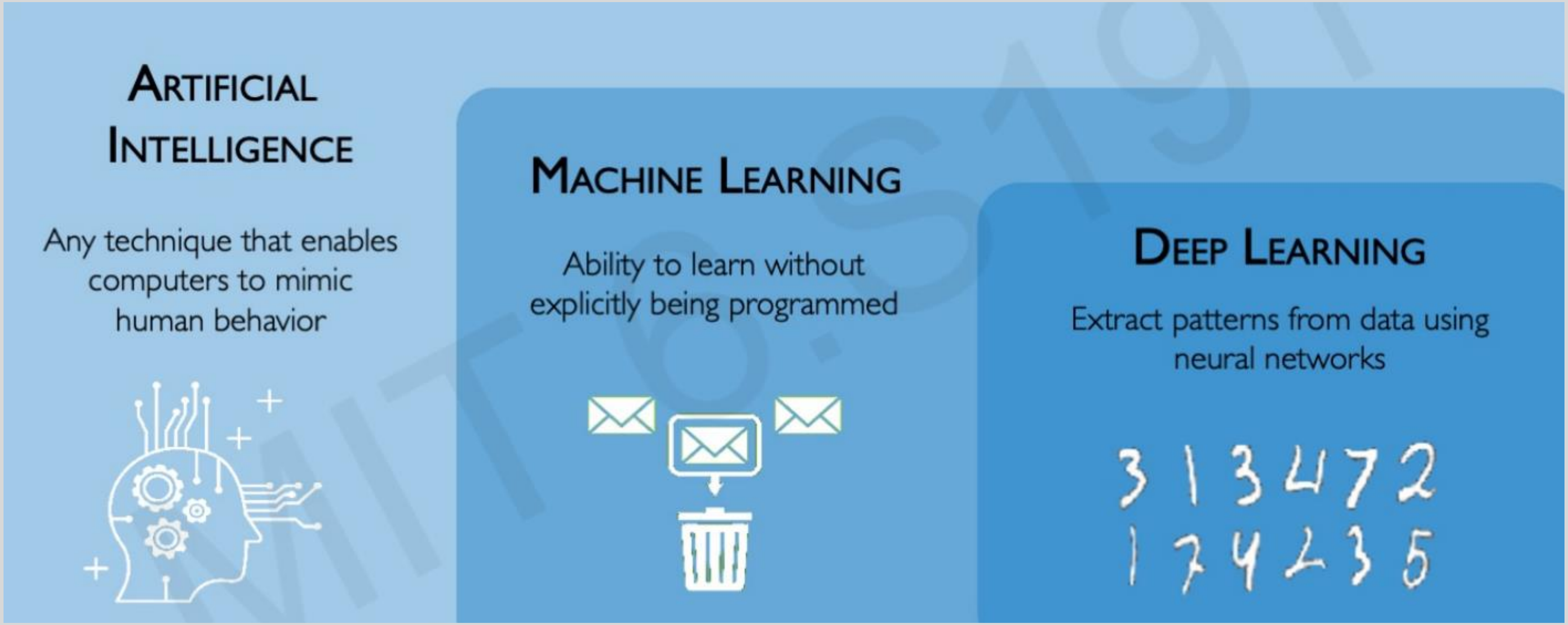
MODULE 4: INTRODUCTION TO DEEP LEARNING (DL)

BA713 - Machine Learning & AI

CONTENTS

DEEP LEARNING (DL)

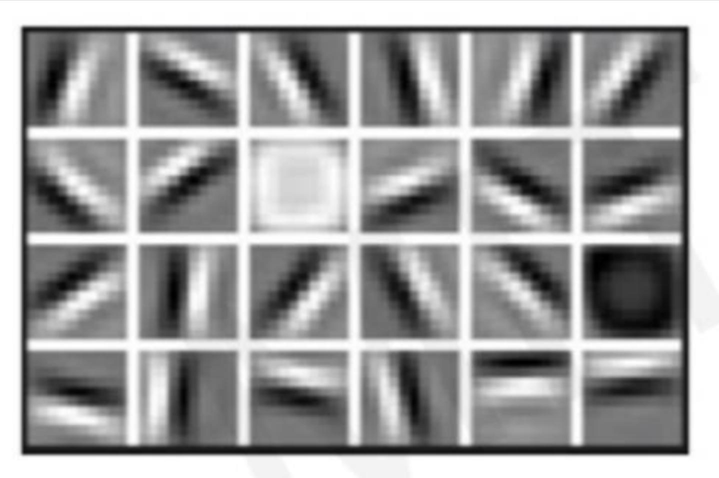


Source: introtodeeplearning.com

WHY DO WE NEED DEEP NEURAL NETWORKS?

