Institute of Data

2023

Data Science and AI

Module 9 Part 2

Deep Learning

Institute of Data

Agenda: Module 9 Part 2

- Neural Networks and Deep Learning – overview
- Deep Learning– Basics
- Demo and lab

Neural Networks

- Neural Networks (NNs) are computing systems **inspired** by the neural networks of biological brains.
- A collection of connected units called artificial neurons are the basis of NNs.
- These have proved most useful in applications that are challenging to express with traditional computer algorithms using rule-based programming.
- Deviations from biology such as backpropagation or passing information in the reverse direction with adjusts to the network to reflect behaviour appeared over time.

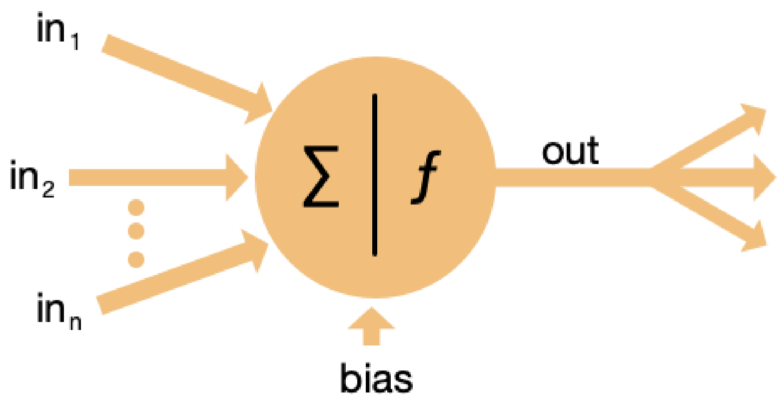
Deep Neural Networks

- DNN is an neural network with a bigger number of **layers** between the input and output layers.
- The network moves through the layers computing the probability of each output.
- The DNN finds the correct mathematical modification to convert the input into the output regardless of being **a linear or a non-linear relationship**.
- For example, a DNN trained to recognise dog breeds analyses a given image and compute the probability of a particular breed for the dog in the image.

Neurons

- A NN consists of **simple elements** called artificial neurons that receive **input**, and compute the **output** depending on the input and **activation**

- Artificial neurons mimic the working of a biophysical neuron with inputs and outputs but is not a biological neuron model.



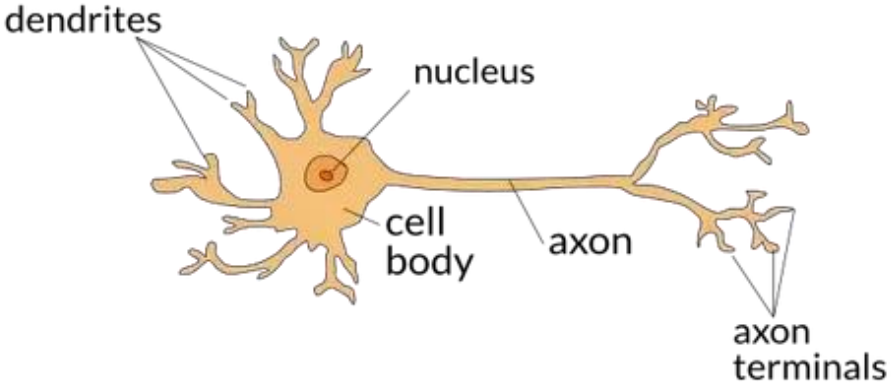


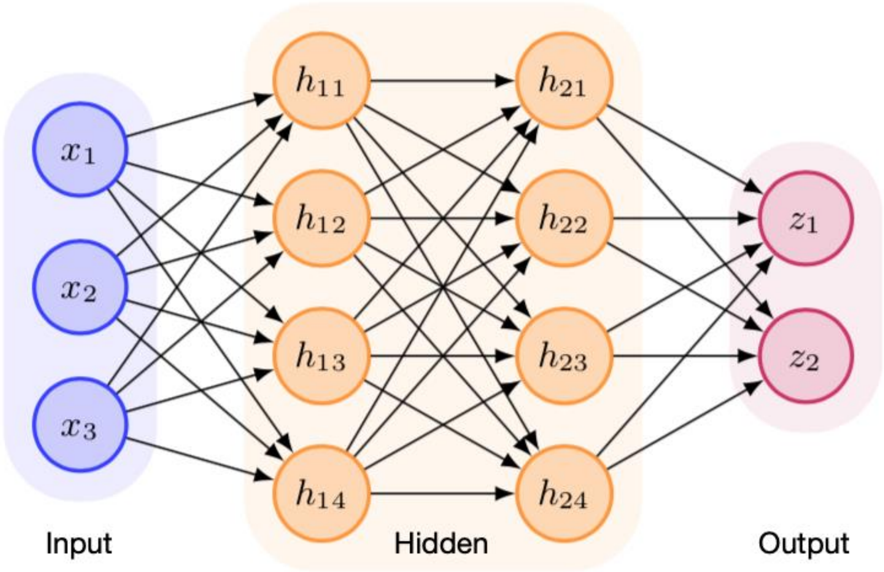
Image: www.quora.com

Feed Forward (FF) and back propagation

- The initial NNs were simple single layer networks
- Feed information from the input to the output
- Trains NNs through backpropagation with **supervised learning**
- The error is often some variation of the difference between the input and the output (like Mean Standard Error)

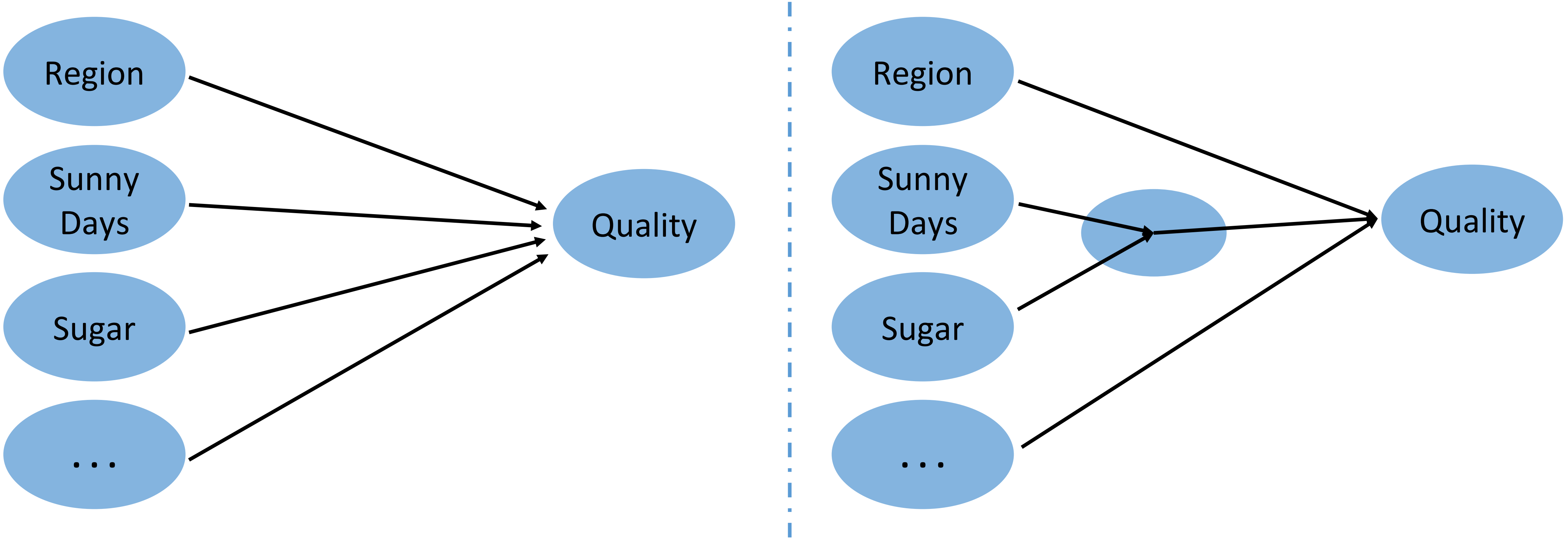
Neural Networks

- Adding more layers to NNs has produced very impressive results
- The structure and connectivity between layers varies widely and still open for research
- The network is made by connecting the output of specific neurons to the input of other neurons forming a directed, **weighted graph**

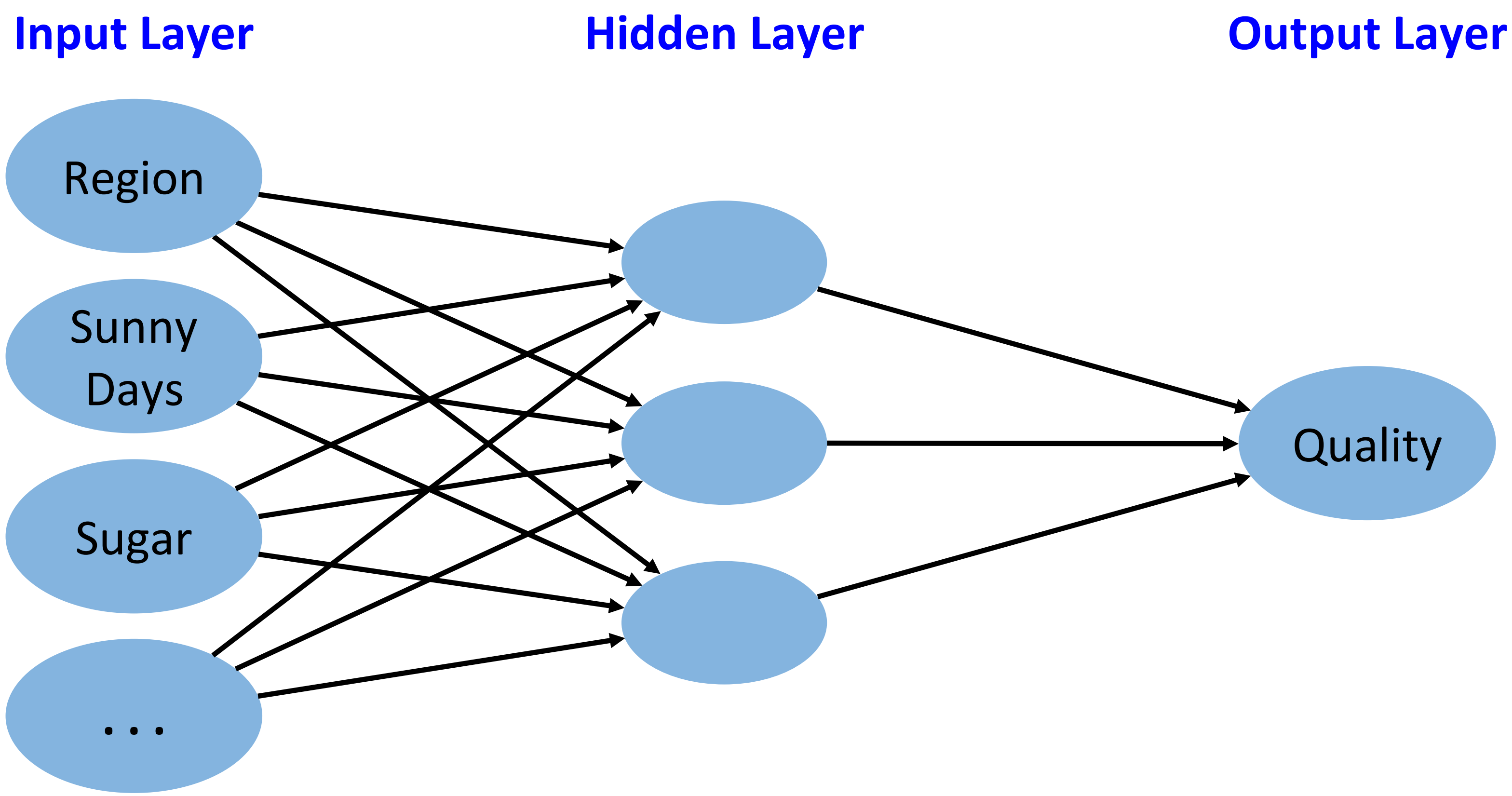


Interaction between inputs

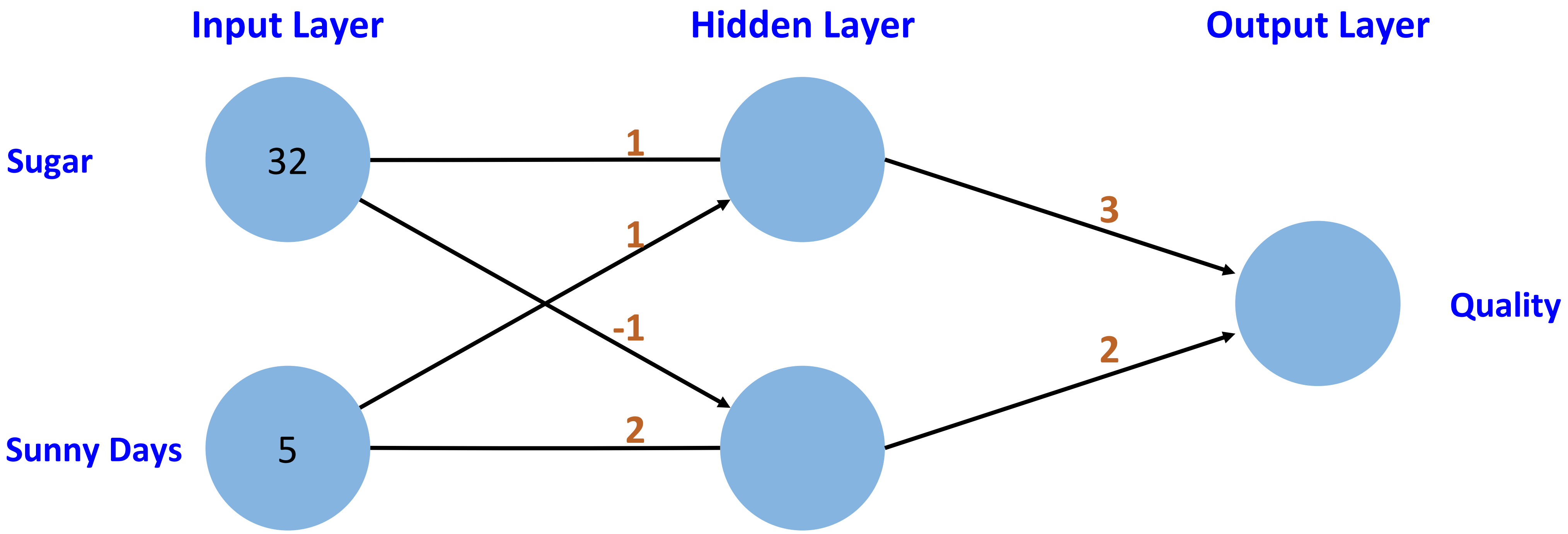
- Model with **no** interaction between inputs
- Model with interaction



