MODULE 13: KEY CONCEPTS IN AI & DL (REVIEW)

BA713 - Machine Learning & AI

VARIOUS APPROACHES TO AI

- AI→ ***"The science and engineering of making intelligent machines, especially intelligent computer programs".*** - John McCarthy

Narrow (Weak) AI:
- **Perform specific tasks, not learn new ones**
- **Train data programmed algorithms**
- **Google Assistant, Siri, Alexa**

Generalized (Strong) AI:
- **Machine with general intelligence like a human being**
- **Learn from experience, solve new problems**
- **AI-based Robot**

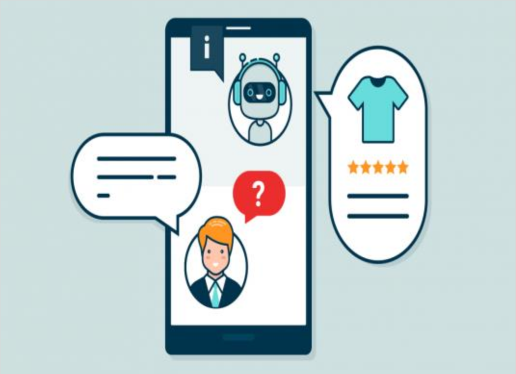
Super (Conscious) AI:
- **Human level consciousness**
- **Self-aware**
- **Not created yet→ difficult to measure consciousness**

Source: https://builtin.com/artificial-intelligence

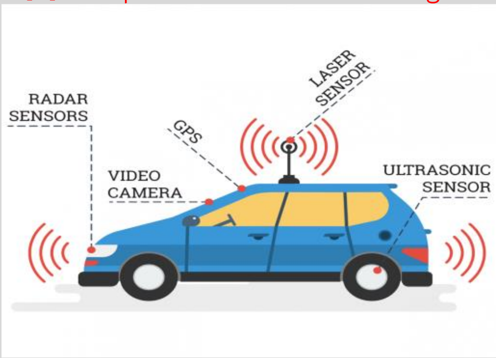
SOME APPLICATIONS OF AI

[1] Speech Recognition: Alexa, Siri

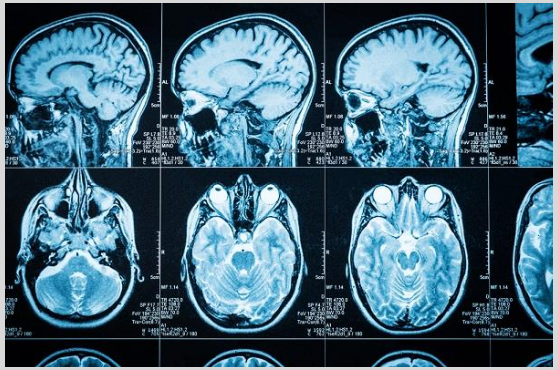
[2] Customer service: Chatbots (E-commerce sites)



[3] Computer Vision: Self-driving cars



[4] Computer Vision: Medical Imagining



[5] Recommendation Systems: Online shopping

[6] AI Stock Trading

[7] Advanced Security Systems : Network security, Credit card Fraud detection, spam detection

AI, MACHINE LEARNING & DEEP LEARNING

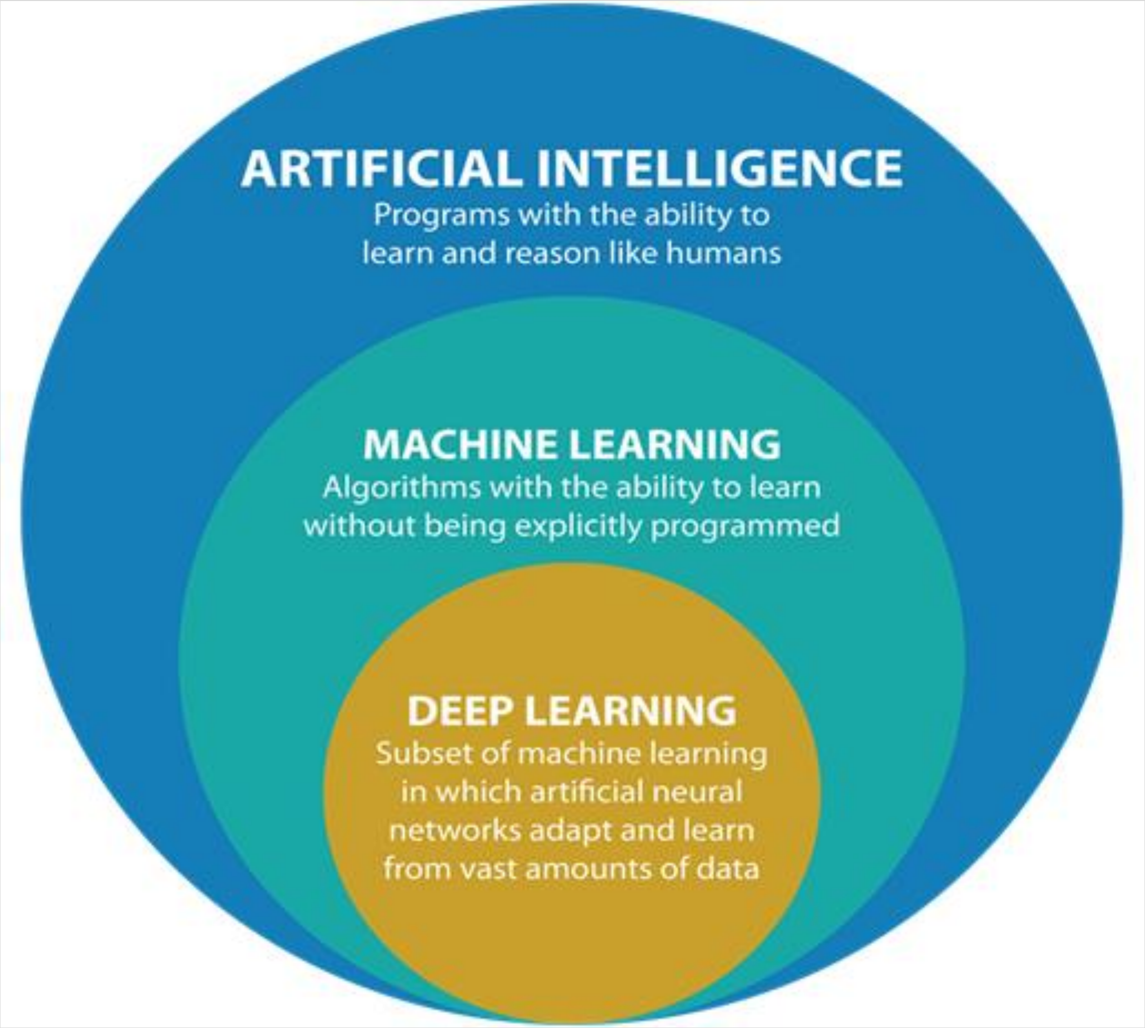
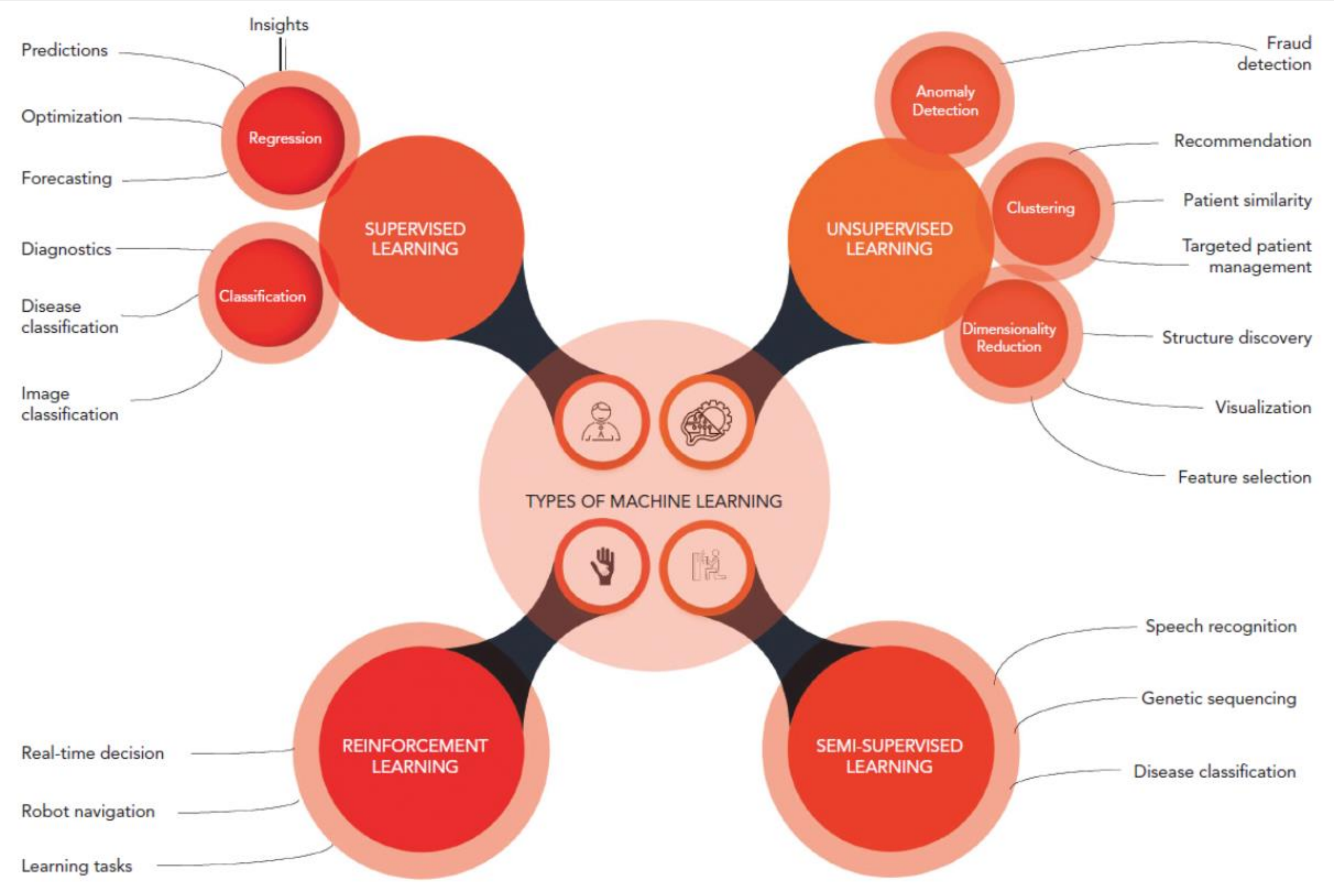


Image source: https://datacatchup.com/artificial-intelligence-machine-learning-and-deep-learning/

HOW DOES AN AI LEARN?



Source : https://www.ecrjournal.com/image-gallery/11492/9473/Types_of_Machine_Learning_with_Examples_of_Respective_Use

POPULAR OPEN DATA SOURCES

- Popular open data repositories
  - UC Irvine Machine Learning Repository (UCI)
  - Kaggle datasets
  - Amazon's AWS datasets

- Meta portals (they list open data repositories)
  - Data Portals
  - OpenDataMonitor
  - Quandl

- Other pages listing many popular open data repositories
  - Wikipedia's list of Machine Learning datasets
  - Quora.com
  - The datasets subreddit

ML/DL PIPELINE

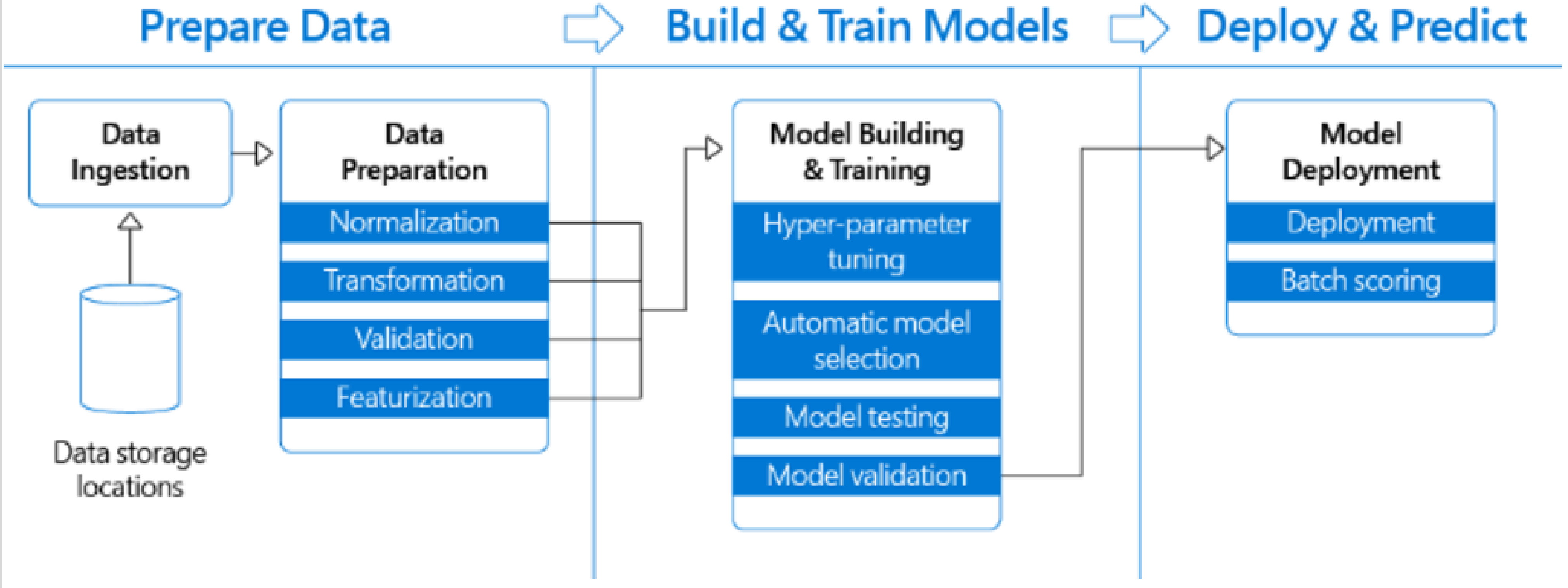


Image Source: https://medium.com/microsoftazure/how-to-accelerate-devops-with-machine-learning-lifecycle-management-2ca4c86387a0

SHALLOW NEURAL NETWORKS OVERVIEW

Perceptron is one of the simplest Artificial Neural Network (ANN) architectures, invented in 1957 by Frank Rosenblatt.

