DEEP LEARNING WITH KERAS

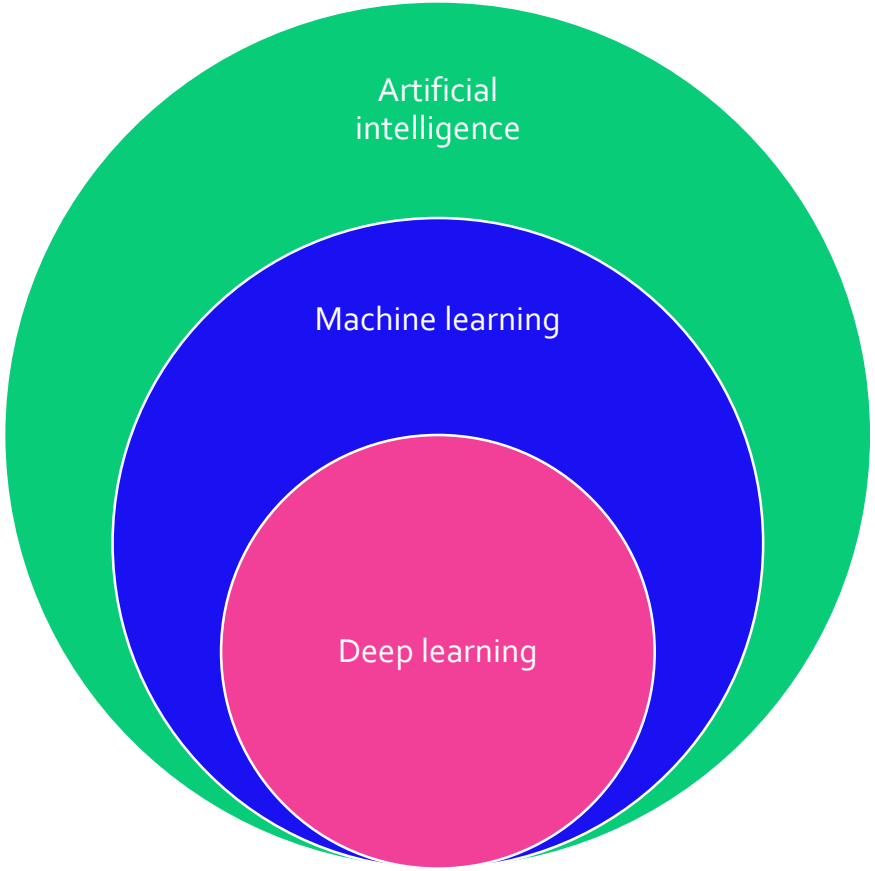
Presented by

Bharathi Raja Asoka Chakravarthi

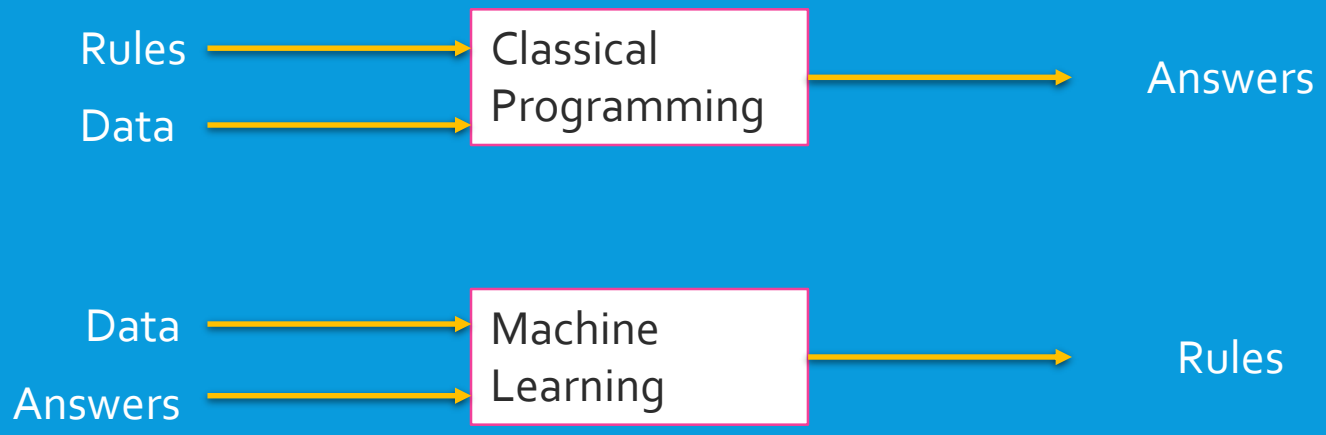
OUTLINE

- Introduction: What is deep learning?
- The mathematical building blocks of neural network
- Fundamentals of machine learning
- Deep learning for text and sequences
- Advanced deep learning practices
- Practical session: Bharathi and Bernardo