- Increasing amount of data
- Increased features
- Time consuming→ impractical

Low level features



Lines and Edges

Mid level features

Eyes and nose and ears

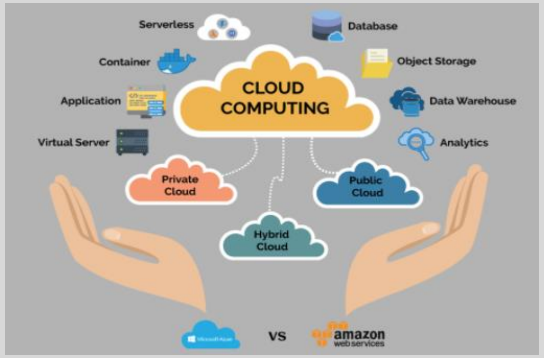
High level features

Facial Structure

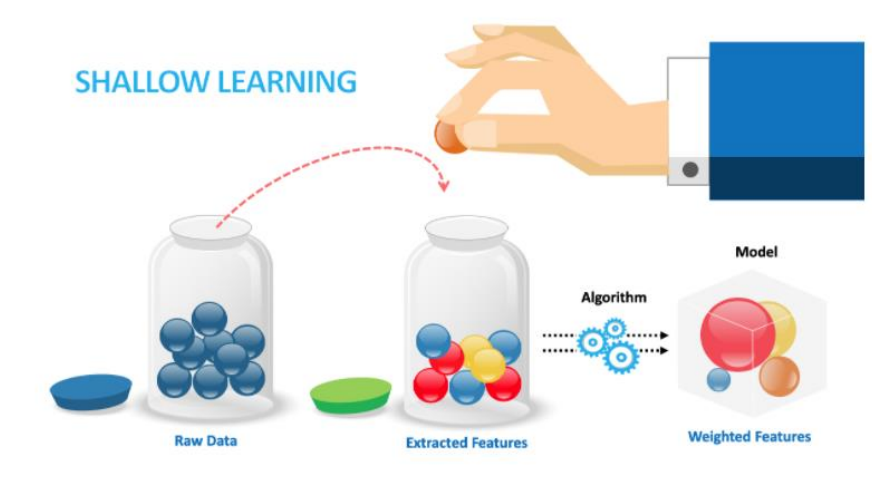
Source: introtodeeplearning.com

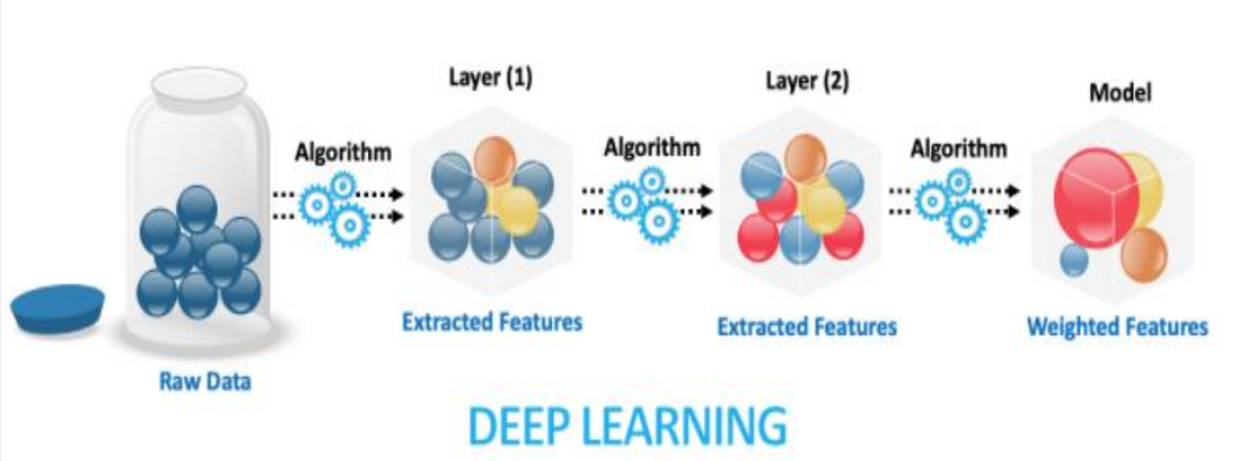
WHY DEEP LEARNING (DL)?

- **Big Data→** Cloud, AWS, Azure, etc
- **Hardware improvement→** Graphical Processing Units or GPUs, Parallel processing
- **Software Improvement→** improved techniques, toolkits, python, keras, pytorch, tensorflow, etc