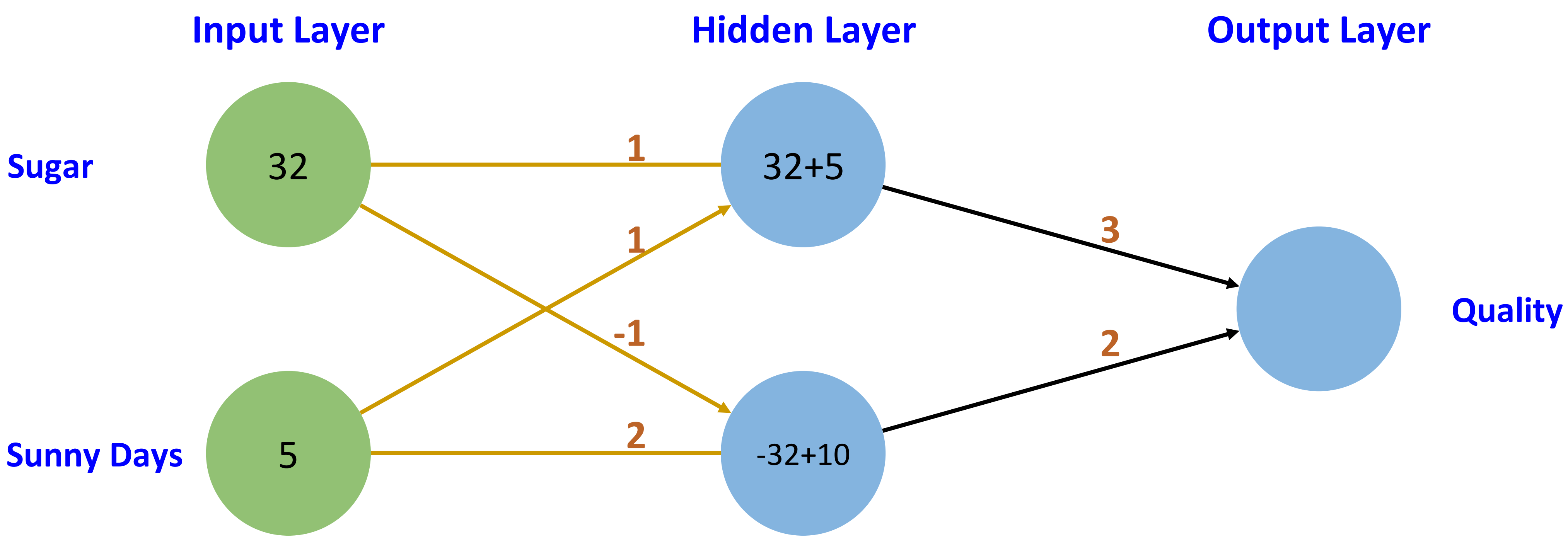
Interactions between inputs can be very complex



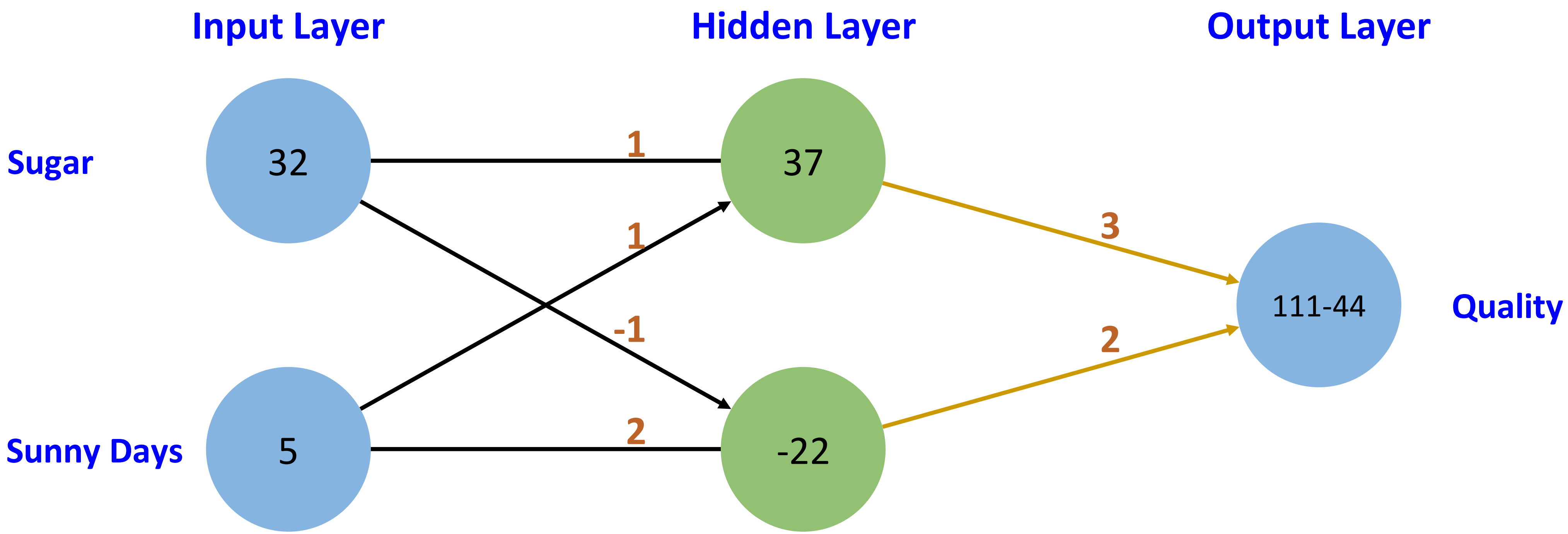
Forward Propagation



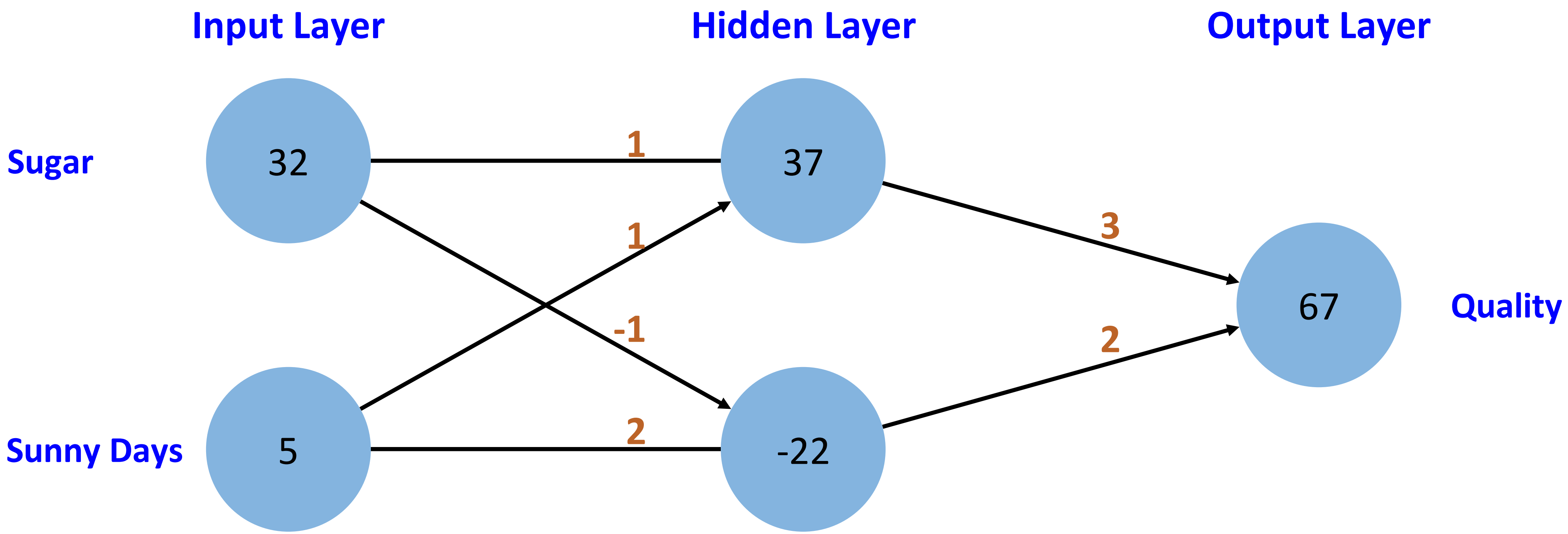
Forward Propagation



Forward Propagation



Forward Propagation

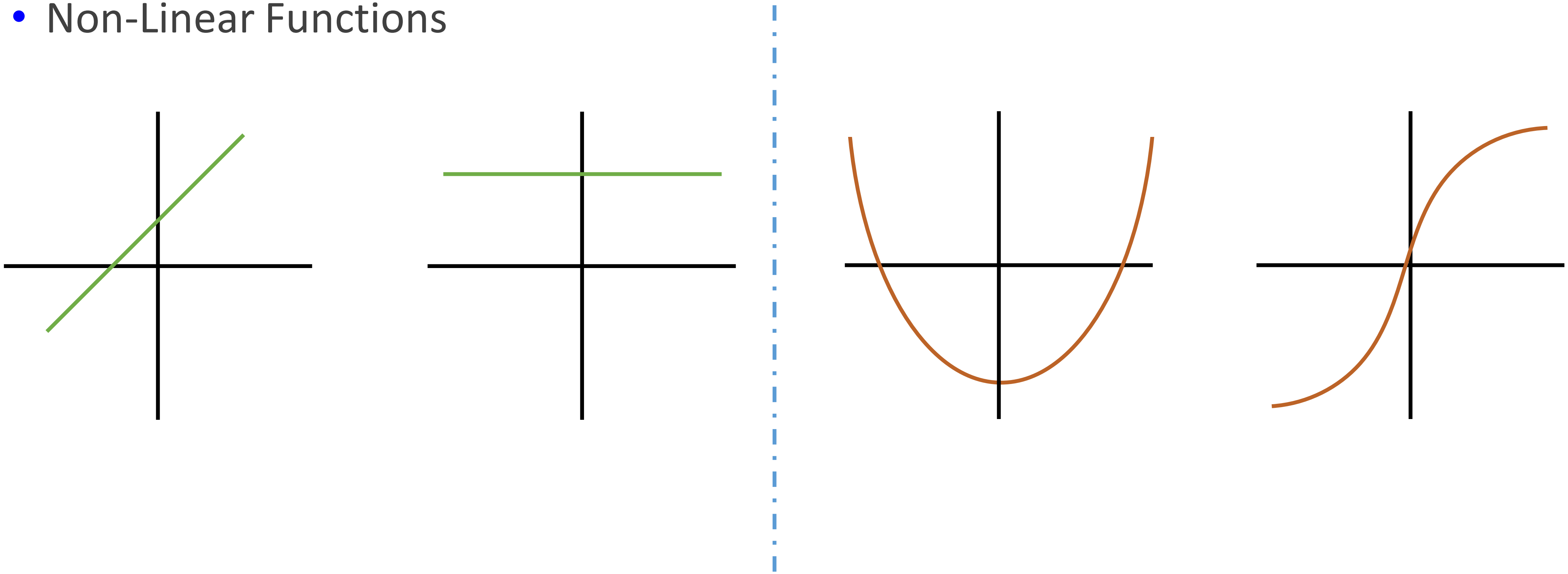


Forward Propagation

- Can be as simple as multiply then add process.
- Forward propagation for individual data point each time.
- The prediction for the data point is the output.