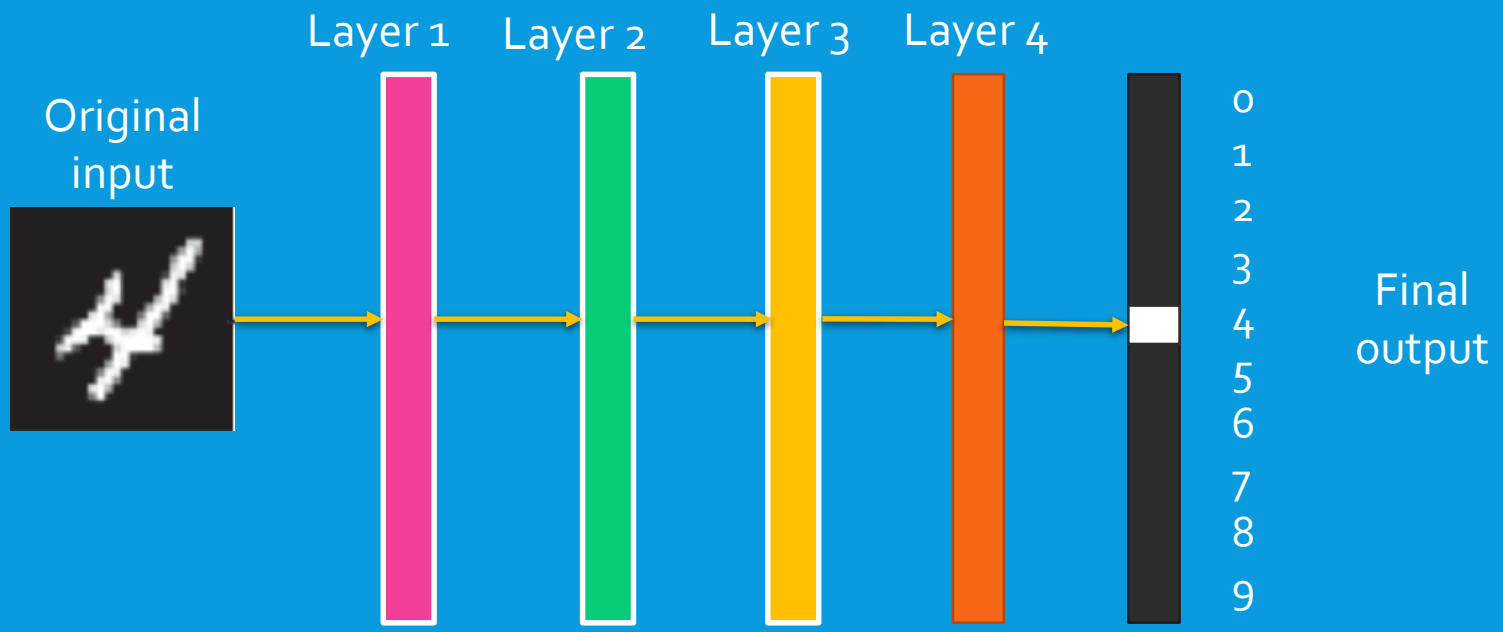
WHAT IS DEEP LEARNING?