SHALLOW LEARNING VERSUS DEEP LEARNING





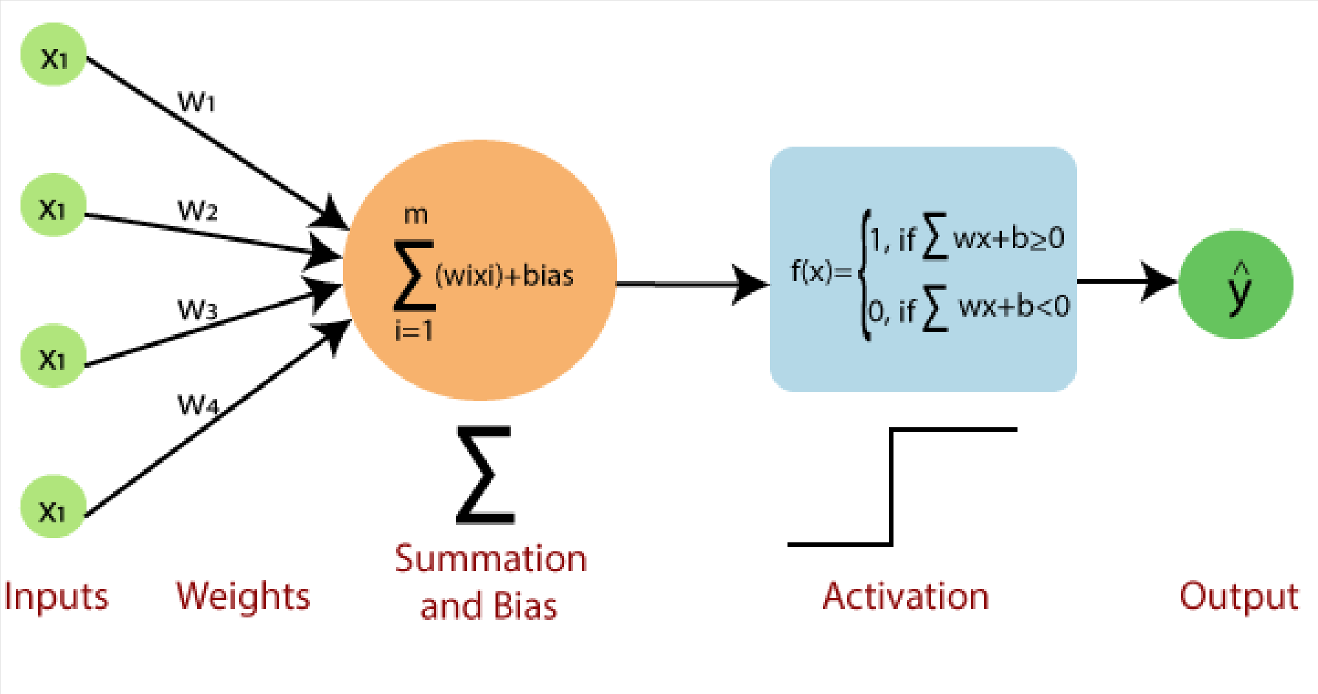
"**Shallow Learning**" is a type of machine learning where we learn from data described by pre-defined features.

A "**Deep learning**" algorithm automatically learns these features/patterns along with their weights.

Source: https://www.linkedin.com/pulse/introduction-shallow-machine-learning-ayman-mahmoud/

STRUCTURAL BUILDING BLOCK FOR DEEP NEURAL NETWORKS

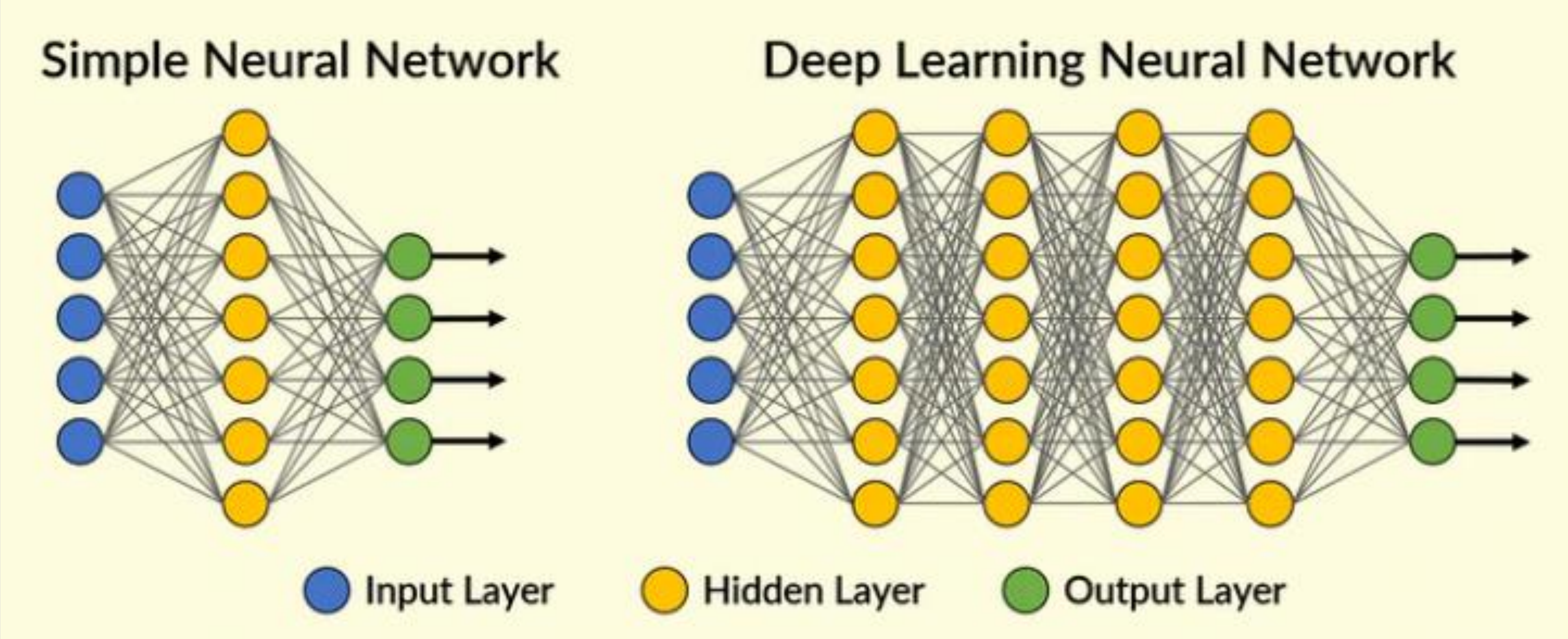
(Perceptron)



