs of perceptrons into a network

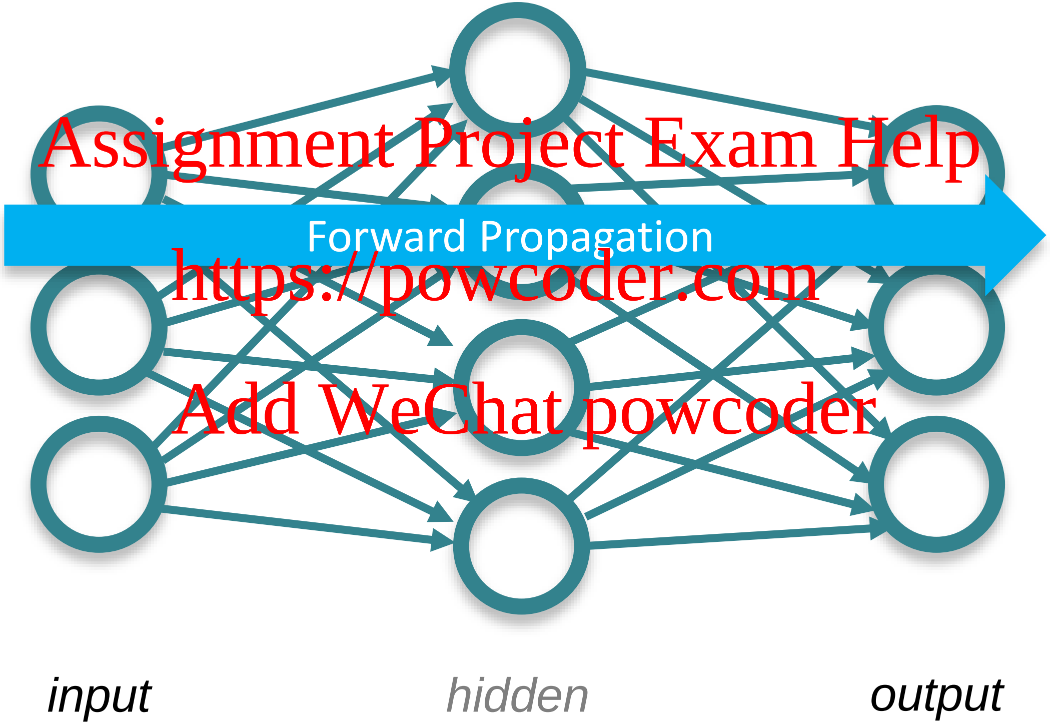


An artificial neural network is composed of an input layer, an output layer, and one or more hidden layers in between



Forward propagation uses the current network parameters to compute a prediction for each training data

Incorrect prediction (or prediction error) will be used to teach the network to change the weights of its connections

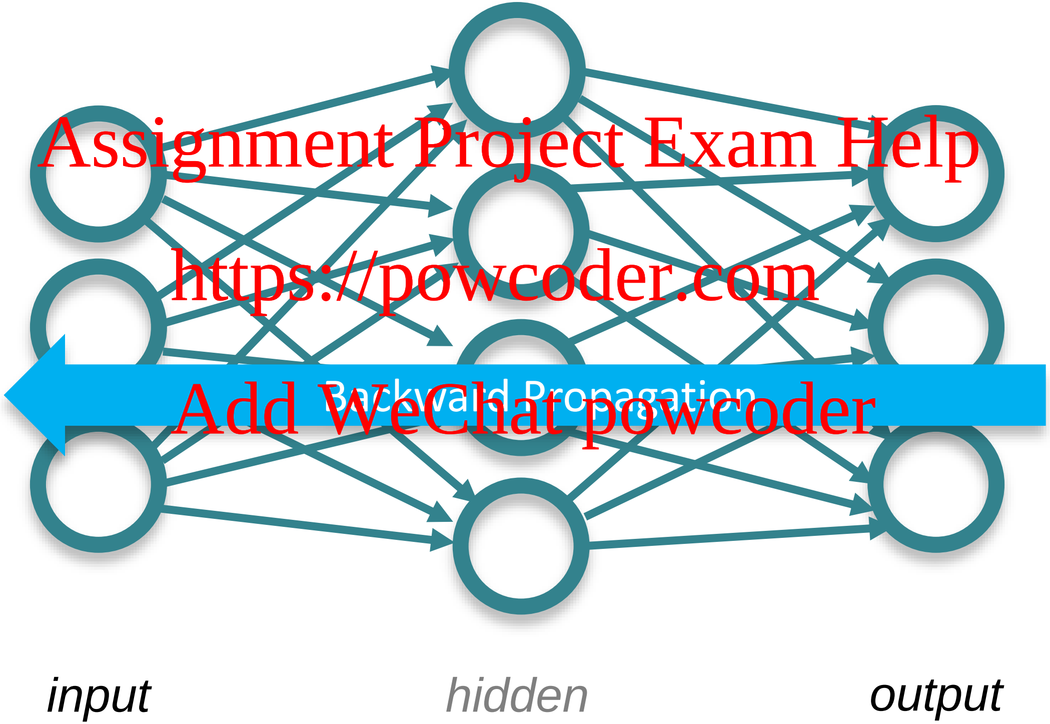


The labels of the training dataset are used to determine if the network made a correct prediction or not

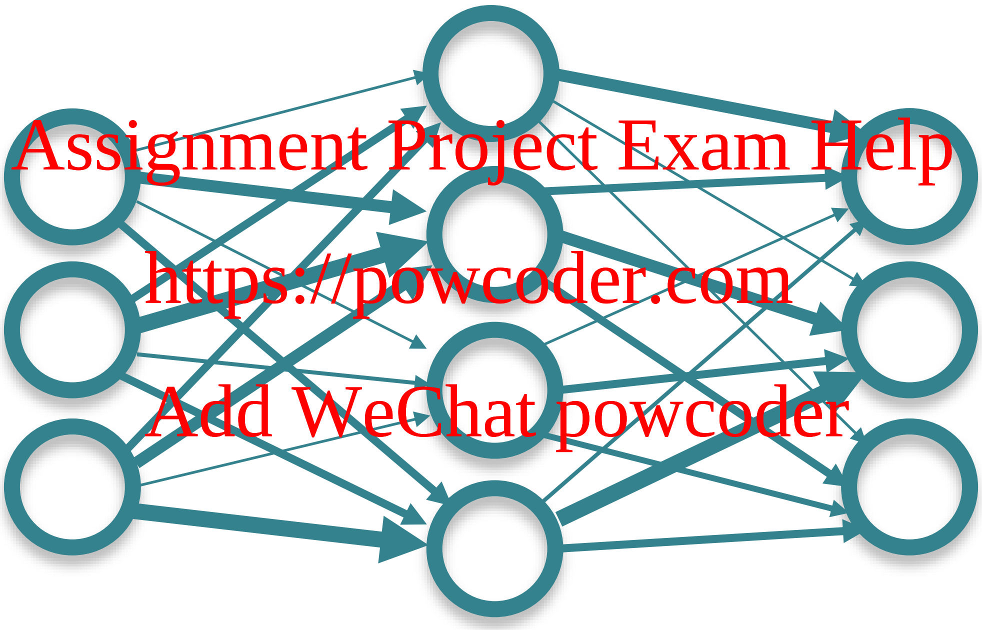
Backward propagation uses the prediction error to update the weights of the connections between neurons

Gradient descent is used to decide whether to increase or decrease the edge weights

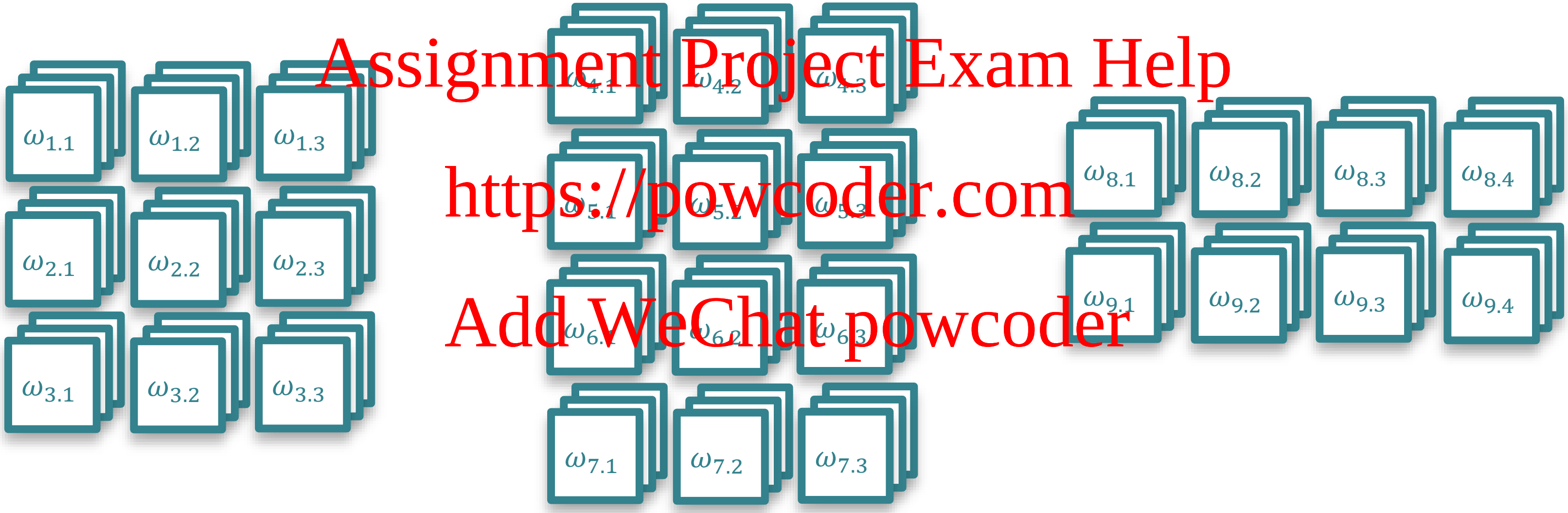
Training rate is used to decide how much to increase or decrease



Forward & backward propagations are repeated for each training data until the weights of the network become stable

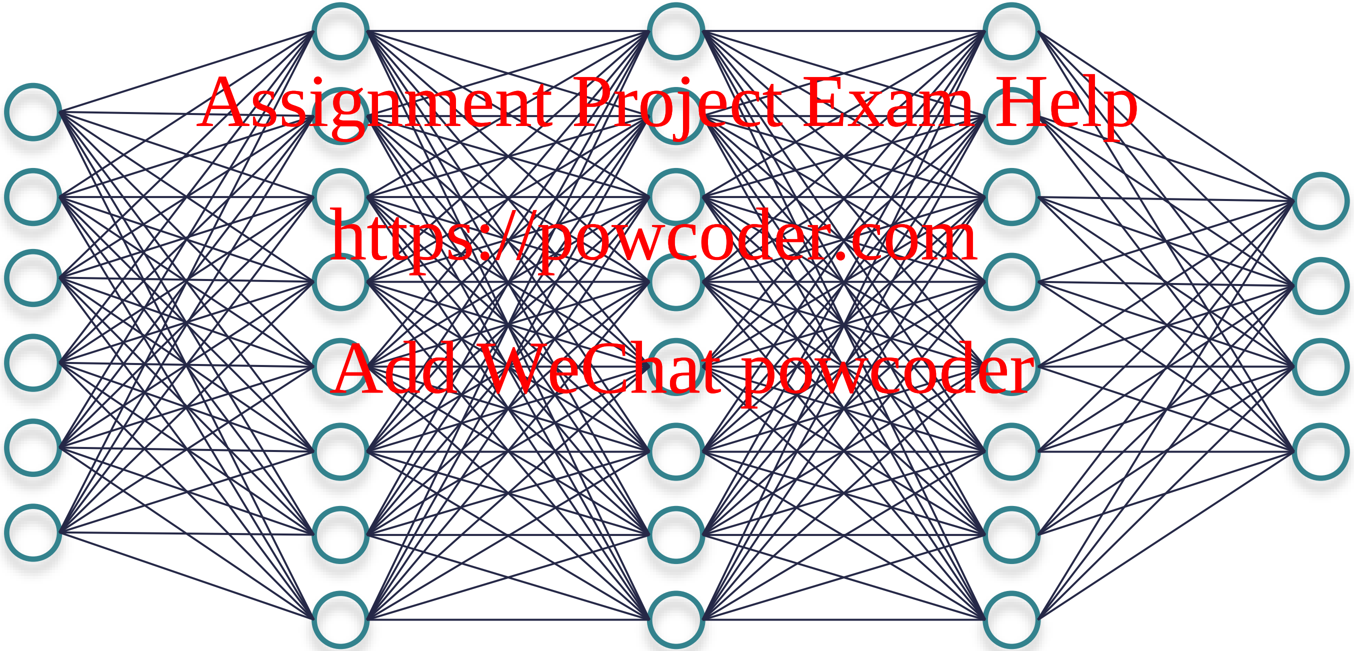


The network of nodes and edges are typically represented using much more computationally efficient data structures



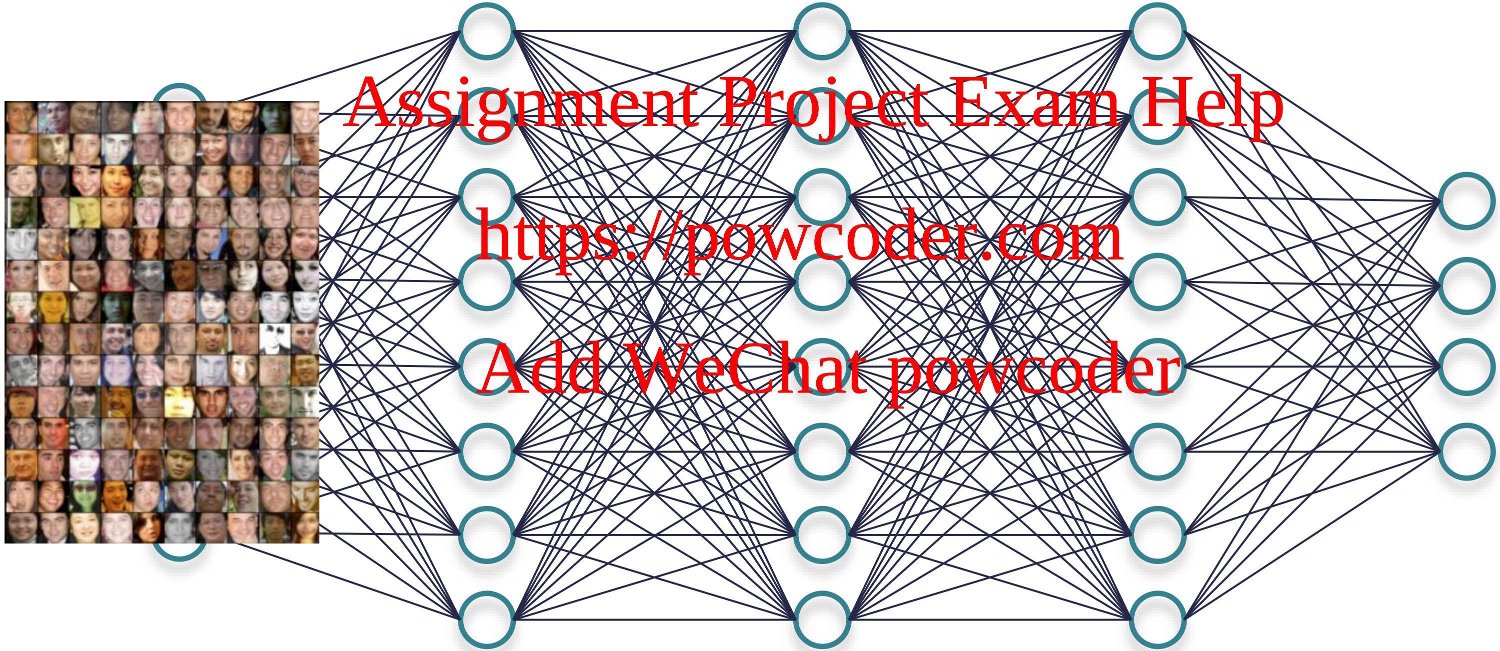
Deep Neural Network

Deep neural networks have more hidden layers allowing them to model progressively more complex functions