MACHINE LEARNING

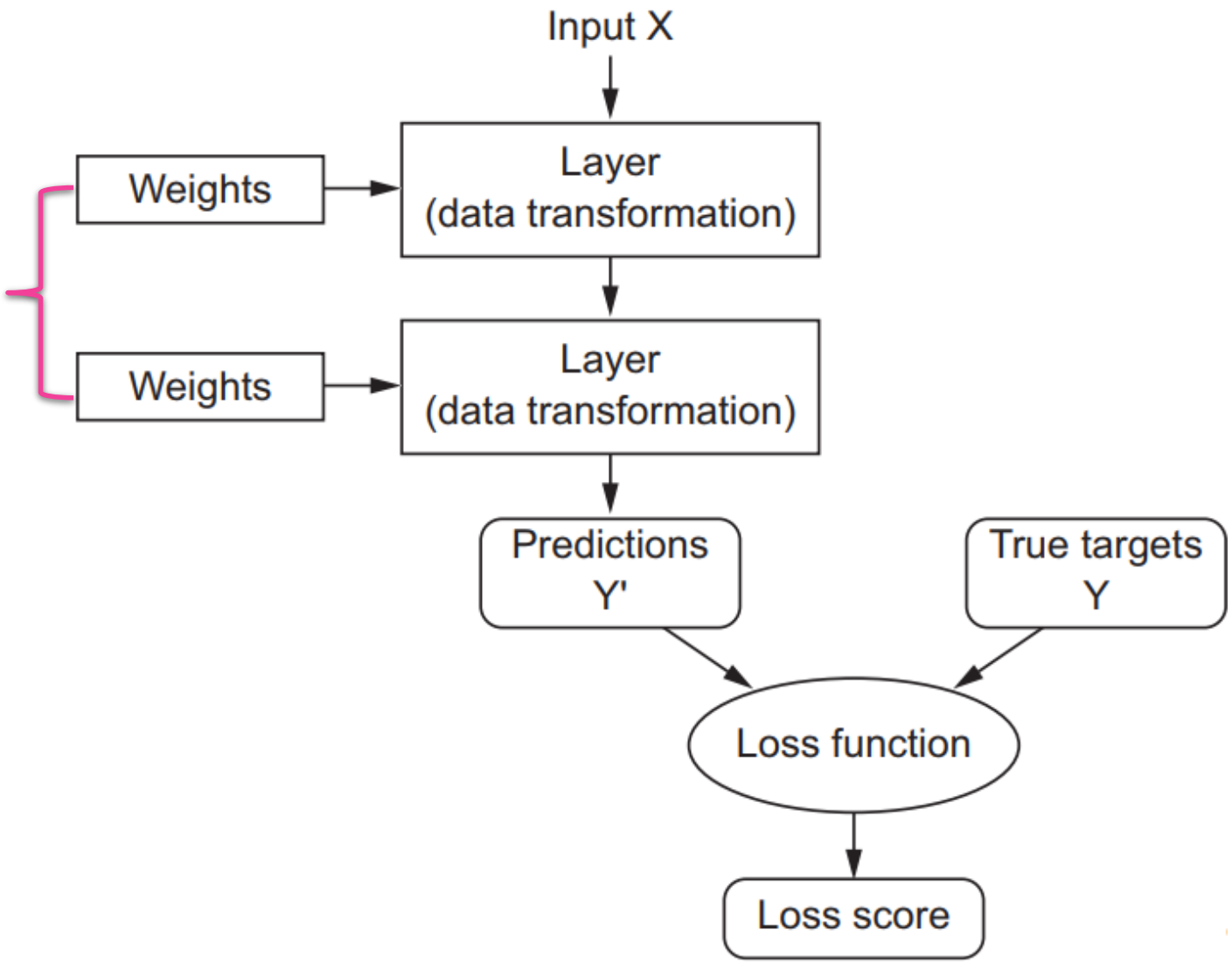


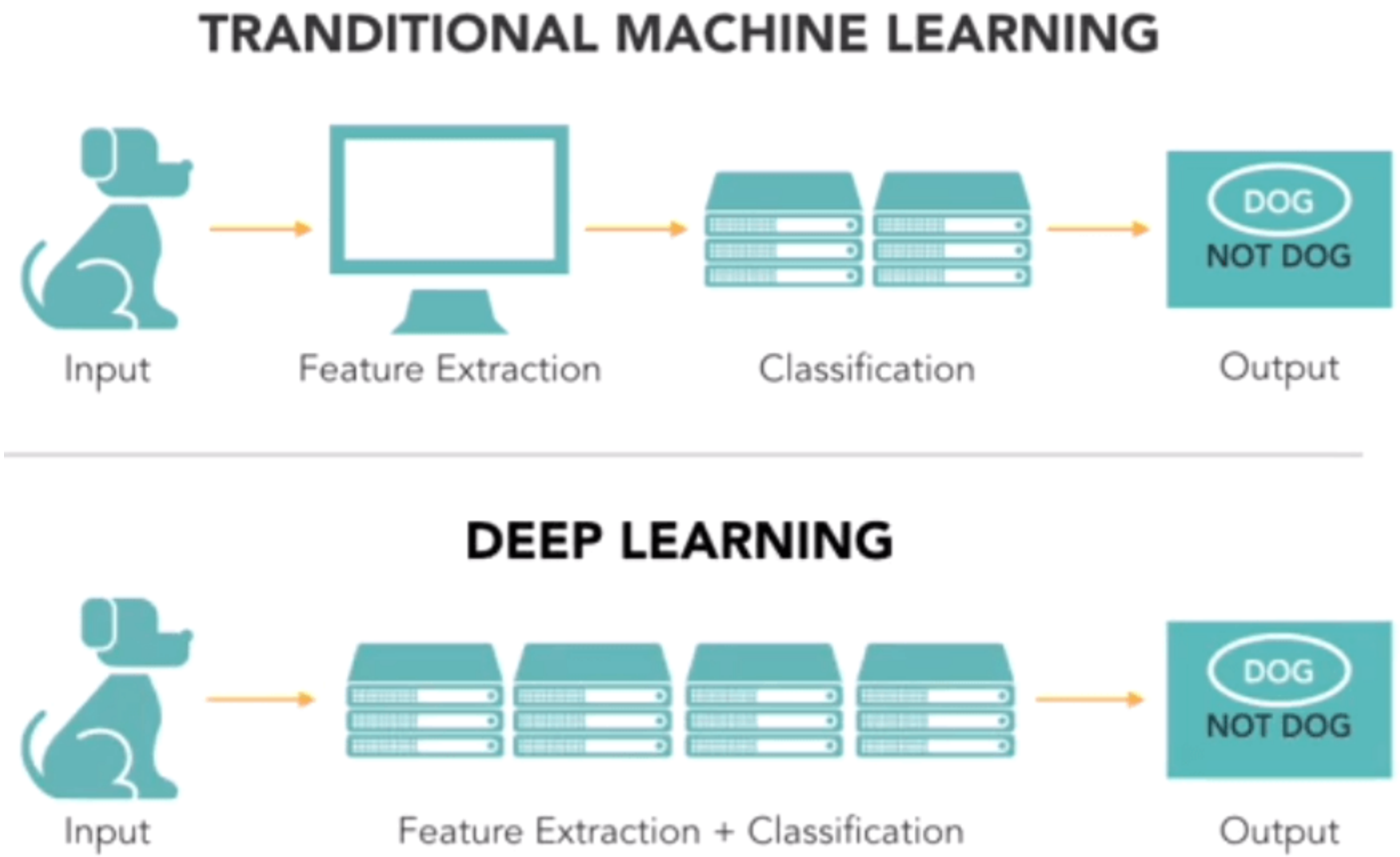
DEEP LEARNING



DEEP LEARNING

- Goal: Finding the right values for the weights





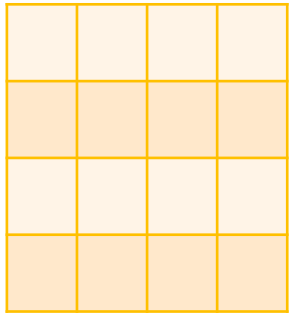
https://www.guru99.com/images/tensorflow/083018_0454_MachineLear5.png

MATHEMATICAL BUILDING BLOCKS OF NEURAL NETWORK

- A tensor is a generalization of vectors and matrices to potentially higher dimensions.
- Scalar (0D tensors)

- Vectors (1D tensors)

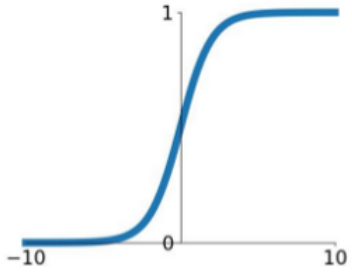
- Matrices (2D tensors)