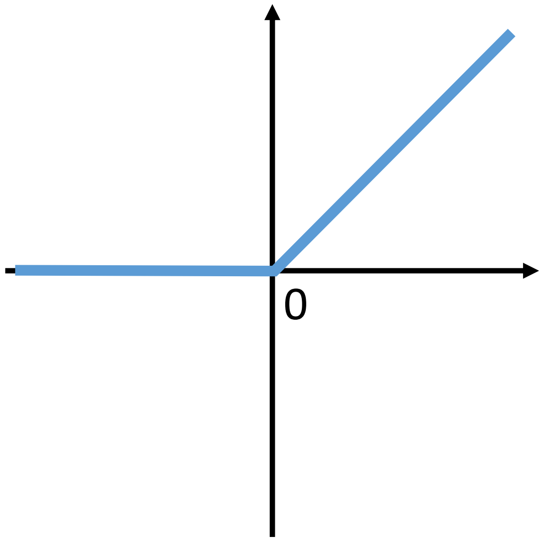
Activation Function

- Linear Functions
- Non-Linear Functions



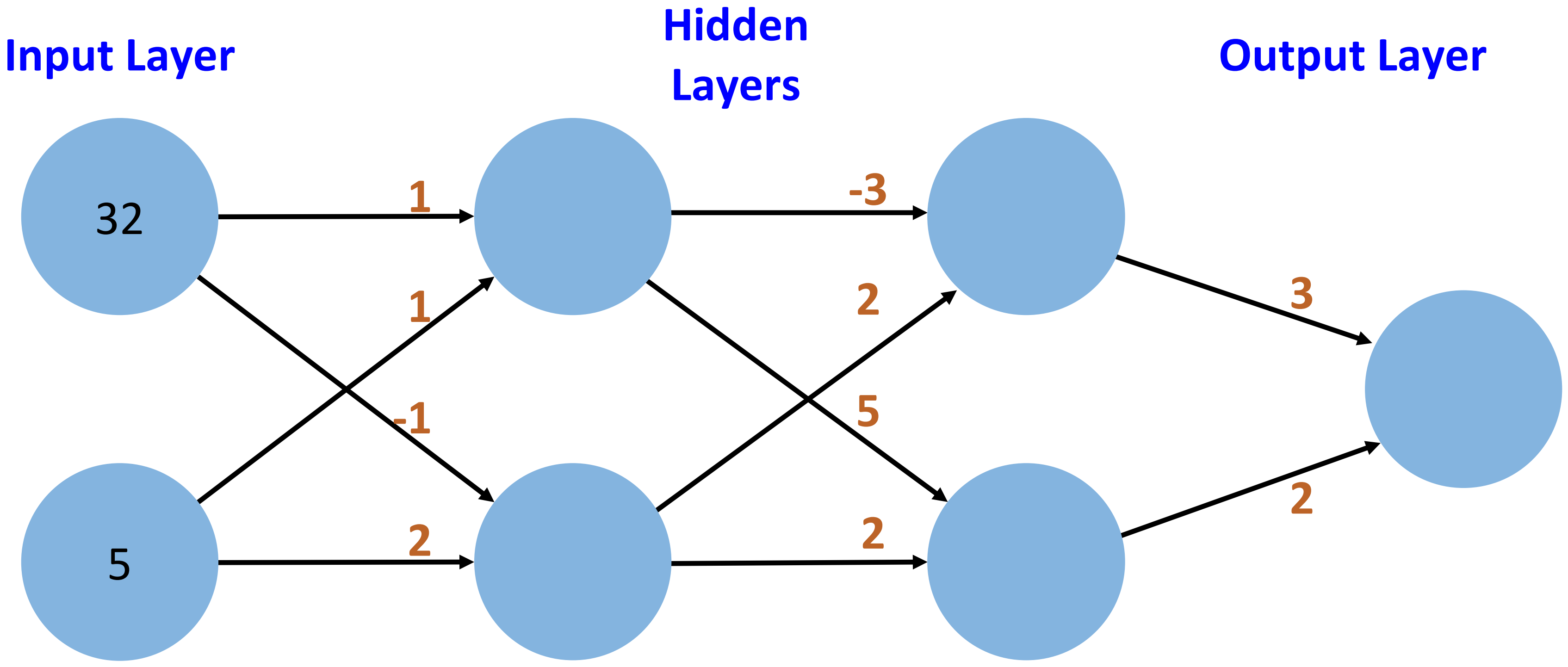
Activation Function

- Applied to the node's inputs to compute the node's output
- ReLU (**Re**ctified **L**inear **A**ctivation)



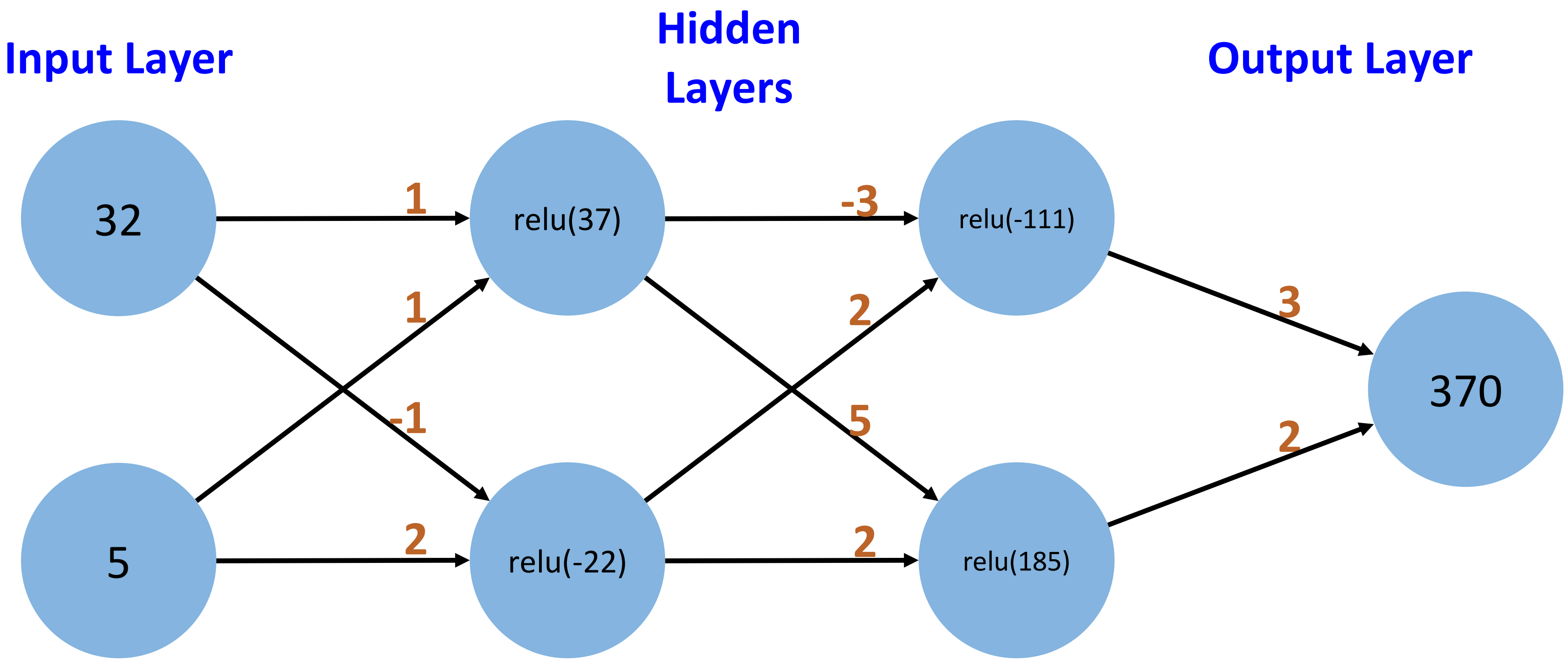
$$RELU(x) = \begin{cases} 0 & if \quad x < 0 \\ x & if \quad x \geq 0 \end{cases}$$

Multiple Hidden Layers



Calculate with ReLU Activation Function

Multiple Hidden Layers



Representation Learning

- Deep networks create **internal representations** of patterns in the data
- Partially replace the need for **feature engineering**
- Later layers create **more sophisticated representations** of raw data

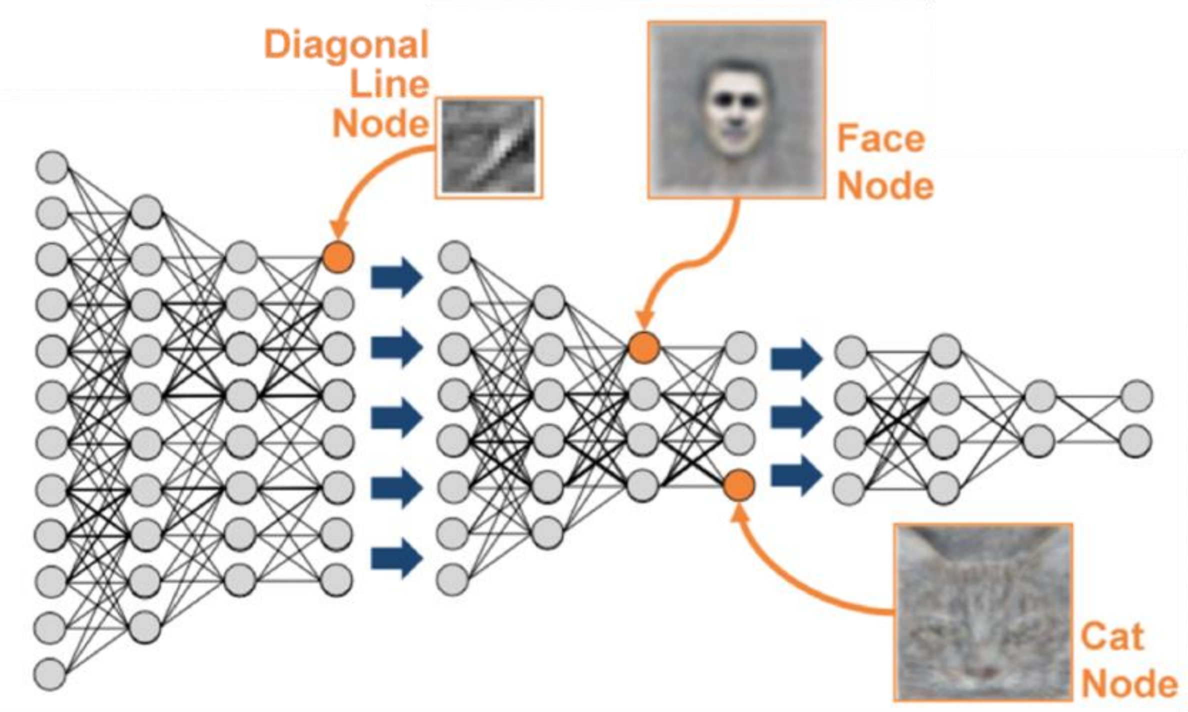
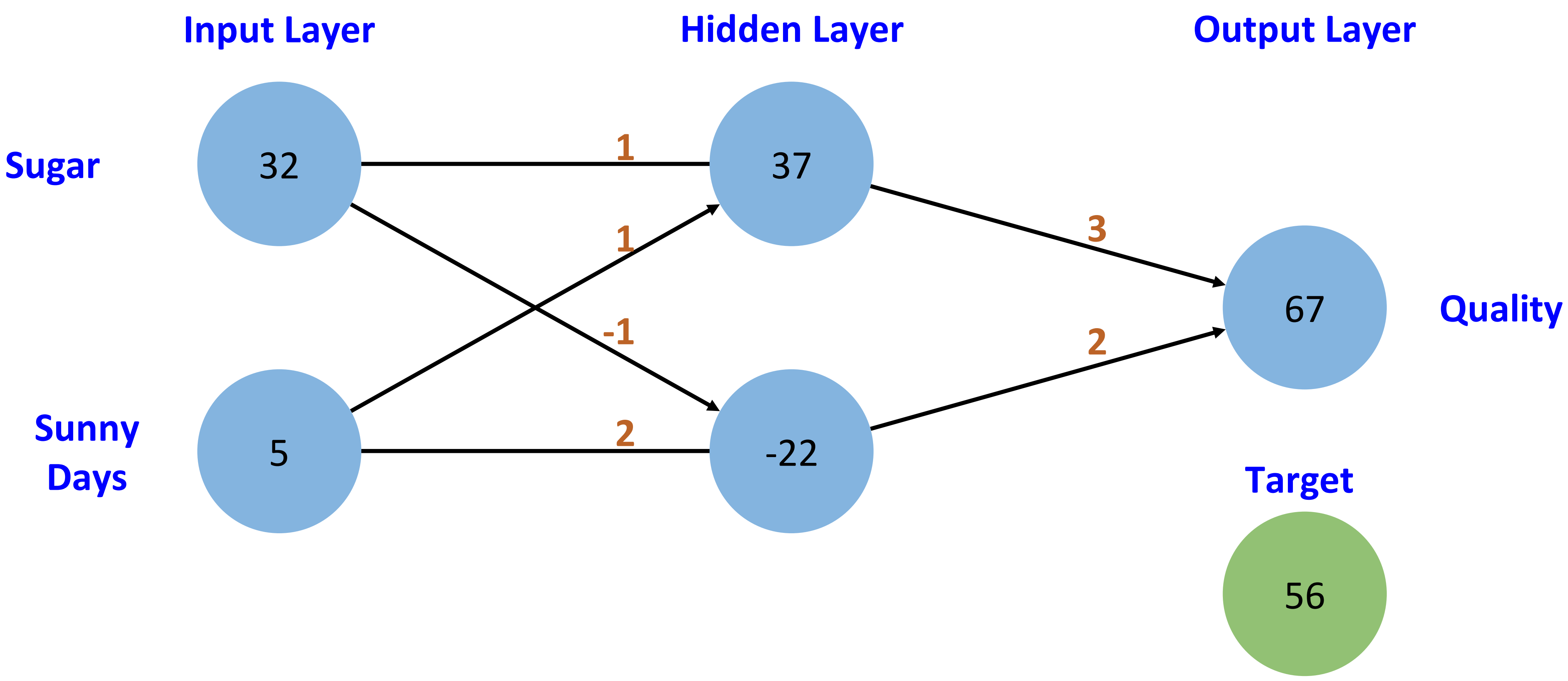