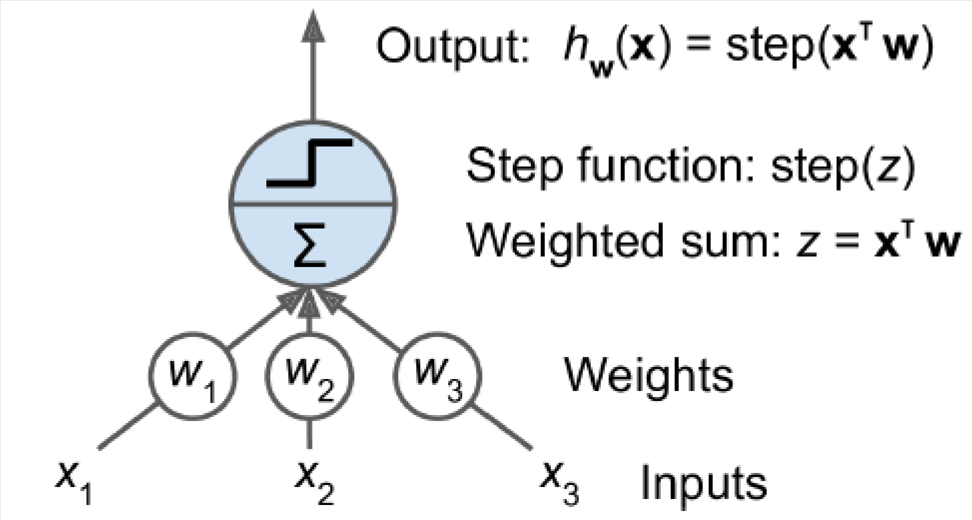


Figure a: Simple Perceptron/Threshold Logic Unit (TLU)/Linear Threshold Unit (LTU)

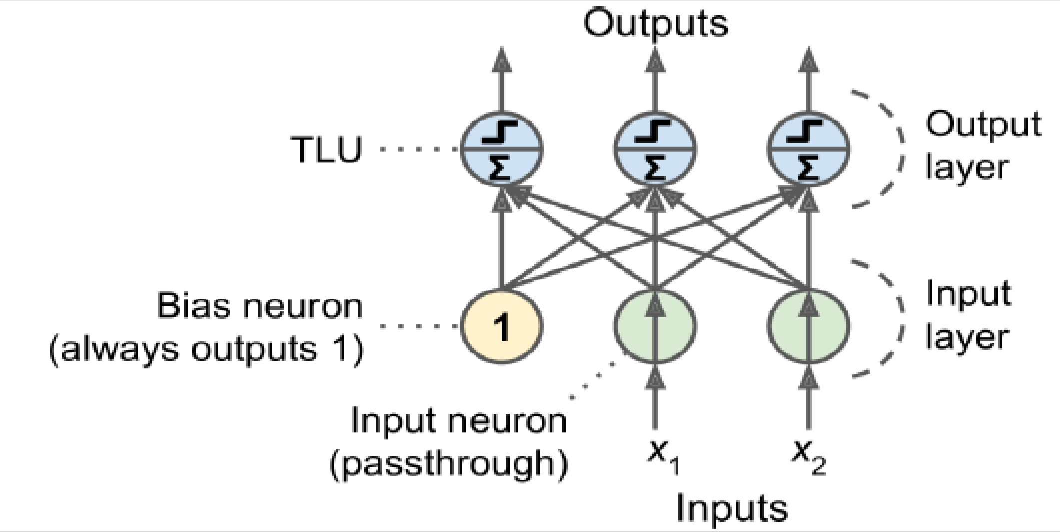


Figure b: Simple Perceptron with 2 input neurons, one bias neuron & 3 output neurons

Weighted sum of inputs: $z = w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{x}^T\mathbf{w}$

$$h_{\mathbf{W},\mathbf{b}}(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b})$$

W→ Weight matrix
b→ Bias vector
X→ feature matrix
ø → activation function

Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

GRADIENT DESCENT

- generic optimization algorithm capable of finding optimal solutions to a wide range of problems
- Tweak parameters iteratively in order to minimize a cost function (loss or error of a MLP)
- measures the local gradient of the error function w.r.t the **parameter vector θ**, and it goes in the direction of descending gradient
- Once the gradient is zero, you have reached a minimum
- start by filling θ with random values→ **random initialization**
- improve it gradually, taking one baby step at a time, each step attempting to decrease the cost function, until the algorithm converges to a minimum.

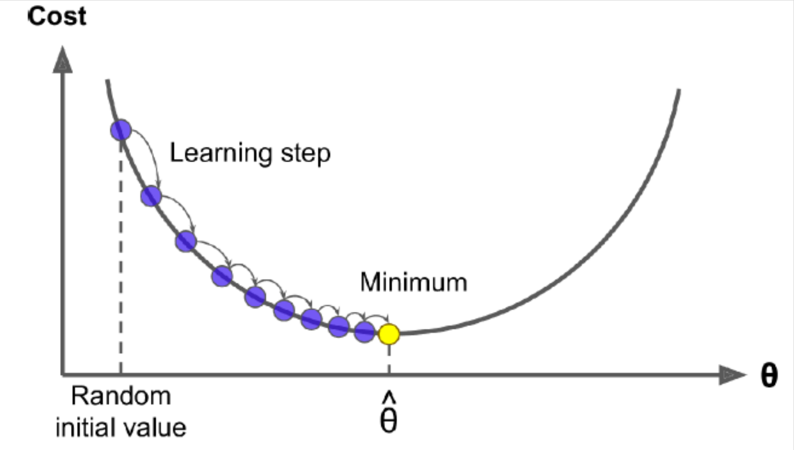


Figure a: Gradient Descent Global Minimum

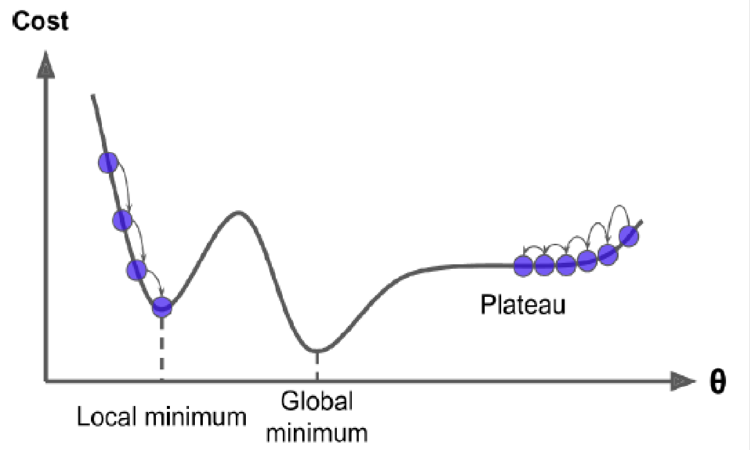
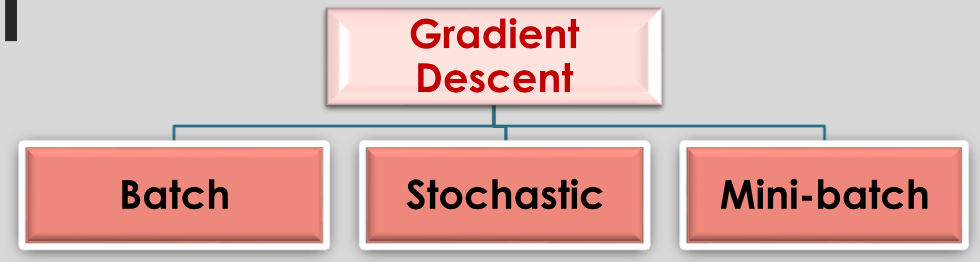


Figure b: Gradient Descent pitfalls

Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

FLAVORS OF GRADIENT DESCENT



Batch/ Full Gradient Descent:

- uses the whole batch of training data at every step→ slow for large datasets

Stochastic Gradient Descent:

- picks a random instance in the training set at every step and computes the gradients based only on that single instance→ faster for large datasets
- Random nature→ the cost function will bounce up and down, decreasing only on average
- once it gets to the minimum, there it will continue to bounce around, never settling down→ good but not optimal
- ensure to shuffle the instances during training such that they are not sorted by label

Mini-batch Gradient Descent:

- computes the gradients on small random sets of instances called mini-batches
- Performance boost, less erratic with larger mini-batches

Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

BACKPROPAGATION TRAINING ALGORITHM FOR MULTI-LAYER PERCEPTRON (MLP)

Input layer:

1) One mini-batch at a time (e.g. 64 instances)

2) Go through full train set multiple times

3) Each pass is an **epoch**

Hidden layer(s):

1) Pass mini-batch to hidden layer

2) Output of all neurons passed to next layer until it reaches output layer

Output layer:

1) Make predictions for each mini-batch

2) Measure the error by comparing desired output and generated output

3) Compute how much error each output connection has contributed

Backpropagation:

1) Backpropagate the error gradient until input layer of the network

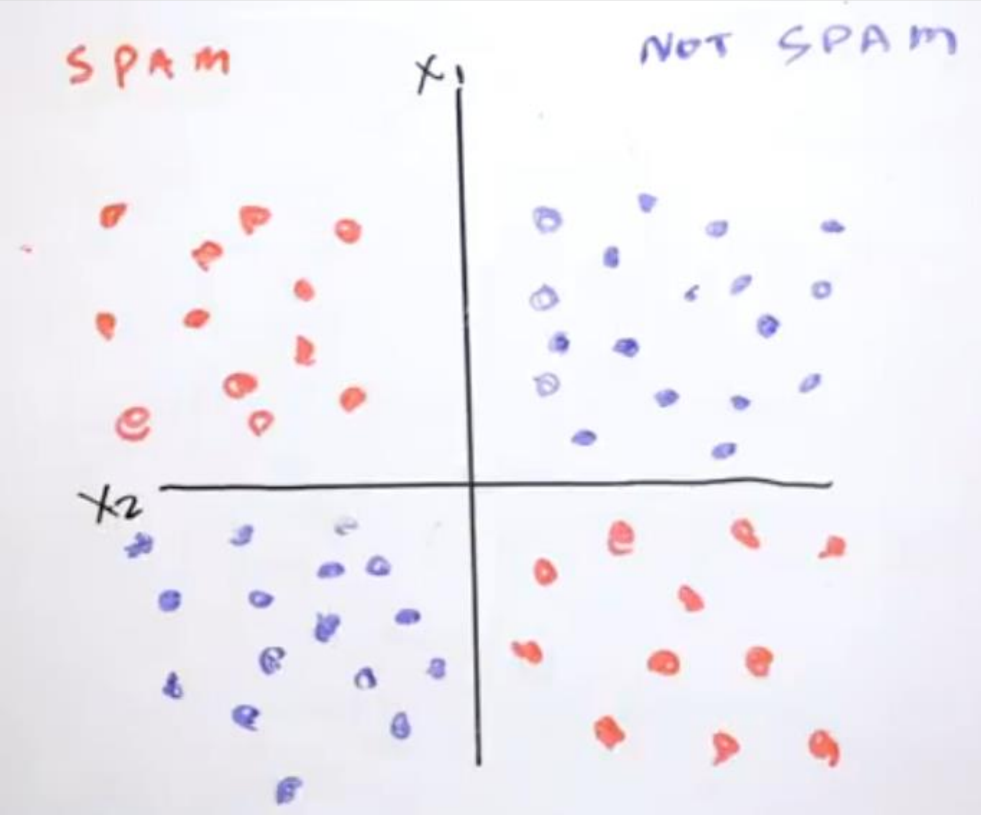
Gradient Descent:

1) Tweak all connection weights in network using the error gradients

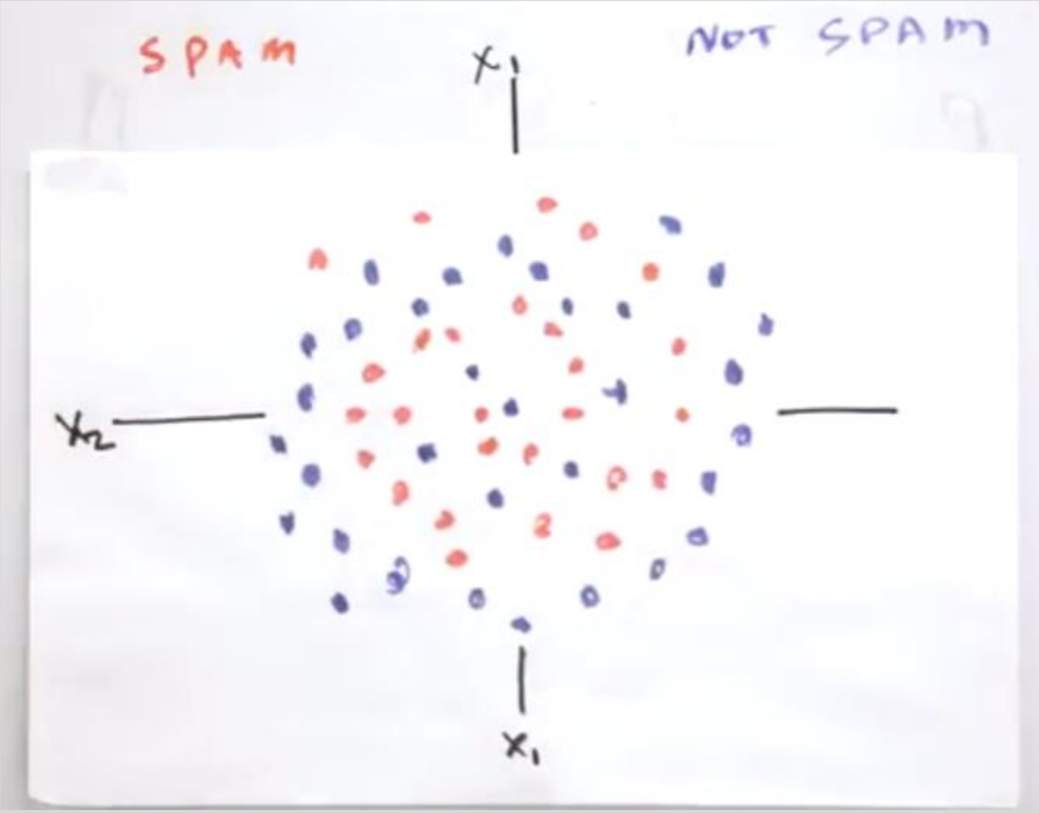
2) Restart the training process

Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

WHY DO WE NEED NEURAL NETWORKS?