Activation Functions

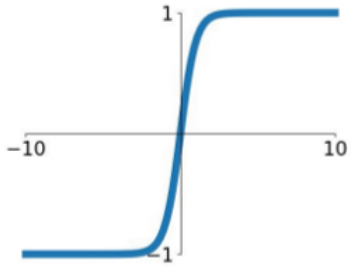
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



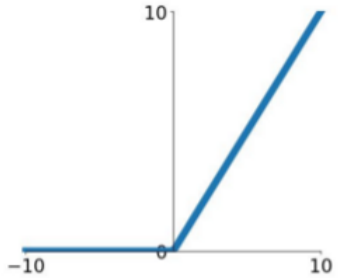
tanh

$$\tanh(x)$$



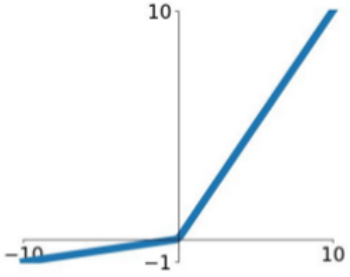
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

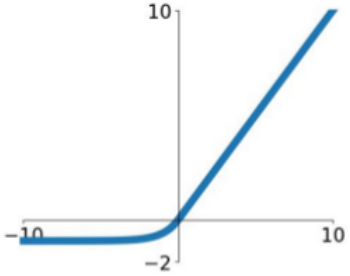


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

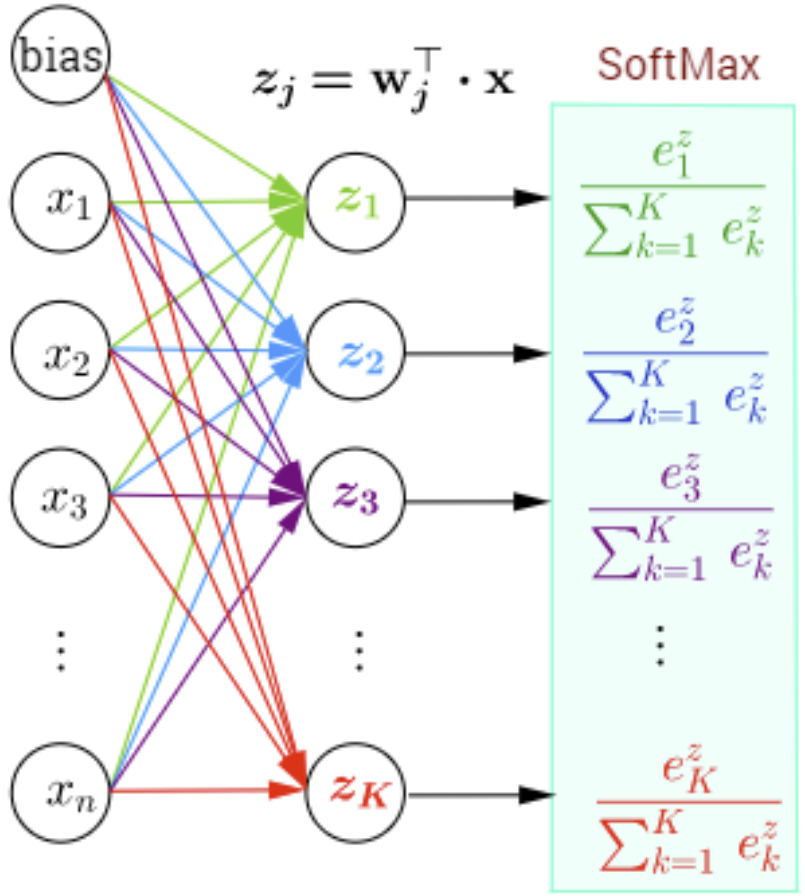
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



https://medium.com/@krishnakalyan3/introduction-to-exponential-linear-unit-d3e2904b366c