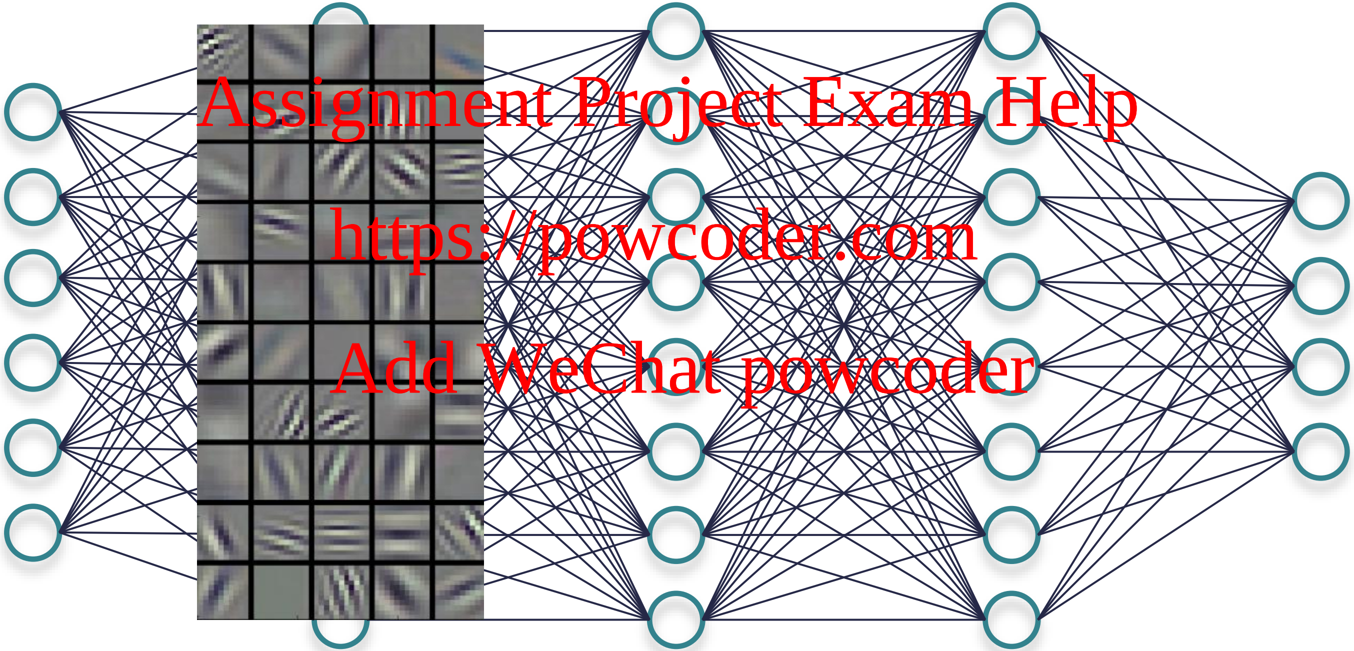


multiple hidden layers

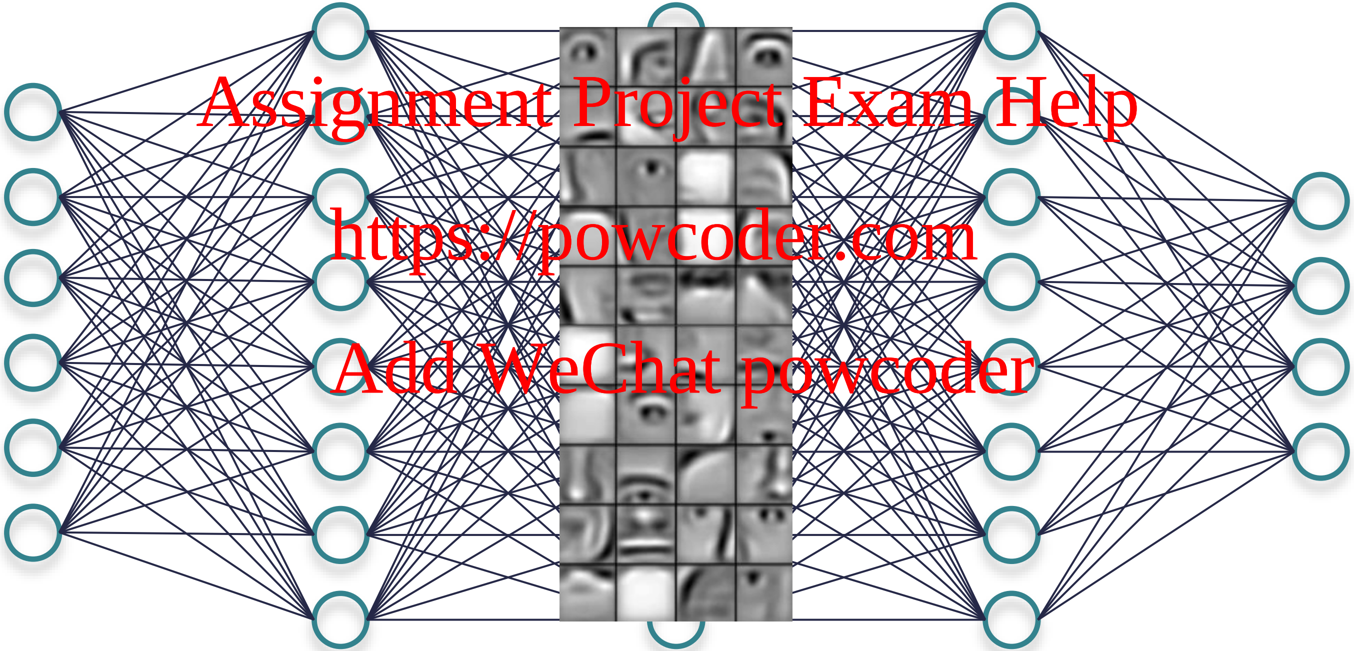
A face recognition deep neural network is trained by feeding to the input layer a set of labelled images of human faces



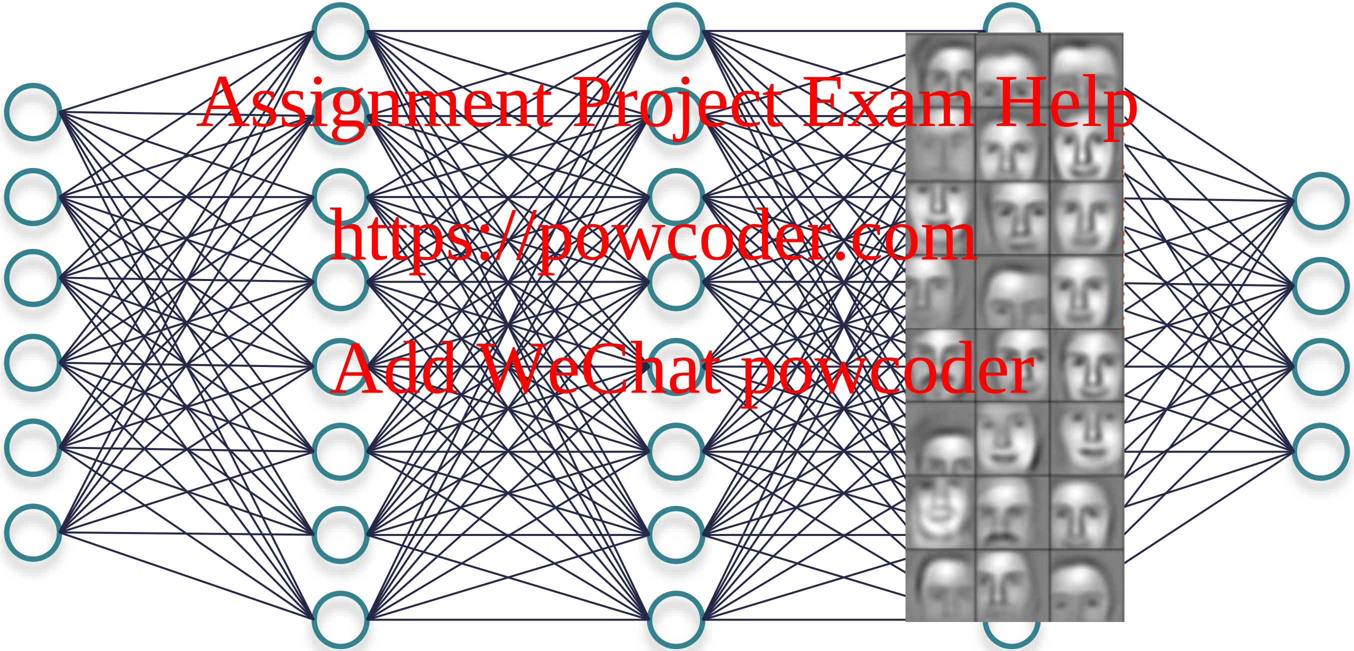
The first hidden layers would learn to detect geometric primitives, e.g. horizontal/vertical/diagonal lines



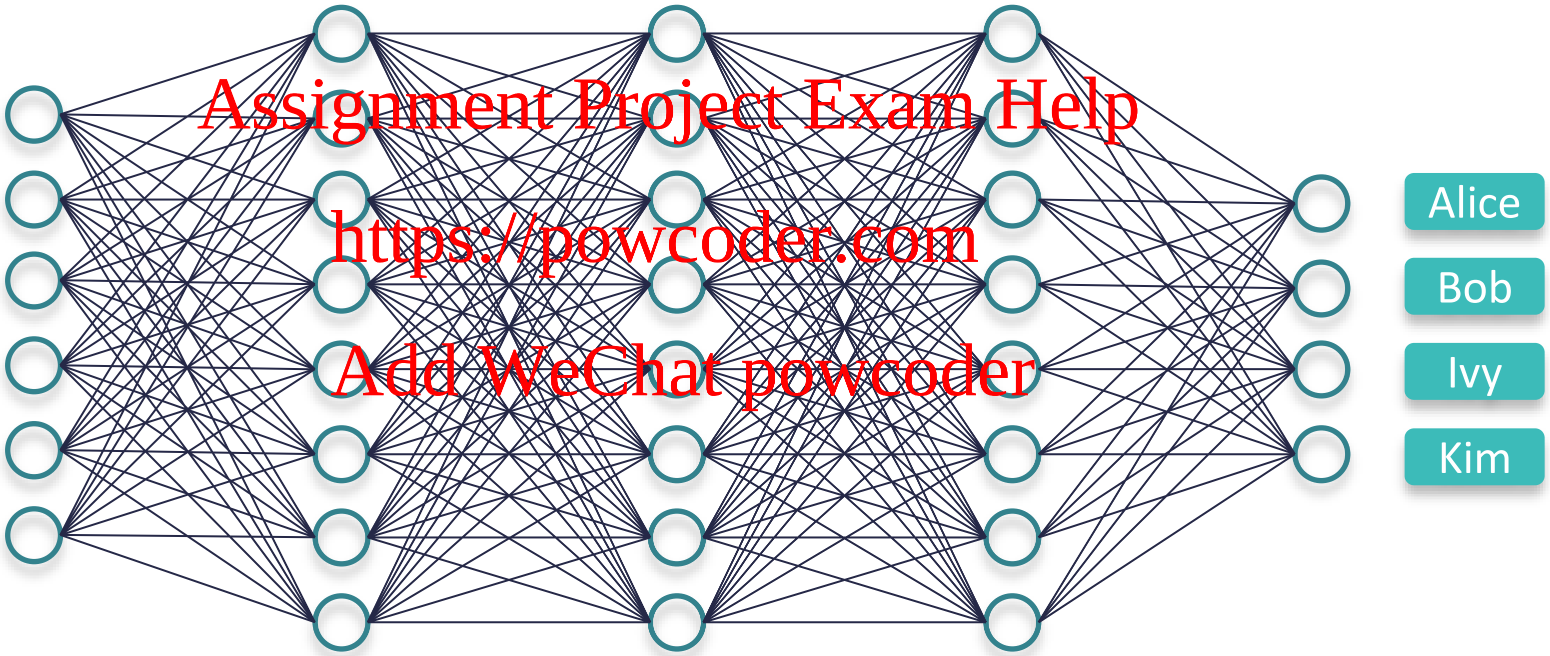
The middle hidden layers would learn to detect more complex facial features (e.g. eyes, noses, mouths)



The final hidden layers would learn to detect the general pattern for entire faces

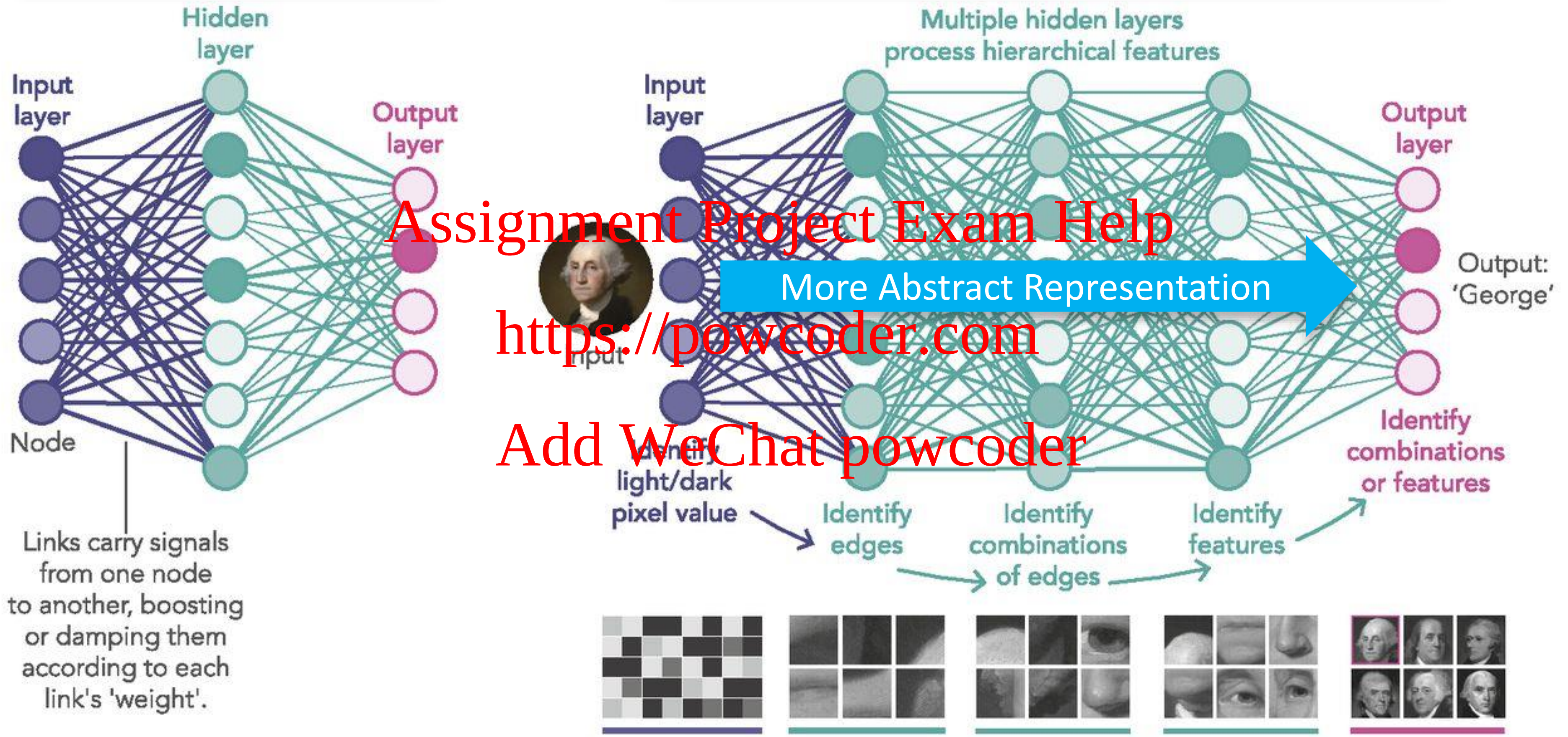


The output layer would learn to detect the most abstract representation of a person (e.g. the name of the person)

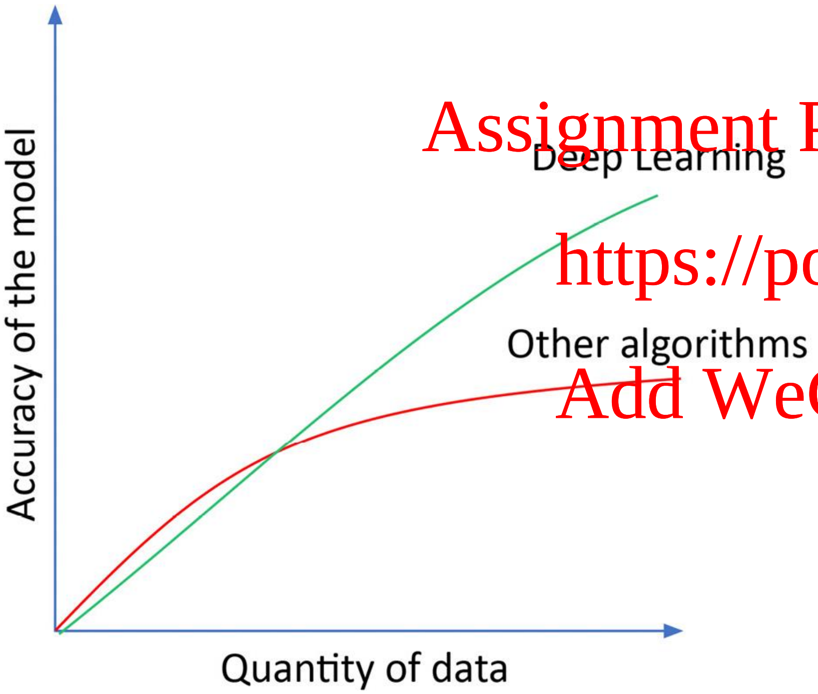


Simple Neural Network

Deep Neural Network



Tensor processing units (TPUs) are being developed to further accelerate the performance of deep learning



- Deep learning's popularity is due to its accuracy
- It has achieved higher accuracy levels than other algorithms have ever before for complex data problems such as natural language processing (NLP)
  - Requires, and actually capitalize on, vast amount of data to achieve an optimal solution
  - Also requires considerable computing power to be able to process such large amount of data without taking weeks or more to be trained

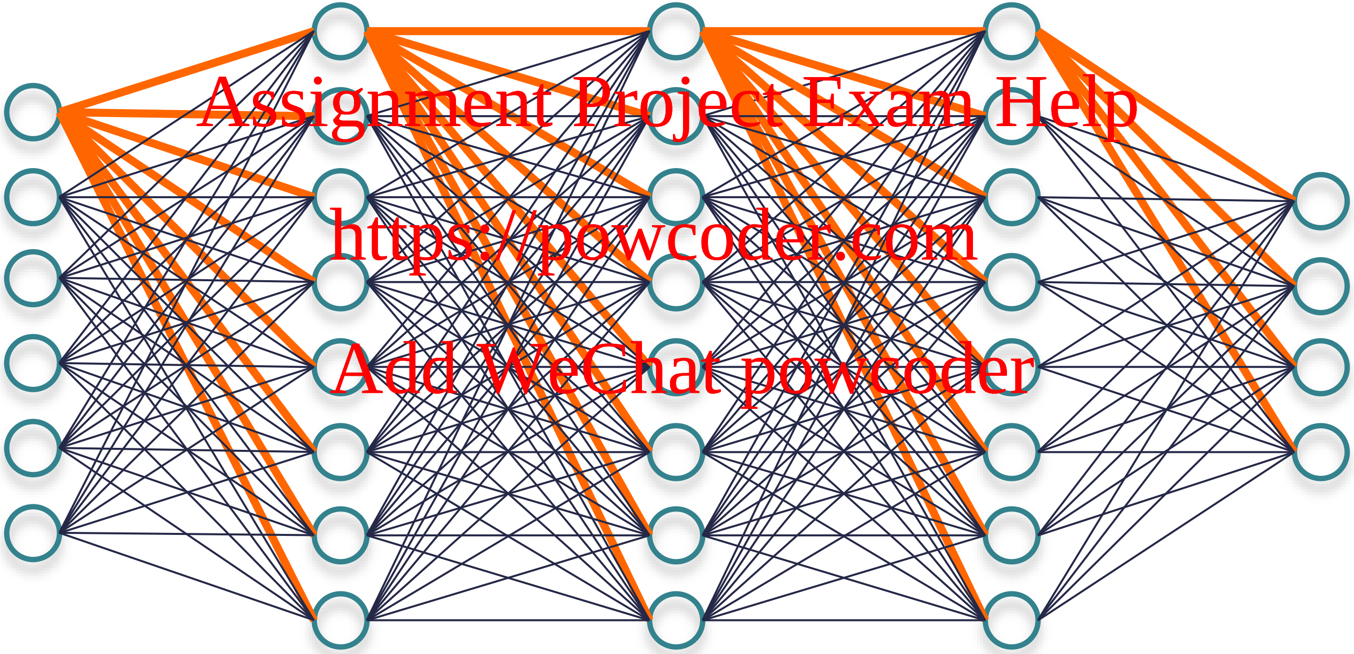
Deep Learning Architectures

Deep Learning Architectures

- Multilayer Perceptron (MLP)
  - the standard network architecture used in most basic neural network applications
- Convolutional Neural Networks (CNN)
  - a network architecture that works well for images, audios, and videos
- Recurrent Neural Networks (RNN)
  - a network architecture that works well for processing sequences of data over time
- Generative Adversarial Networks (GAN)
  - a technique where we place two opposing neural networks in competition with one another in order to improve each other's performance
- Deep Reinforcement Learning (RL)
  - a technique for providing reward signals when multiple steps are necessary to achieve a goal