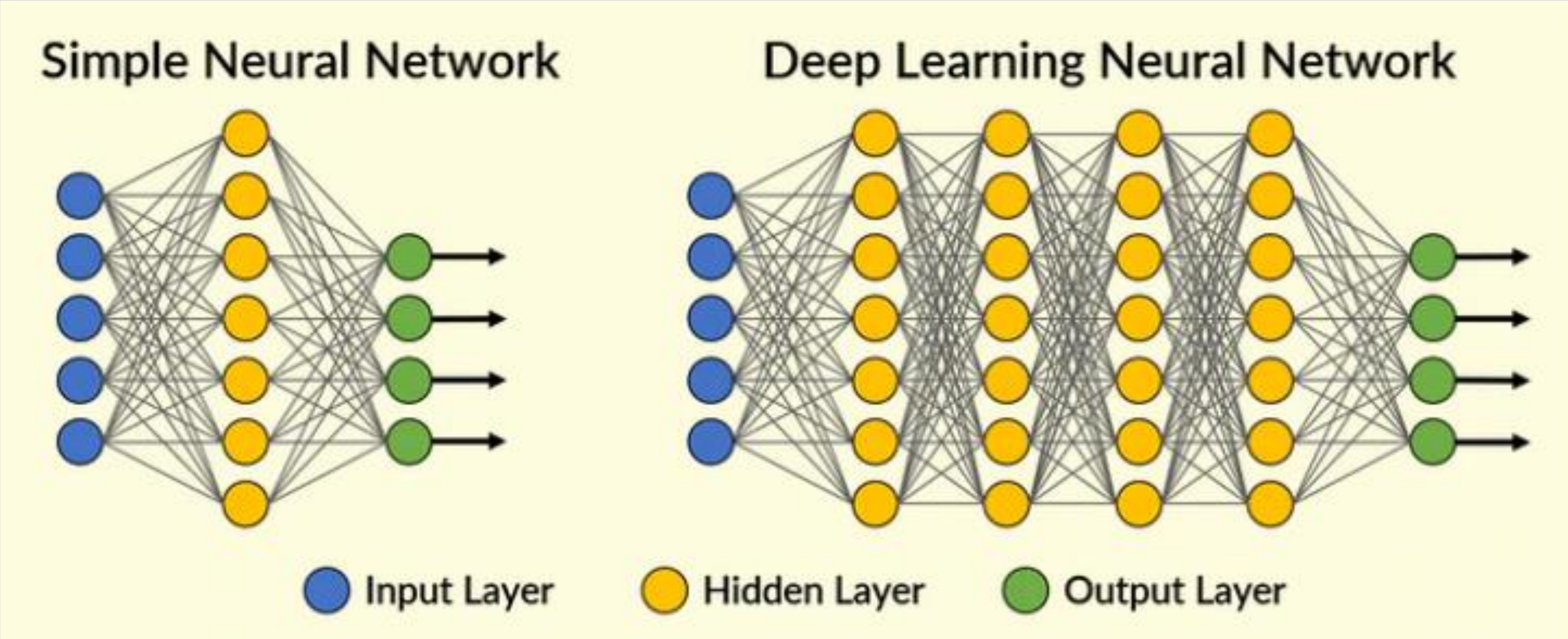


Simple non-linear problem



Complex non-linear problem

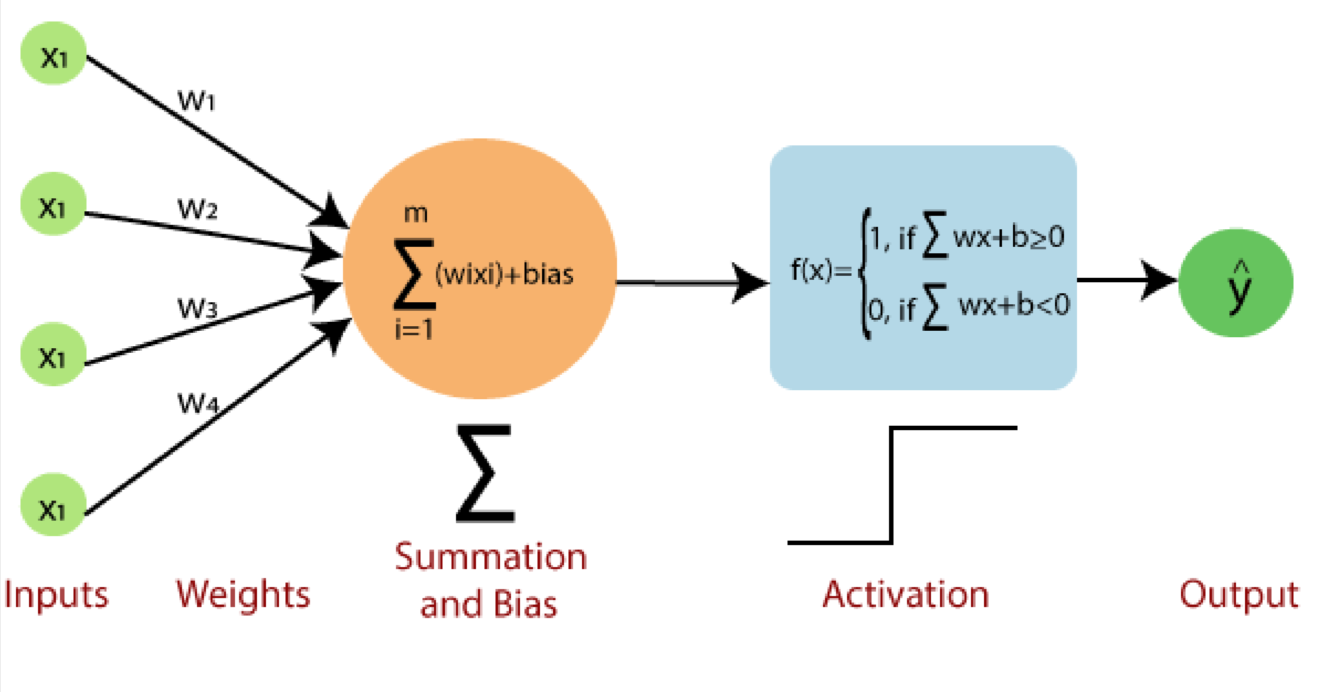
Source: https://developers.google.com



Source: https://www.securityinfowatch.com/

STRUCTURAL BUILDING BLOCK FOR DEEP NEURAL NETWORKS

(Perceptron)



$$h_{\mathbf{W},\mathbf{b}}(\mathbf{X}) = \phi(\mathbf{XW} + \mathbf{b})$$

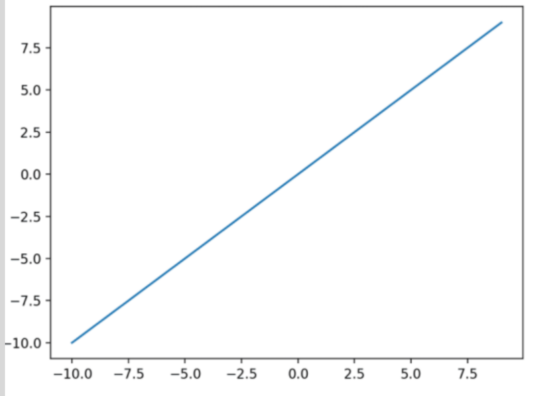
W→ Weight matrix
b→ Bias vector
X→ feature matrix
Ø → non-linear activation function
h→ Output
XW→ Linear combination of inputs
XW= $\sum_{i=1}^{m} x_i \, w_i$

Source: https://medium.com/nerd-for-tech/flux-prediction-using-single-layer-perceptron-and-multilayer-perceptron-cf82c1341c33

ACTIVATION/TRANSFER/SQUASHING FUNCTION

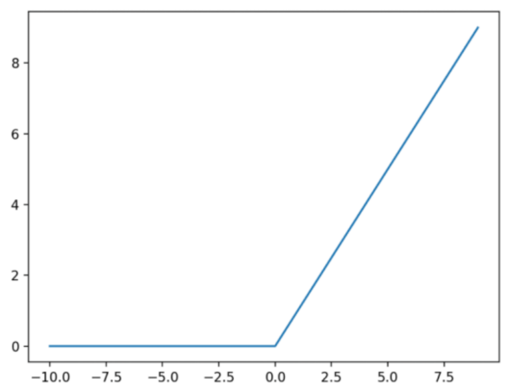
- defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network

$$f(x) = w^T x + b$$



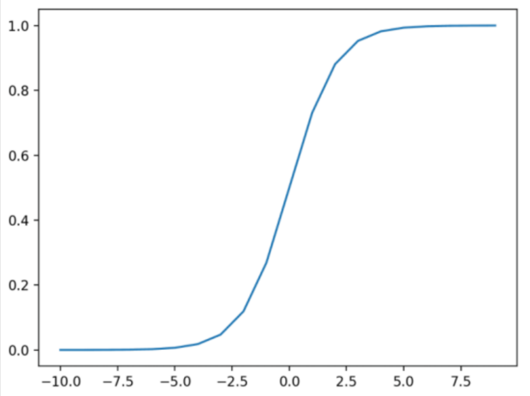
Linear

$$f(x) = max\,(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases}$$



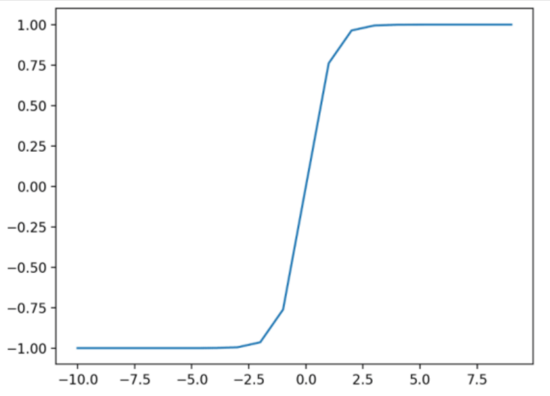
Rectified Linear Unit(ReLU)

$$f(x) = \left(\frac{1}{(1 + exp^{-x})} \right)$$



Sigmoid

$$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right)$$

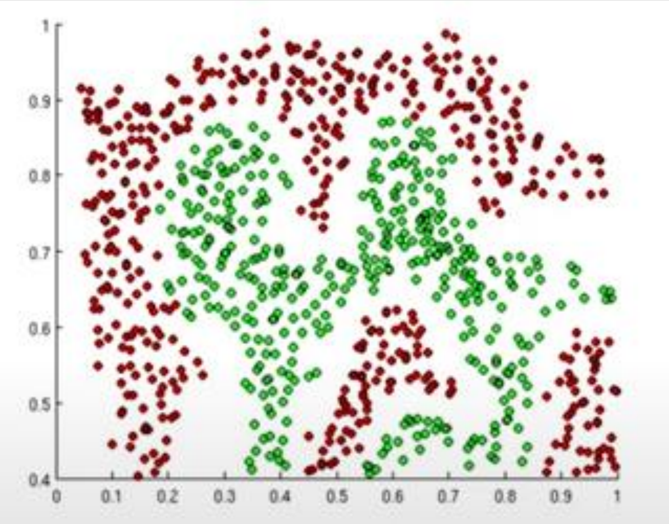


Hyperbolic Tangent (Tanh)

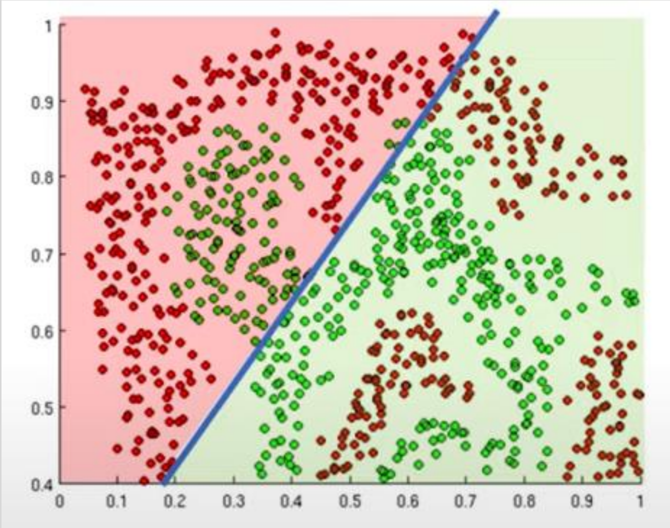
Source: https://machinelearningmastery.com/

IMPORTANCE OF ACTIVATION FUNCTIONS

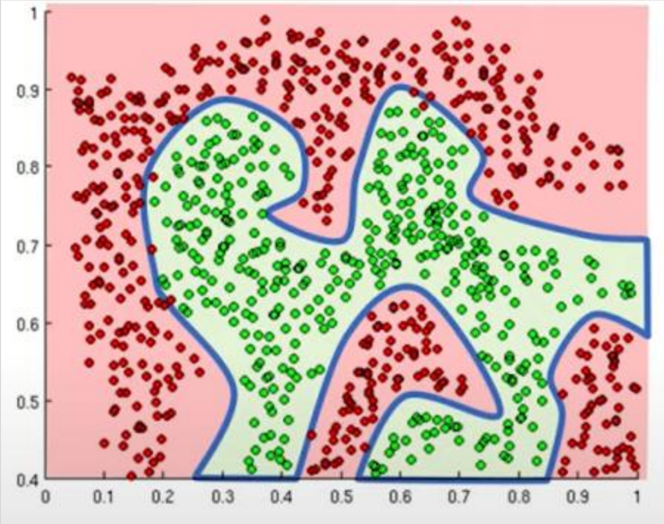
- Non-linear function
- perform complex computations in the hidden layers and then transfer the result to the output layer
- Introduce non-linearities into the network



Identify spam (red) and non-spam (green)



Linear activation functions produce linear decisions no matter the size of the network



Non-linearities allow us to approximate complex functions

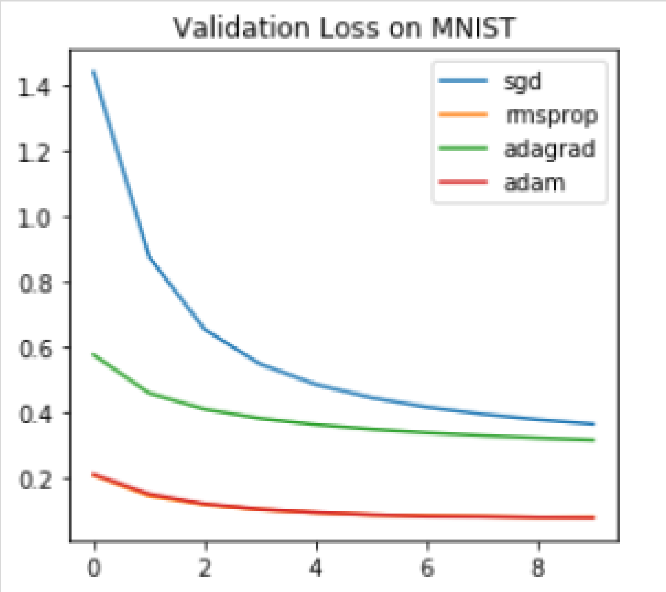
Source: introtodeeplearning.com

ADAPTIVE LEARNING RATES FOR DL MODELS

- Adapt to the landscape
- Not fixed→ can change based on the gradient
- Based on size of the weight
- Or Learning speed
- https://keras.io/api/optimizers/