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

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



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

KERAS & TENSORFLOW OVERVIEW

Keras: Keras

- deep learning high-level API written in Python, running on top of the machine learning platform TensorFlow
- highly-productive interface
- provides essential abstractions and building blocks for developing DL solutions
- fast experimentation

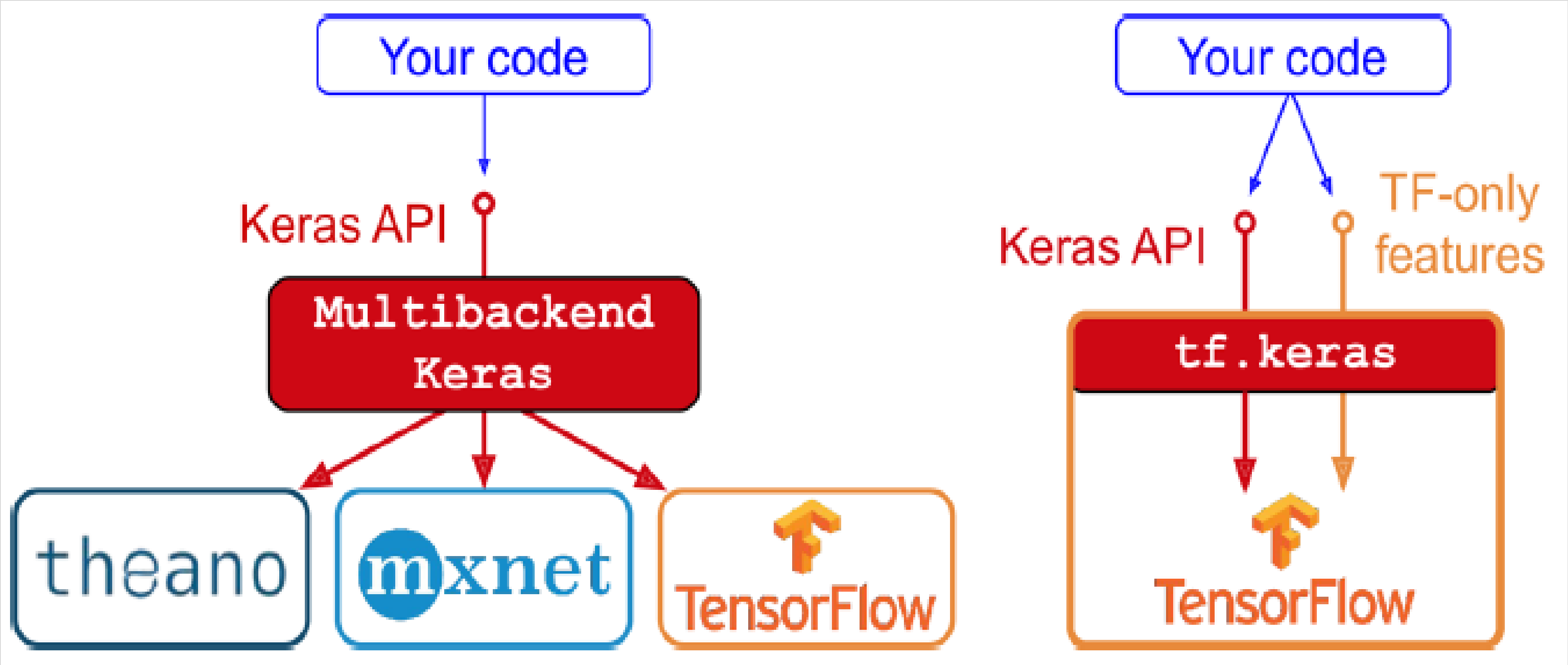
Tensorflow: TensorFlow

- end-to-end, open-source machine learning platform owned by Google
- infrastructure layer for Deep Learning programming
- Working with GPU/CPU/TPU
- with state-of-the-art tools, libraries, and community→ build and deploy DL models
- Scaling computation to many devices→ clusters of GPUs

Source: https://keras.io/about/

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

ANATOMY OF A DEEP NEURAL NETWORK (DNN)

- **Layers**, which are combined into a **network** (or **model**)
- The **input data** and corresponding **targets**
- The **loss function**, which defines the feedback signal used for learning
- The **optimizer**, which determines how learning proceeds