Image: theanalyticsstore.ie

Reach the Target



Reach the Target

Actual value			56
Predicted value			67
Error	Predicted - Actual	67 - 56	11

More Data

Actual	Predict	Error (Predict - Actual)	Squared Error
56	67	11	121
67	50	-17	289
149	148	-1	1
117	99	-18	324
29	9	-20	400
23	42	19	361
64	28	-36	1296
57	39	-18	324
36	7	-29	841
56	23	-33	1089
		Total Squared Error	5046
		Mean Squared Error	504.6

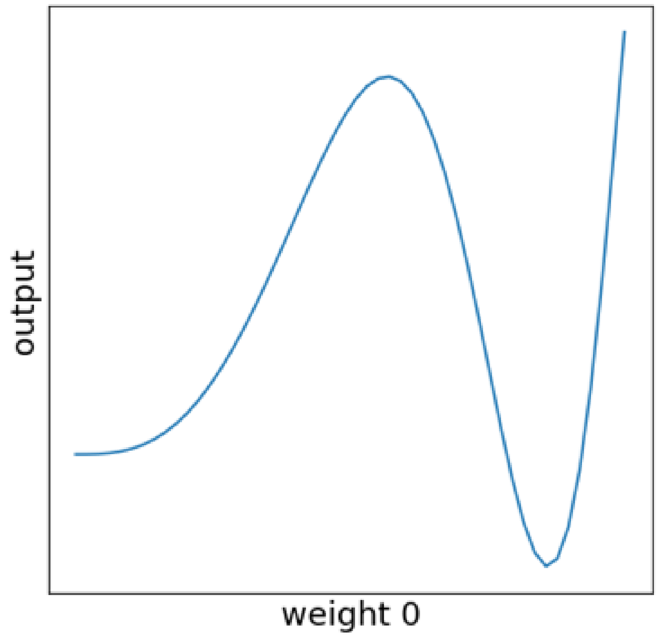
Loss functions

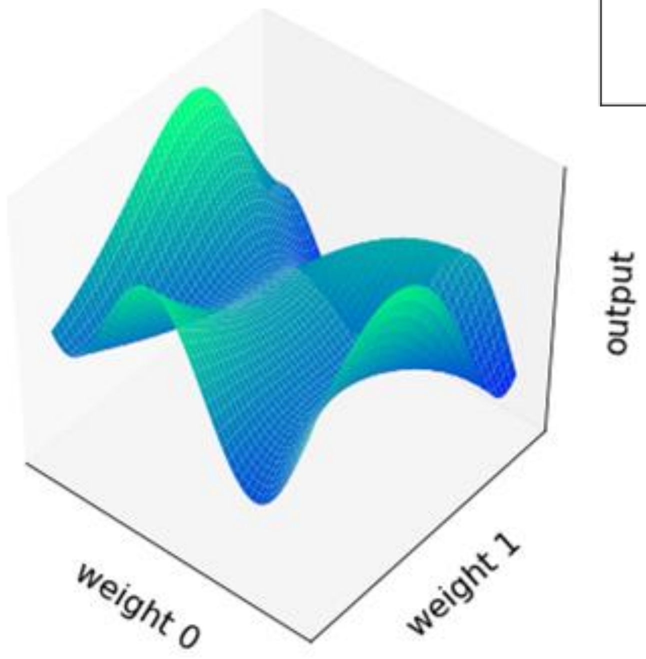
- The objective is to find the **weights** that give the **lowest value** for the loss function.
- The computation gets more complicated for multiple points.
- A Loss Function
  - Serves as a measure of the predictive performance of a model
  - Calculates a single number from the predictions' errors of many data points
- A better model has a **lower** loss function value
- The typical approach to obtain a lower value is to use **Gradient Descent**

Gradient Descent

- The Goal is to find a lower point a curve, surface or multi-plane

1. Start at a random point
2. Find the slope
3. Move a step down
4. Repeat 2 and 3 until the slope is zero





Gradient Descent

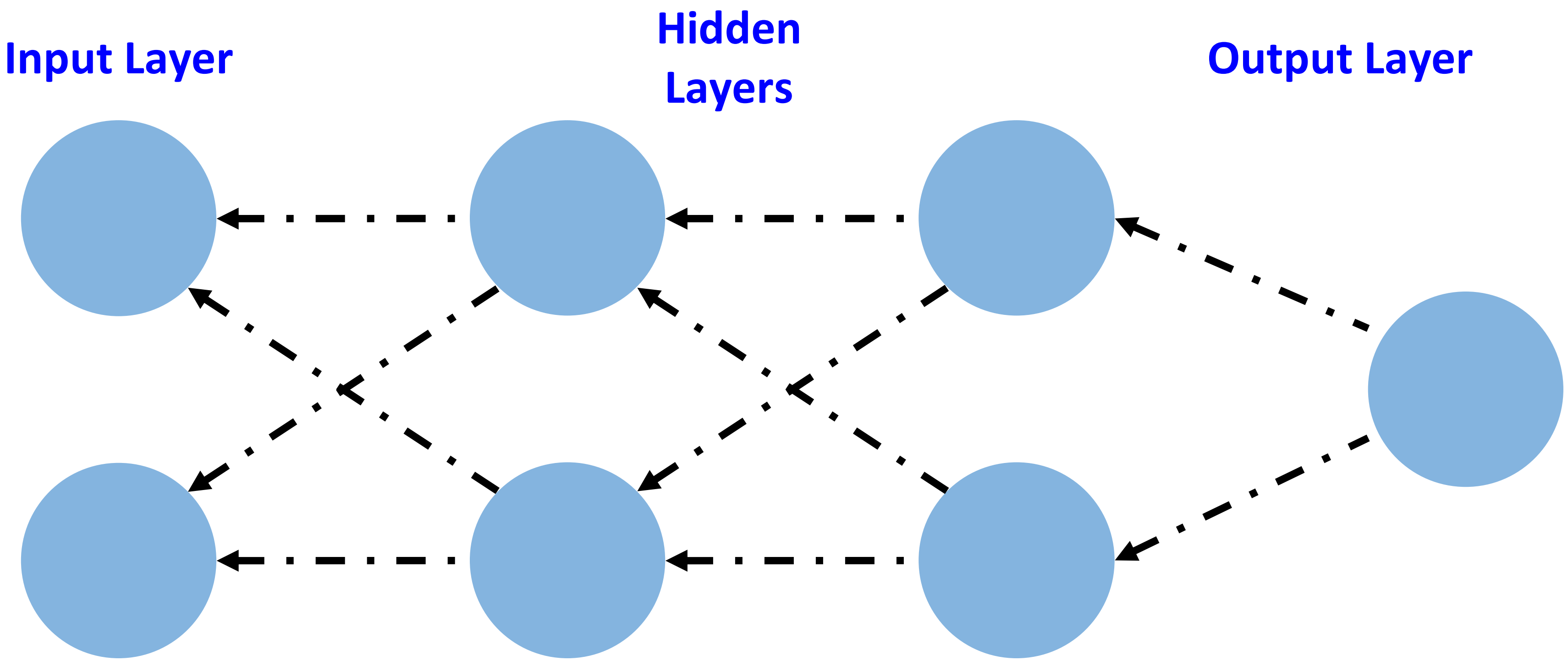
- Gradient descent aims to **update all weights** in a neural network based on the prediction error

$$w'_x = w_x - l\left(\frac{\partial Error}{\partial w_x}\right)$$

Where	
w_x'	New weight
w_x	Old weight
l	Learning rate
$\partial Error / \partial w_x$	Derivative of Error w.r.t. weight

- **Backpropagation** is an algorithm enabling the gradients to be calculated efficiently, applying the chain rule of calculus

Backpropagation (of Errors)



Demo: Neural Networks Basics

- Purpose
  - Understand the basic calculations and mechanics of NN

- Materials
  - Jupyter Notebook (Demo-9-Neural_Networks_Basics)

TensorFlow

- **Open-source** software library from Google for data flow programming and various tasks.
- **Symbolic math library** also used for machine learning applications such as neural networks.
- TensorFlow has a very active and large community of developers and users. Its GitHub has over 100k stars
- TensorFlow can run on multiple CPUs (Central Processing Units), GPUs (Graphical Processing Units) and TPU (Tensor Processing Units).
- In Jan 2018, Google announced TensorFlow 2.0 beta.
- There are alternatives for TensorFlow such as PyTorch, MXNet and CNTK.

Demo: TensorFlow Playground

- Purpose
  - Play with TensorFlow Playground
  - Visualise the structure of NN
  - Understand the workings go NN

- Resource
  - [TensorFlow Playground]

- Materials
  - Jupyter Notebook (Demo-9-TensorFlow_Playground)

Keras

- Open-source neural network Python library
- **Runs on top of** TensorFlow, Theano , Microsoft Cognitive Toolkit
    - TensorFlow supports Keras at its core library since 2017
- Enables fast experimentation with deep neural networks focusing on simplicity and modularity
- Keras was designed to be an interface opposed to a standalone framework
- It offers a high level and intuitive abstraction to ease the development of deep learning models regardless of the backend

Keras - Basic Code Structure

- The principal data structure is a model, a way to organise layers
- The simplest type of model is the **Sequential** model, a linear stack of layers

```
from keras.models import Sequential
# Set up the model architecture
model = Sequential()
from keras.layers import Dense
# Add the first hidden layer
model.add(Dense(15, activation = 'relu', input_shape = (n_cols, )))
# Add the second hidden layer
model.add(Dense(5, activation = 'relu'))
# Add the output layer
model.add(Dense(1, activation = 'linear'))
```

Keras - Basic Code Structure - continued

```
# Compile the model
model.compile(optimizer = 'adam', loss = 'mse', metrics = ['mse’])
# Fit the model
history = model.fit(X_train, y_train, validation_split = 0.25,
   epochs = 1000)
```

Demo: NN with Keras

- Purpose
  - Understand the building blocks of Keras to practice with Neural Networks

- Materials
  - Jupyter Notebook (Demo-9-Keras)

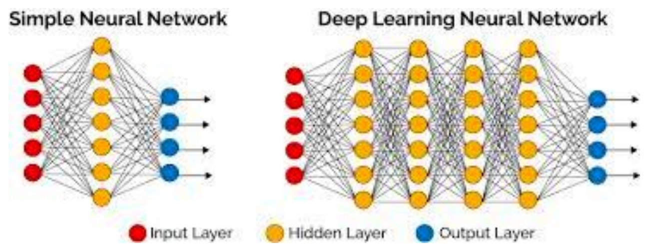
Lab 9.1: NN with Keras

- Purpose
  - Use Keras to practice with Neural Networks

- Materials
  - Jupyter Notebook (Lab-9_1)

Deep Learning – Recap

- Deep Learning (DL) is a **Machine Learning** technique that extracts **patterns** from **data**.
- DL uses multi-layer Neural Network (NN).
- It can be implemented **relatively easy** using **Python**, **TensorFlow** and **Keras**.
- Because of the strength of DL, it can potentially **discover features automatically**.
- The **foundations** of DL, which is NN, is **not new**. However, the availability of large volume of **data**, powerful **computing resources**, readily available **tools** and active **community** has propelled DL to the forefront of all ML techniques and, almost, all technologies.
- In spite of recent achievements of DL, we need to realise that the **big fundamental questions** about Artificial Intelligence (**AI**) remain **unanswered**.



Deep Learning – Recent achievements

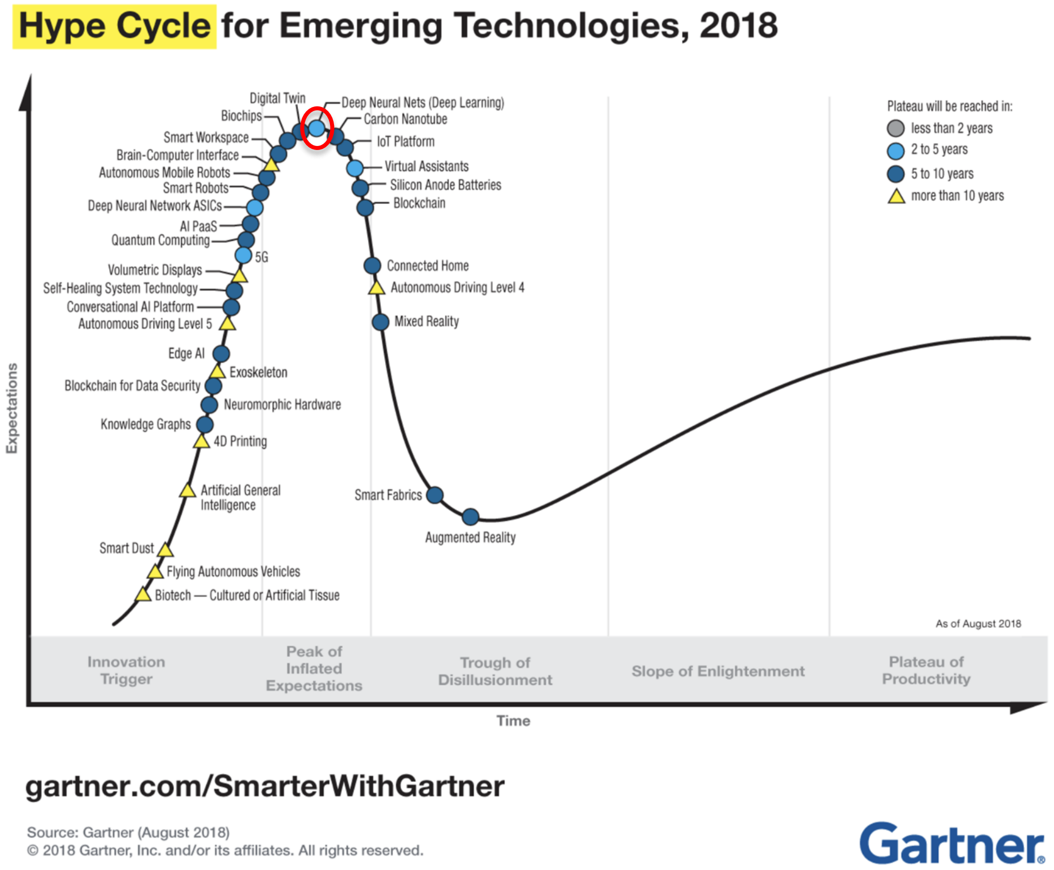
- Face recognition
- Image classification
- Speech recognition
- Text-to-speech generation
- Handwriting transcription
- Machine translation
- Medical diagnosis
- Driverless cars (Note that DL is used only for perception)
- Digital assistants
- Ads, search, social recommendations
- Game playing with deep Reinforcement Learning (RL)