Multilayer Perceptron (MLP)

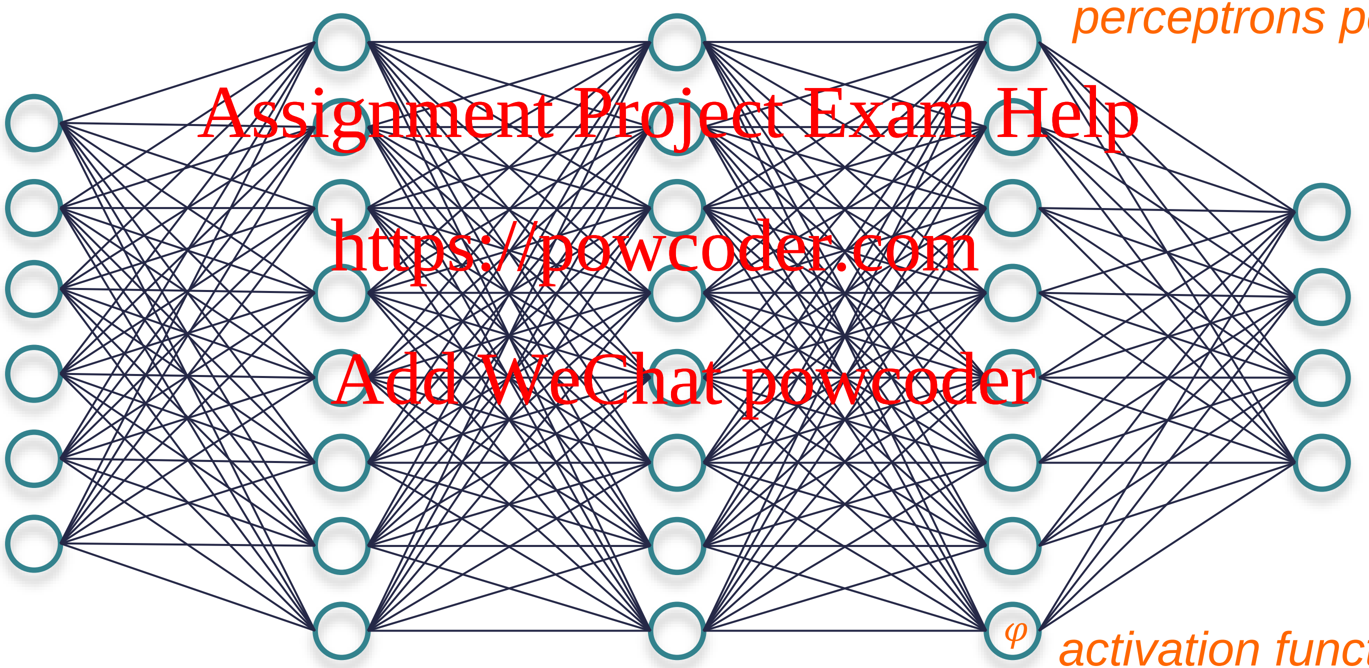
Each perceptron in the preceding layer can be connected to every perceptron in the subsequent layer



Perceptrons in any preceding layer are only ever connected to the perceptrons in a subsequent layer

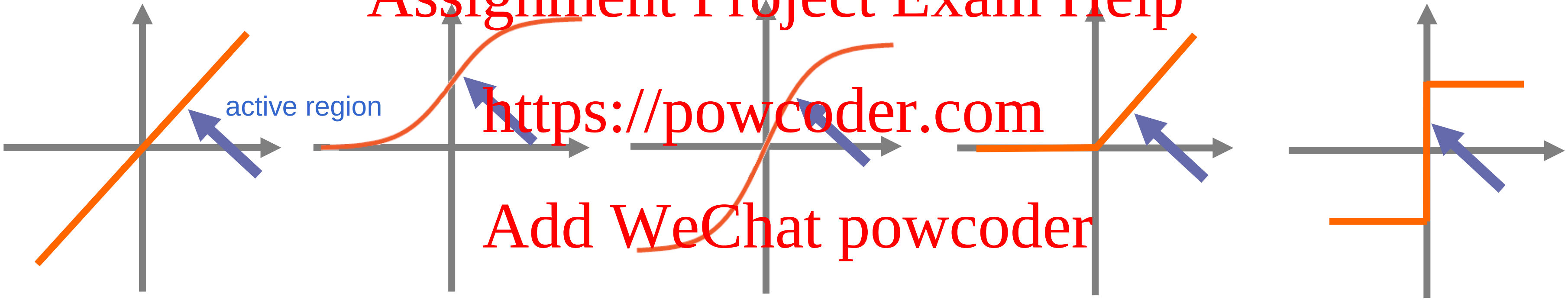
there is no cycle or loop in the connections of the graph of perceptrons

Architecture parameters include inputs, outputs, number of layers, perceptrons per layer & the activation functions



Activation Functions

non-linearity is what allows deep neural networks to model complex functions

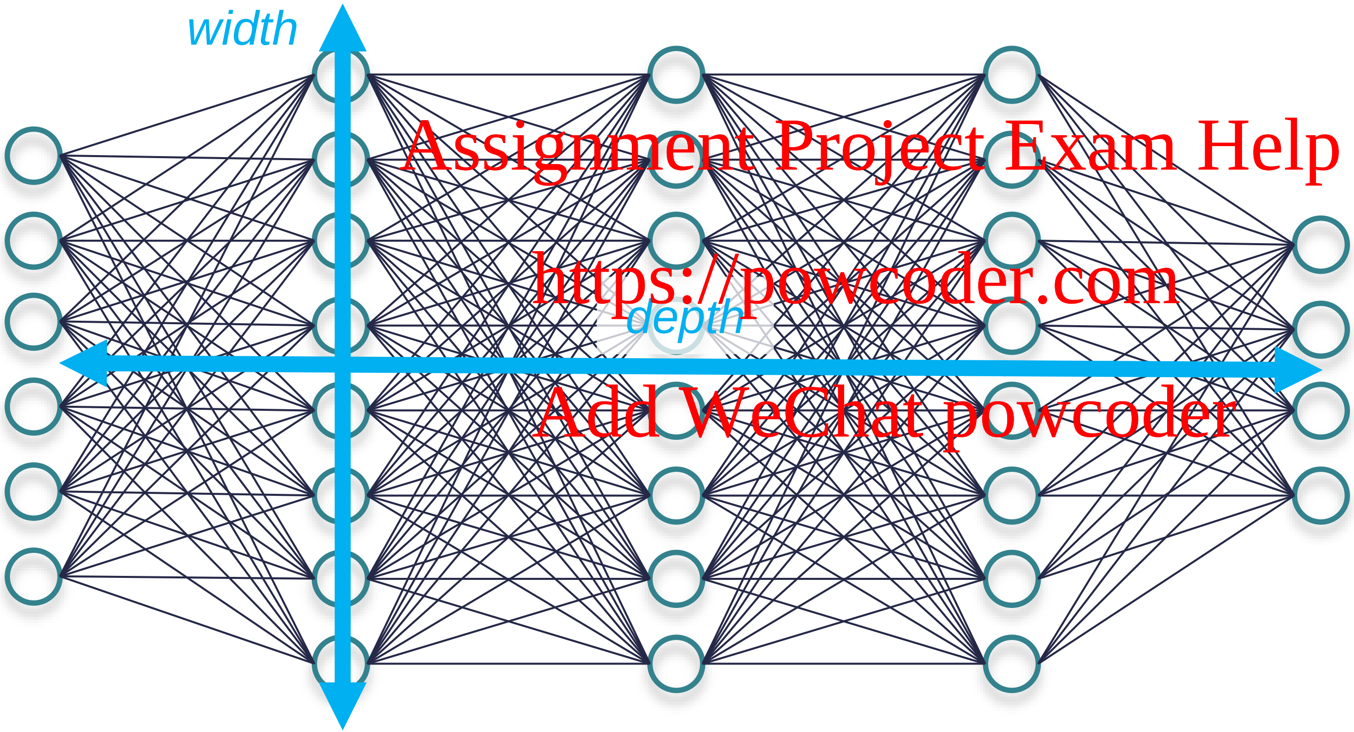


Linear | *Logistic (sigmoid)* | *Hyperbolic Tangent (tanh)* | Rectified Linear Unit (*ReLU*) | Step *(binary)*

Interpretation of the Activation Functions

Activation Function	Mathematical Representation	Value Range	Remark
Linear	$\sigma(x) = cx$	-∞ to +∞	Not possible to use in backward propagation as the derivative of the function is a constant and has no relation to the input x. All layers of the neural network collapse into one – becoming simply a linear regression model.
Logistic (sigmoid)	$\sigma(x) = \frac{1}{1 + e^{-x}}$	0 to 1	Normalizes an input real value. Good for classifier. For x above 2 or below -2, tends to bring the prediction value y to the edge of the curve, very close to 1 or 0. This enables clear predictions. Vanishing gradient problem for high/low x values.
Hyperbolic Tangent (tanh)	$\sigma(x) = \frac{2}{1 + e^{-2x}} - 1$	-1 to +1	Zero centered—making it easier to model inputs that have strongly negative, neutral, and strongly positive values. Gradient strength stronger than sigmoid providing more optimized solution; otherwise, like the Sigmoid function.
ReLU (Rectified Linear Unit)	$\sigma(x) = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x \geq 0 \end{cases}$	0 to +∞	Making the activation sparse and efficient. Converge faster than other functions (i.e. computationally efficient) speeding up network training. No vanishing gradient problem. Use softmax for classification. Use linear function for regression.
Step	$\sigma(x) = \begin{cases} -1 \ if \ x < 0 \\ 1 \ if \ x \geq 0 \end{cases}$	-1 to +1	A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the perceptron is activated and sends exactly the same signal to the next layer.

Each added perceptron increases the network complexity and therefore the required processing power

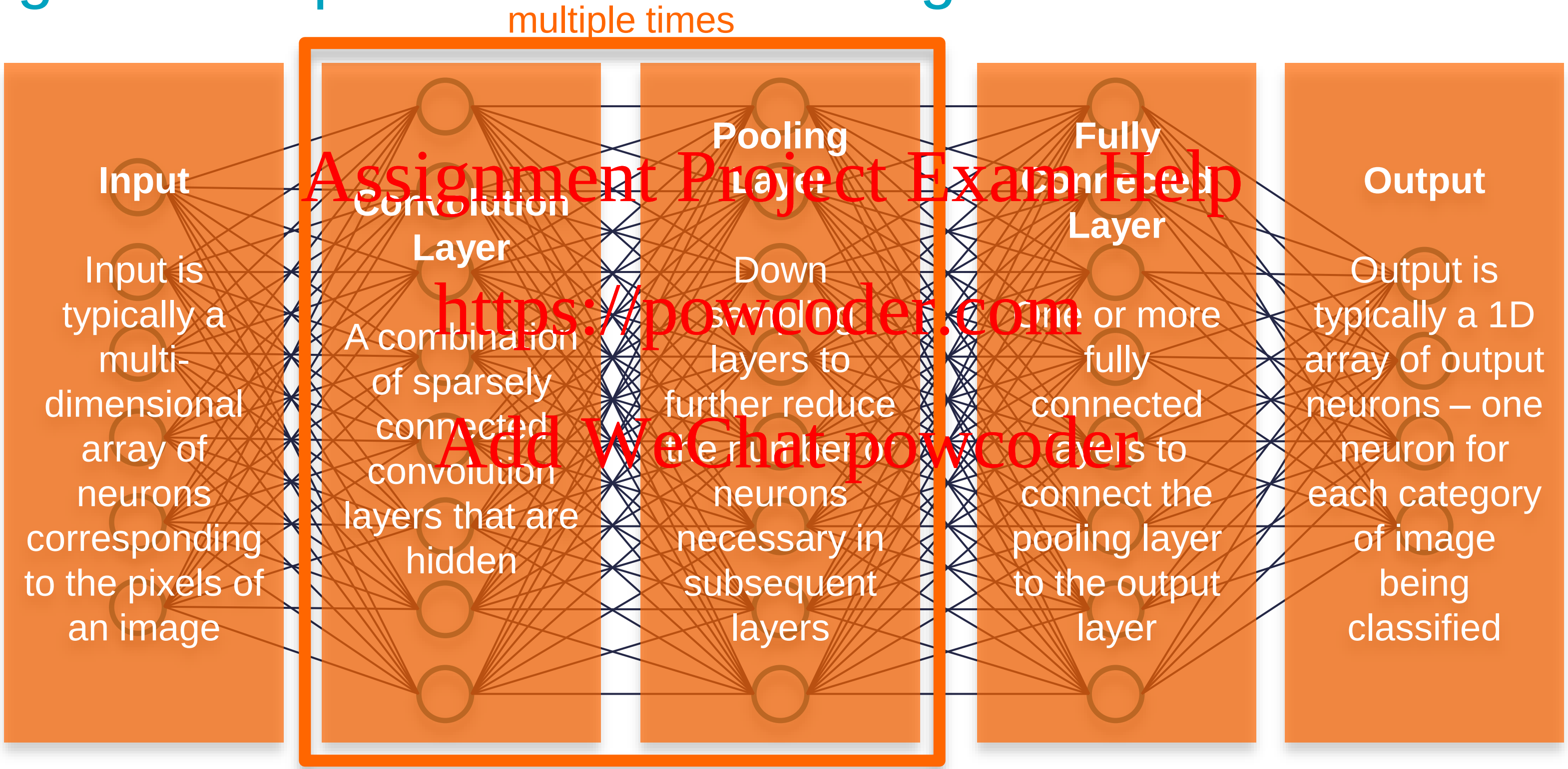


The increase in complexity is not linear to the number of perceptrons added

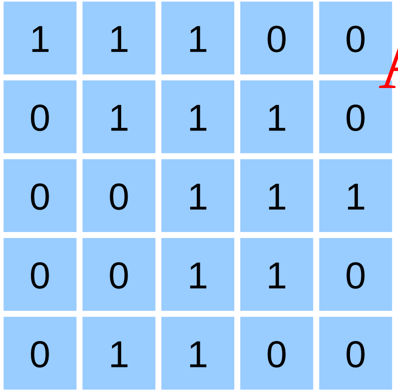
Increase in width and depth leads to an explosion in complexity and training time for large neural networks

Convolutional Neural Networks (CNN)

A CNN is a type of deep neural network architecture designed for specific tasks like image classification



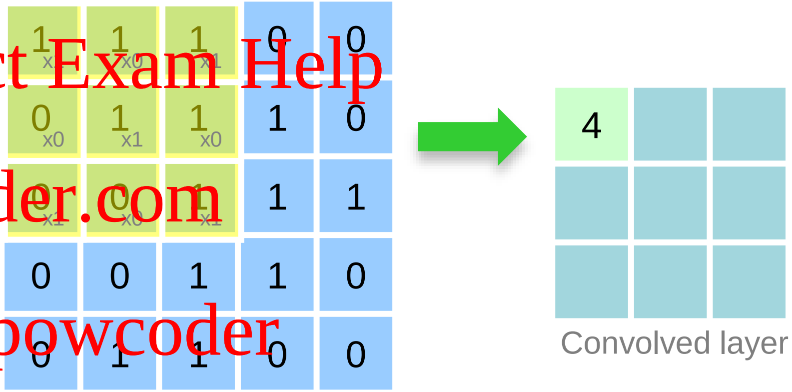
Convolution is a technique to extract visual features from an image in small chunks



Input layer

Input image is represented as a matrix of neurons

A filter (or kernel) extracts feature over a region in the image defined by its dimensions (as a bounding box)

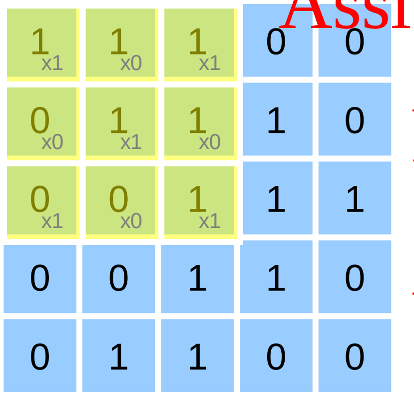


Convolved layer

Conceptually, the filter will move across the image and perform mathematical operations on individual regions of the image.