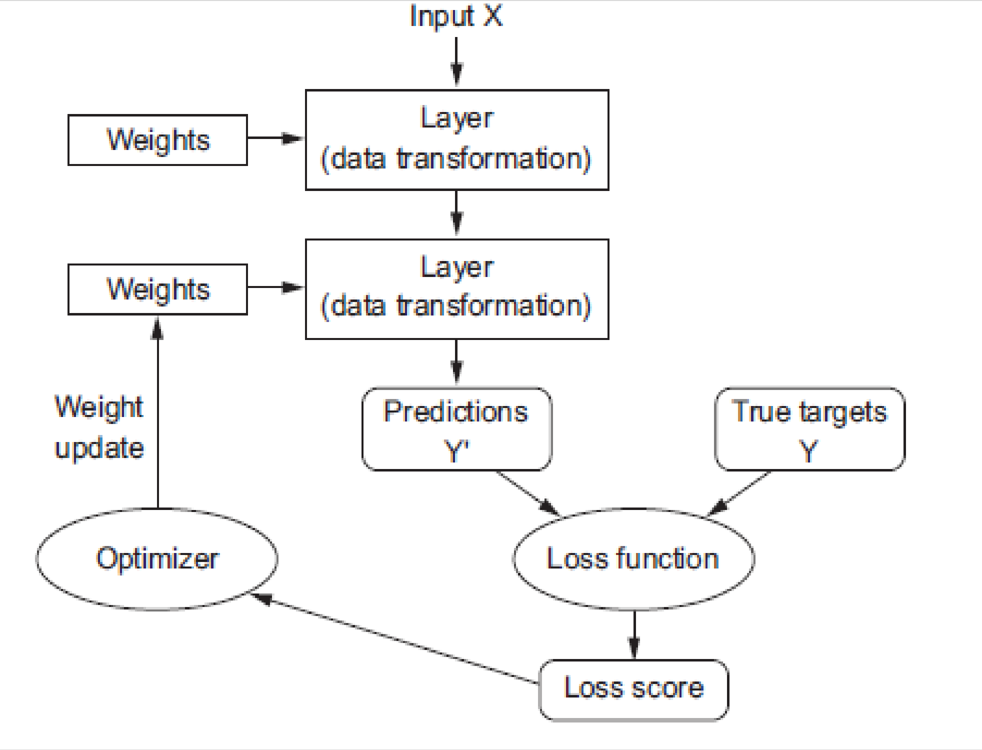


Figure: Relationship between the network, layers, loss function, and optimizer

Source : Deep Learning with Python by François Chollet

Layers: the building blocks of deep learning

- data-processing module that takes as input one or more tensors and that outputs one or more tensors
- frequently layers have a state→ the layer's weights, one or several tensors learned with stochastic gradient descent, which together contain the network's knowledge
- Densely/ Fully connected layers→ *Dense* class in Keras

- **Simple vector data**→ stored in 2D tensors → *(samples, features)*→ DNN model
- **Sequence data**→ 3D tensors→ *(samples, timesteps, features)*→ LSTM Model
- **Image data**→ 4D tensors→ *2D convolution layers (Conv2D)*→ CNN Model

```
from keras import layers
layer = layers.Dense(32, input_shape=(784,))
```

A dense layer with 32 output units

Source : Deep Learning with Python by François Chollet

Models: networks of layers

- linear stack of layers, mapping a single input to a single output
- Keras functional API→ more complex
- Write models from scratch

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(784,)))
model.add(layers.Dense(10, activation='softmax'))
```

Simplest model type→ Sequential API

```
input_tensor = layers.Input(shape=(784,))
x = layers.Dense(32, activation='relu')(input_tensor)
output_tensor = layers.Dense(10, activation='softmax')(x)

model = models.Model(inputs=input_tensor, outputs=output_tensor)
```

Using Functional API

Source : Deep Learning with Python by François Chollet

Loss functions and optimizers: configuring the learning process

Loss/objective function:

- minimize the loss/error→ measure of success

Optimizer:

- Determines how the network will be updated based on the loss function
- variant of stochastic gradient descent (SGD).

```
from keras import optimizers

model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss='mse',
              metrics=['accuracy'])
```

Optimizer RMSprop

Loss function-MSE

Source : Deep Learning with Python by François Chollet

Network/Model training & evaluation

- Pass numpy array of input data and target label (x_train, y_train) into fit() method

```
# x_train and y_train are Numpy arrays
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

- Evaluate your test loss and metrics in one line:

```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

- Or generate predictions on new data:

```
classes = model.predict(x_test, batch_size=128)
```

Source : https://keras.io/about/

GRADIENT DESCENT OPTIMIZATION

- Deep Neural Networks learn through Gradient Descent Optimization

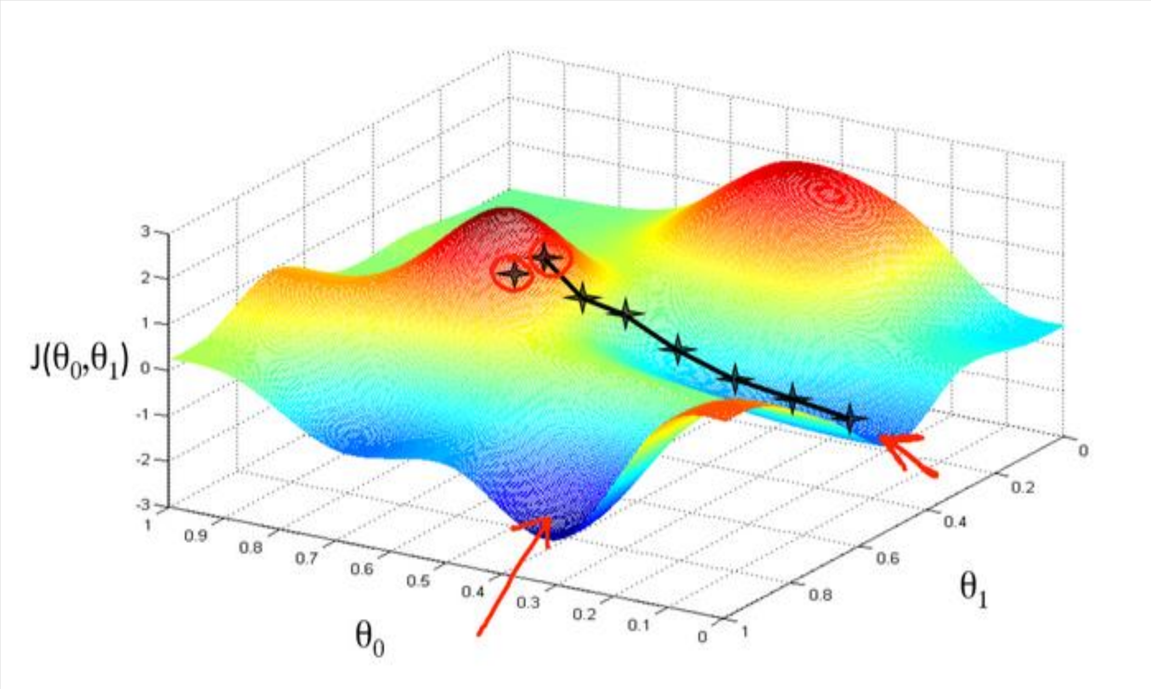


Figure a: Gradient Descent example

Gradient Descent Algorithms

1. **Batch gradient descent**
2. **Stochastic gradient descent**
3. **Mini-batch gradient descent**

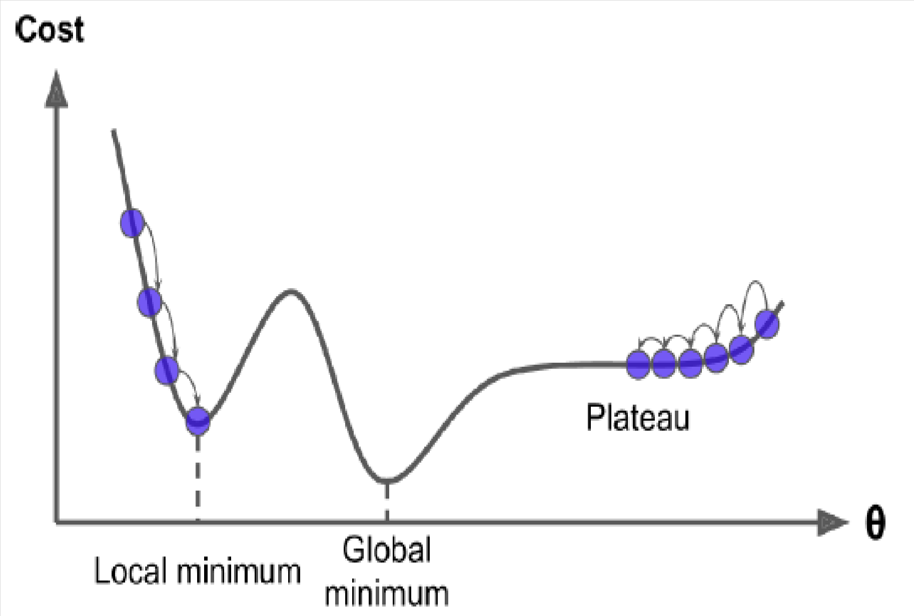


Figure b: Gradient Descent pitfalls

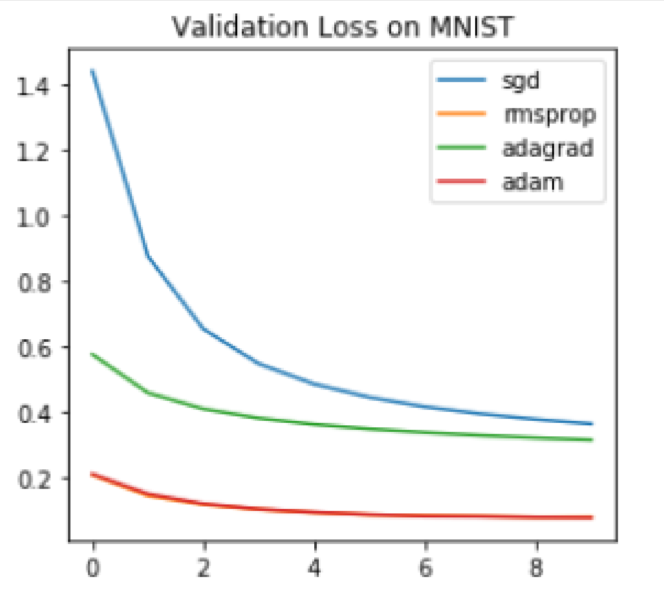
Source : https://machinelearningmind.com/2019/10/06/gradient-descent-introduction-and-implementation-in-python/