Deep Learning – A brief history

- 1943: Neural Networks
- 1957:Perceptron
- 1969: Minsky & Papert (1969) pricked the neural network balloon
- 1971: A paper by A. G. Ivakhnenko described a deep network with 8 layers
- 1974-86: Backpropagation, Recurrent NN
- 1986: The term Deep Learning was introduced by Rina Dechter
- 1989-98: Convolutional NN, MNIST digits dataset, Long Short-Term Memory (LSTM) NN
- 2006: “Deep Learning” papers by Geoff Hinton et al
- 2009: ImageNet dataset
- 2012:AlexNet, Dropout technique
- 2014: Generative Adversarial Networks (GANs)
- 2014: DeepFace by Facebook
- 2016: AlphaGo
- 2017: AlphaZero
- 2018: Google BERT pre-trained NLP model
- 2021: DALL-E
- 2022: ChatGPT

Deep Learning – The hype

- Some of the excitement about DL can be attributed to the recurring phenomena of the Hype Cycle.
- Gartner releases a yearly hype cycle report that tracks hyped technologies.
- Deep Learning recently on the top of the hype cycle



Common types of NNs

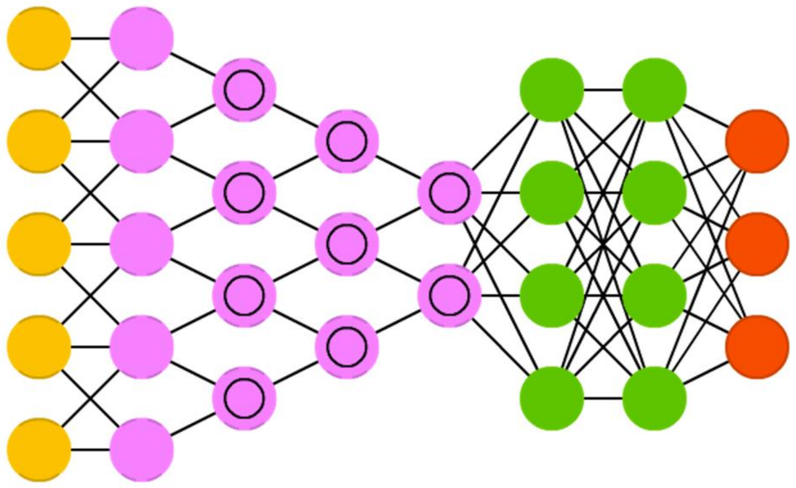
- Feed Forward and Perceptrons
- Convolutional Neural Networks
- Recurrent Neural Networks
- Long/Short-Term Memory
- Transformers
- Many others

Feed Forward (FF) and Perceptrons (P)

- Very straightforward
- Feed information from the input to the output
- Each layer is made of either input, hidden or output cells in parallel

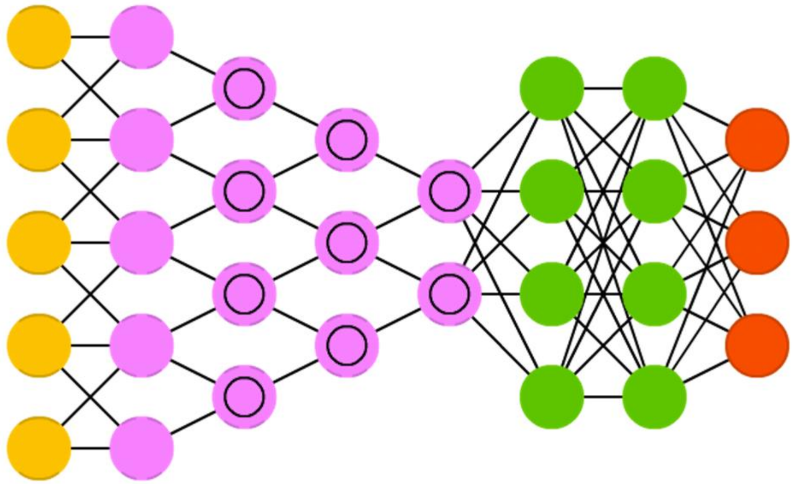
Convolutional Neural Networks (CNN)

- Mainly used for **image processing** but adaptable for other input like audio
- The network classifies the data from an image (outputs “cat” for a picture with a cat picture and “dog” for a dog picture)



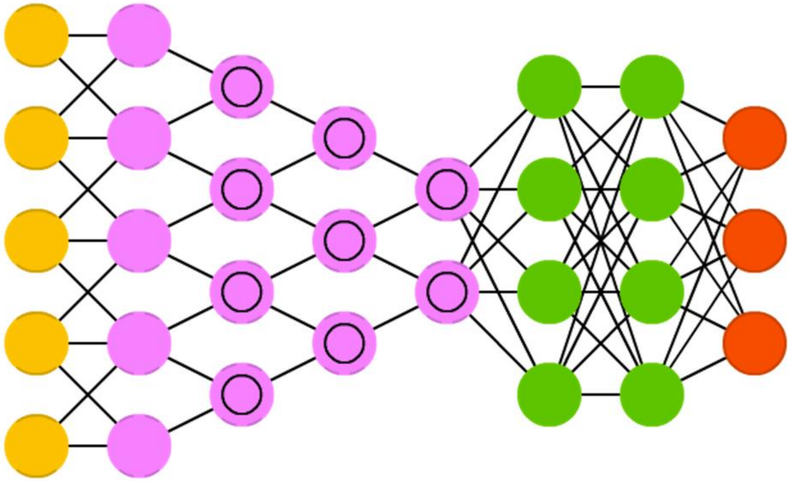
Convolutional Neural Networks (CNN)

- Start with an input “scanner” not intended to parse all the training data at once
- Then get the next block (move the scanner one pixel to the right)
- This input data then feeds through convolutional layers instead of standard layers (not all nodes have connections to all other nodes)



Convolutional Neural Networks (CNN)

- Each node only concerns itself with not more than a few neighbouring cells
- These layers tend to shrink as they become deeper (divisible by two or powers of two are common)
- Also often feature pooling layers to filter out details



Convolutional Neural Networks

- Overview
- Terminology
- Data Representation
- Architecture
- Convolution
- Padding
- Strides
- Dilation

Overview

- Convolutional Neural Network (CNN) is a kind of deep neural networks most commonly applied to analysing **visual content**.

- Convolutional networks are inspired by biological processes that resemble the organisation of the animal **visual cortex**.

- Individual cortical neurons respond to stimuli only in **a restricted region** of the visual field known as the receptive field.

Overview

- CNNs use less pre-processing compared to other image classification algorithms.

- The independence from prior knowledge and human effort in feature design is a significant advantage.

- Some of the Applications are in image and **video recognition**, recommender systems, image classification, medical image analysis, and natural language processing.

Terminology

- **Convolution** is a function derived from two functions (***f*** and ***g***) that expresses how the shape of one function is modified by the other function.
- The term convolution refers to both the process of computing it and the resulting function.
- In regular terms, convolution tries to find changes so it can identify **continuity** or **edges**.
  - Bright and dark
  - Lines or curves

Data Representation

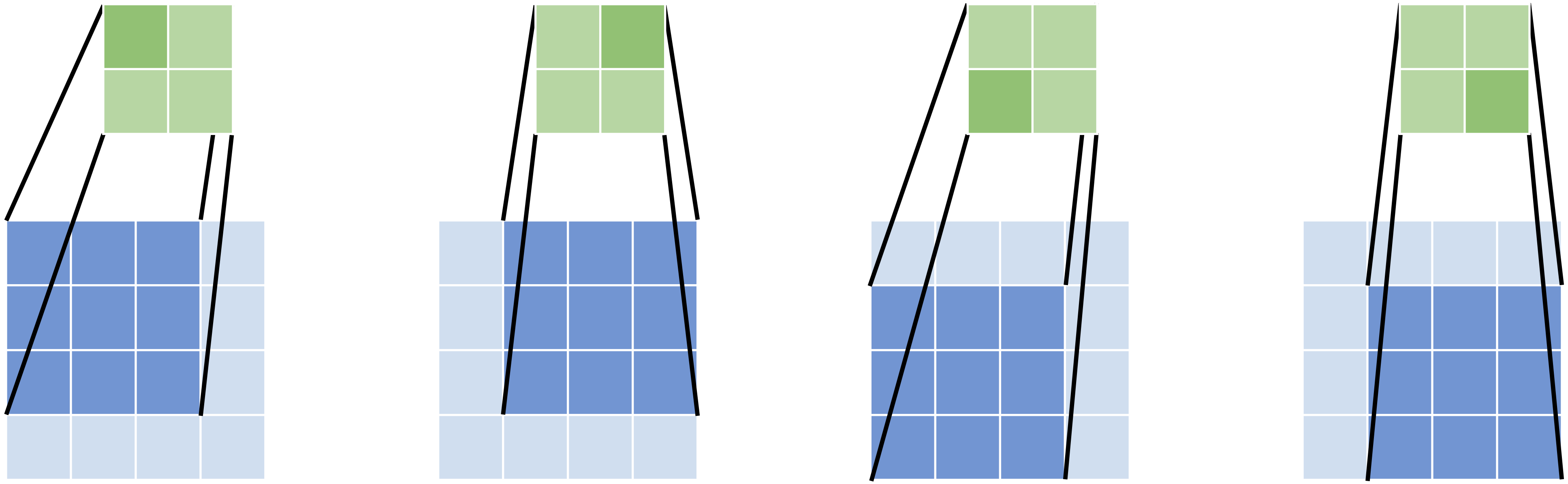
- Some neural networks linearly handle input data even when has a distinct logical representation
  - E.g. Images are 2D but are translated into a **1D representation**
- CNNs maintain the relationship between **adjacent data elements**
  - E.g. Images keep their 2D structure

Architecture

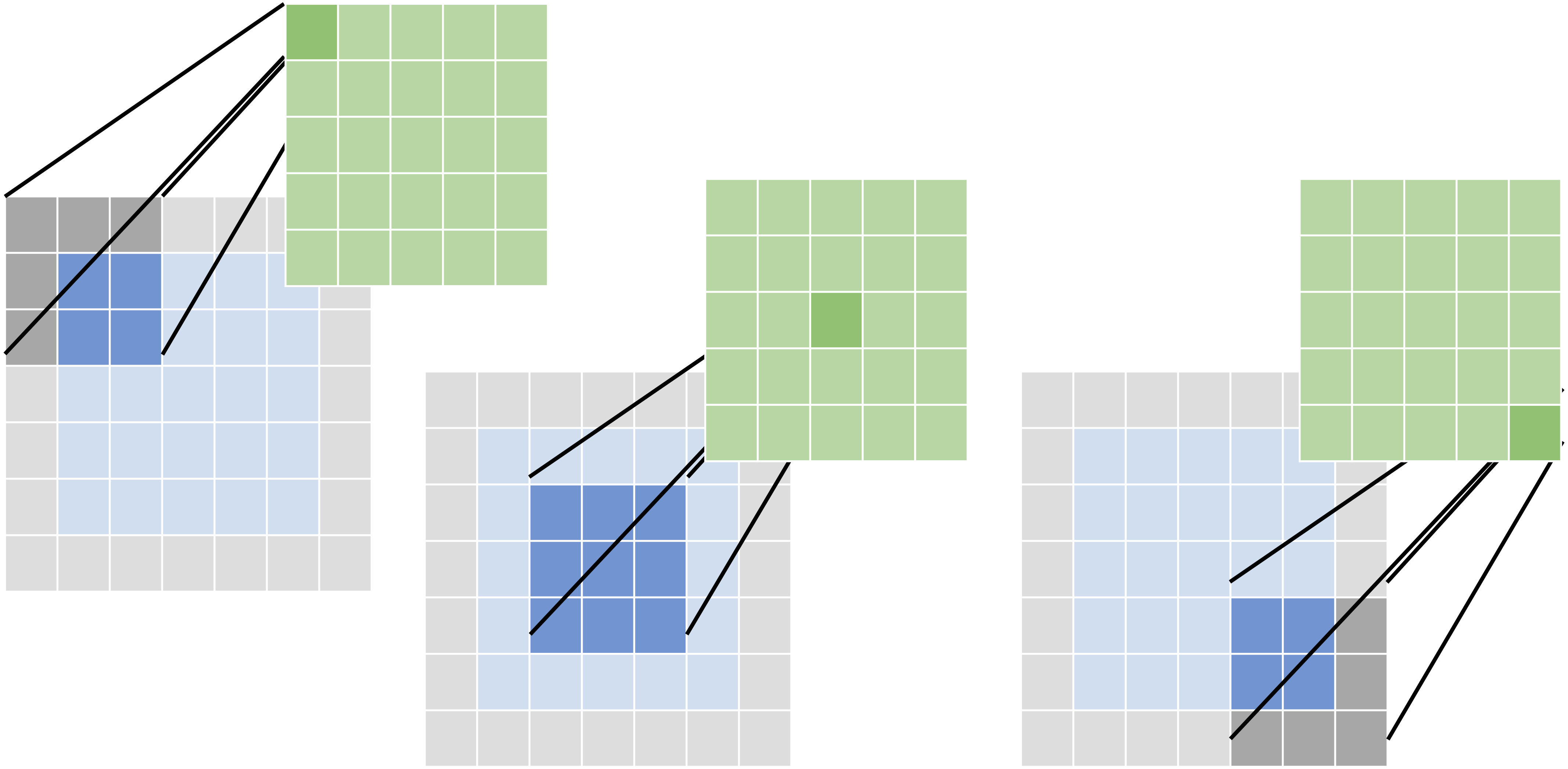
- In typical fully connected NNs all neurons of a layer are connected to **all** neurons of neighbour layers