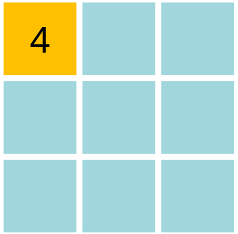
Each neuron in a convolution layer is responsible for a small cluster of neurons in the preceding layer

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix



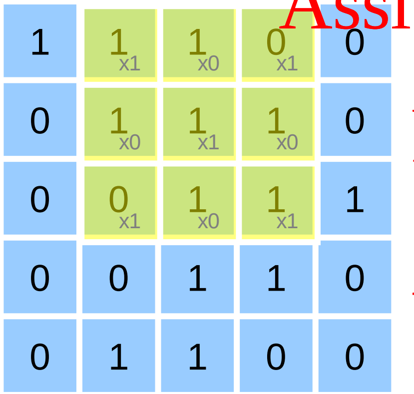
Convolution over the input image

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 4$$



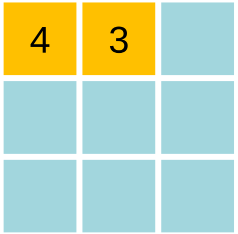
Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix



Convolution over the input image

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 3$$



Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix

1	1	1 x1	0 x0	0 x1
0	1	1 x0	1 x1	0 x0
0	0	1 x1	1 x0	1 x1
0	0	1	1	0
0	1	1	0	0

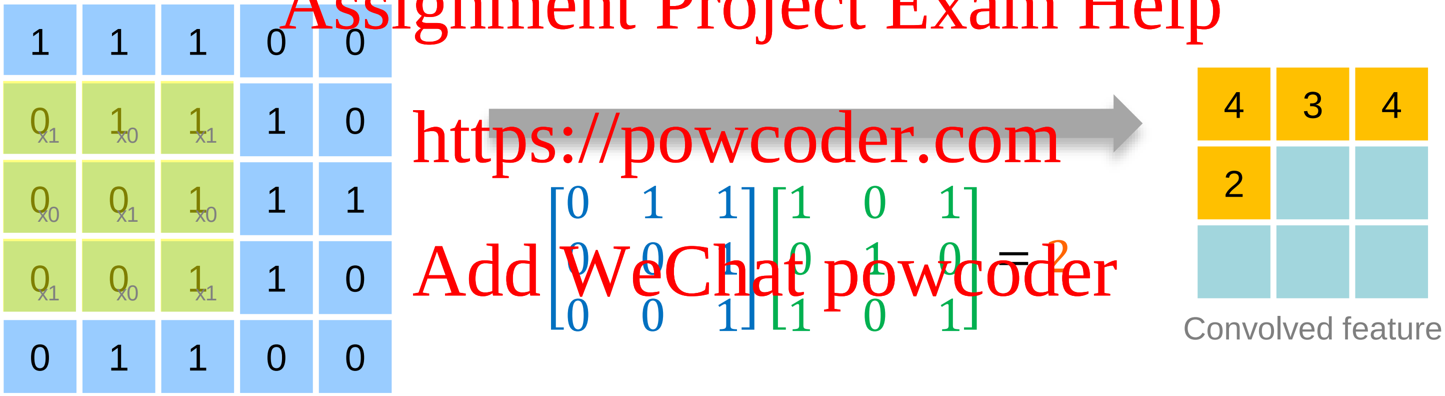
Convolution over the input image

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 4$$

4	3	4

Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix

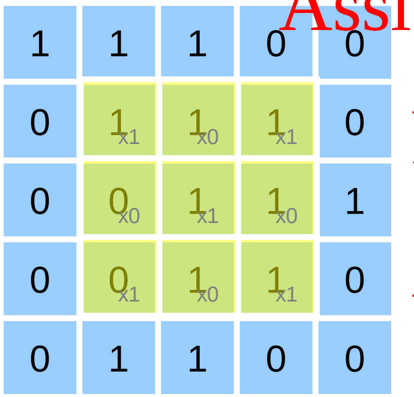


$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 2$$

Convolution over the input image

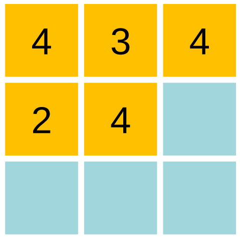
Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix



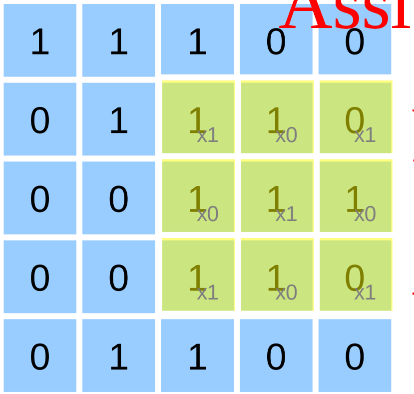
Convolution over the input image

$$\begin{bmatrix}1 & 1 & 1\\0 & 1 & 1\\0 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 1\\0 & 1 & 0\\1 & 0 & 1\end{bmatrix} = 4$$



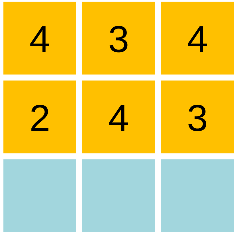
Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix



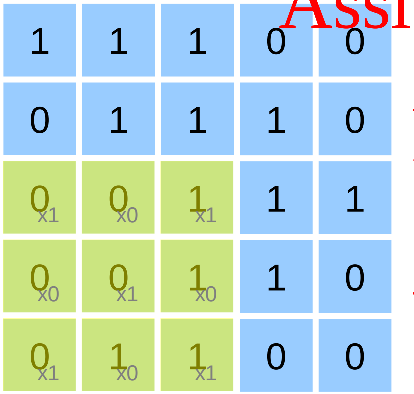
Convolution over the input image

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 3$$



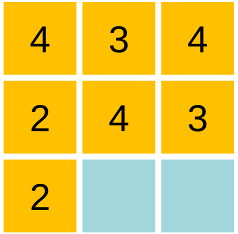
Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix



Convolution over the input image

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 2$$



Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix

1	1	1	0	0
0	1	1	1	0
0	0_{x1}	1_{x0}	1_{x1}	1
0	0_{x0}	1_{x1}	1_{x0}	0
0	1_{x1}	1_{x0}	0_{x1}	0

Convolution over the input image

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 3$$

4	3	4
2	4	3
2	3	

Convolved feature

Convolution feature is obtained by multiplying image values bounded by the filter by the convolution matrix

1	1	1	0	0
0	1	1	1	0
0	0	1 ×1	1 ×0	1 ×1
0	0	1 ×0	1 ×1	0 ×0
0	1	1 ×1	0 ×0	0 ×1

Convolution over the input image

$$\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 0\\ 1 & 0 & 0\end{bmatrix}\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 1\end{bmatrix} = 4$$

4	3	4
2	4	3
2	3	4

Convolved feature

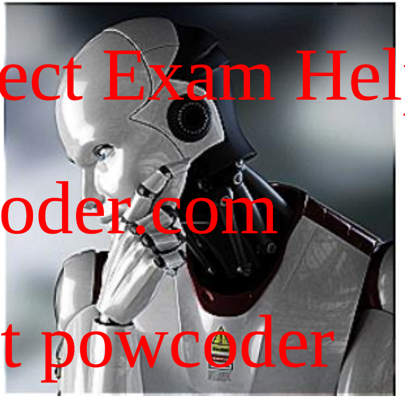
Filters mathematically modify the input of a convolution to help detect certain types of features in the image

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

Identity

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Blur



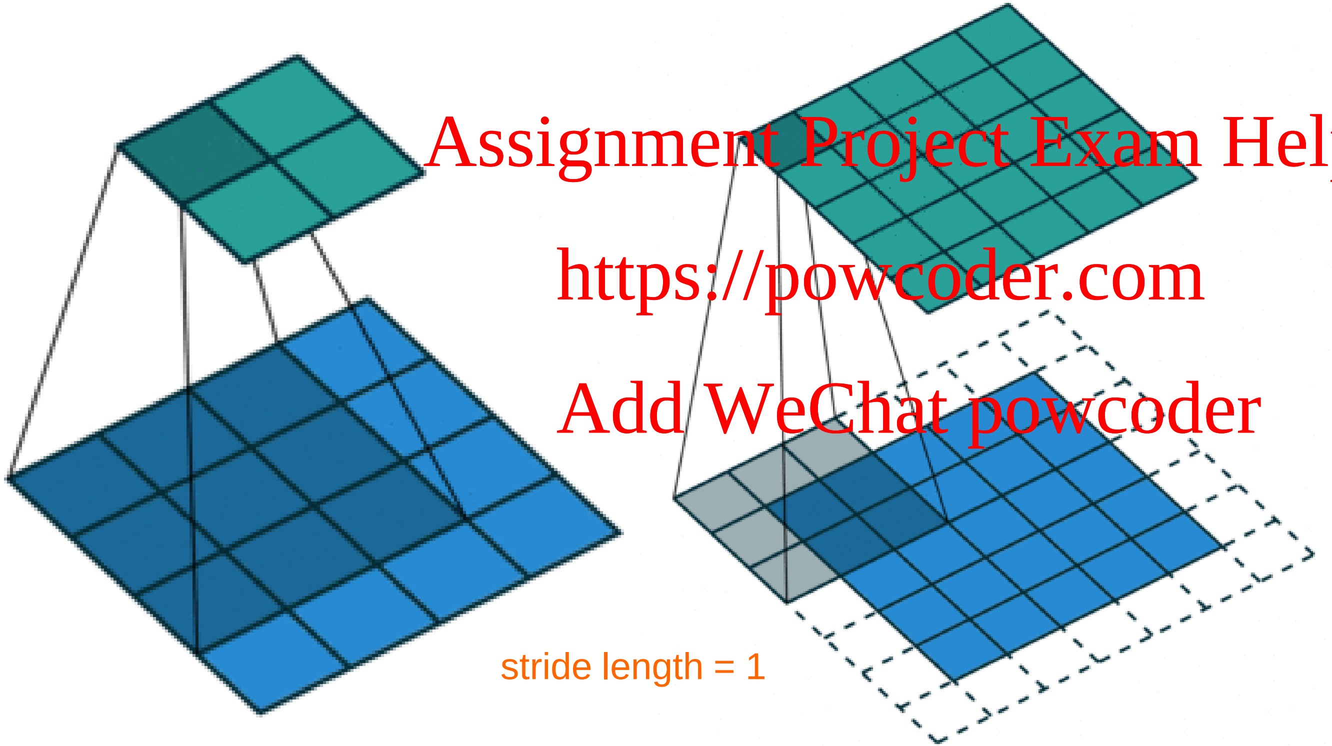
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

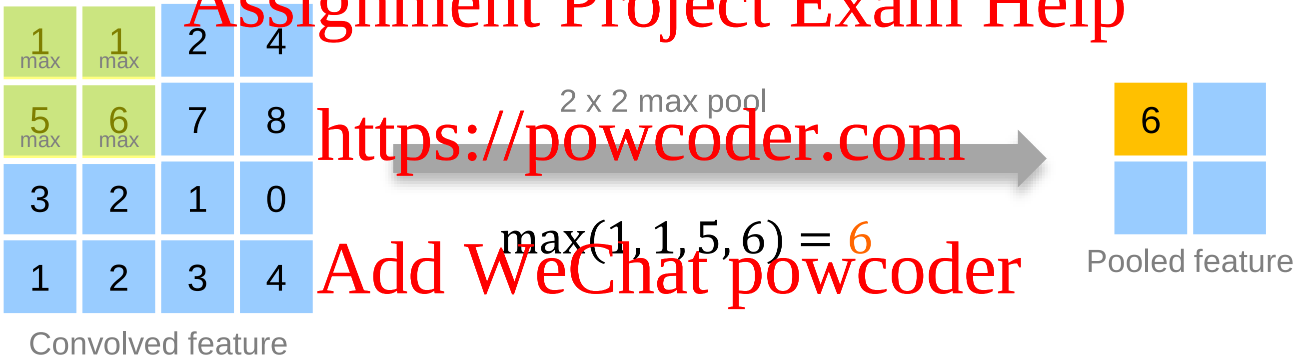
Edge

The edges of an image can be padded with 0-valued pixels to fully scan the original image and preserve its dimensions

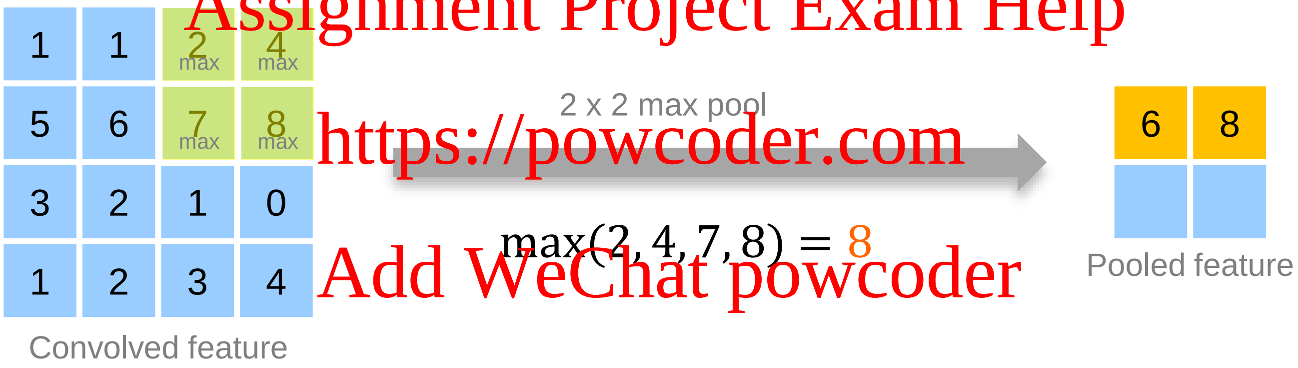


- In practice, we don't explicitly define the filters that our convolutional layer will use
- We instead parameterize the filters and let the network learn the best filters to use during training

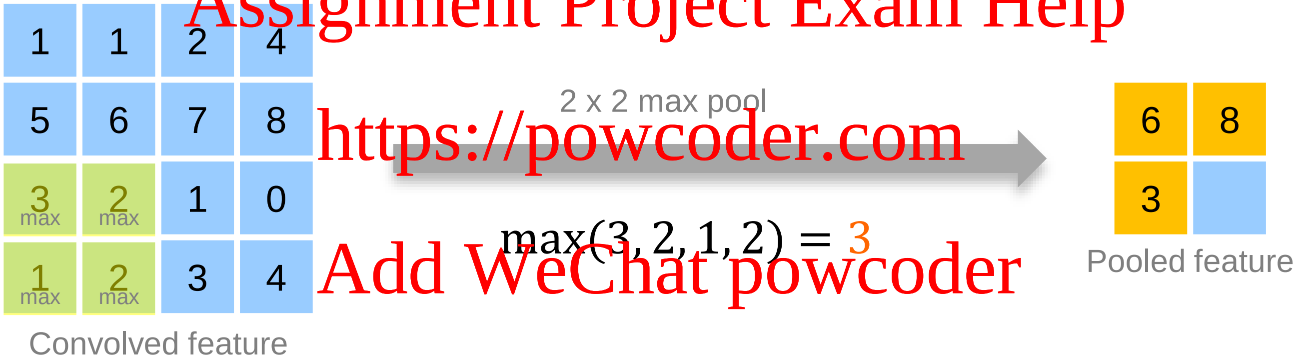
Pooling reduces the number of neurons in the previous convolution layer while retaining the most important info



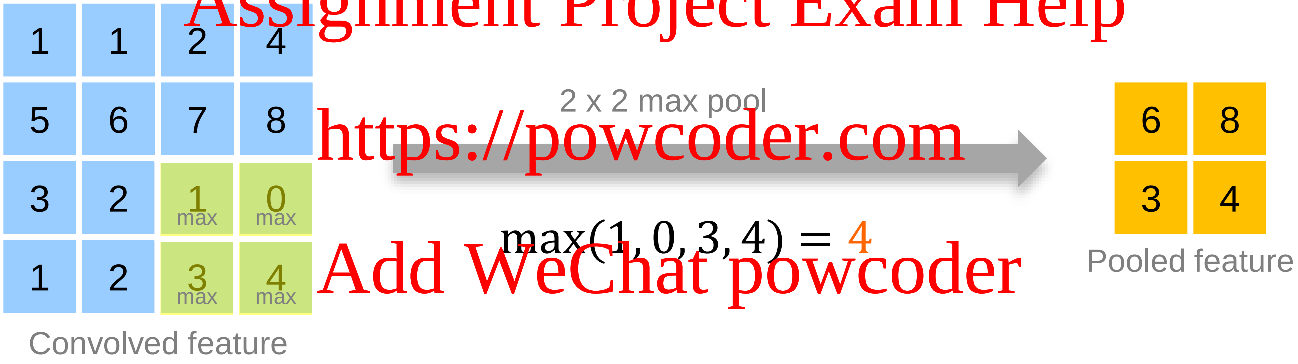
Pooling reduces the number of neurons in the previous convolution layer while retaining the most important info



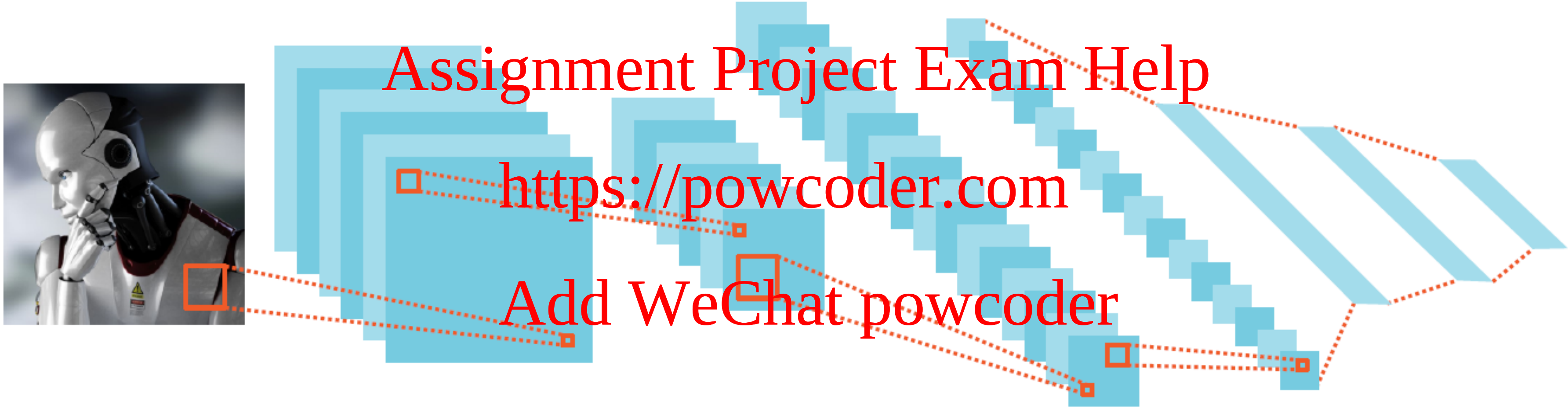
Pooling reduces the number of neurons in the previous convolution layer while retaining the most important info