Popular Adaptive Keras Optimizers

1. **Adagrad**
2. **Adadelta**
3. **RMSprop**
4. **Adam**



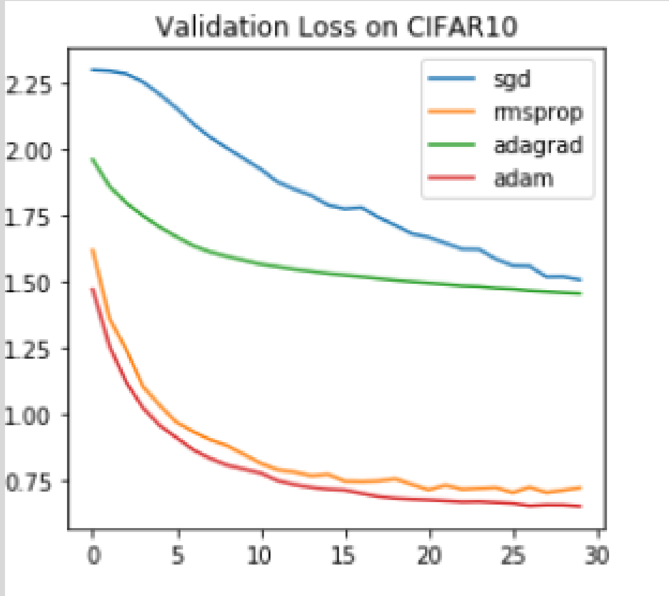
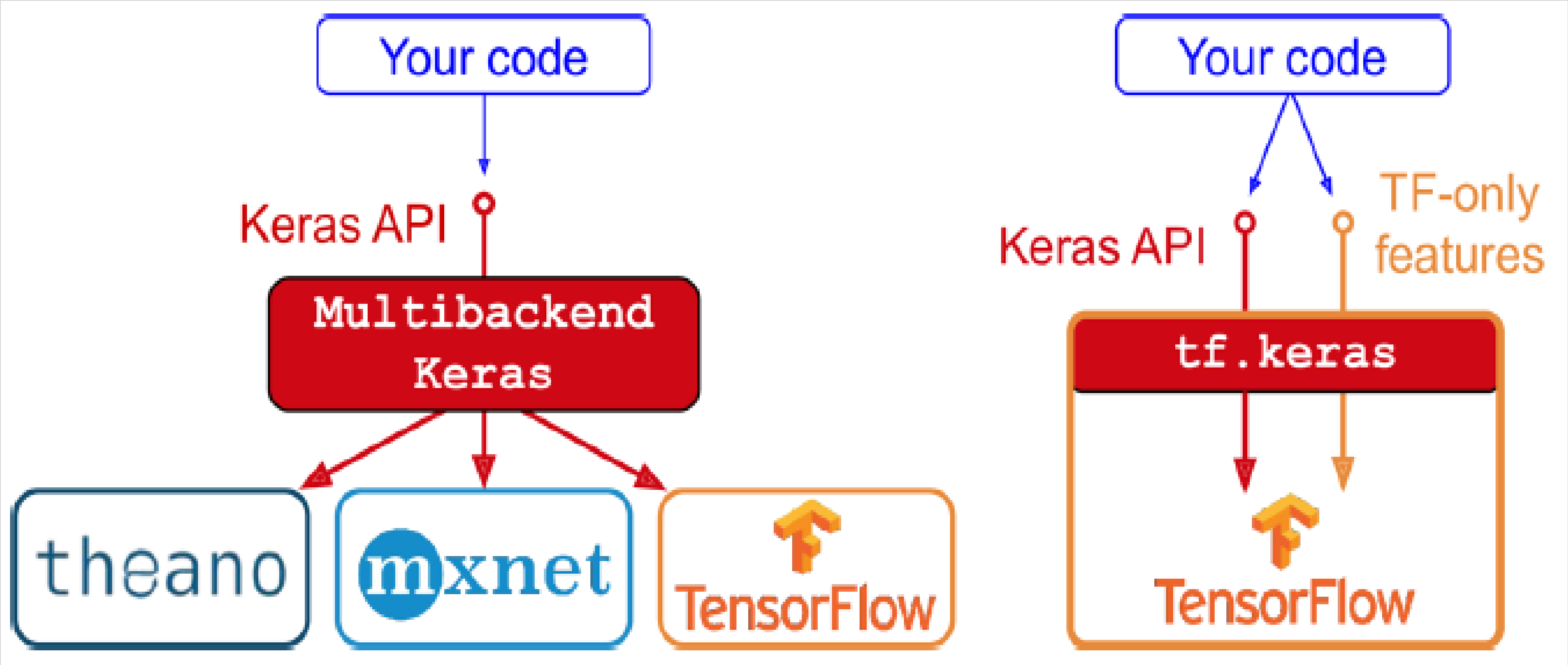


Image source: https://heartbeat.fritz.ai/an-empirical-comparison-of-optimizers-for-machine-learning-models-b86f29957050

IMPLEMENTATIONS OF THE KERAS API

- Keras is a high-level Deep Learning API that allows you to easily build, train, evaluate, and execute all sorts of neural networks.
- Its documentation (or specification) is available at **https://keras.io/.**



Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

KEY NEURAL NETWORK ARCHITECTURES

- Vanilla Neural Network (NN)
- Densely Connected Network or Deep Neural Network (DNN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Generative Adversarial Network (GAN)
- Deep Autoencoder (DAE) & Variants
- Variational Autoencoder (VAE)
- Conditional Variational Autoencoder (CVAE)
- Deep Q Network (DQN)

DNN CLASSIFIER

Binary-classification

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(num_input_features,)))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy')
```

Multi-classification

```
model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(num_input_features,)))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(num_classes, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy')
```

Source : Deep Learning with Python by François Chollet

1D, 2D and 3D Convnet (CNN)

1D CNN

```
import keras

from keras.layers import Conv1D

model = keras.models.Sequential()

model.add(Conv1D(1, kernel_size=5, input_shape = (120, 3)))

model.summary()
```

2D CNN

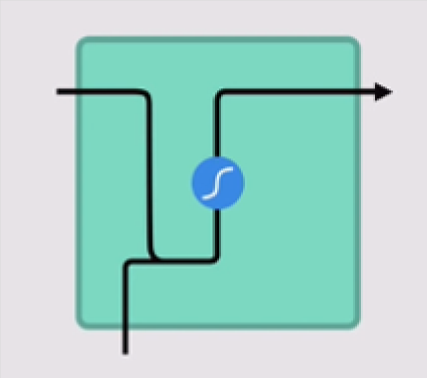
```
import keras

from keras.layers import Conv2D

model = keras.models.Sequential()

model.add(Conv2D(1, kernel_size=(3,3), input_shape = (128, 128, 3)))

model.summary()
```

3D CNN

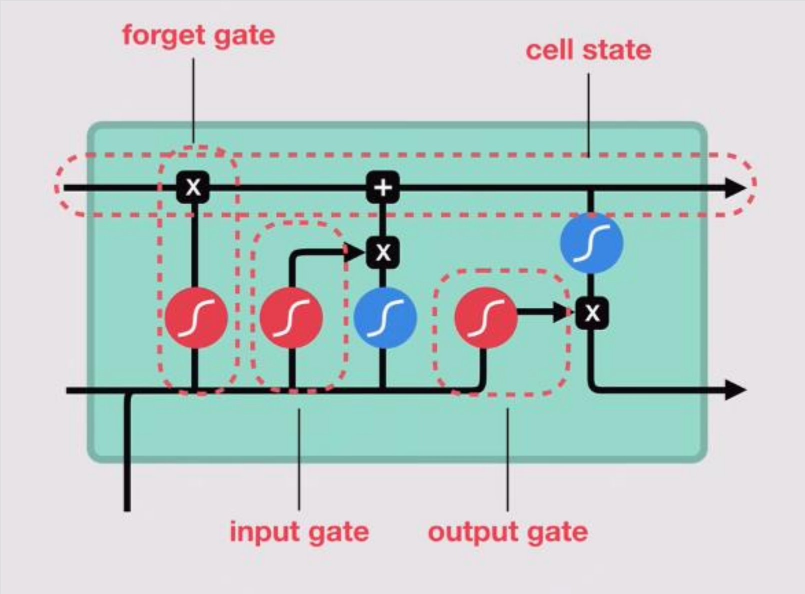
```
import keras

from keras.layers import Conv3D

model = keras.models.Sequential()

model.add(Conv3D(1, kernel_size=(3,3,3), input_shape = (128, 128, 128, 3)))

model.summary()
```

Source: https://www.programmersought.com/article/63014851723/

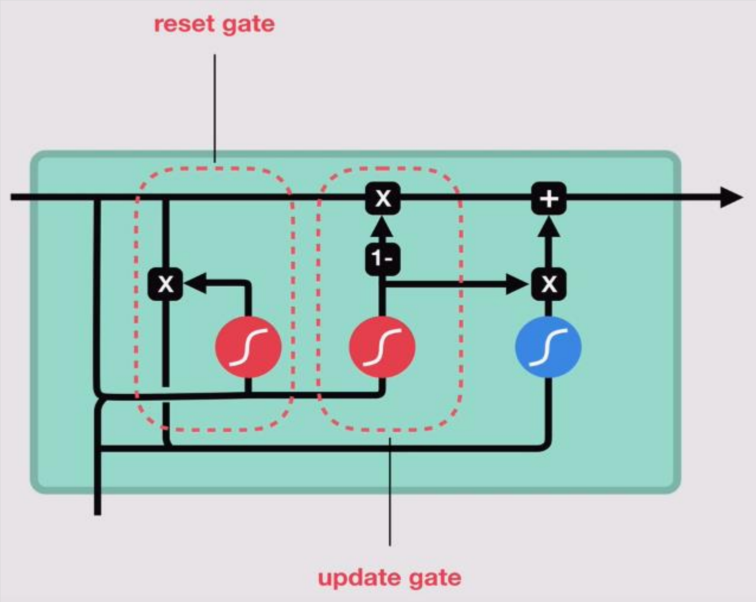
RNN AND ITS VARIANTS



Vanilla RNN



LSTM



GRU

Recurrent neural network (RNN)

Following is a single RNN layer for binary classification of vector sequences:

```
model = models.Sequential()
model.add(layers.LSTM(32, input_shape=(num_timesteps, num_features)))
model.add(layers.Dense(num_classes, activation='sigmoid'))

50model.compile(optimizer='rmsprop', loss='binary_crossentropy')
```

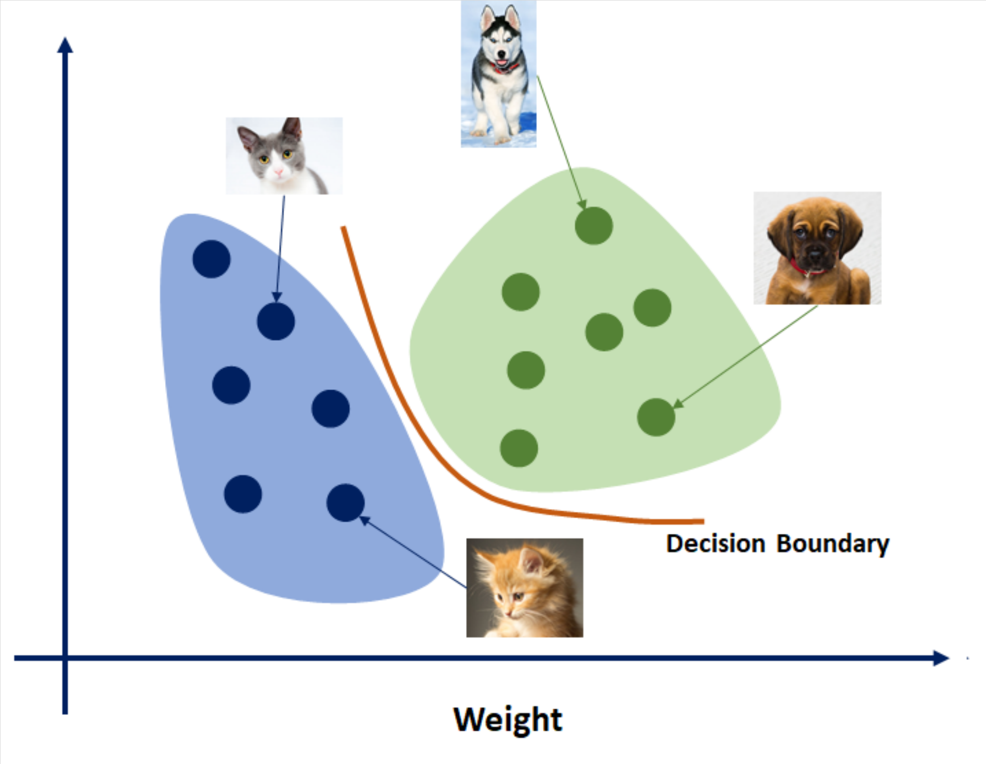
And this is a stacked RNN layer for binary classification of vector sequences:

```
model = models.Sequential()
model.add(layers.LSTM(32, return_sequences=True,
               input_shape=(num_timesteps, num_features)))
model.add(layers.LSTM(32, return_sequences=True))
model.add(layers.LSTM(32))
model.add(layers.Dense(num_classes, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy')
```

Source : Deep Learning with Python by François Chollet

GENERATIVE VERSUS DISCRIMINATIVE MODELS

Generative Model



Discriminative Model

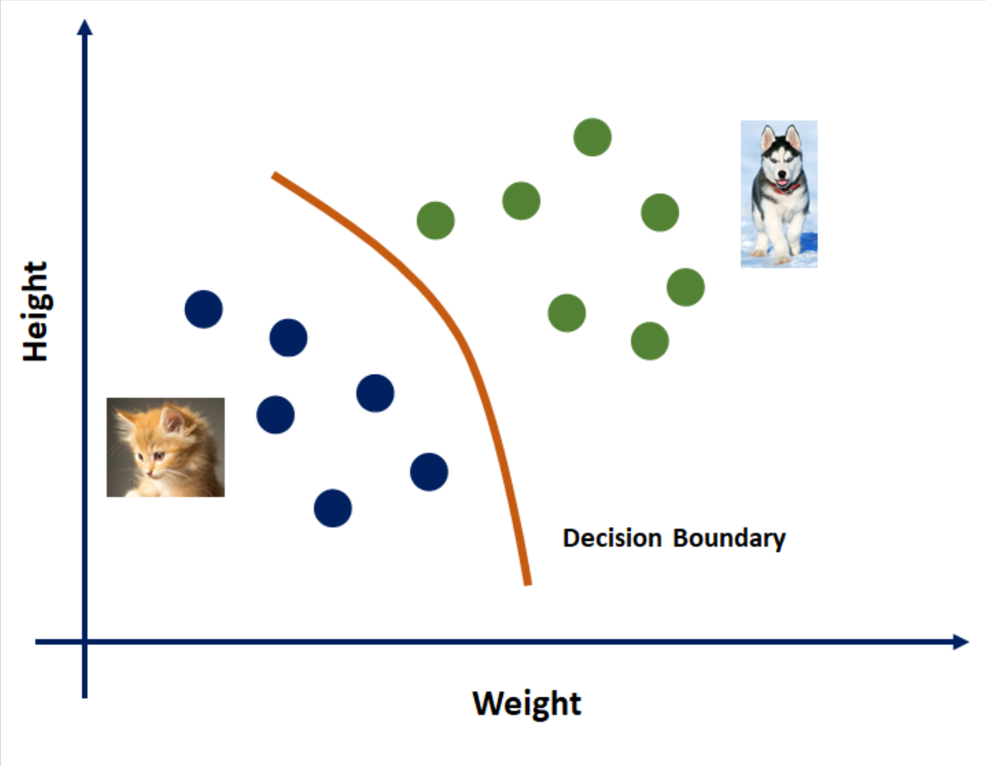
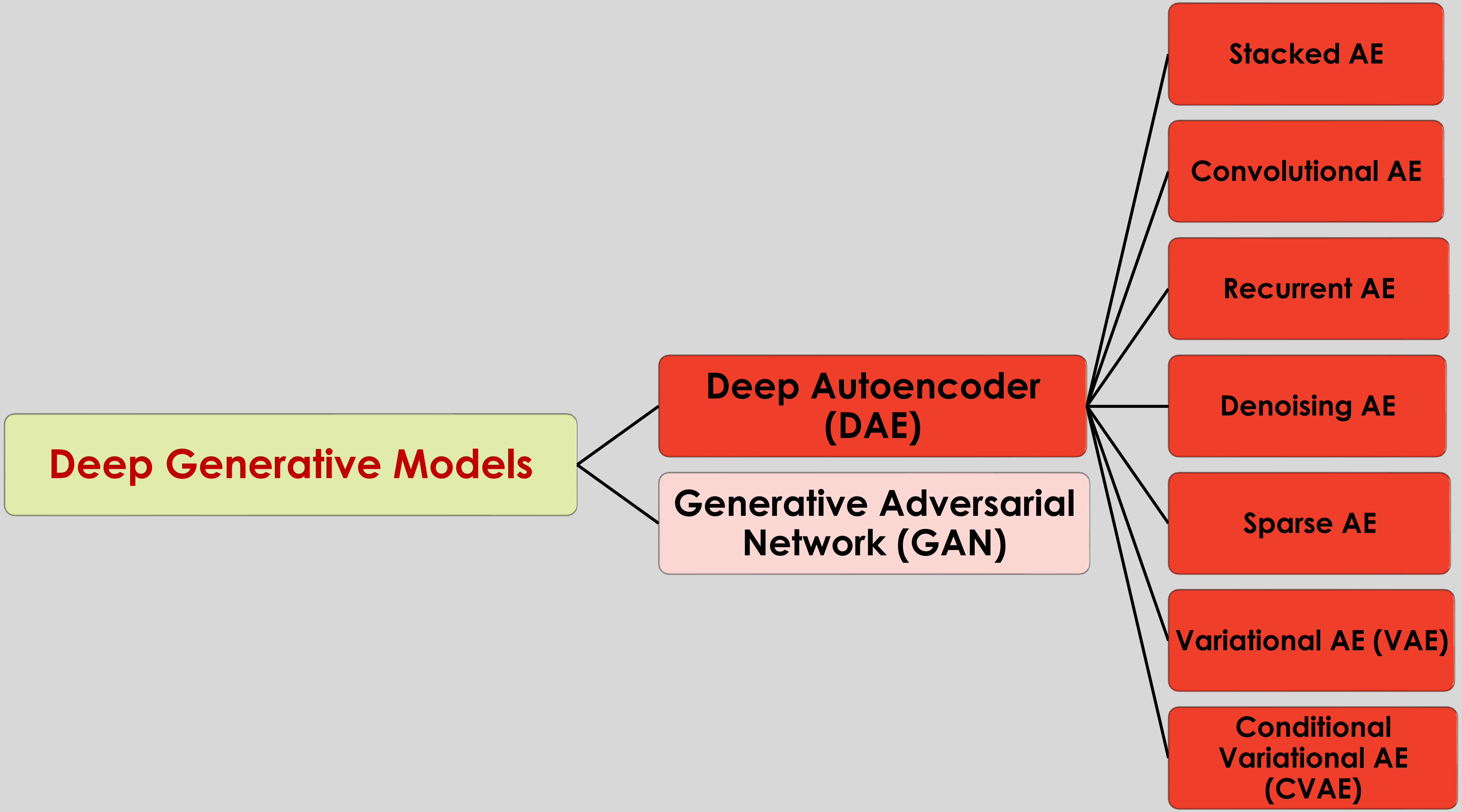


Image Source: https://medium.com/@jordi299/about-generative-and-discriminative-models-d8958b67ad32

DEEP GENERATIVE MODELS CLASSIFICATION

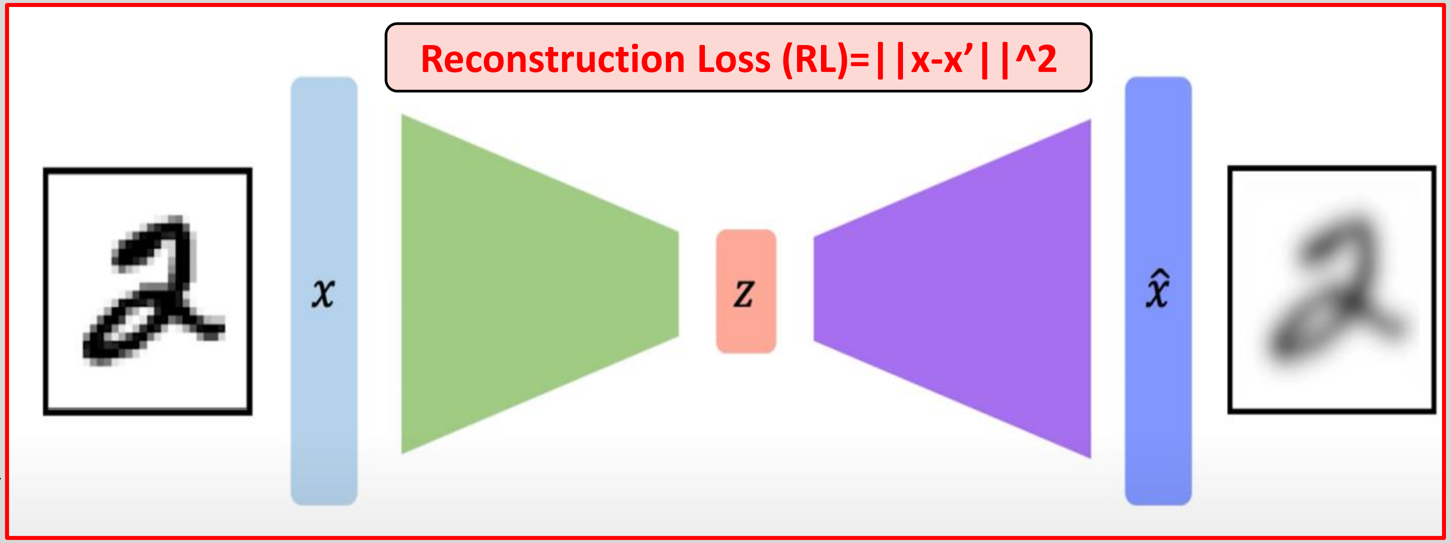


Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

DAE VERSUS VAE

Accepts input, compresses it and recreates the original input

DAE



Assumes source data has underlying probability distribution (Gaussian). Generates new data from lower dimensional latent space

VAE

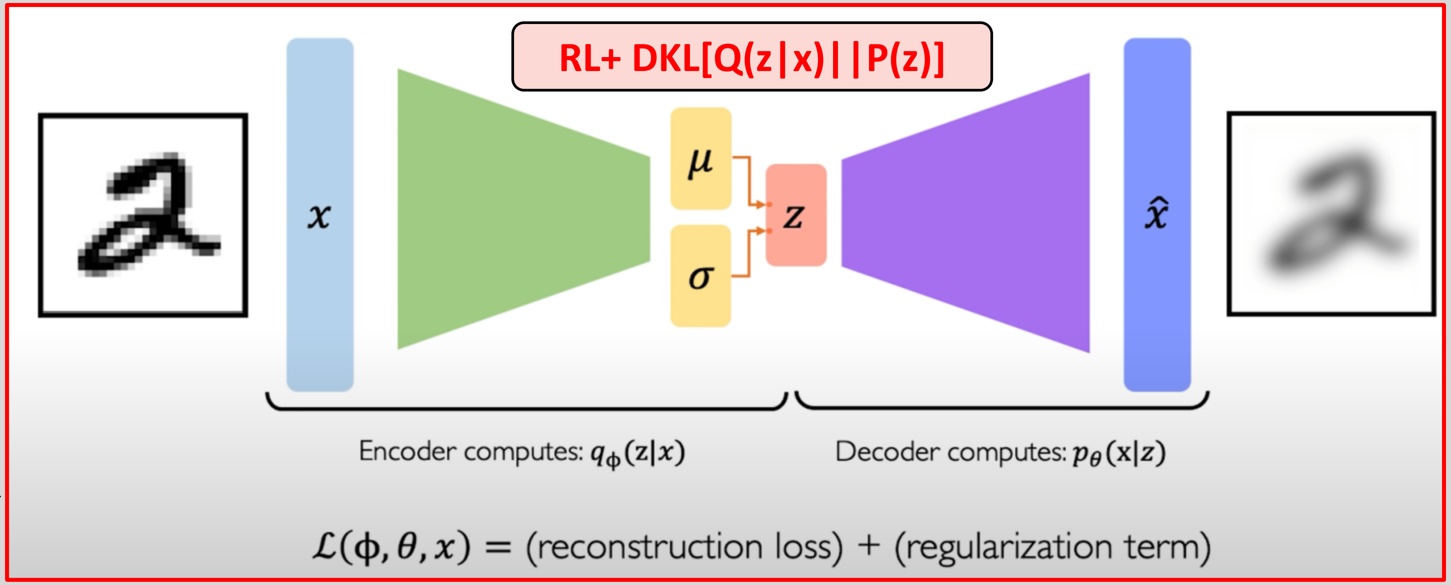
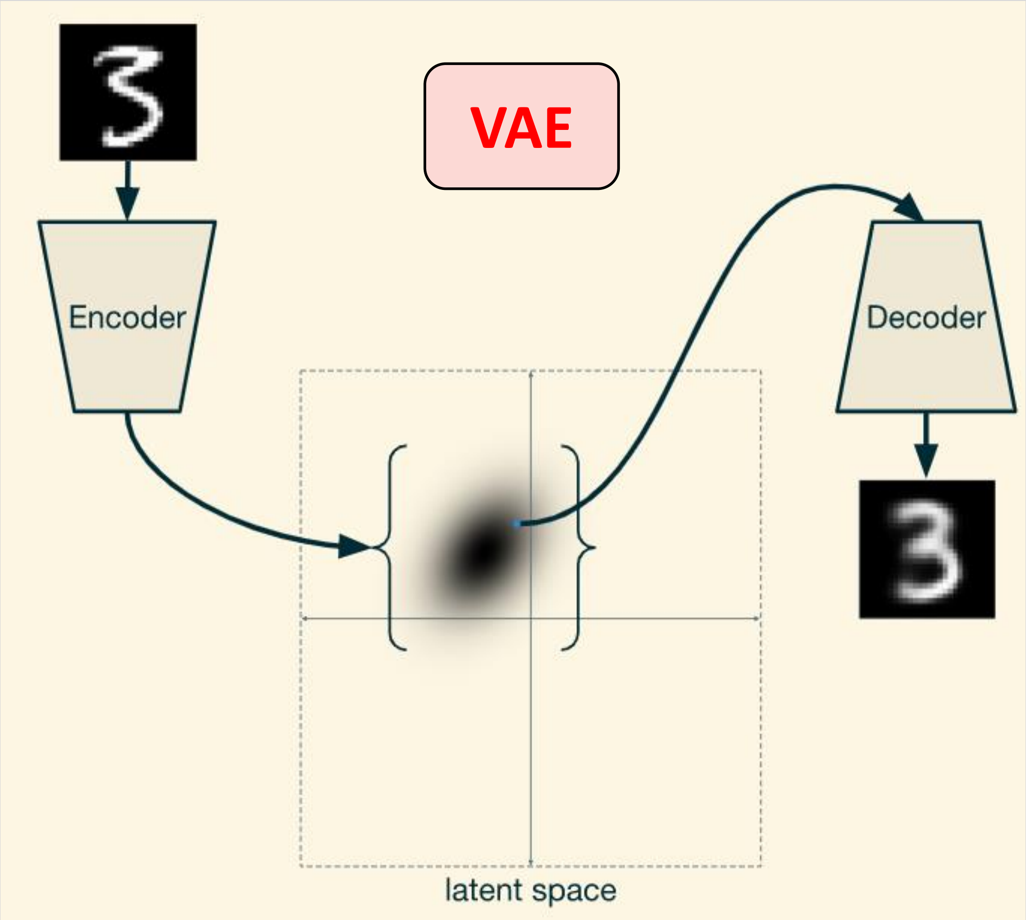
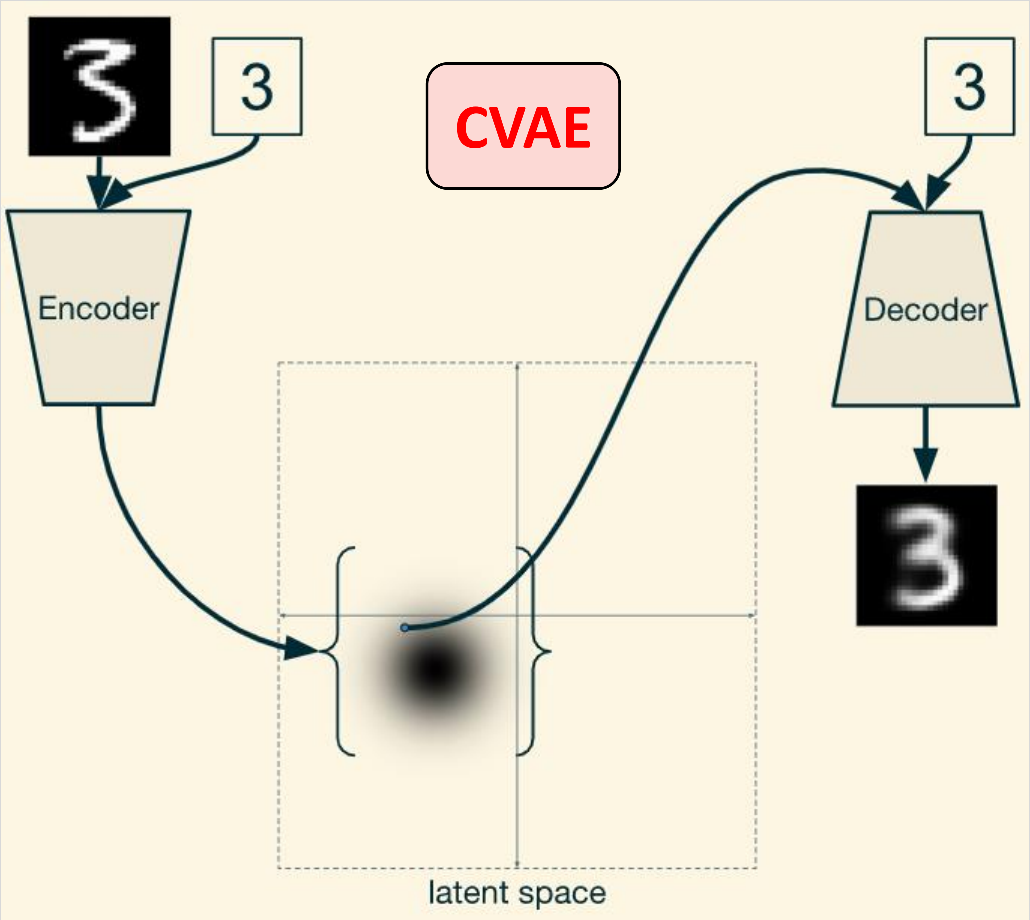


Image Source: introtodeeplearning.com

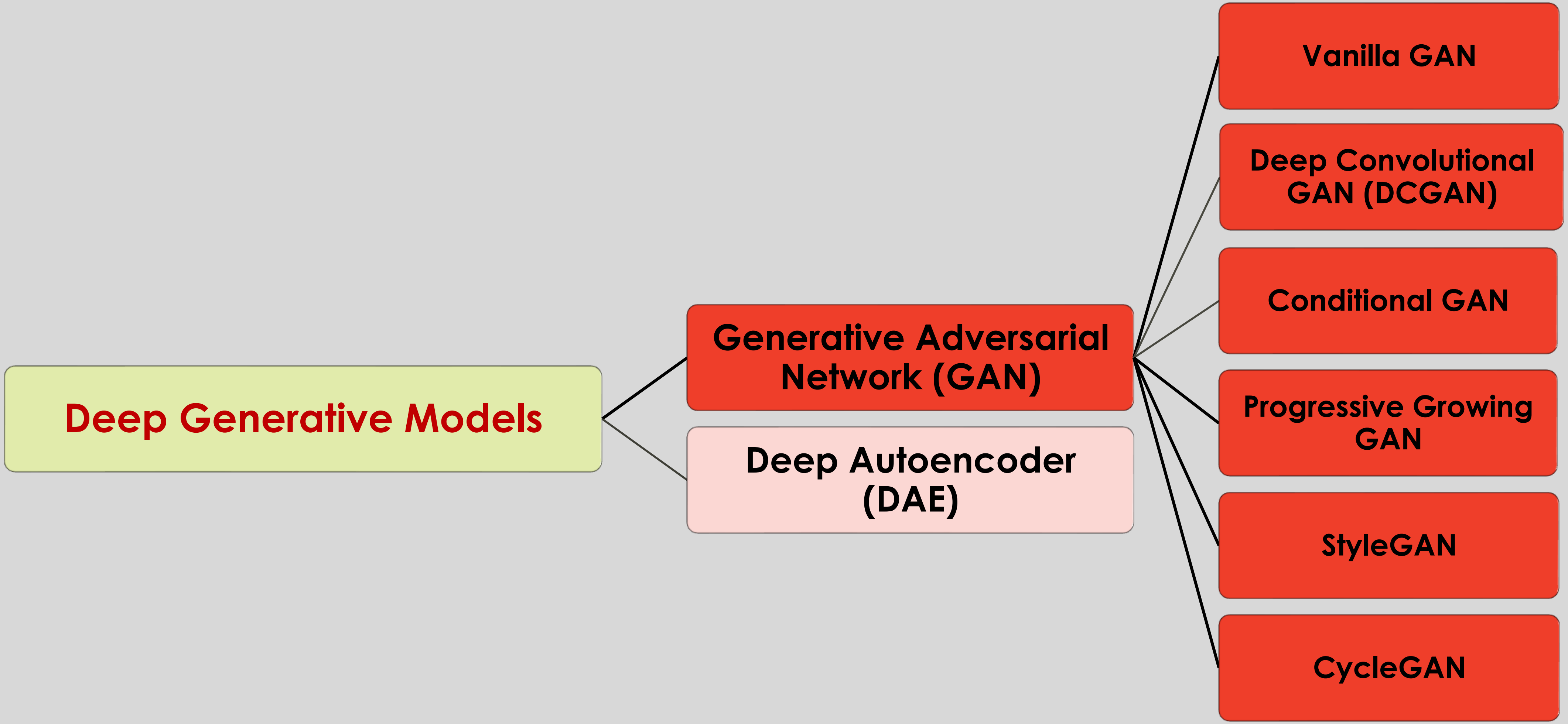
VAE VERSUS CVAE





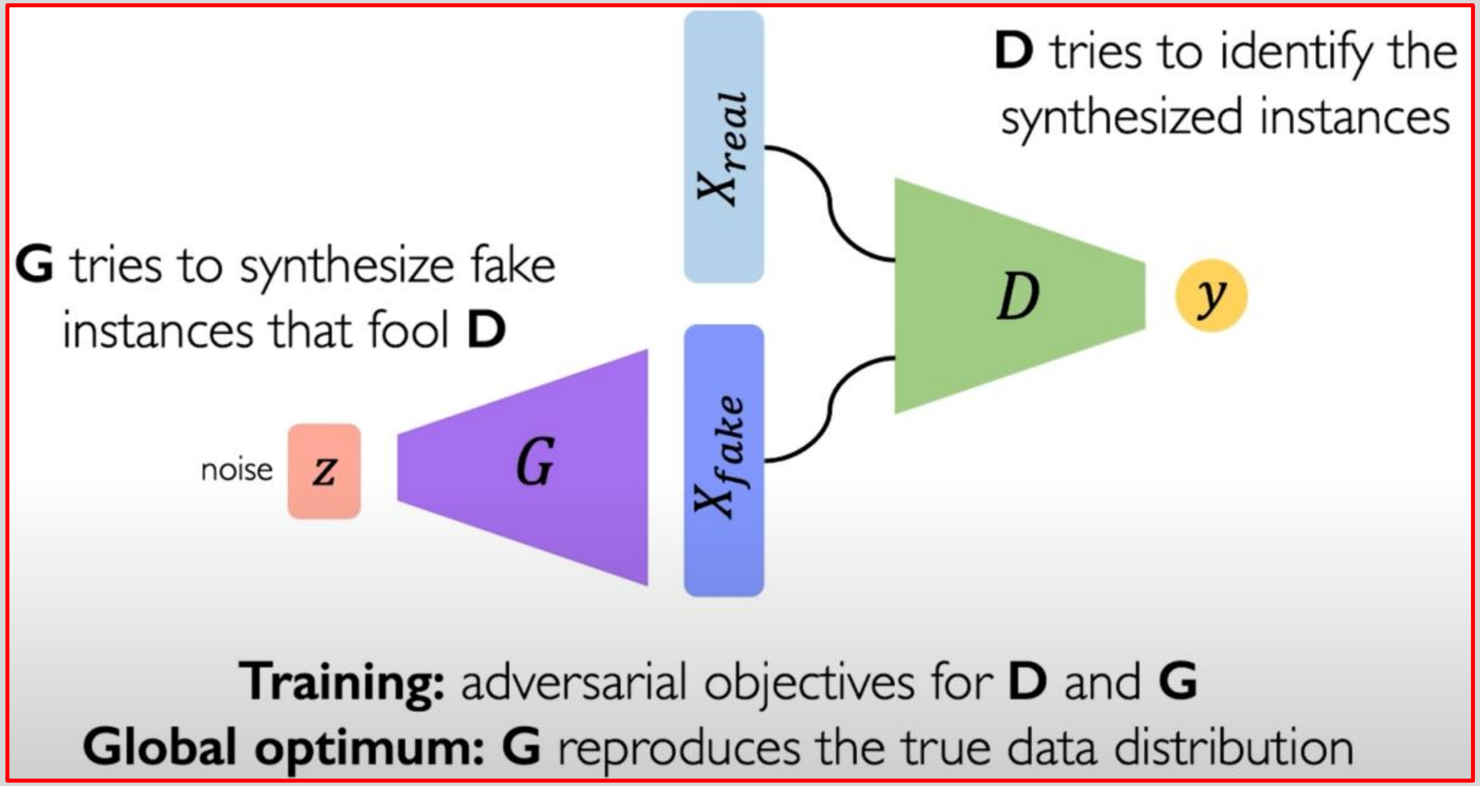
Source: https://ijdykeman.github.io/ml/2016/12/21/cvae.html

DEEP GENERATIVE MODELS CLASSIFICATION



Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

WORKING OF A GAN MODEL



$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[\ \log D(G(\mathbf{z})) + \log\left(1 - D(\mathbf{x})\right)\ \right]$$

Source: introtodeeplearning.com

REINFORCEMENT LEARNING

- **Learning by interacting with the environment**
- How a child or infant learns
- Interactions with the environment→ useful information
- Different from supervised learning where the algorithm is told what actions to take
- trial and error
- Discover the actions with the highest reward by itself
- Decision making problem

Examples:

- learning to drive a car, aware of environment, take actions
- The tic-tac-toe game