- There are alternative forms of connection between layers, and not all neurons are necessarily connected in CNNs

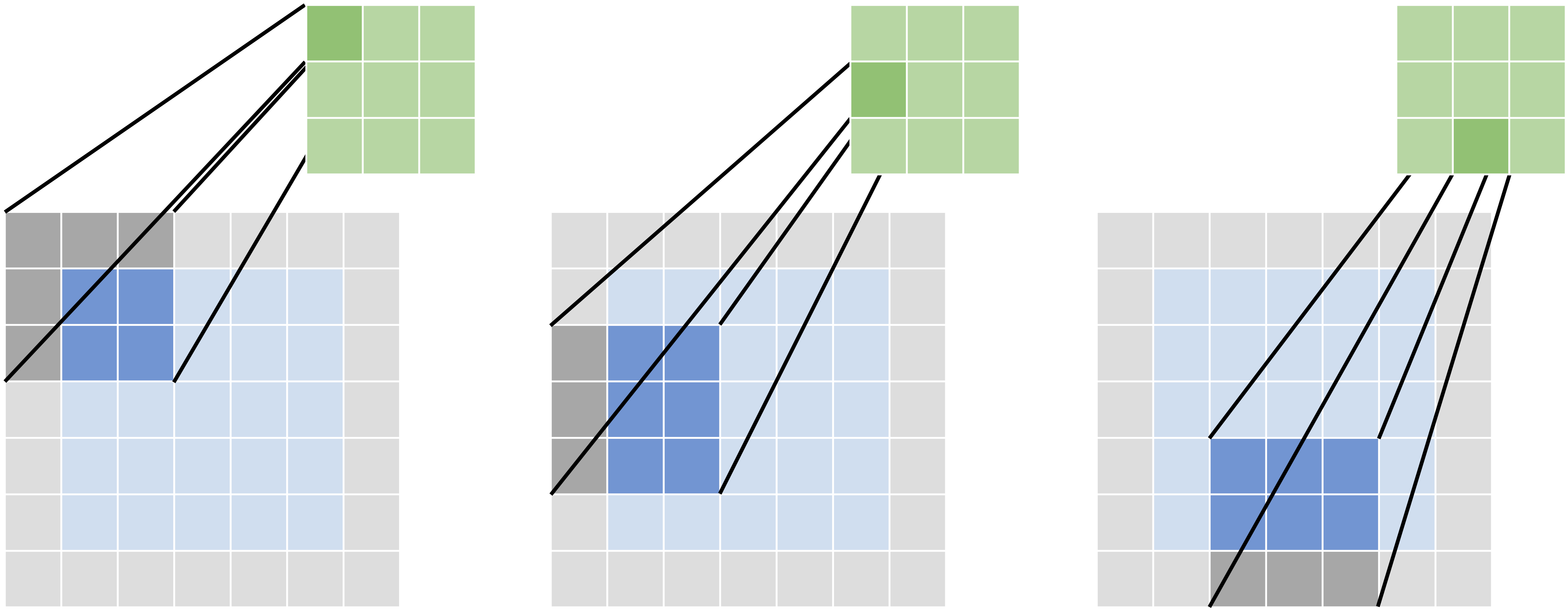
Convolution



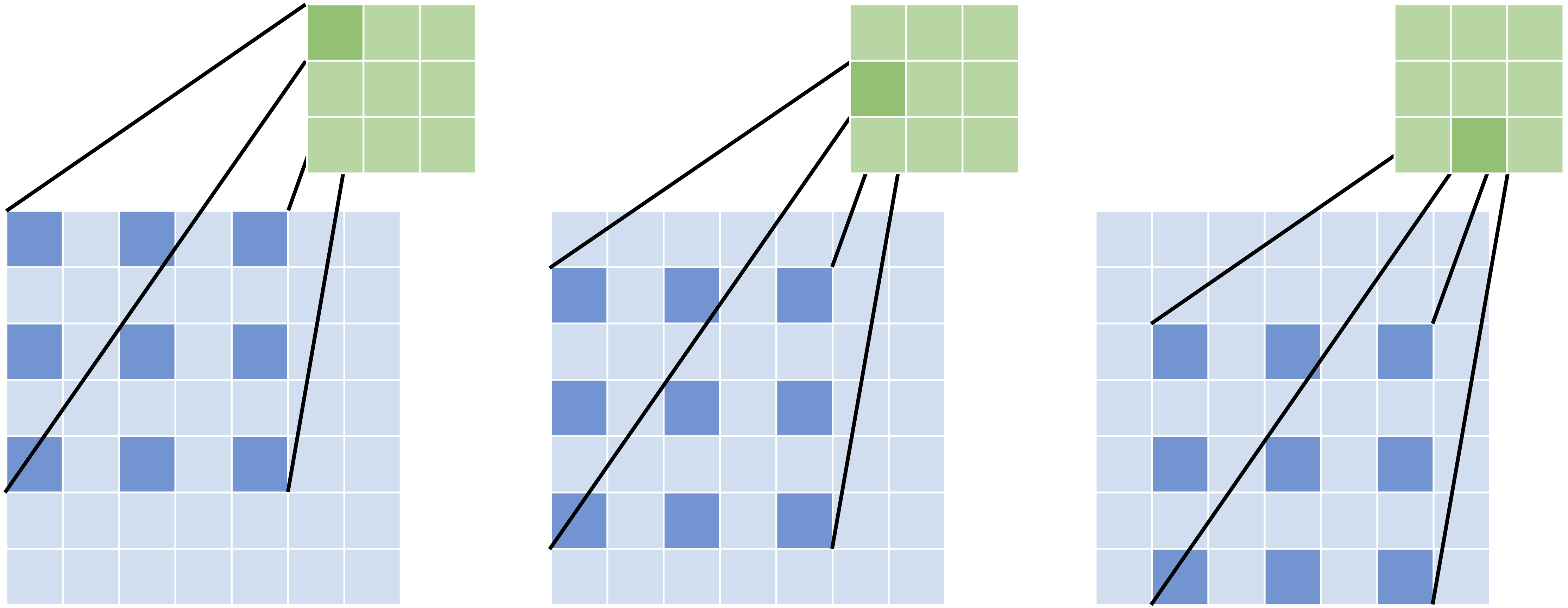
Padding



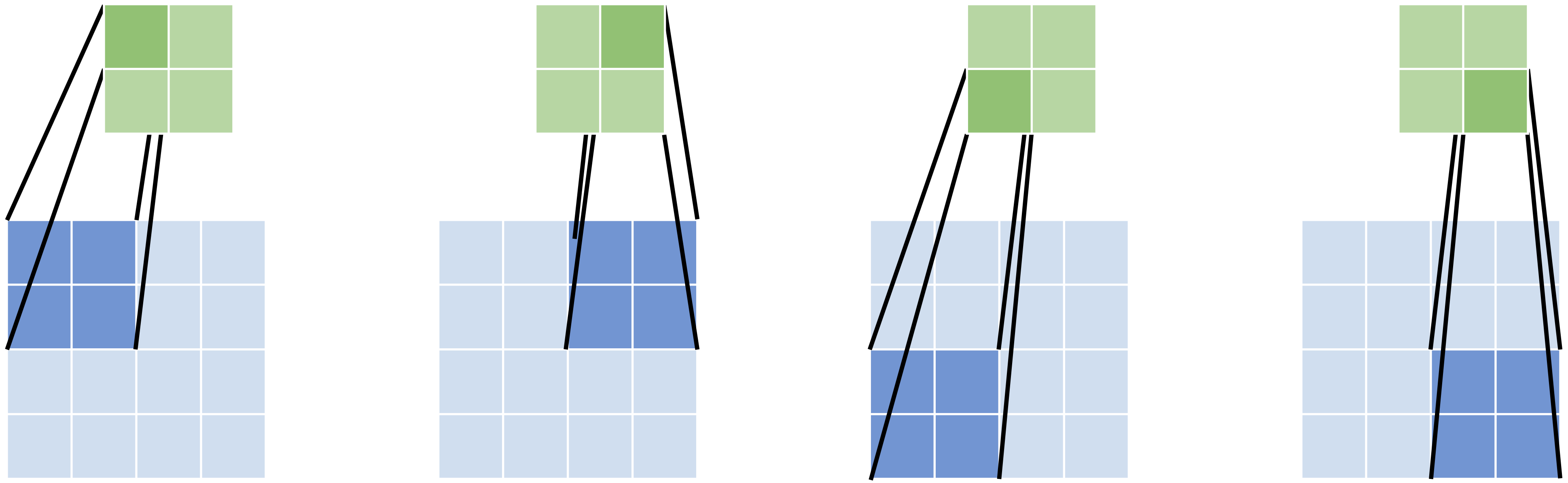
Strides



Dilation



Max Pooling

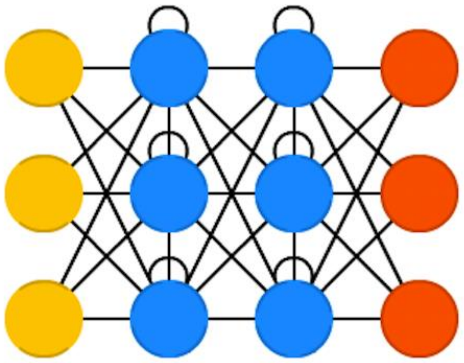


Other Neural Networks

- Recurrent Neural Networks (RNN)
- Long Short-Term Memory (LSTM)
- Transformer Neural Networks
- More on attention

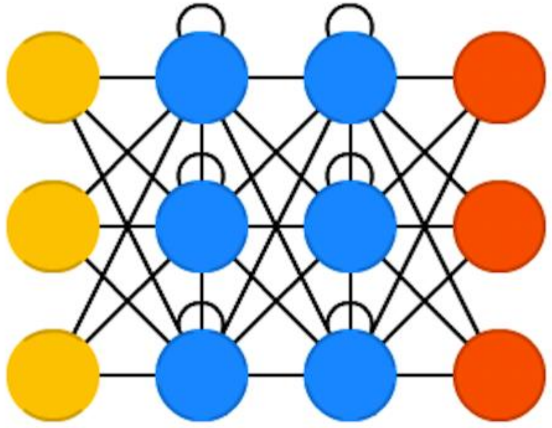
Recurrent Neural Networks (RNN)

- FFs with a **time twist**: not stateless
- Have connections between passes, connections through time
- Neurons are fed information from the previous layer and of **themselves** at the previous pass
- The order of input and train matters: feeding “A” then “B” may yield different results to feeding “B” then “A”



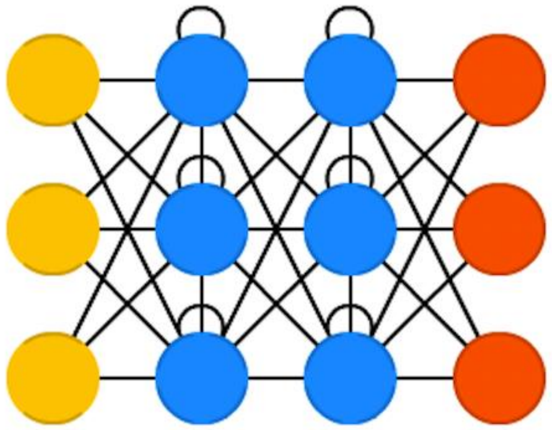
Recurrent Neural Networks (RNN)

- Many fields can use RNNs in principle as most forms of data that do not have a timeline, so the time-dependent weights are used for what came before in the **sequence**, not actually from what happened seconds before
- RNNs are a good choice for advancing or completing the information, such as **autocompletion**



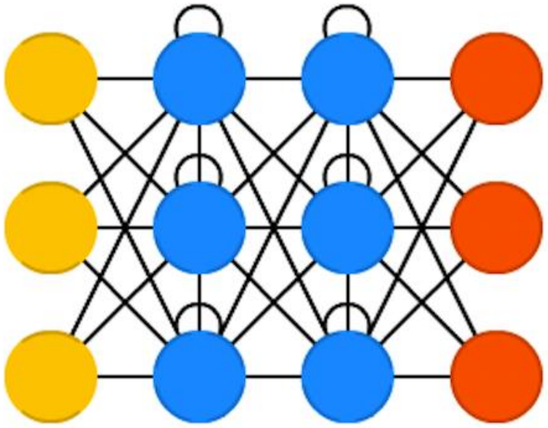
Long/Short-Term Memory (LSTM)