Pooling reduces the number of neurons in the previous convolution layer while retaining the most important info



CNNs work well for a variety of tasks including image recognition, image processing, image segmentation, video analysis, and natural language processing



input image

a convolution layer with multiple filters with a matrix output for each filter

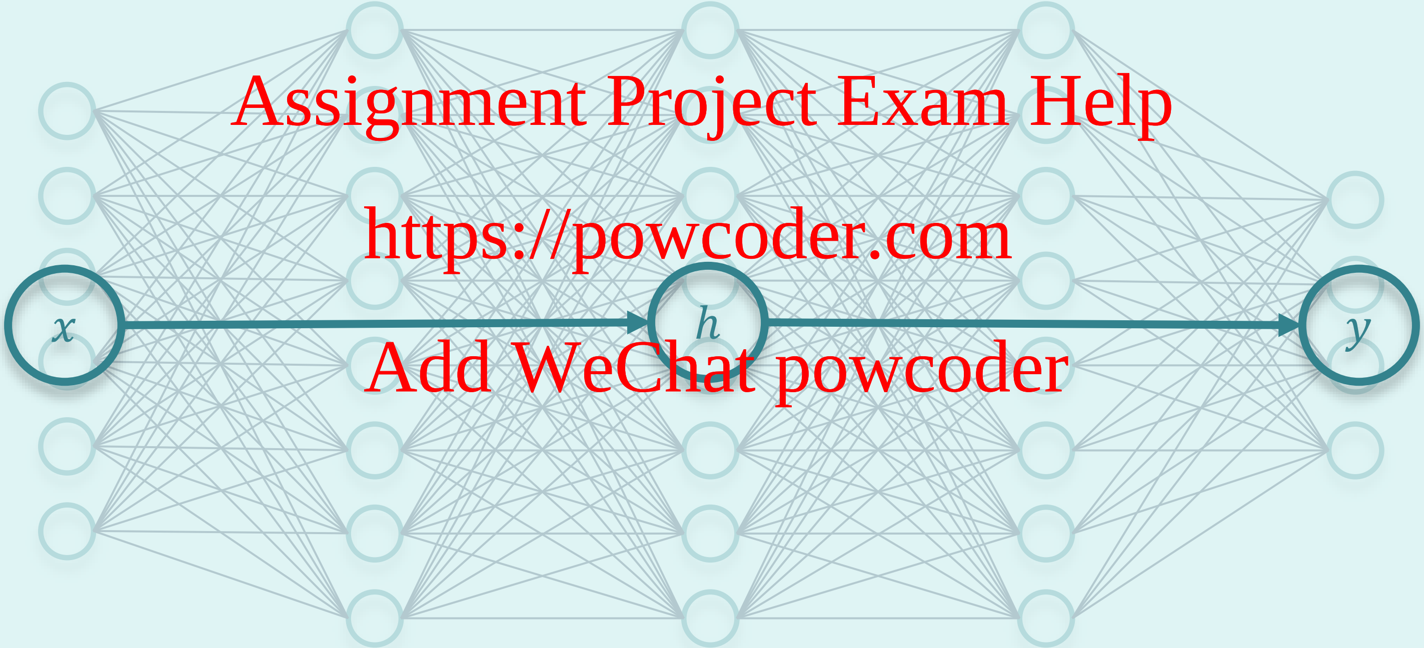
a pooling layer produces a down sample feature matrix for each convolution filter

repeat the convolution and pooling steps multiple times using previous features as input

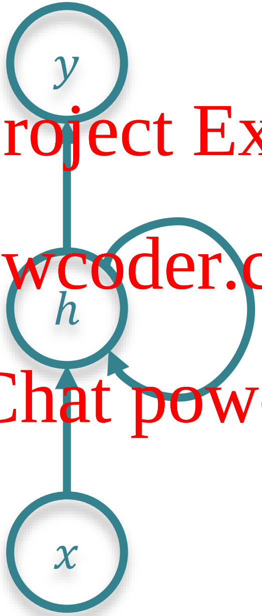
A few fully connected layers to classify the image and produces the classification prediction

Recurrent Neural Networks (RNN)

In a feed-forward neural network, input x flows through one or more hidden layers of neurons h, to the output y

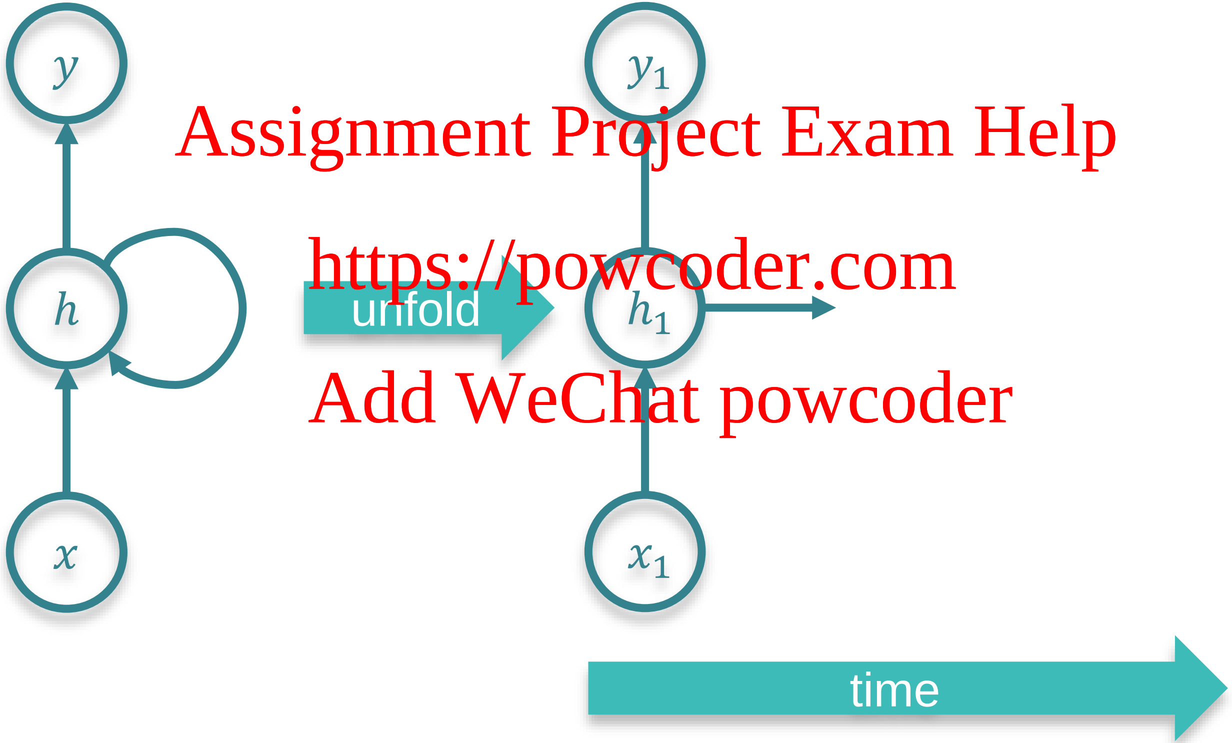


Unlike feed-forward neural networks, RNN's contain feedback loops

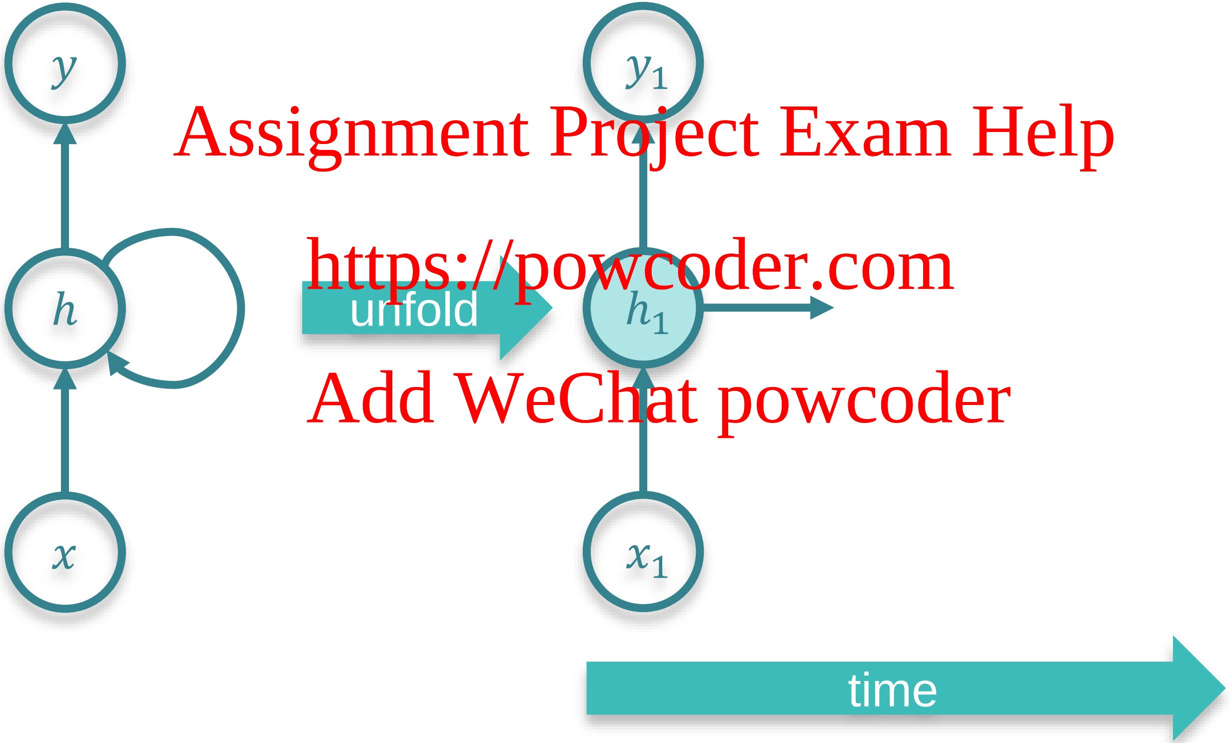


Can operate effectively on sequences of data with variable input length

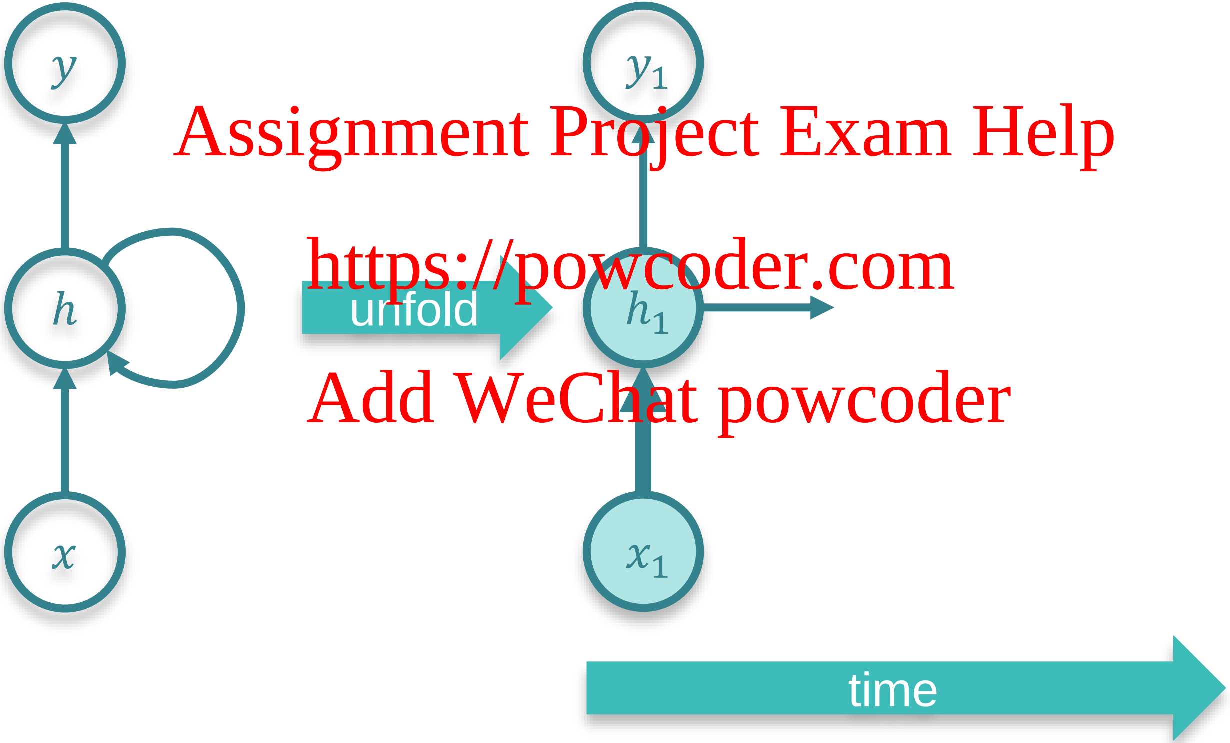
Observing the RNN over time, the hidden node h_1 uses the input x_1 to produce the output y_1



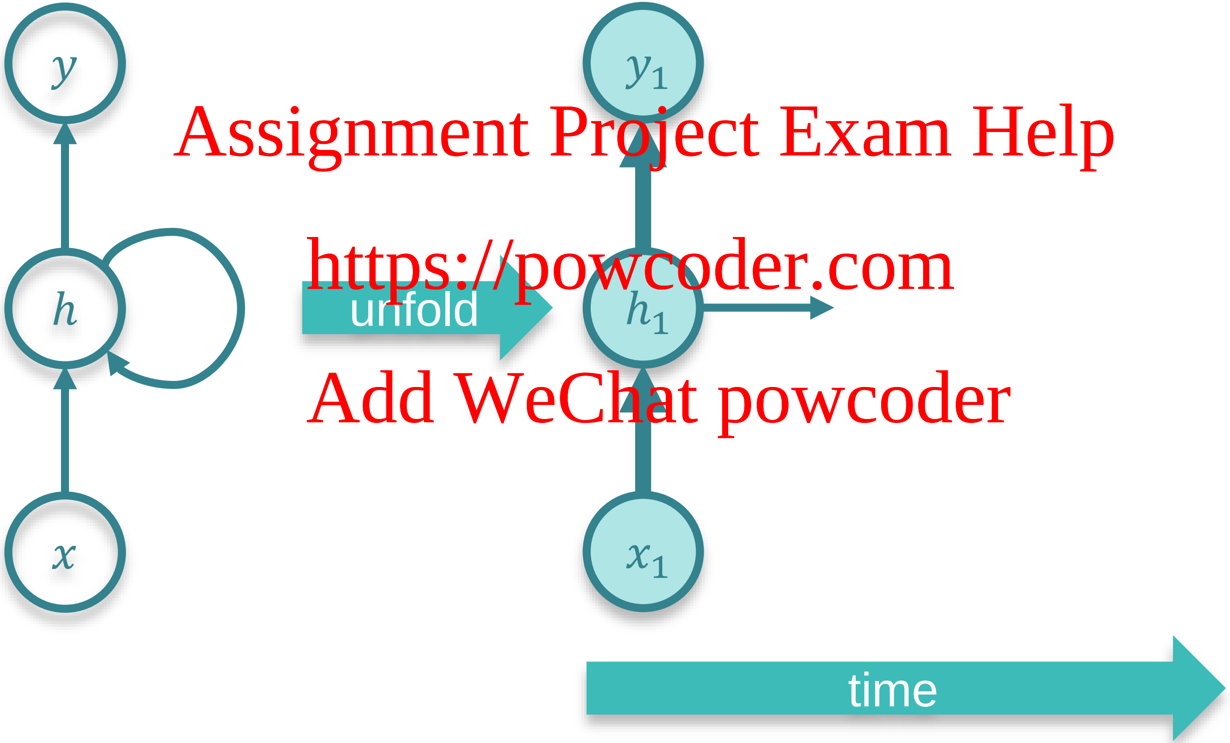
Observing the RNN over time, the hidden node h_1 uses the input x_1 to produce the output y_1



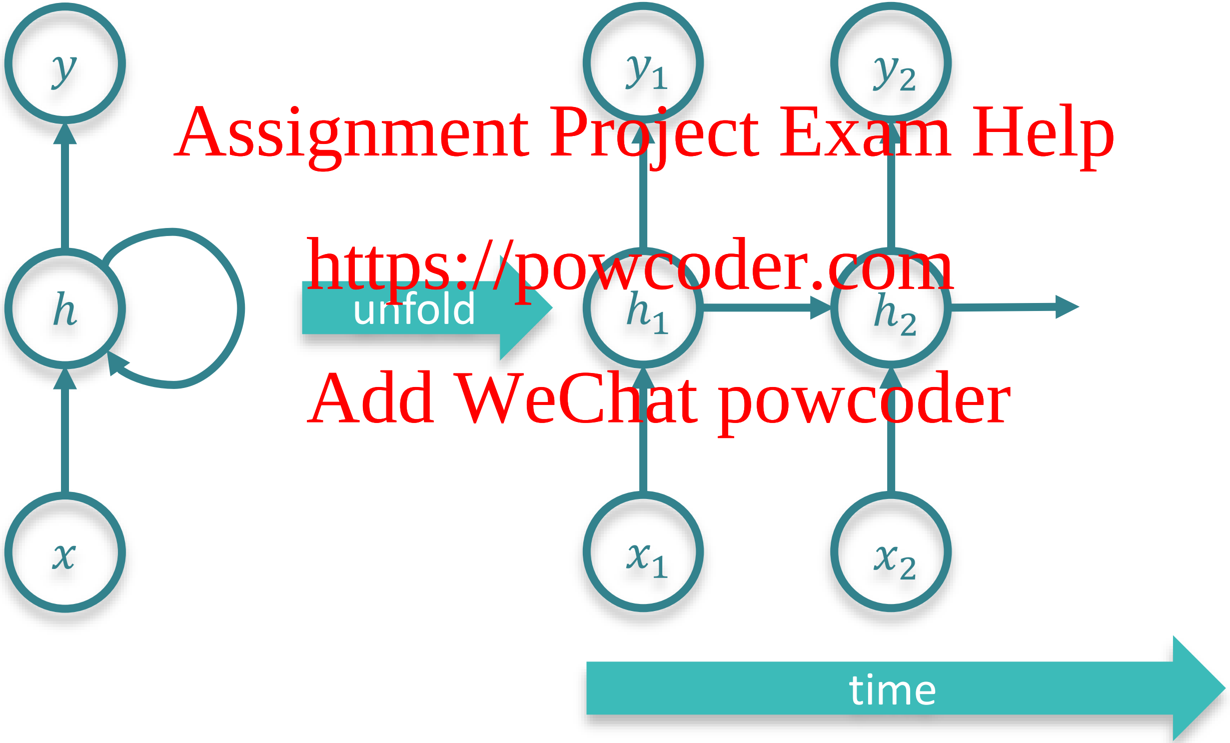
Observing the RNN over time, the hidden node h_1 uses the input x_1 to produce the output y_1