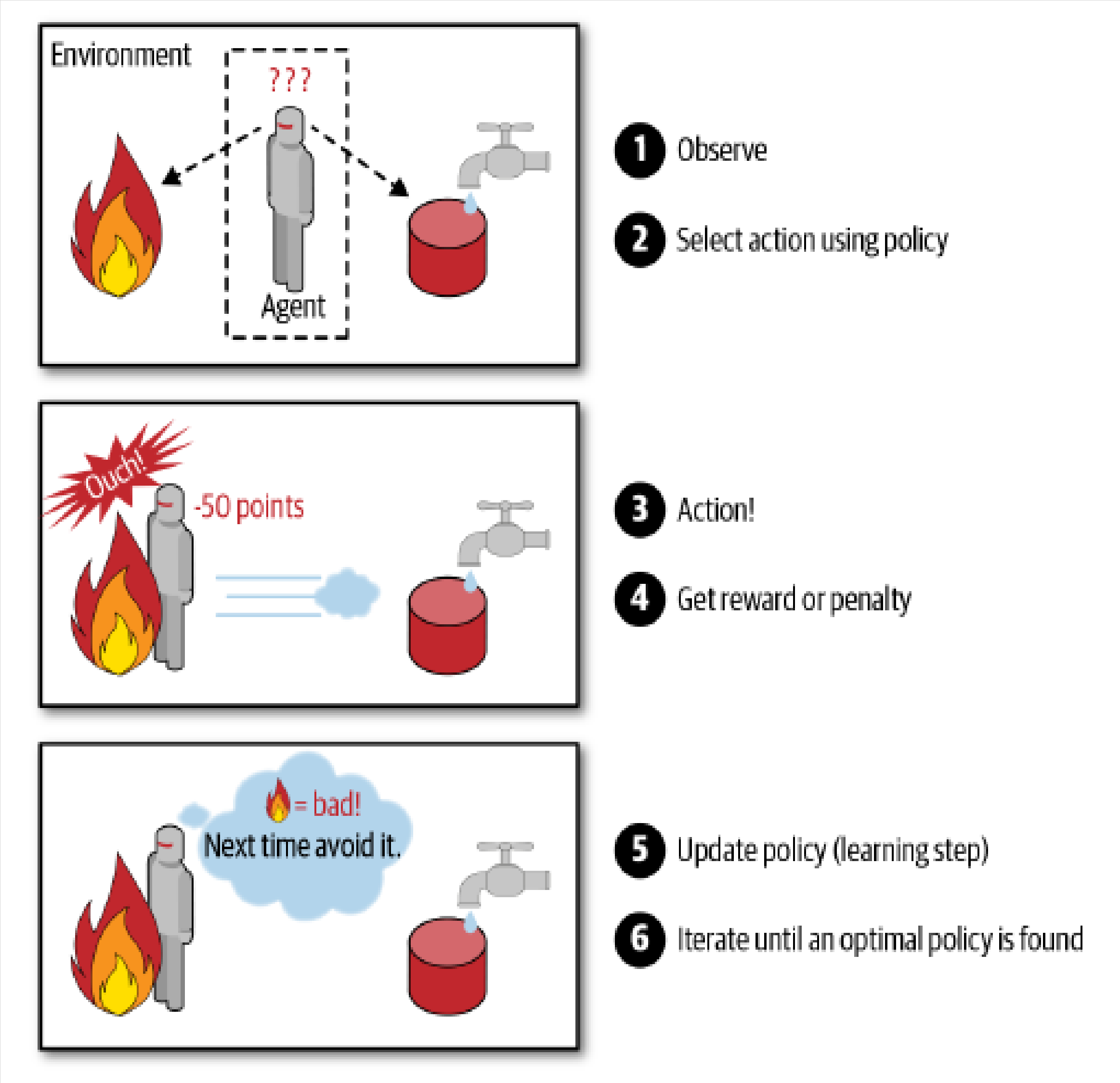
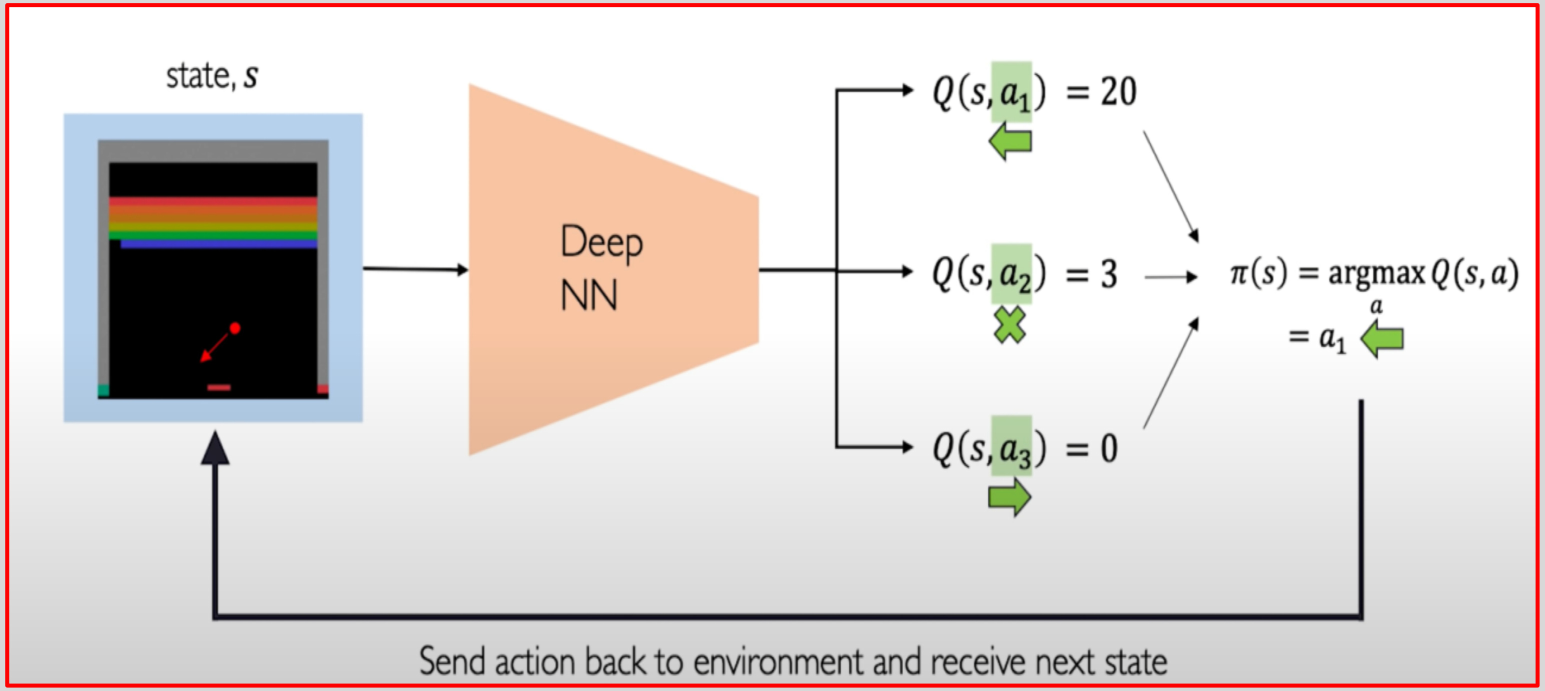


Figure Source: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Source: https://towardsdatascience.com/drl-01-a-gentle-introduction-to-deep-reinforcement-learning-405b79866bf4

DEEP Q-NETWORK (DQN): VALUE or Q LEARNING

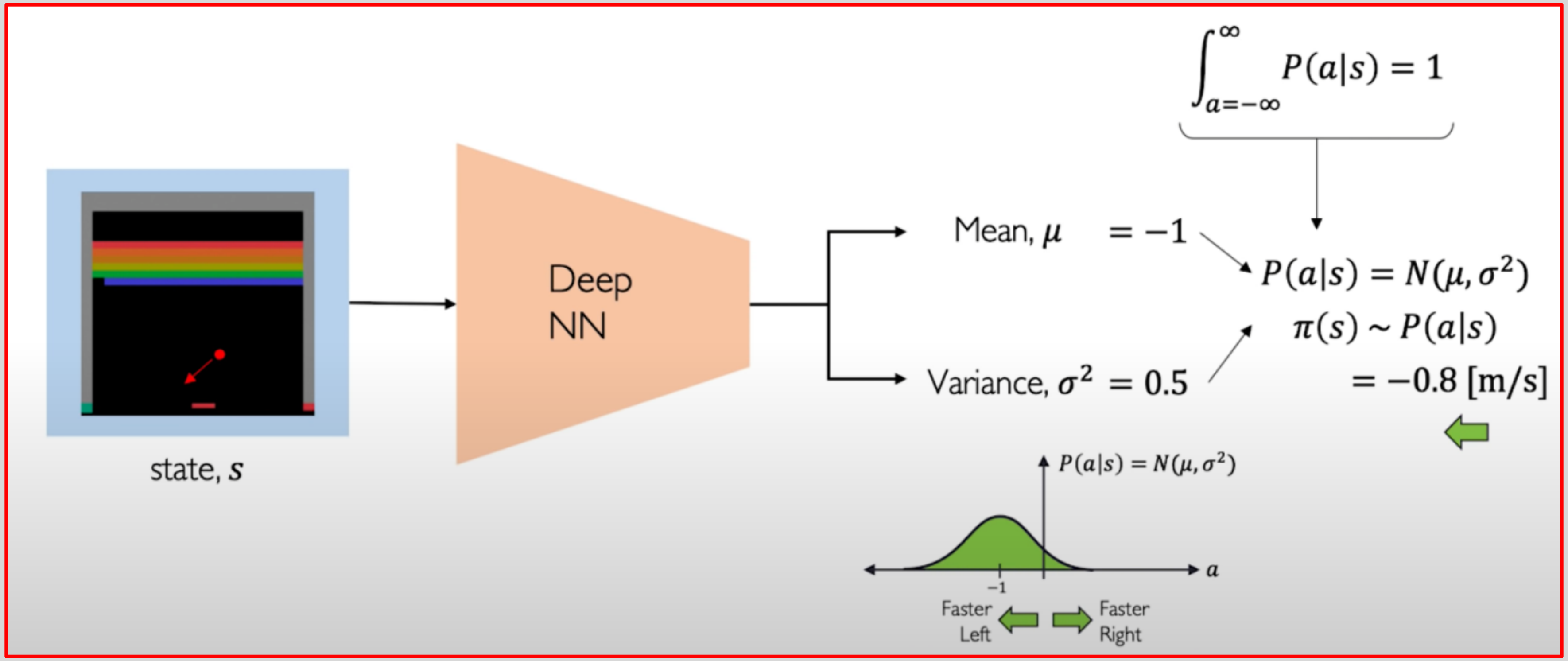
- Using Q-learning or Value Learning
- A DNN is used to learn the Q-function and then used to infer the optimal policy
- Example of **Atari Breakout video game**
- Input to DNN is state "**s**" and output Q-value for the three possible Actions "**a**"
- Actions→ move left, right or stay in same place
- To infer the optimal policy, select the action that maximizes the Q-value



Source: introtodeeplearning.com

POLICY GRADIENTS LEARNING

- Model the continuous action space with Policy Gradient method
- Instead of predicting the probability of an Action, given a possible State, there will be an infinite number of Actions in this case
- Output distribution is Gaussian with a mean and variance value→ only two outputs



Source: introtodeeplearning.com

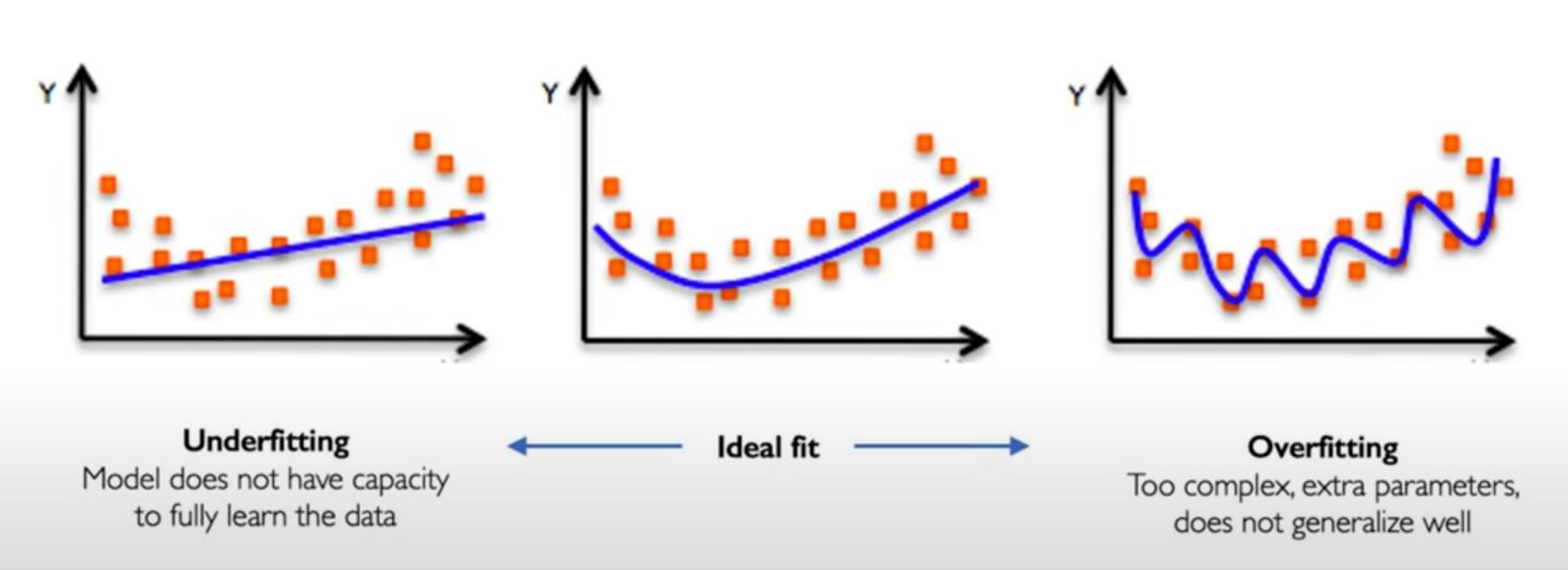
COMMON PROBLEMS IN ML/DL

DATA RELATED ISSUES

- Insufficient training data
- Non-representative training data
- Poor quality training data
- Irrelevant features
- Data mismatch during evaluation

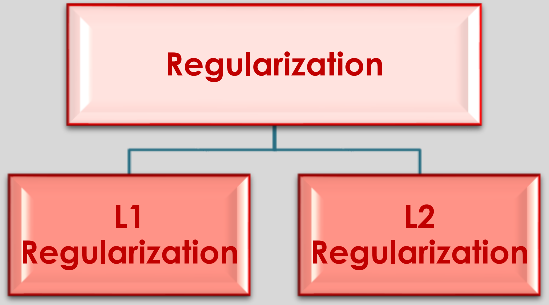
Source: : Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

OVERFITTING & UNDERFITTING PROBLEM



Source: introtodeeplearning.com

REDUCE OVERFITTING IN DL MODELS

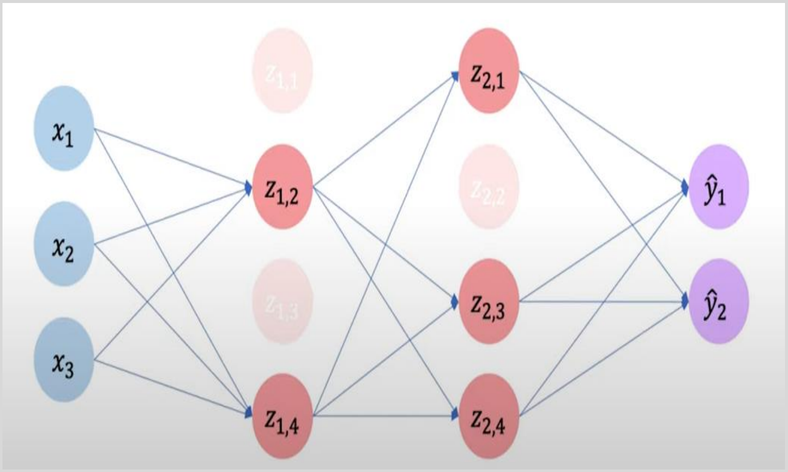


Regularization:

- Put constrains on optimization problem
- Reduce model complexity
- Improves generalization on unknown test data

Early stopping:

- Stop training a DL model before it overfits



Dropout:

- Randomly drop some activations→ results of some nodes (e.g., 50%)

Fig. Source: introtodeeplearning.com

REDUCE UNDERFITTING IN DL MODELS

- Powerful model such as Ensembles
- Feature Engineering
- Reduce Regularization constraints