- Each neuron in a LSTM network has a memory cell and three gates: input, output and forget
  - The gates safeguard the information by stopping or allowing its flow
  - The input gate determines how much data from the previous layer gets stored
  - The output layer has the job of deciding how much the following layer gets to know about the state of this node
  - The forget gate controls the retention of data between layers



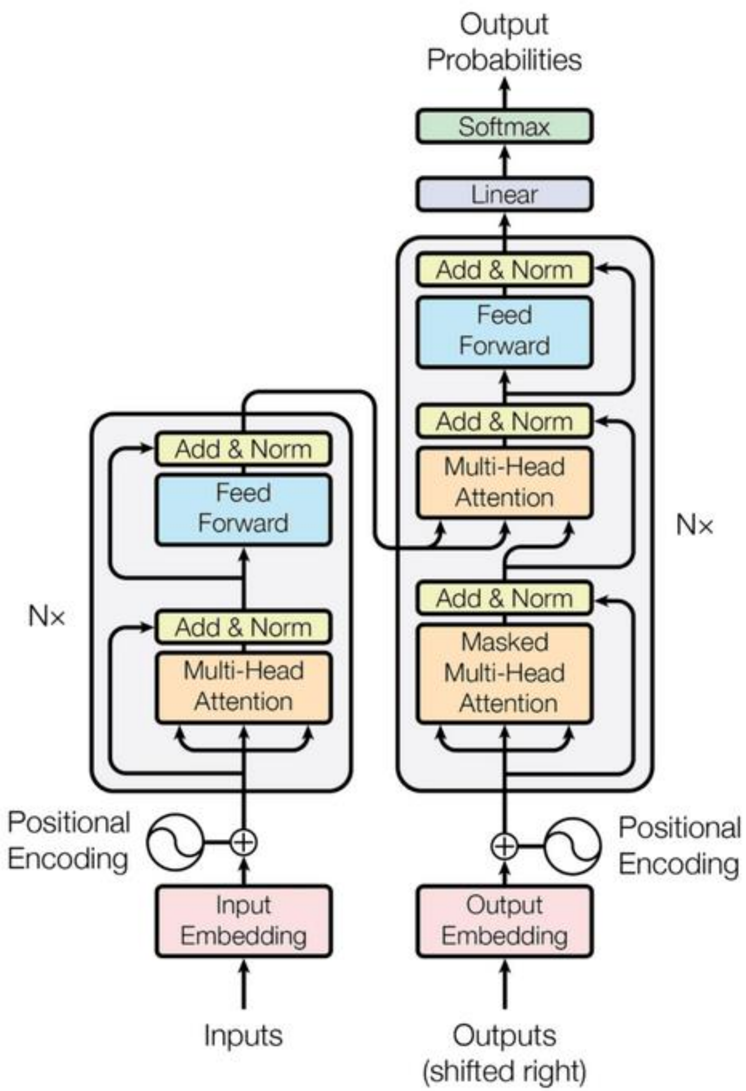
Long/Short-Term Memory (LSTM)

- These are inspired mostly by circuitry, not biology
- It typically requires more resources to run as each of the gates has a weight to a cell in the previous neuron



Transformer Neural Networks

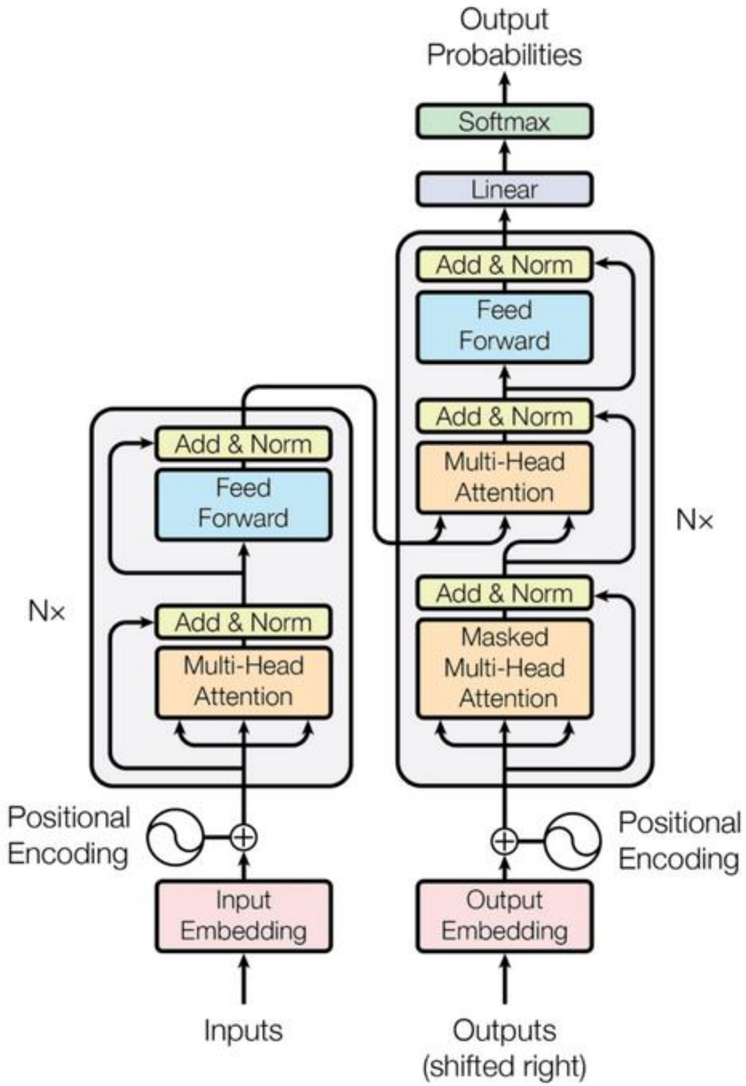
- Originally used for sequence-sequence models (e.g. language translation)
- Seen to be a general-purpose model for representing sequences and performing a wide variety of tasks
- Encoder blocks (left) are used in BERT (used in Google search)
- Decoder blocks (right) are used in GPT models



"[Attention is All You Need](#)"

Transformer Neural Networks

- Components:
  - Embeddings of tokens into numerical vectors
  - Positional encoding (modelling the index of each token)
  - Residual connections and layer normalisation (speed up training)
  - Multi-Head Attention and Feed Forward (create multiple context vectors in parallel and combine them, faster than RNNs of equivalent size)
  - Nx repetition of blocks (each repetition enables longer-term memory of the sequence)



"Attention is All You Need"

More on attention

- The mechanism outputs a context vector from a weighted combination of hidden value vectors. Unlike traditional feed-forward networks the weights are different for each input.
- This provides an efficient way to capture long-term dependencies between inputs.
- Viewing attention weights can provide some insight of ‘where’ the network focuses for each input to the network.

	I	love	you
je	0.94	0.02	0.04
t'	0.11	0.01	0.88
aime	0.03	0.95	0.02

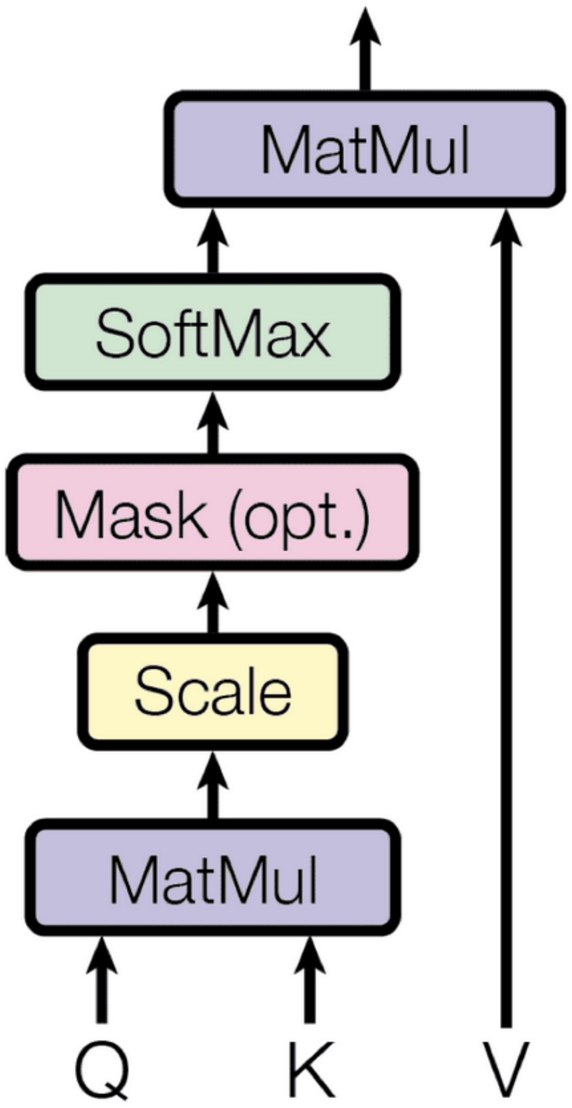
https://en.wikipedia.org/wiki/Attention_(machine_learning)

In the above English-French translation example, the second word to be generated has 88% of attention on “you”, so the token t’ is output.

More on attention

In transformers each attention block learns 3 matrices that project inputs into vectors q, k and v:

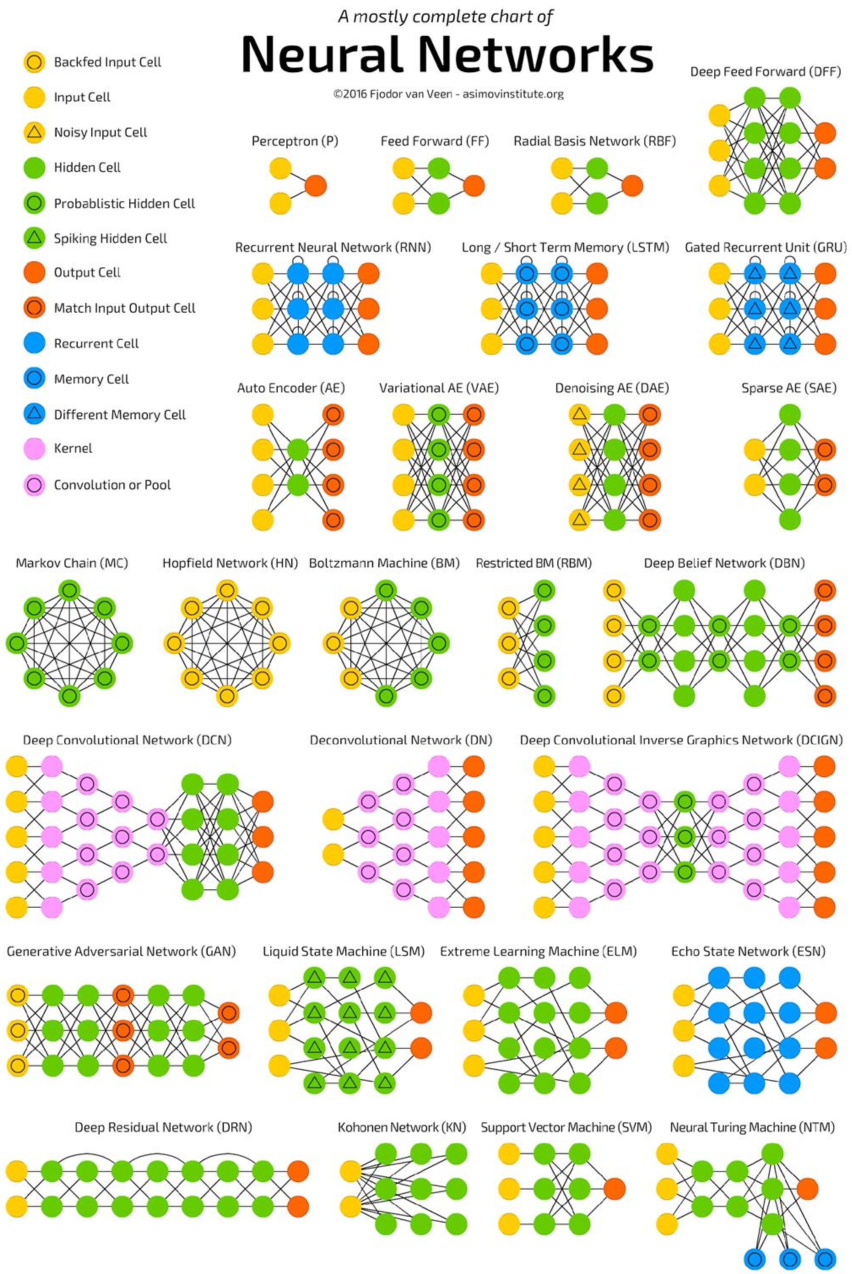
- Query: the current input we are focusing on
- Key: a (preceding) input that is compared to the query
- Value: captures the relevance of the query-key relationship for the current input
- The query and key are compared (scaled dot product), then the resulting weights (scaled to sum to 1) are applied as a linear combination of the value vectors



"Attention is All You Need"

Many Others NN types

- See Appendix for further details



[© 2016 Fjodor van Veen-asimovinstitute.org]

Other Aspects when Training

- Epoch, Batch and Iteration
- Dropout
- Batch Normalisation
- Kernel Regularisation

Deep Learning – Epoch, batch and iteration

- An **epoch** represent one iteration over the entire dataset.
- We divide the dataset into a number of **batches**.
- Iteration is passing data within a batch through the network
- Trade-off regarding the **batch size**
  - Larger batch size = Faster
  - Smaller batch size = (empirically) better generalisation

Dropout

- Dropout is a **regularisation** technique where we randomly remove some nodes in the network to address **overfitting**
- Most deep learning frameworks come with a dropout layer technique
- The reduction in the number of parameters at each step of training has the effect of **regularisation**
- Dropout has shown better performance of neural networks on supervised learning
- In each learning step
  - Select a subset of the units
  - Ignore it in the forward pass
  - And in the back-propagation of error

Batch Normalisation

- Normalises the activation of the preceding layer at each batch
  - i.e. uses a transformation that keeps the activation standard deviation close to 1 and the mean activation close to 0
- Addresses the problem of **internal covariate shift**
- Also acts as a regulariser, in some cases eliminating the need for Dropout
- Achieves similar accuracy with less training steps thus speeding up the training process

Kernel Regularisation

- Apply **penalties** on layer parameters during optimisation
- The **loss function** incorporates the penalties
- The regularisation penalises **peaky weights** and makes sure that all the inputs are considered

Key Theoretical Questions in Deep Learning

- **Architecture design**
  - Are there principled ways to design networks?
    - How many layers?
    - Size of layers?
    - Choice of layer types?
    - What **classes of functions** can be approximated by a feedforward neural network?
    - How does the architecture impact **expressiveness**?
- **Optimisation**
  - What does the **error surface** look like?
  - How to **guarantee** optimality?
  - When does **local descent** succeed?
- **Generalisation**
  - How well do deep networks **generalise**?
  - How should networks be **regularised**?
  - How to prevent **under or overfitting**?