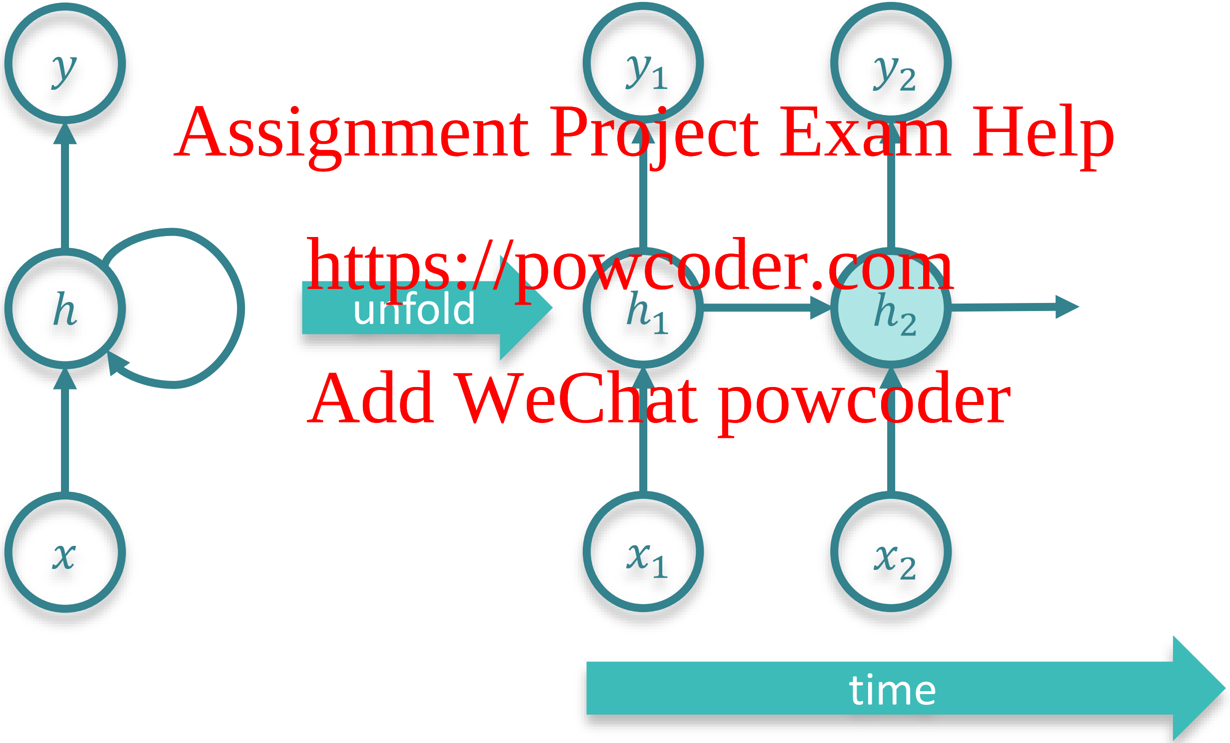
Observing the RNN over time, the hidden node h_1 uses the input x_1 to produce the output y_1



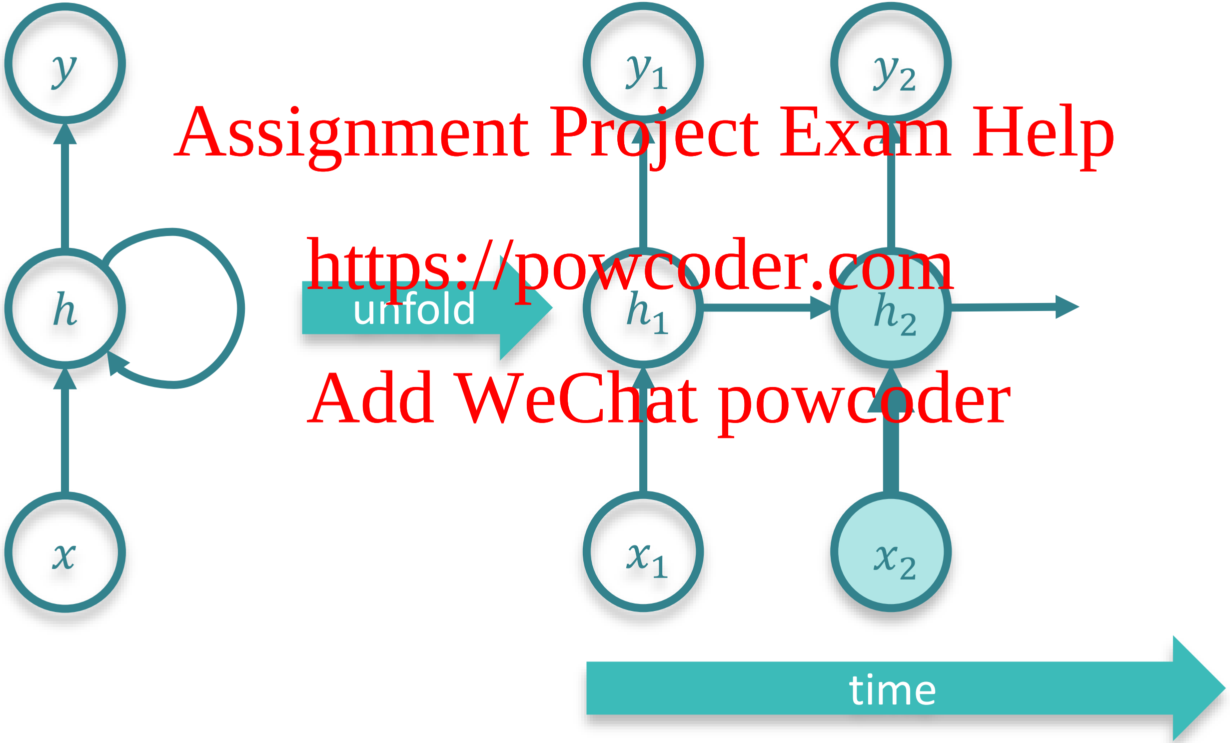
RNN uses knowledge of its previous state as an input for its current prediction giving the network a short-term memory



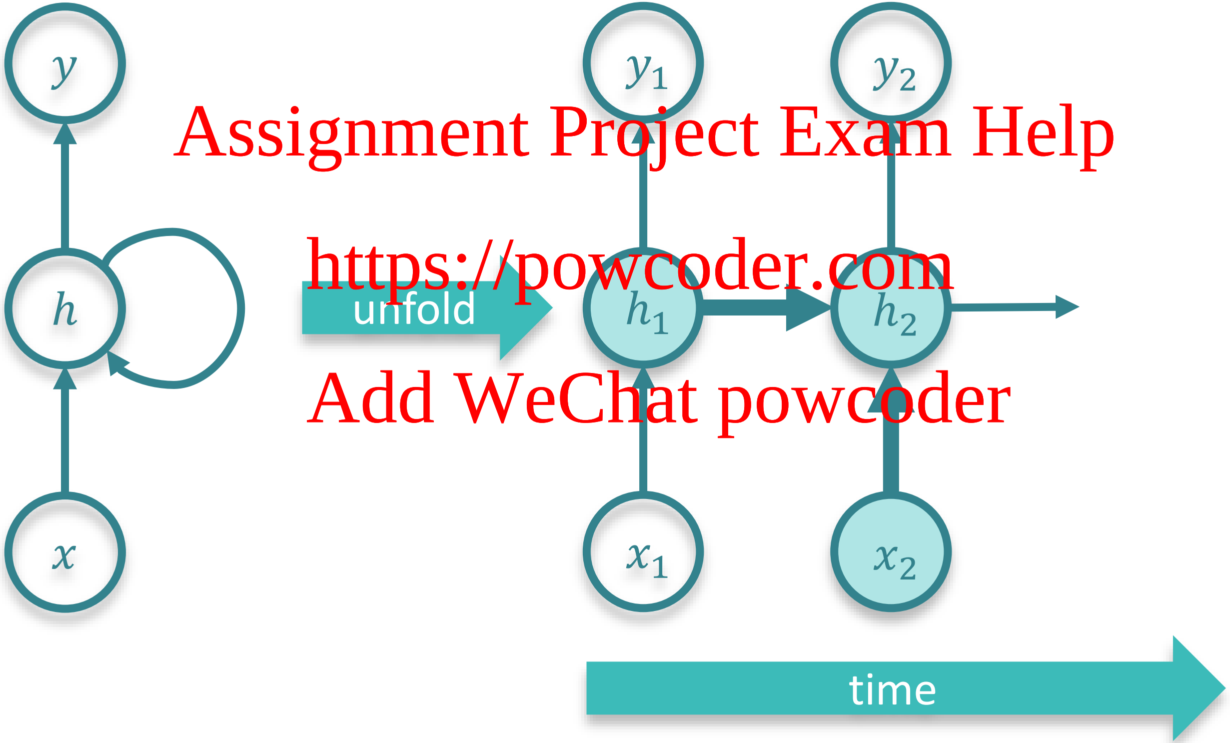
RNN uses knowledge of its previous state as an input for its current prediction giving the network a short-term memory



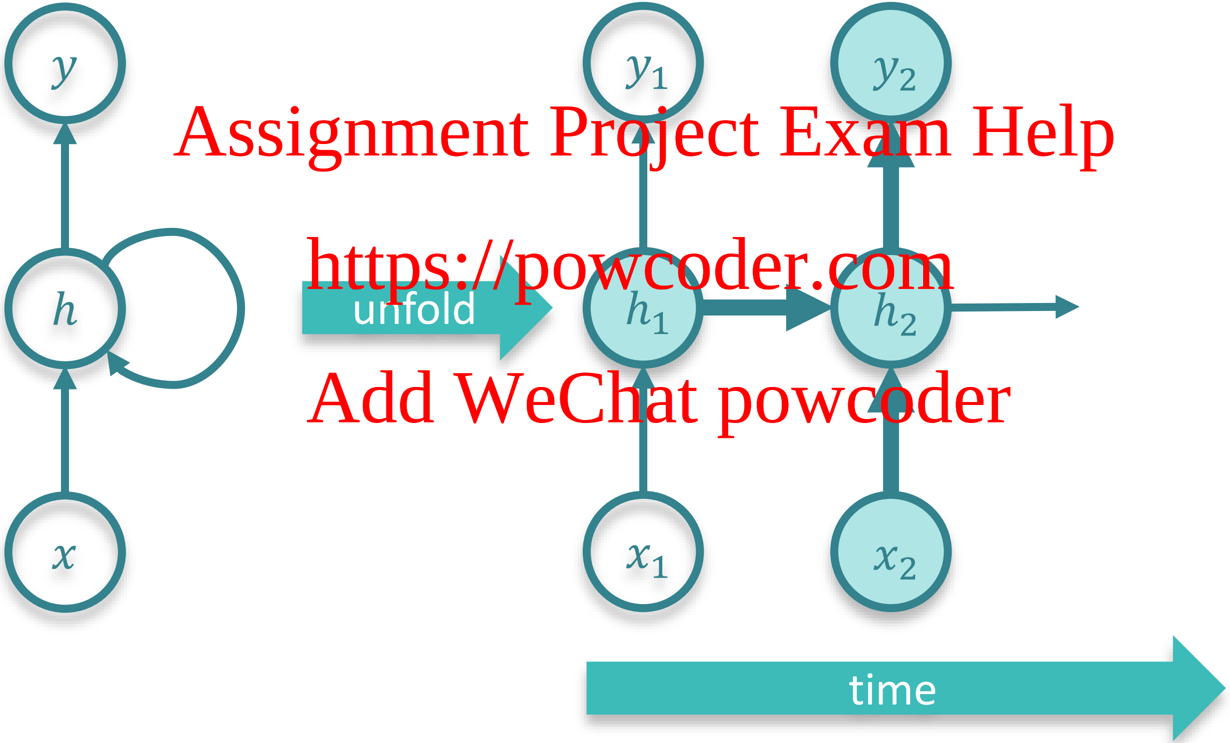
RNN uses knowledge of its previous state as an input for its current prediction giving the network a short-term memory



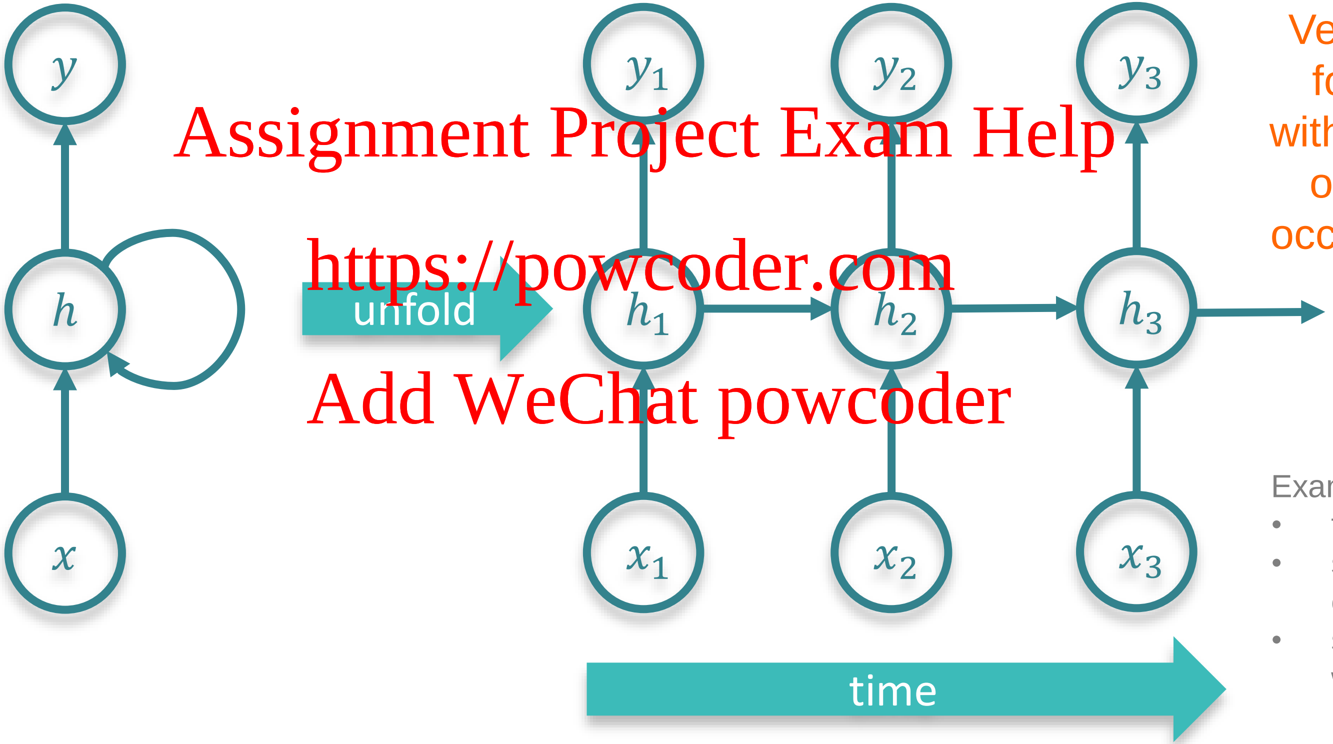
RNN uses knowledge of its previous state as an input for its current prediction giving the network a short-term memory



RNN uses knowledge of its previous state as an input for its current prediction giving the network a short-term memory



The process of using previous state in current prediction can be repeated an arbitrary number of times



Very effective for working with sequences of data that occur over time

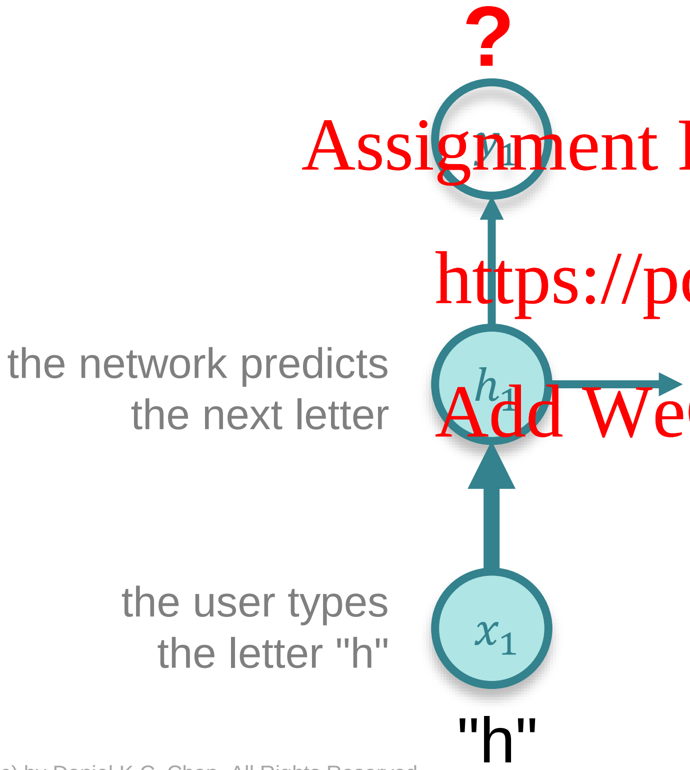
Examples:

- time-series data
- sequence of characters
- sequence of words

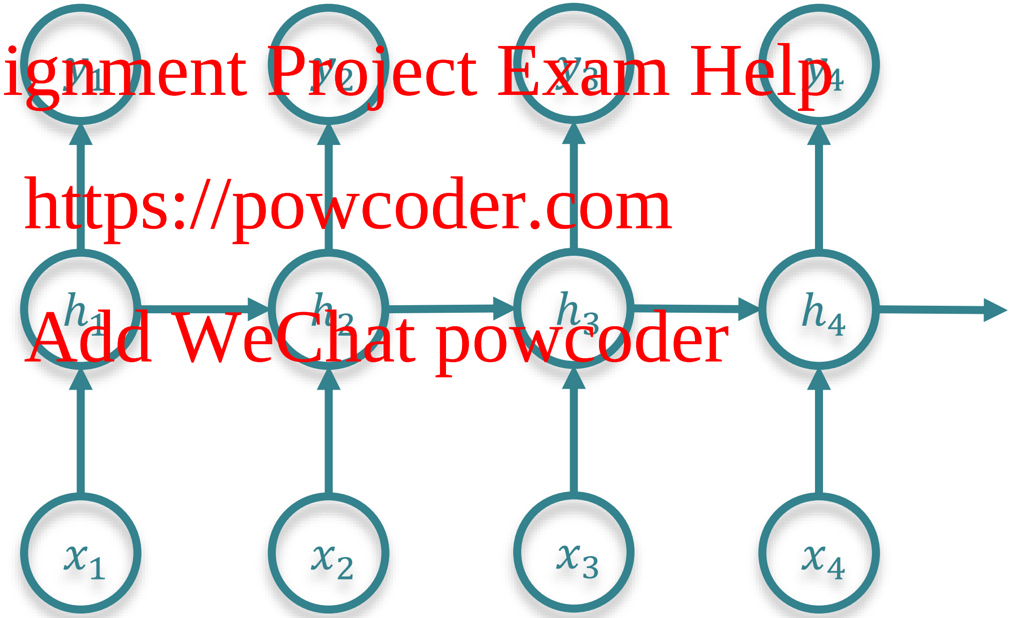
RNN Applications

- Time-series data
  - Changes in stock prices
- Natural language processing
- Speech recognition
- Language translation
- Conversion modelling
- Image captioning
- Visual Q&A

Predicting the next letter a person is likely to type, the letter just typed and all previous typed letters are important



Predicting the next letter a person is likely to type, the letter just typed and all previous typed letters are important



Predicting the next letter a person is likely to type, the letter just typed and all previous typed letters are important

"i" is predicted based on previous training examples that included the word "hi"

the network predicts the next letter

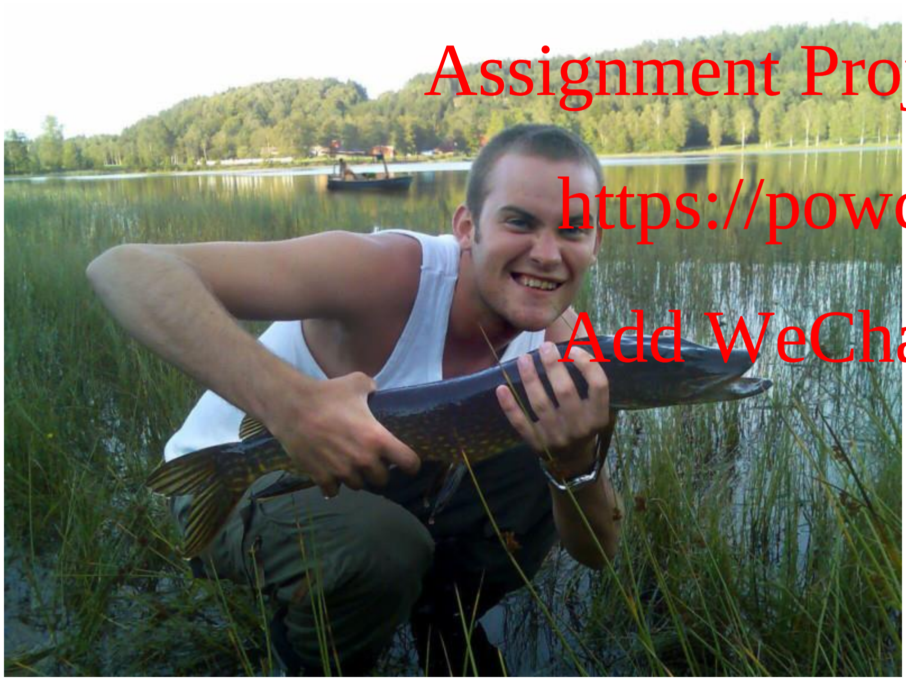
the user types the letter "h"

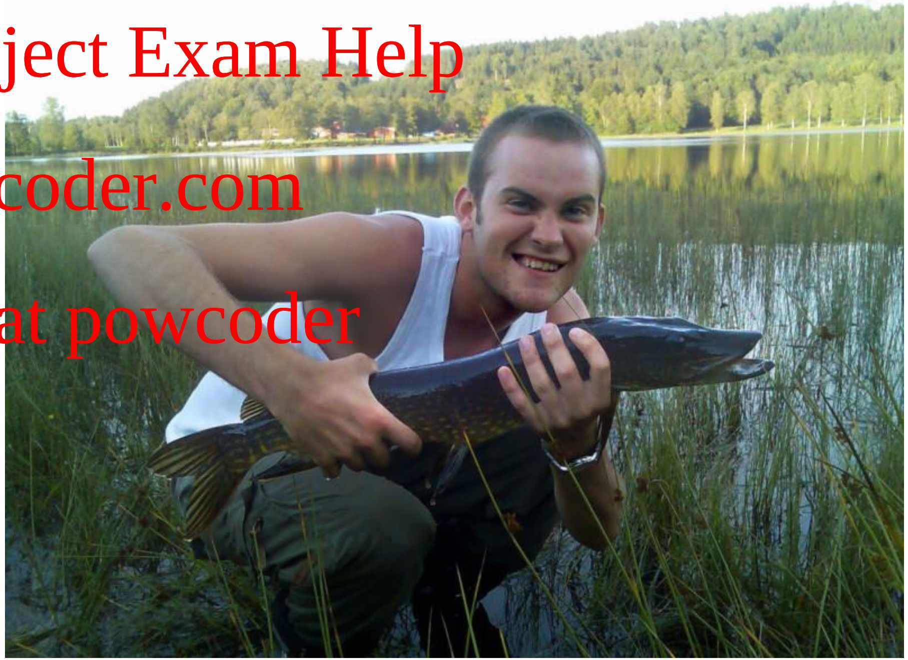
Generative Adversarial Networks (GAN)

Which photo is fake?

Which photo is fake?

Which photo is fake?





Is this fake?

Is this fake?

Is this fake?

Is this fake?

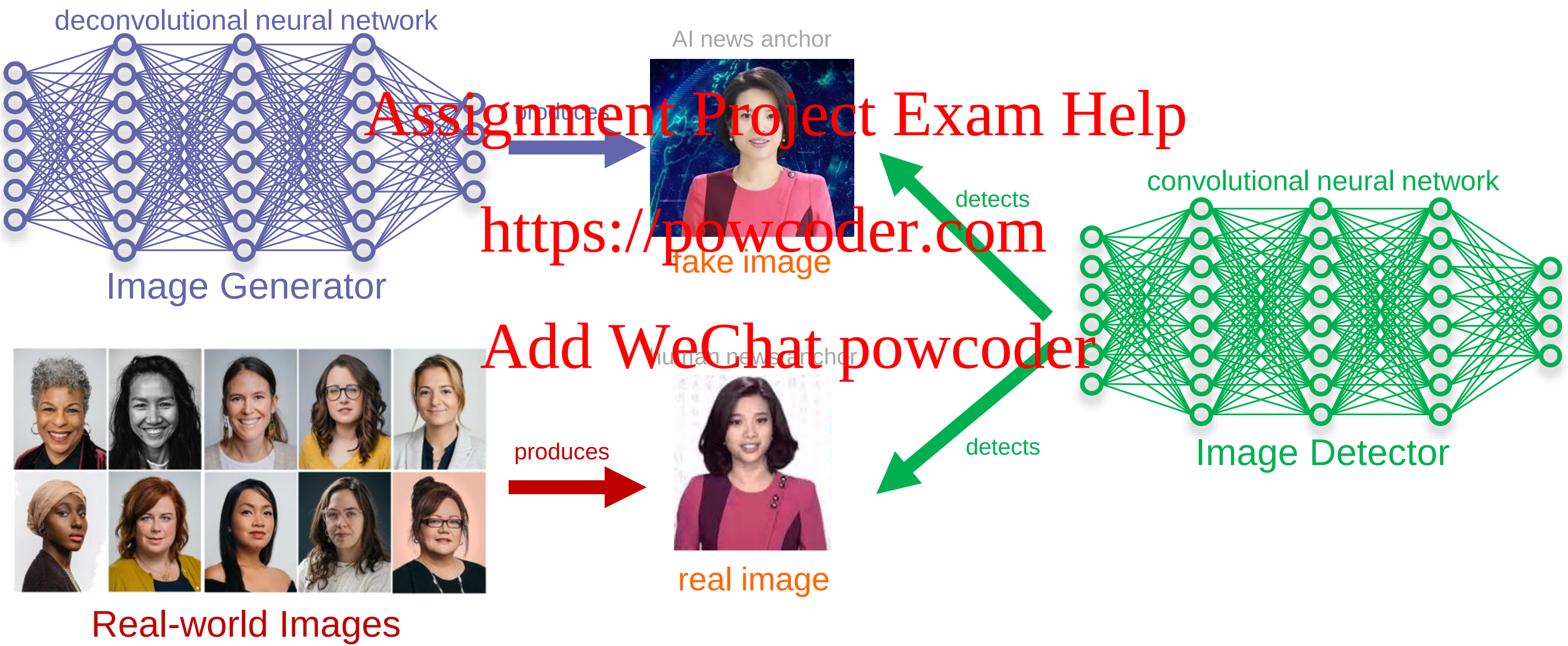
https://www.youtube.com/watch?v=9NSVdQfD0K8&feature=youtu.be

Is this fake?

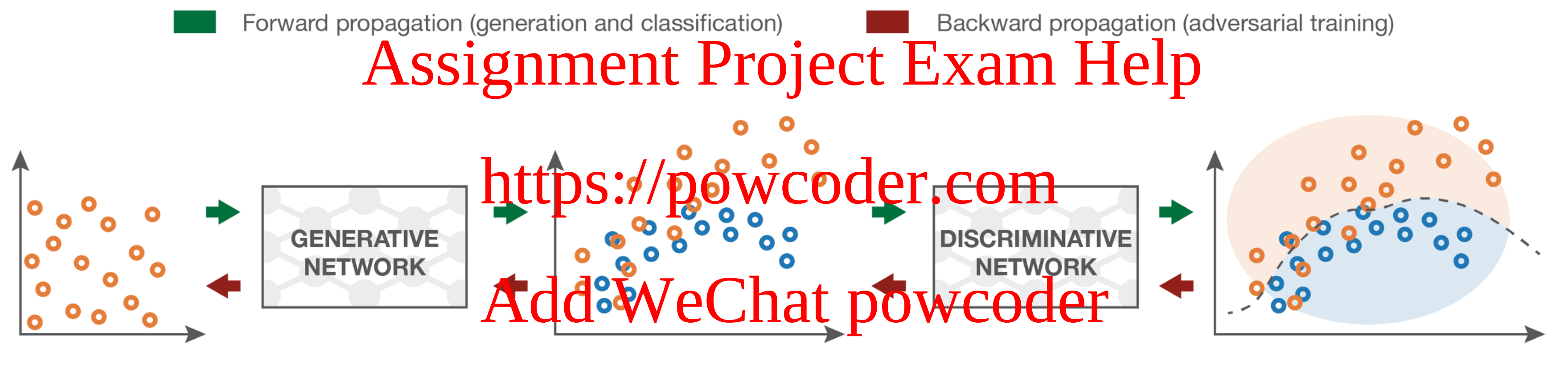
The Generative Adversarial Network (GAN) is a combination of two deep learning neural networks

The two networks are adversaries in the sense that they are both competing to beat one another

Generative Adversarial Network (GAN) can be deployed for image generation and image enhancement



The generator & discriminator networks can be trained jointly as a max-min game



Input random variables.

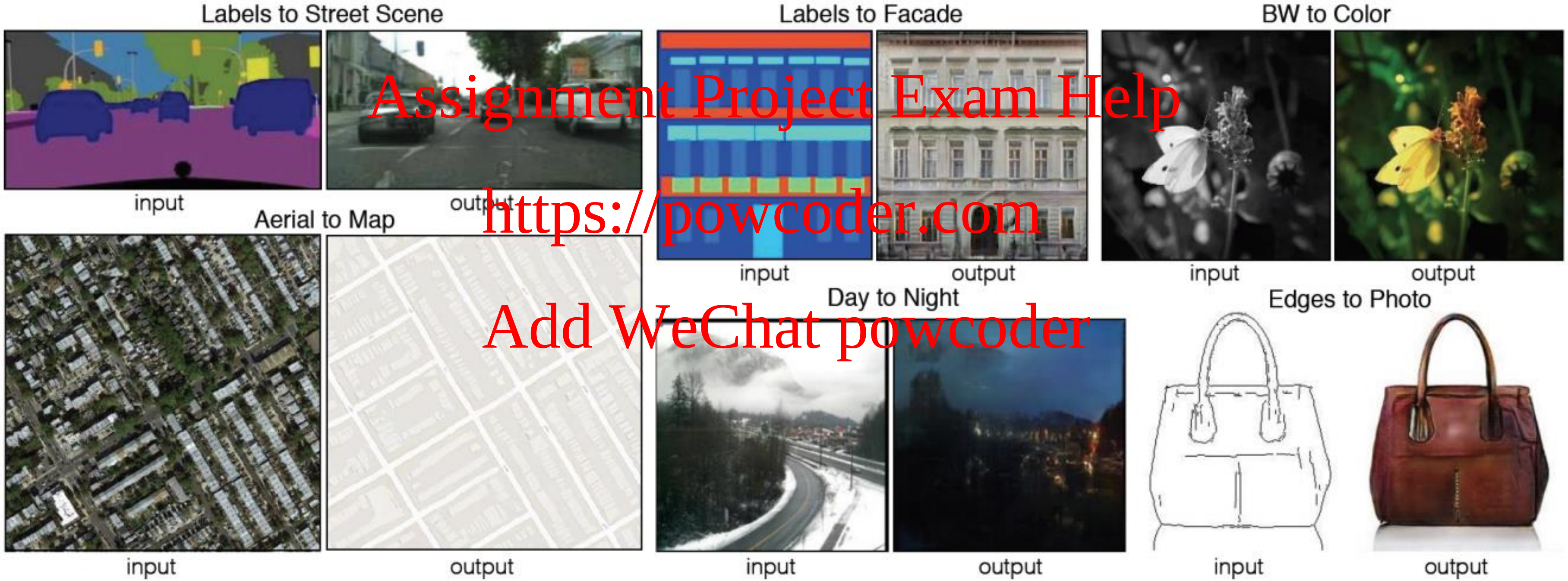
The generative network is trained to **maximise** the final classification error.

The generated distribution and the true distribution are not compared directly.

The discriminative network is trained to **minimise** the final classification error.

The classification error is the basis metric for the training of both networks.

GAN Applications: Image-to-Image Translation



GAN Applications: Text-to-Image Synthesis

This small bird has a pink breast and crown, and black primaries and secondaries

This magnificent fellow is almost all black with a red crest and while cheek patch

The flower has petals that are bright pinkish purple with white stigma

This white and yellow flower have thin white petals and a round yellow stamen

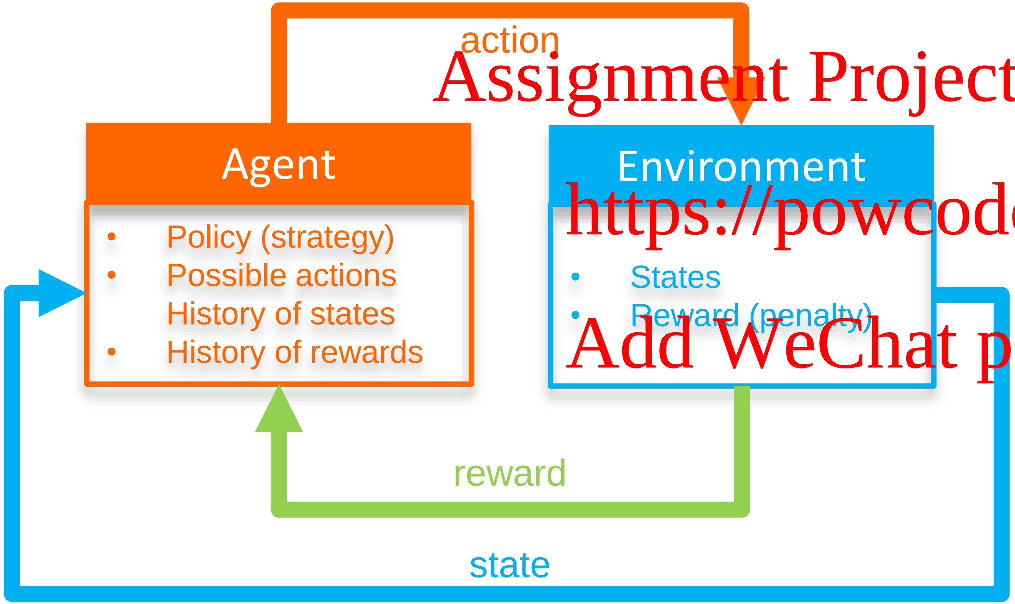
GAN Applications: Video & Speech Synthesis

https://www.youtube.com/watch?v=o2DDU4g0PRo

https://www.youtube.com/watch?v=HJcdVjkqiW8

Deep Reinforcement Learning

Reinforcement learning is a general purpose framework for making optimal decisions using experiences



- An agent has the capacity to act
- Each action influences the agent's future state
- Success is measured by a scalar reward signal
- The goal is to select actions to maximise future reward

A self-driving taxi operates in an area that can be represented by a 5x5 grid with 4 pick-up/drop-off locations