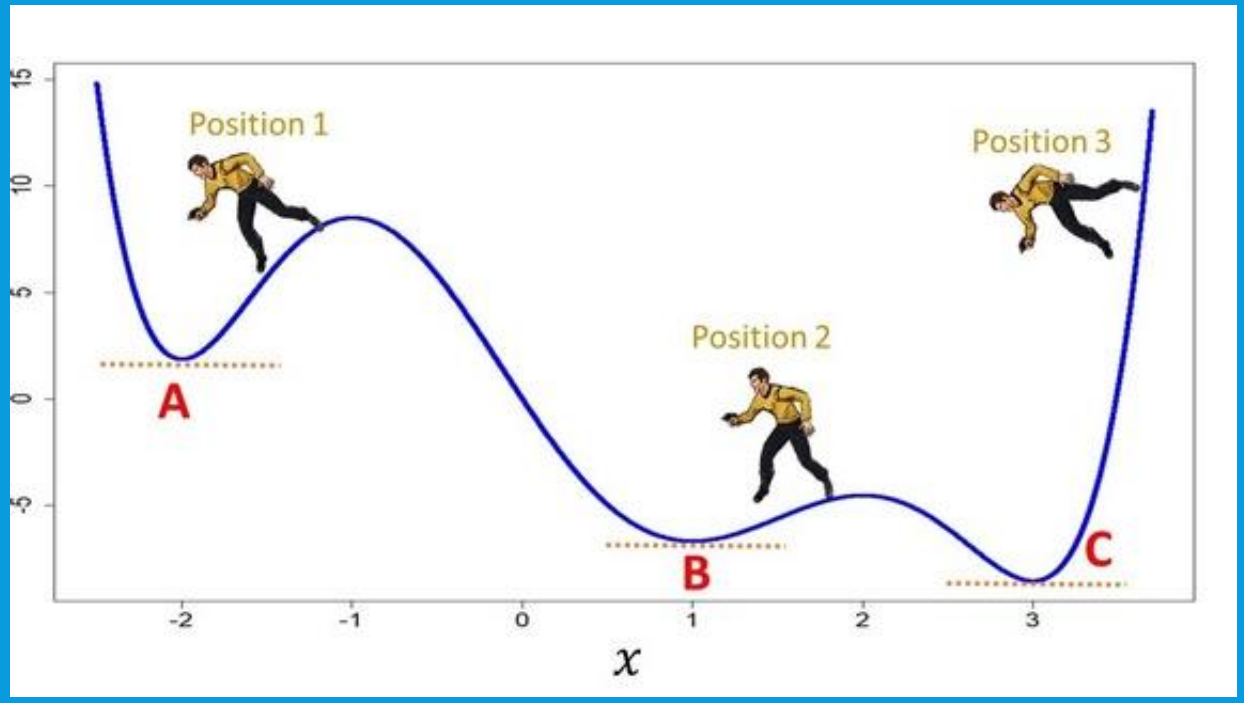
Multi-Class Classification with NN and SoftMax Function

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



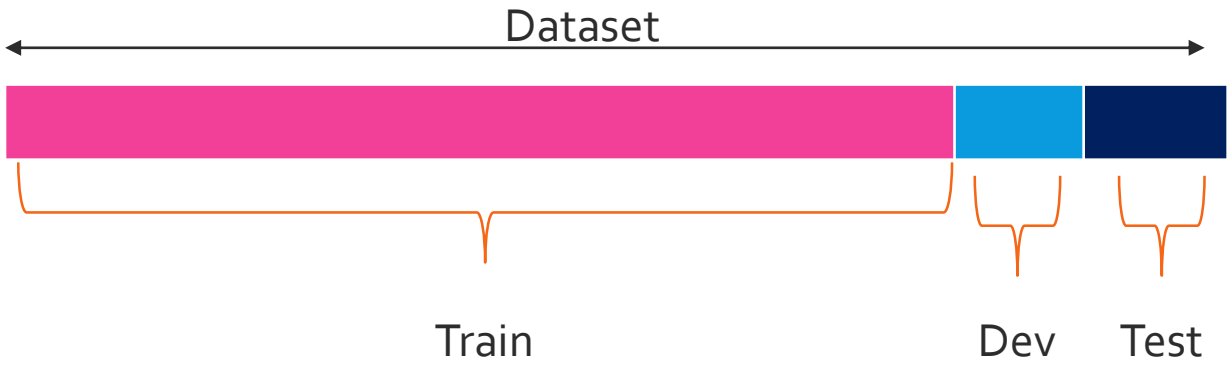
http://rinterested.github.io/statistics/softmax.html