- Hyperparameter optimization

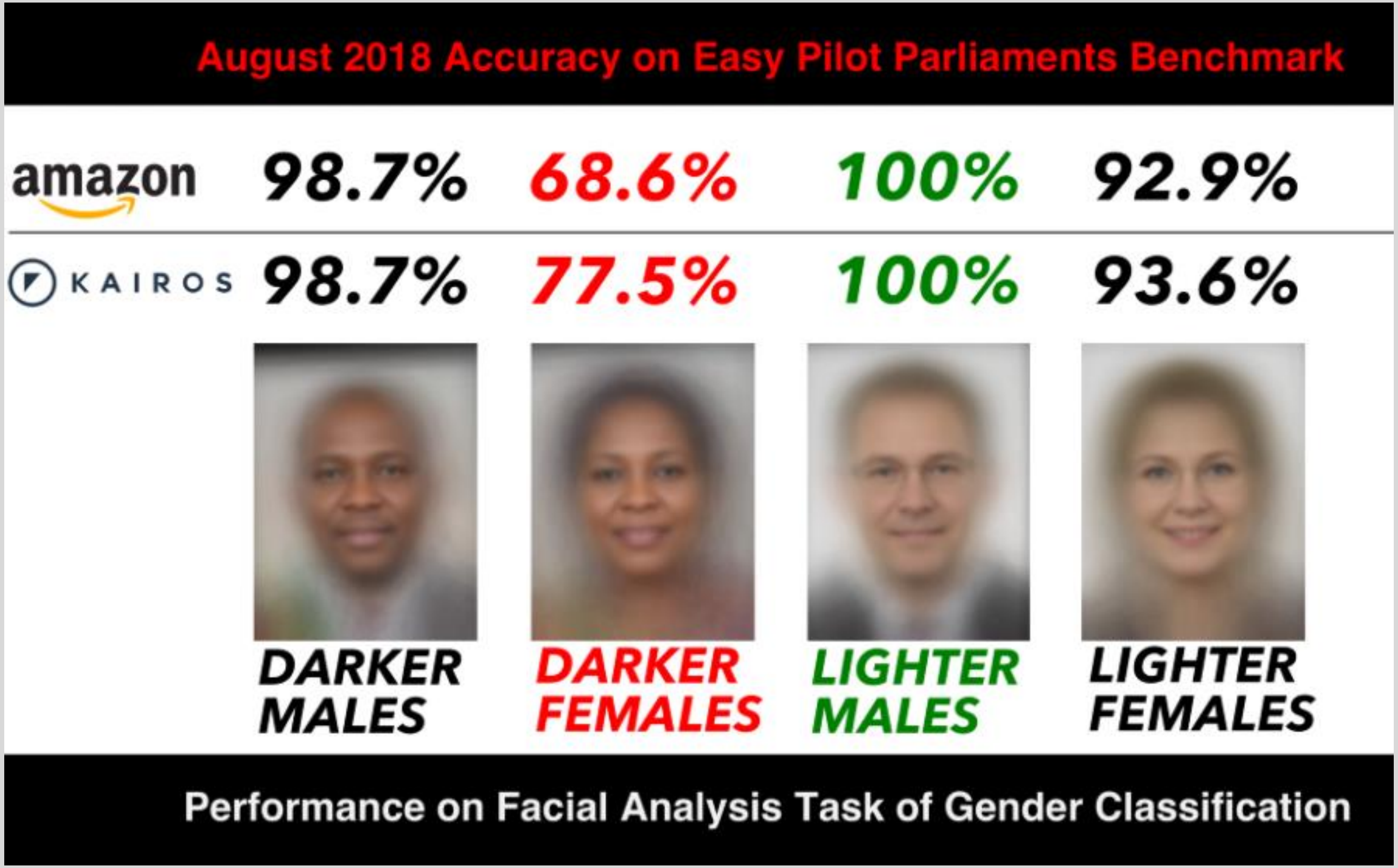
Source: : Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

WHAT IS AI BIAS?

- An anomaly in the output of an AI algorithm
- Exist in many shapes and forms
- prejudiced assumptions
- Introduced at any stage of the AI pipeline
- **Bias** is present inherently in the world around us, in our society
- We cannot directly solve the bias in the world
- We can take precautions to remove bias from different datasets used to train the AI algorithms
- Since AI has the potential to help humans to make fair decisions
- We need to work towards the **fairness** of the AI algorithms itself
- **Fairness** depends upon the situation in addition to the representation of your values, ethics and legal regulations

Source: https://www.datarobot.com/trusted-ai-101/ethics/bias-and-fairness/

BIAS IN FACIAL DETECTION



Source: https://medium.com/@Joy.Buolamwini/response-racial-and-gender-bias-in-amazon-rekognition-commercial-ai-system-for-analyzing-faces-a289222eeced

MITIGATING AI BIAS

- AWARENESS & UNDERSTANDING OF DATA & AI ALGORITHM
- IMPROVING DATA COLLECTION, RESAMPLING DATA USING GENERATIVE MODELS, REDUCING CLASS IMBALANCE PROBLEM
- IMPROVING HUMAN CENTRIC DESIGN APPROACH
- ESTABLISH A DEBIASING STRATEGY SUCH AS, EVALUATION METRICS, SUBGROUPS AND COMBINATIONS
- MAINTAINING A DIVERSE AI TEAM, ENGAGE FACT-BASED CONVERSATIONS ABOUT POTENTIAL AI BIASES
- MONITORING AND UPDATING YOUR MODEL DURING TRAINING & AFTER DEPLOYMENT
- INVEST MORE IN BIAS RESEARCH, MULTI-DISCIPLINARY APPROACH

Minimizing AI bias is important for AI systems to flourish and increase people's trust in such systems

Source: https://research.aimultiple.com/ai-bias/

LIMITATIONS OF DL

DL LIMITATIONS

1) Misinterpreting what deep-learning models do and overestimating their abilities→ lack theory of mind unlike humans

Figure: Failure of an image-captioning system based on DL

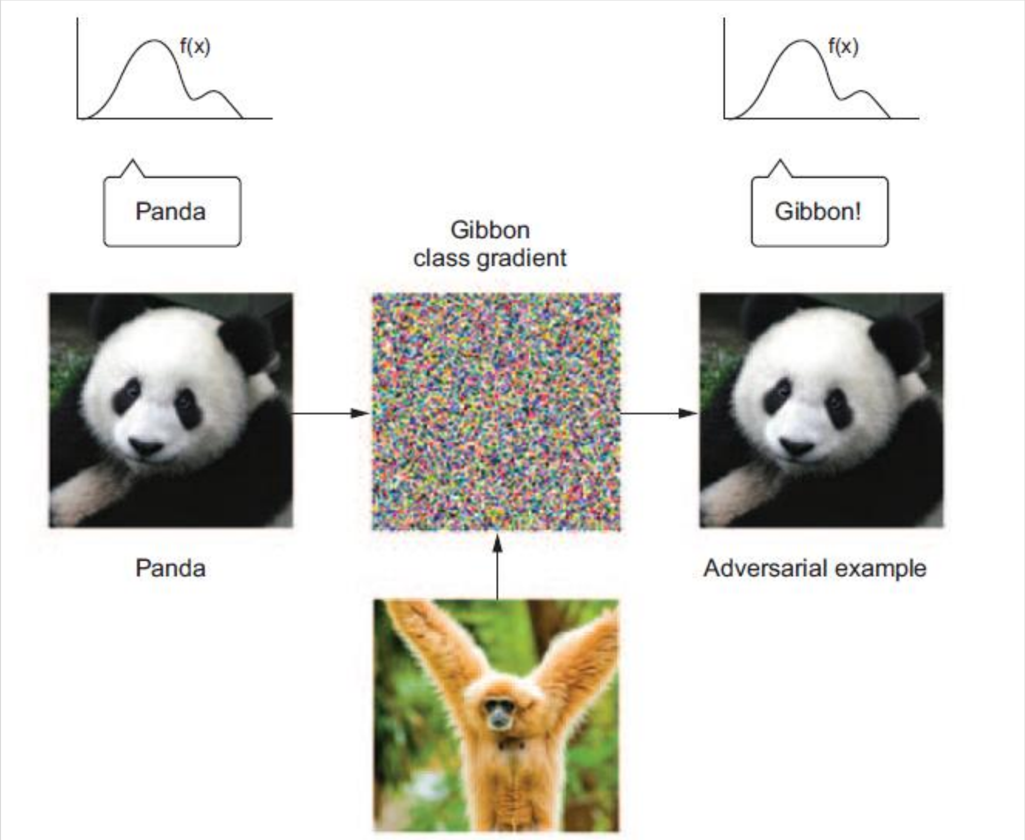


Figure: An adversarial example: imperceptible changes in an image can change a model's classification of the image