ADAPTIVE LEARNING RATES

- 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**



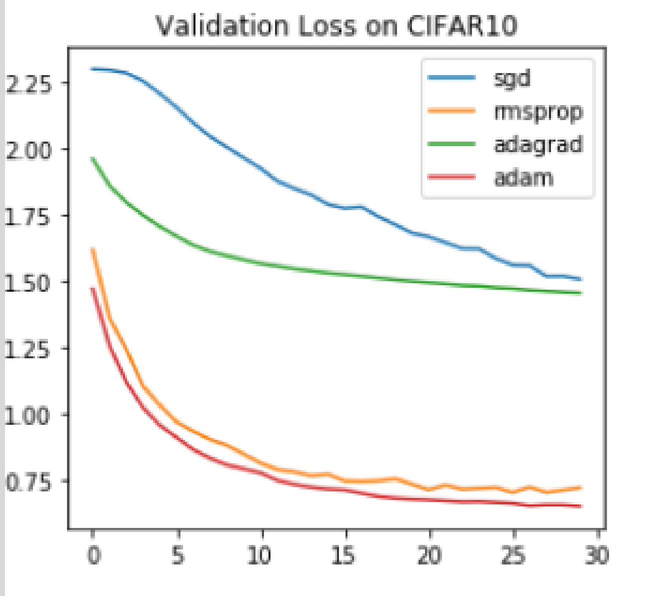
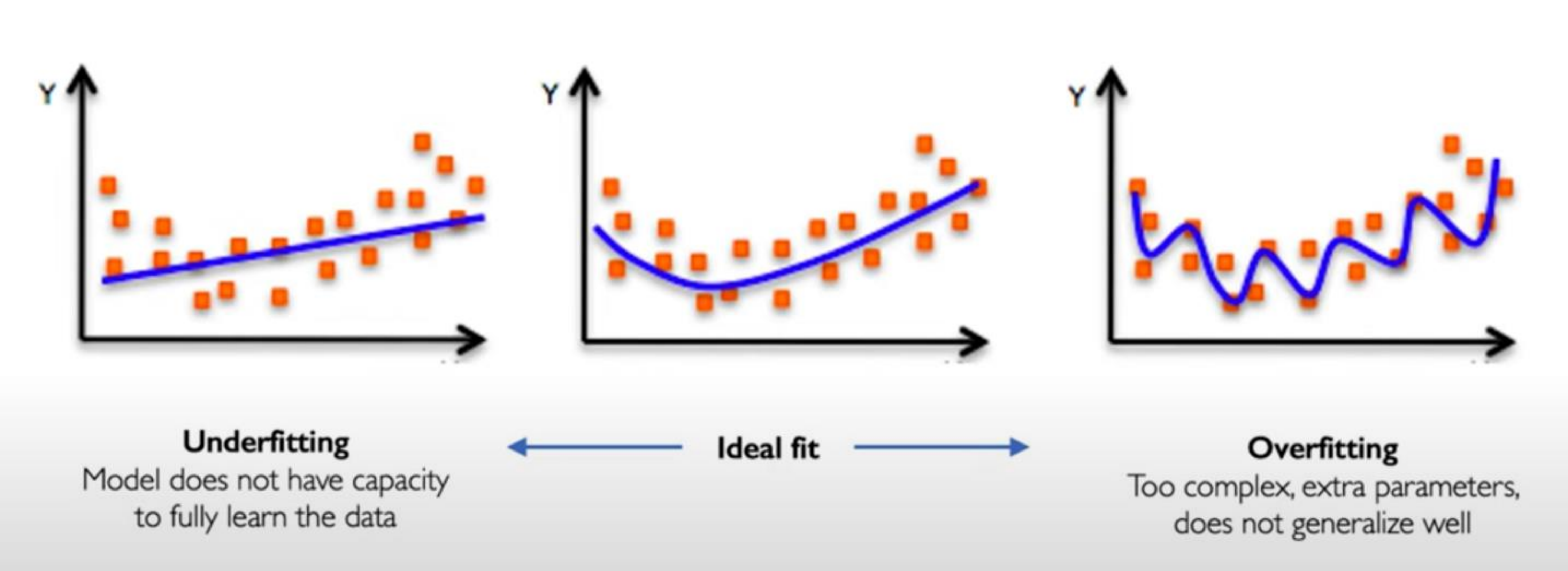


Image source: https://heartbeat.fritz.ai/an-empirical-comparison-of-optimizers-for-machine-learning-models-b86f29957050

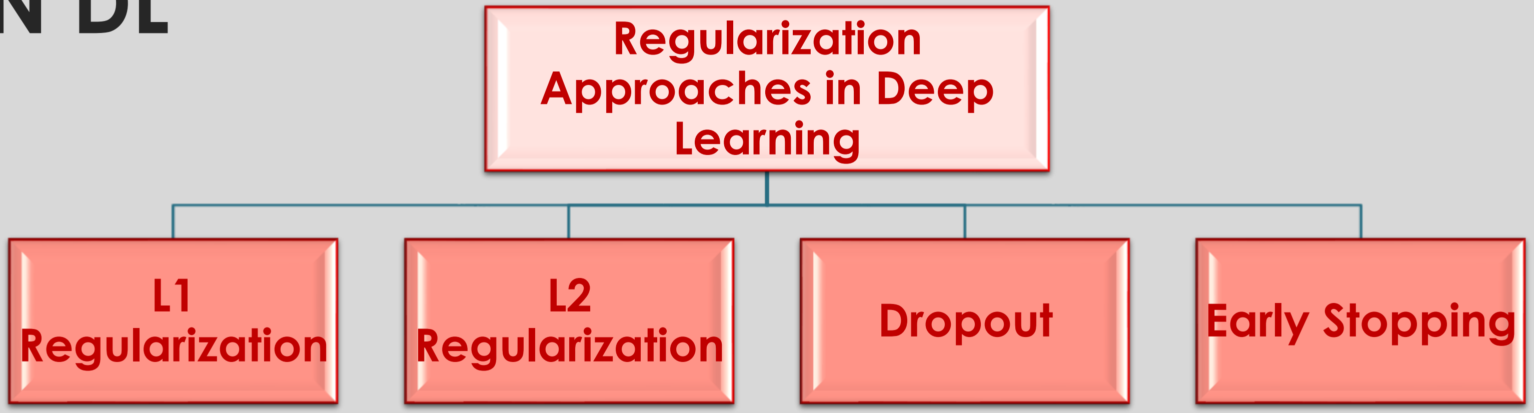
OVERFITTING PROBLEM IN A DNN MODEL



Source: introtodeeplearning.com

REGULARIZATION IN DL

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



L1 & L2 Regularization

- Update the cost function by adding regularization term
- Cost function = Loss (e.g., binary cross entropy) + Regularization term