Other Deep Learning libraries - Theano

- Theano is a Python library and optimising compiler for manipulating and evaluating mathematical expressions
- Theano computations are expressed in a NumPy style syntax and compiled to run efficiently on **CPU and GPU** chip design
- Theano is an **open-source** project mainly developed by the Montreal Institute for Learning Algorithms (MILA) at the Université de Montréal
- Theano has been used in computationally intensive large-scale scientific investigations since 2007
- It is approachable enough to be used in the classroom (University of Montreal's machine learning classes)

Other Deep Learning libraries - PyTorch

- PyTorch is an **open-source** library for machine learning in Python used for applications such as natural language processing originally based on Torch
- **Facebook's** artificial-intelligence research group primarily develops it, and Uber's "Pyro" software for probabilistic programming is built on it.
- PyTorch provides two high-level features
  - **Tensor computation** (like NumPy) with strong GPU acceleration
  - **Deep Neural Networks**

PyTorch

Demo: Convolutional NN Basics

- Purpose
  - Understand the basic calculations and mechanics of CNN

- Materials
  - Jupyter Notebook (Demo-9-CNNs_Basics)

Demo: CNN with Keras

- Purpose
  - Understand Keras’ CNNs

- Materials
  - Jupyter Notebook (Demo-9-Keras_CNNs)

Lab 9.2: CNN with Keras

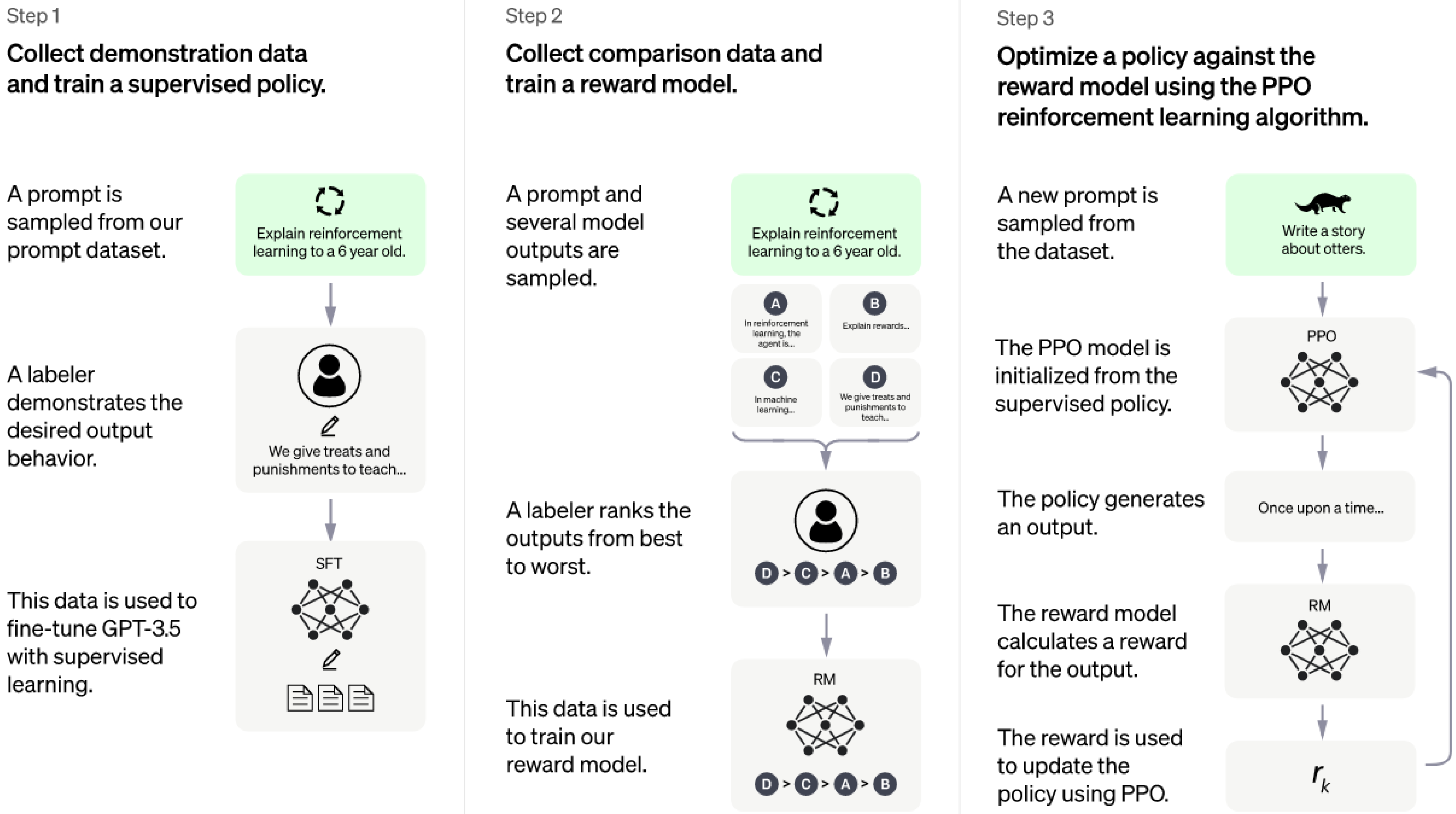
- Purpose
  - Training a Classifier

- Materials
  - Jupyter Notebook (Lab-9_2)

Questions

Appendices

ChatGPT

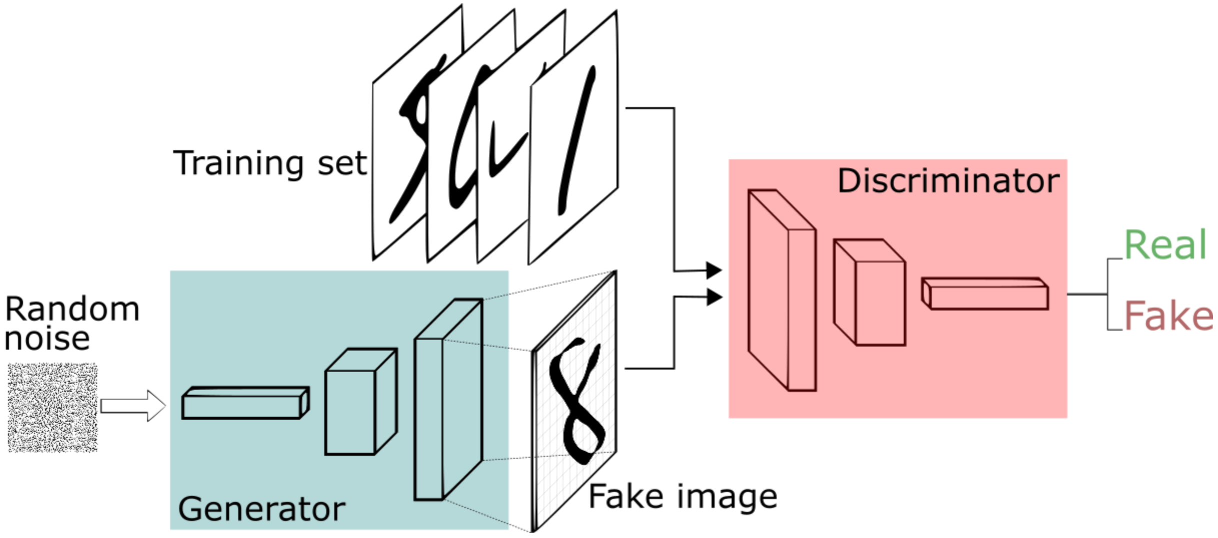


https://openai.com/blog/chatgpt/

Other Generative AI Models

Generative Adversarial Networks (GANs)

- Two separate networks compete against each other:
    - A generator network takes a random input and aims to learn a probability distribution matching that of the training set
    - A discriminator network is learning to classify whether a sample comes from the training set or the generator
- Ideally, the generator should eventually match the true data distribution while the discriminator is optimised but doing no better than random guessing

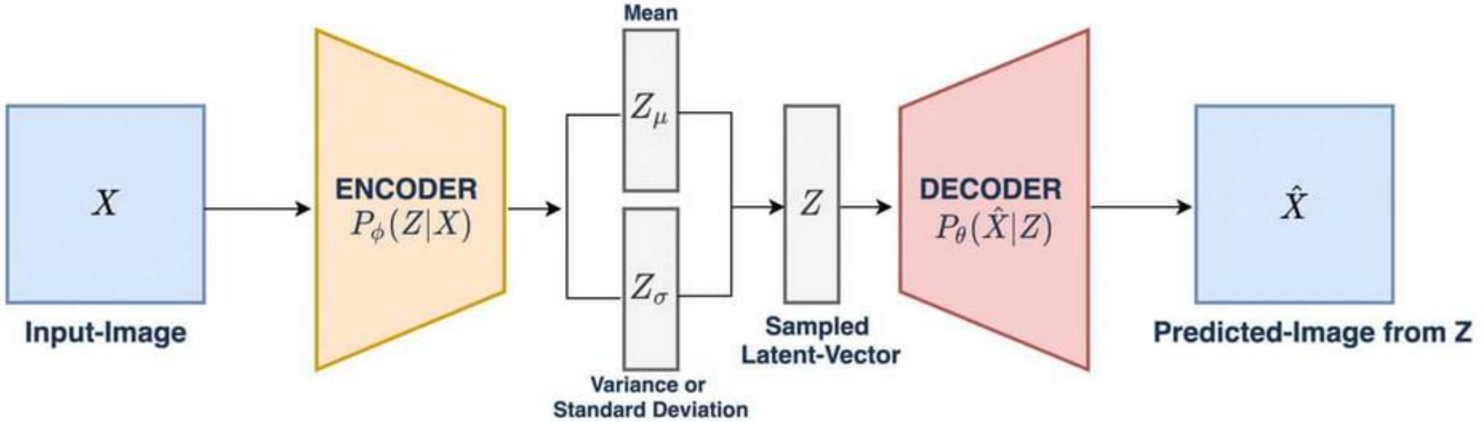


https://sthalles.github.io/intro-to-gans/

Other Generative AI Models

Variational Autoencoders (VAEs)

- Commonly used for image generation/reconstruction
- Encoder: inputs are mapped onto the mean and covariance of a normal distribution in some underlying vector space
- A random vector is then sampled from this normal distribution
- Decoder: provides an output from this random vector

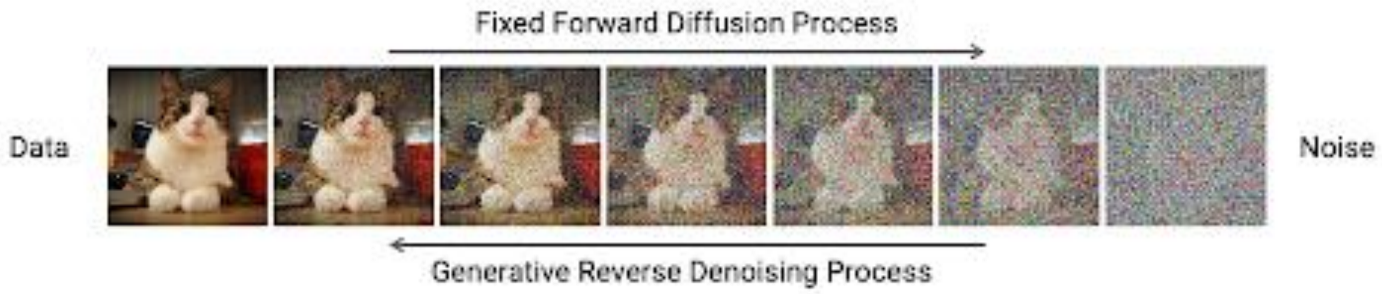


https://learnopencv.com/variational-autoencoder-in-tensorflow/

Other Generative AI Models

Diffusion Models

- Noise is added to an image in stages, then the noise removal process for each stage is learned (e.g. via a convolutional neural network)
- Later given noise, apply this noise removal process to arrive at an image



https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/

- Primarily applied to image and video generation but also natural language generation, time series forecasting
- Demo: https://keras.io/guides/keras_cv/generate_images_with_stable_diffusion/

End of Presentation!