- The State Space (environment) is the set of all possible situations the self-driving taxi (agent) could inhabit
- The state should contain useful information the agent needs to make the right action
- The layout on the left will be used as the training environment where the agent will transport people to four different locations: R (0,0), G (0,4), Y (4,0), B (4,3)

The self-driving taxi is to take a passenger safely from the pick-up to the drop-off point in the minimum time possible

- Assumes that the agent is the only vehicle in the environment
- The current location of the agent is the position having coordinate (3,1)
- Passenger at Y wishes to go to location R
- Total number possible states is 500
  - Passenger at one of the 4 pick-up points
  - Passenger inside the taxi
  - Taxi at 25 (5x5) possible locations
  - 4 possible drop-off points

The self-driving taxi has 6 possible actions: moving the taxi in one of the four directions or pick-up/drop-off passenger

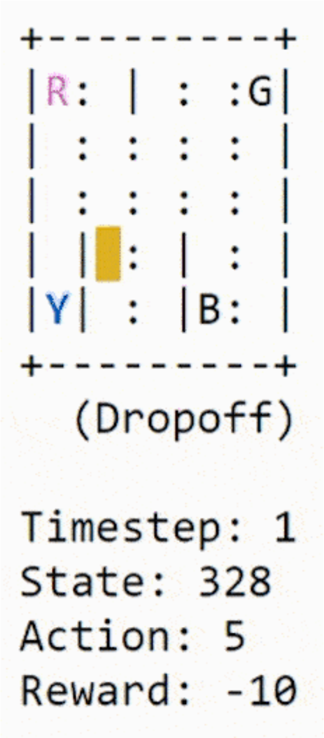
- The agent encounters one of the 500 states and takes an action
- The action space contains 6 possible actions
  - move south (action number 0)
  - move north (action number 1)
  - move east (action number 2)
  - move west (action number 3)
  - pick-up (action number 4)
  - drop-off (action number 5)

The self-driving taxi is reward-motivated and will navigate by trial experiences that come with rewards or penalties

- A high positive reward (e.g. +20 points) for a successful drop-off because this behavior is highly desirable
- A penalty (e.g. -10 points) if it tries to drop-off a passenger in wrong locations
- A slight negative reward (e.g. -1 point) for not making it to the destination after every time-step
    - Slight negative because it is preferred that the agent to reach late instead of making wrong moves

Reinforcement learning will learn a mapping of states to the optimal action to take in that state

- The agent explores the environment and takes actions based on rewards defined in the environment
- The optimal action for each state is the action that has the highest cumulative long-term reward

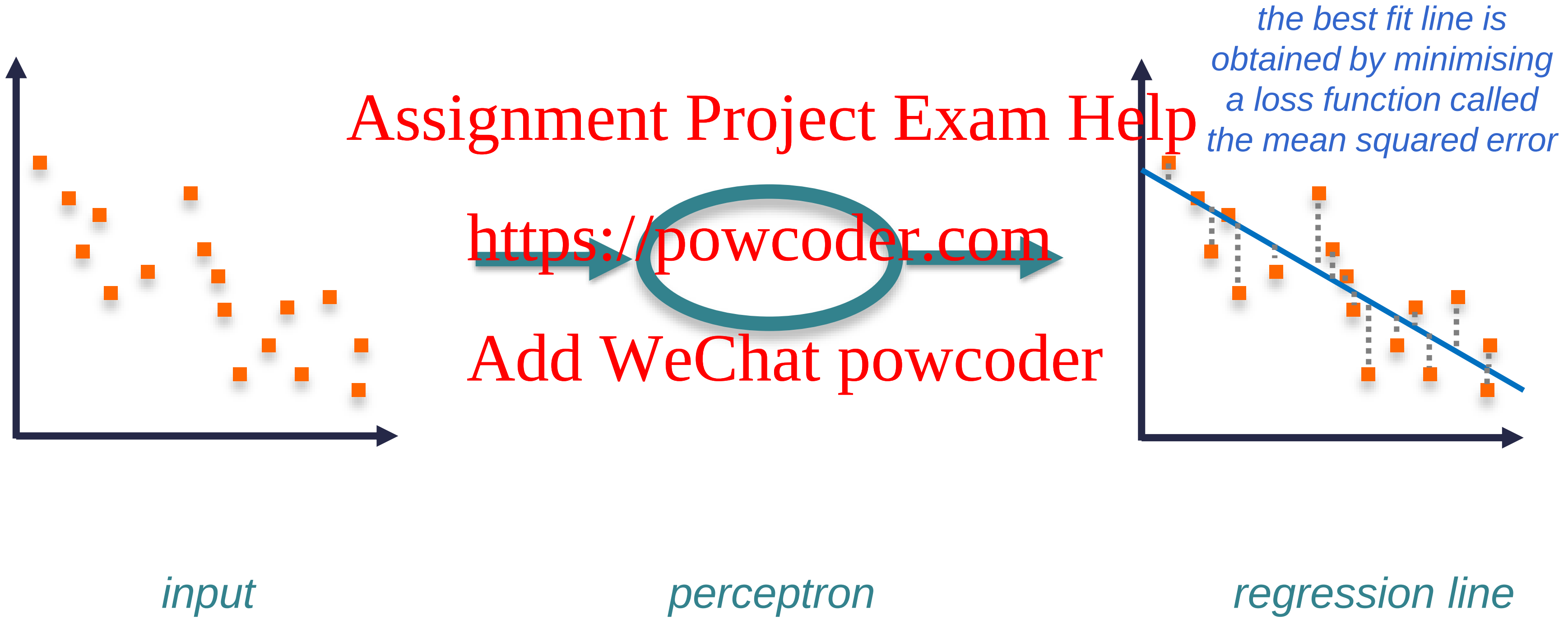


Deep Reinforcement Learning Applications

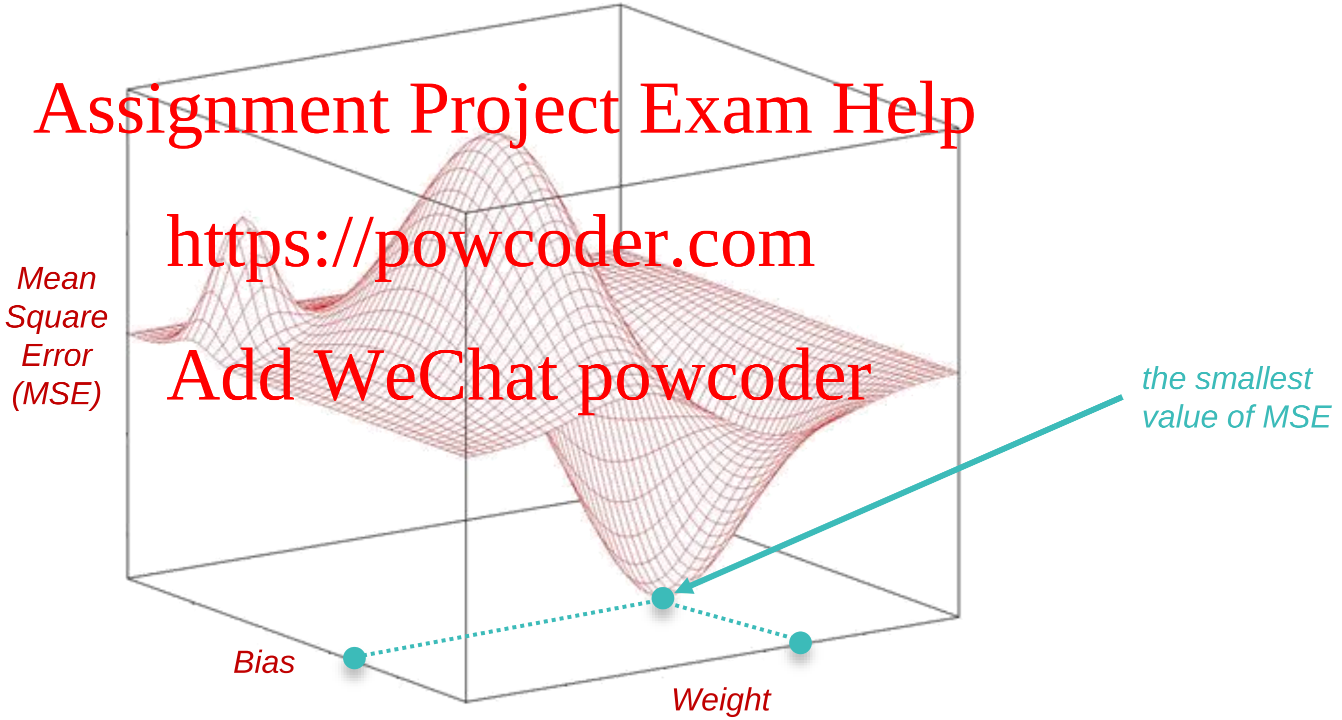
- Robotics for industrial automation
- Business strategy planning
- Machine learning and data processing
- It helps you to create training systems that provide custom instruction and materials according to the requirement of students
- Aircraft control and robot motion control

Gradient Descent Optimization

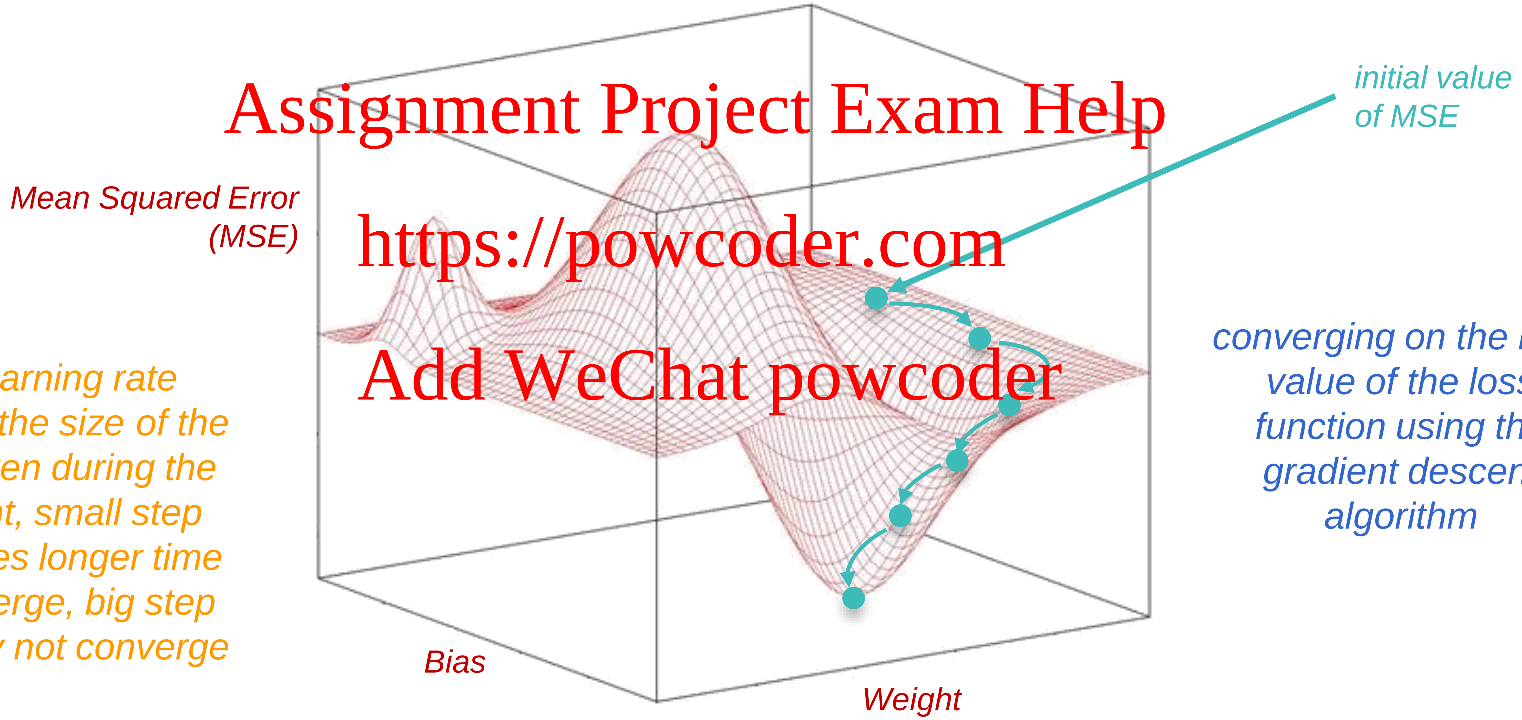
A linear regression problem using a single perceptron with the identity function being the activation function



The objective of the gradient descent optimization is to locate the smallest value of the loss function



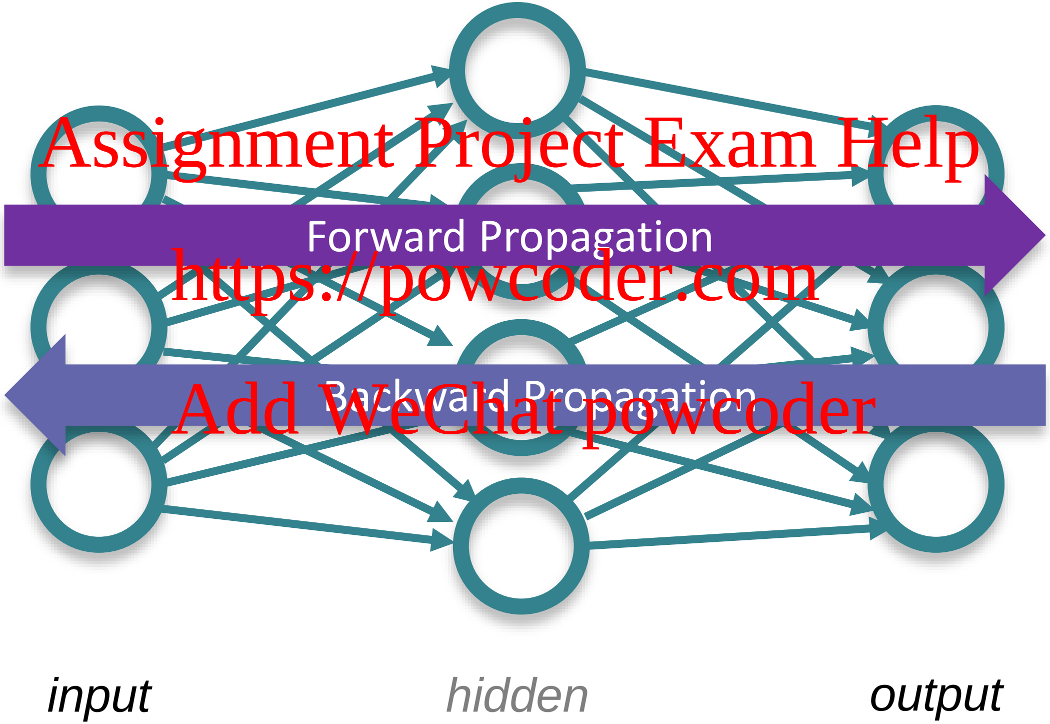
Starting with some random values for the parameters, the optimization walks down the surface towards the direction of the lowest or best possible loss function value



the learning rate controls the size of the step taken during the descent, small step size takes longer time to converge, big step size may not converge

converging on the best value of the loss function using the gradient descent algorithm

Backward propagation allows the weights and biases of the perceptrons to converge to their final values

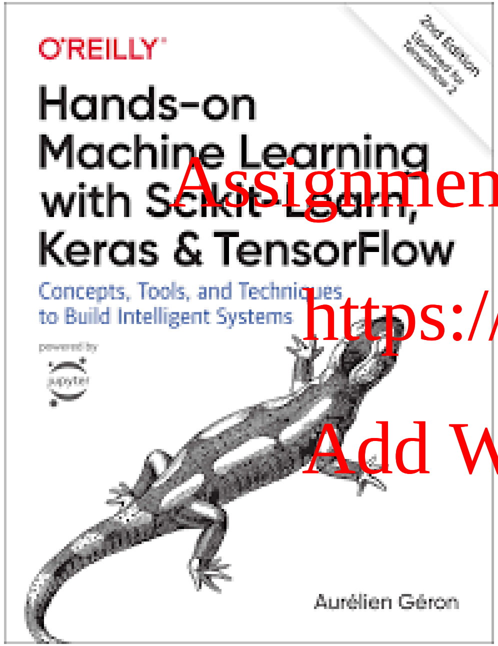


$$Error = Prediction - Actual$$

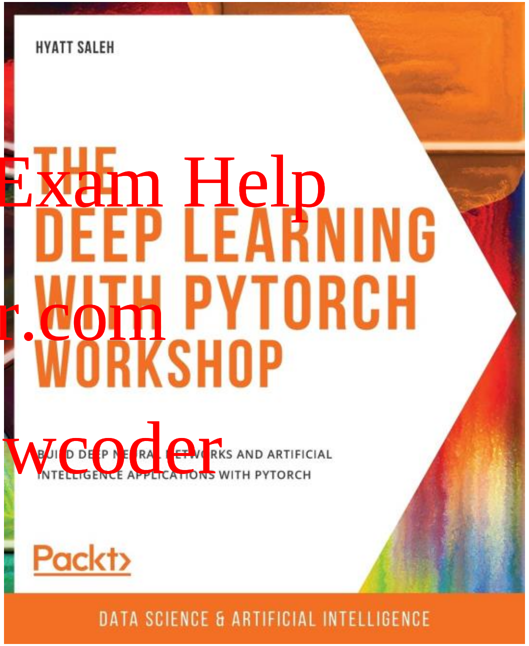
$$Optimization = Adjustment\ of\ Parameters\ using\ an\ Optimizer$$

References

References



"Hands-On Machine Learning with Scikit-Learn and TensorFlow", Aurelien Geron, O'Reilly Media, Inc., 2017



"The Deep Learning with PyTorch Workshop", Hyatt Saleh, Packt Publishing, 2020

References

- "7 Types of Neural Network Activation Functions: How to Choose?" (https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/)
- "Convolutional neural networks, Jeremy Jordan (https://www.jeremyjordan.me/convolutional-neural-networks/)
- “Reinforcement Q-Learning from Scratch in Python with OpenAI Gym”, Satwik Kansal & Brendan Martin (https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/)

THANK YOU