```
from keras import regularizers
model.add(Dense(64, input_dim=64,
                kernel_regularizer=regularizers.l2(0.01)
```

For L2

$$Cost\ function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

For L1

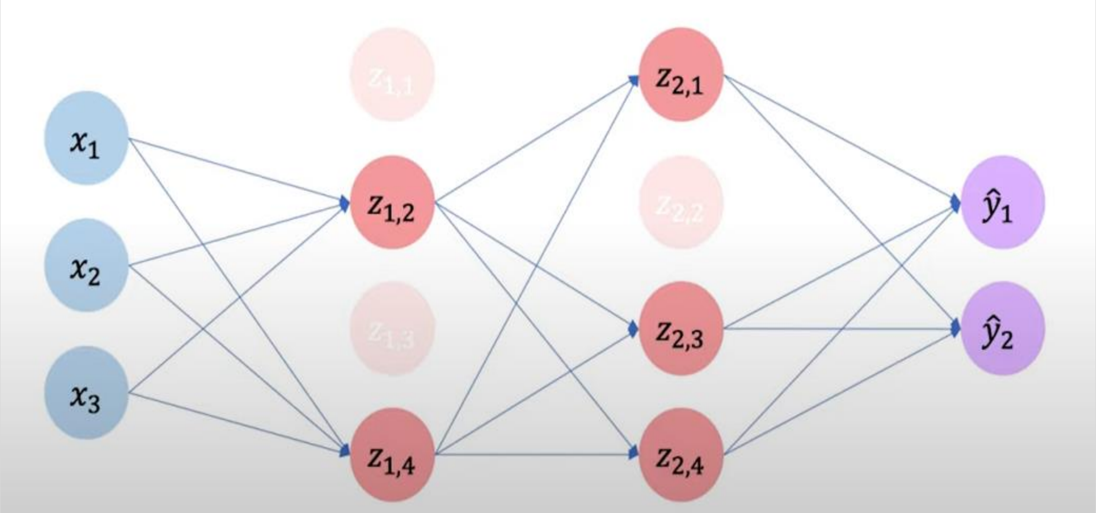
$$Cost\ function = Loss + \frac{\lambda}{2m} * \sum \|w\|$$

λ → Regularization parameter
w → weights

Source: https://www.analyticsvidhya.com/

Dropout

- Randomly drop some activations→ results of some nodes
- Normally 50%
- Rely on only the other set of nodes left
- Dropout layer→ between existing layers
- applies to outputs of the prior layer that are fed to the subsequent layer



Source: introtodeeplearning.com

```
# example of dropout between fully connected layers
from keras.layers import Dense
from keras.layers import Dropout
...
model.add(Dense(32))
model.add(Dropout(0.5))
model.add(Dense(1))
...
```

Source: https://machinelearningmastery.com/

Early Stopping

- Stop training a DL model before it overfits

```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping
```

```
# simple early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)
```

```
# fit model
history = model.fit(trainX, trainy, validation_data=(testX, testy), epochs=4000, ver
bose=0, callbacks=[es])
```

Source: introtodeeplearning.com

Source: https://machinelearningmastery.com/

THANKS!

Do you have any questions?