Source : Deep Learning with Python by François Chollet

LIMITATIONS OF DL

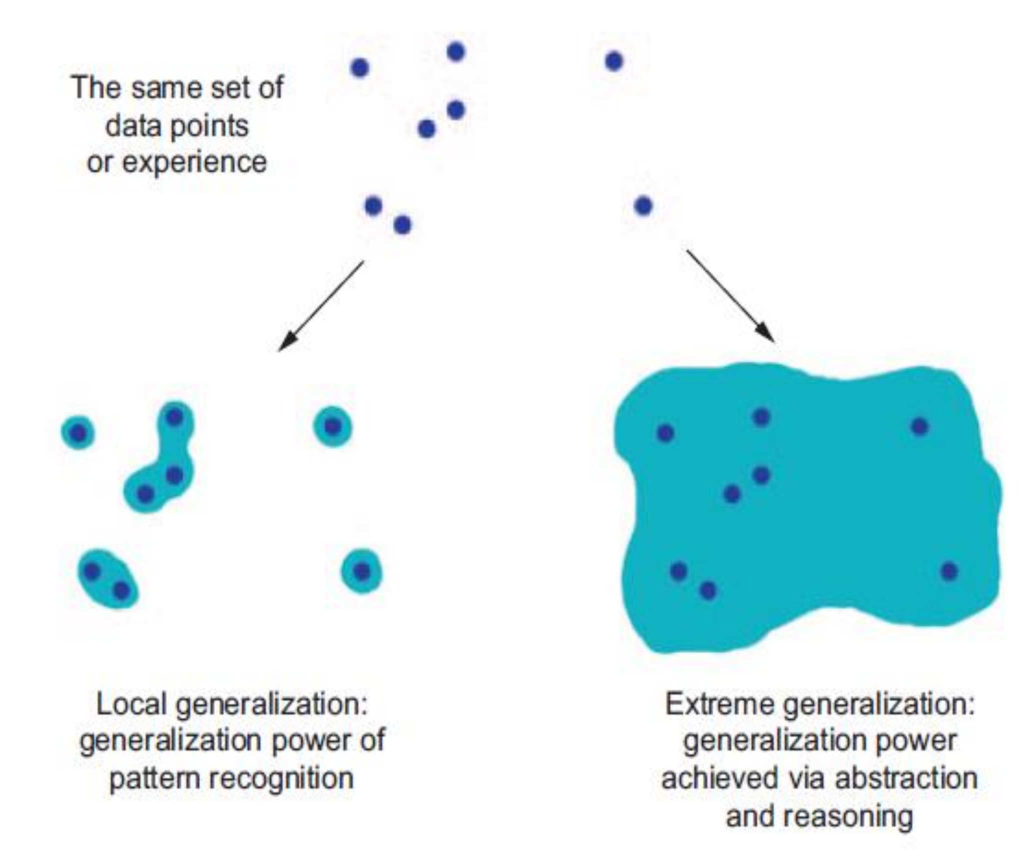
2) Local generalization vs. extreme generalization

Extreme generalization: (Human mind)

- Ability to adapt to novel, never-before-experienced situations using little data or even no new data at all, handle hypothetical situations. For e.g., a pink colored horse

Local generalization: (DL model)

- mapping from inputs to outputs performed by a DNN model
- stops making sense if new inputs differ even slightly from training data
- For e.g., learning to launch a rocket to land on the moon



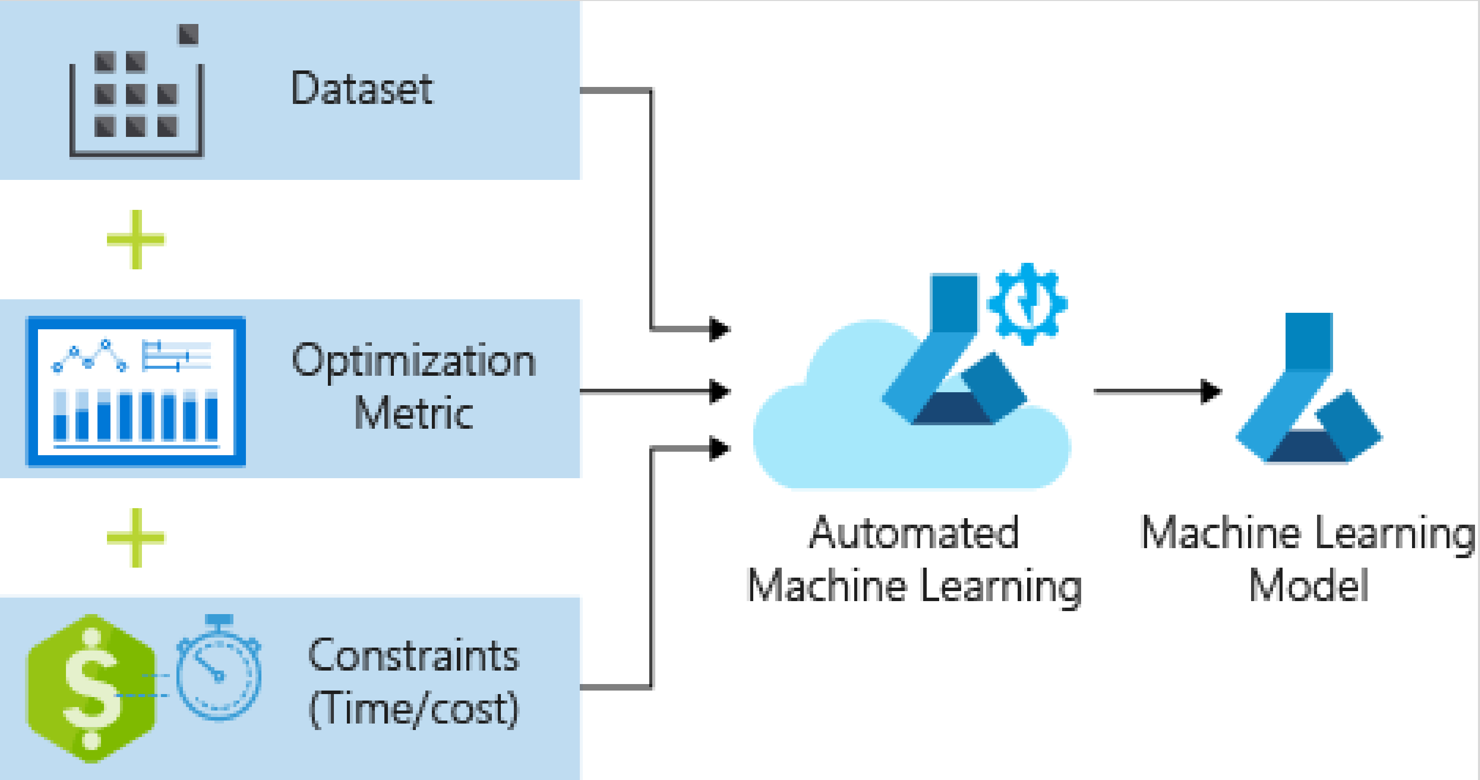
Source : Deep Learning with Python by François Chollet

FUTURE OF DL

- **Extreme generalization:** models closer to general-purpose computer programs with reasoning and abstraction
- **New forms of learning** to improve current AI performance
- **Automated ML/DL:** Models that require less involvement from human engineers, automated pipeline
- **Transfer learning:** Greater and systematic reuse of previously learned features and architectures→ reusable models
- **Explainable Artificial Intelligence (XAI):** framework that aids in understanding and interpreting predictions made by a ML/DL model

Source : Deep Learning with Python by François Chollet

AUTO ML/ AUTO AI



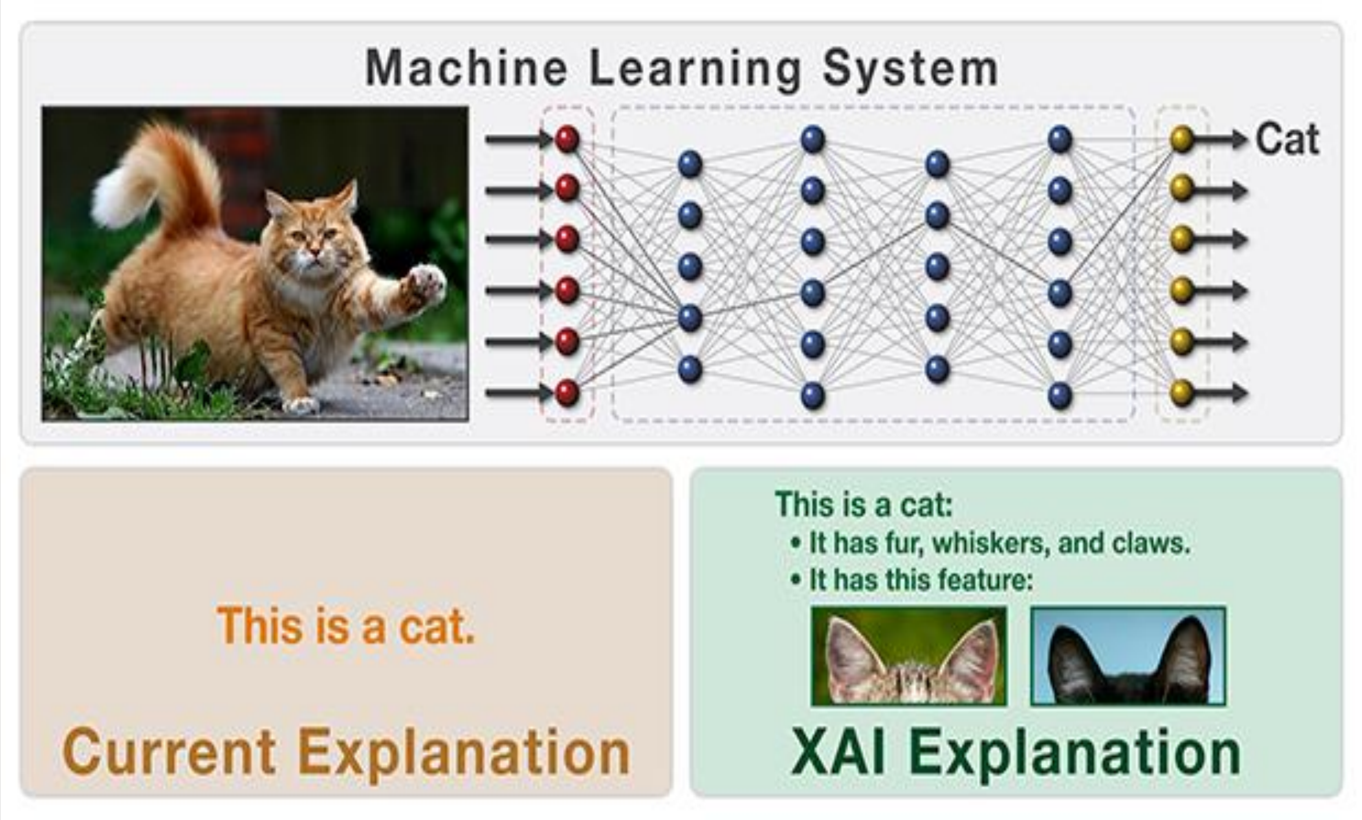
Auto AI Example Frameworks

[Google's Auto ML]
[Microsoft's Auto ML]
[Auto sklearn]
[AutoKeras]

Image Source: https://softwareengineeringdaily.com/2019/05/15/introduction-to-automated-machine-learning-automl/

Explainable Artificial Intelligence (XAI)

- explaining how much each feature contributed to the model predictions



XAI Example Frameworks

Google's Explainable AI

IBM's Explainable AI

Local interpretable model-agnostic explanations (LIME)

SHAP (SHapley Additive exPlanations)

Image Source: https://www.bigdatavietnam.org/2017/11/explainable-artificial-intelligence-xai.html

THANKS!

Do you have any questions?