GRADIENT DESCENT

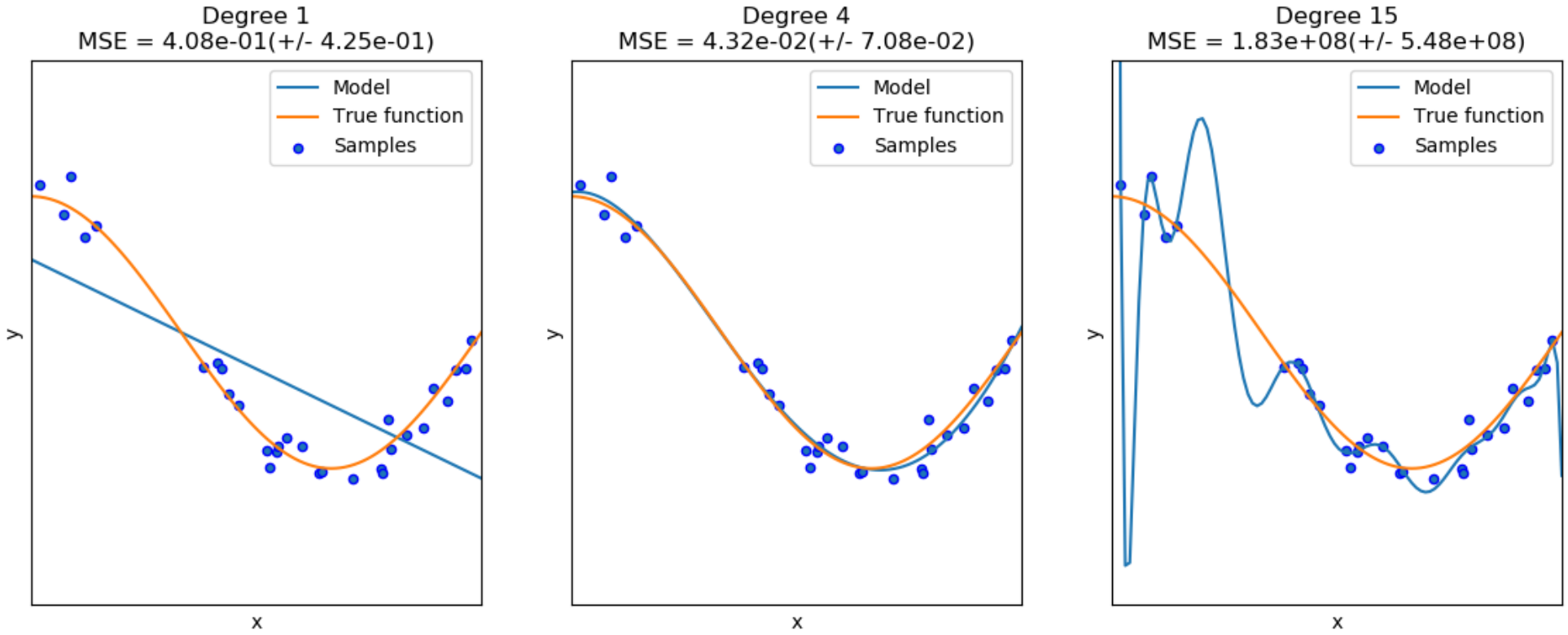


https://qph.fs.quoracdn.net/main-qimg-a820f5e19a9393391572a73868f8ca1a

EVALUATION MACHINE LEARNING MODELS



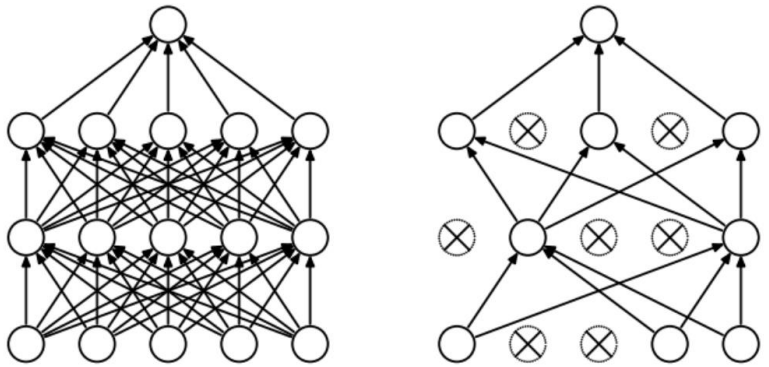
OVERFITTING AND UNDERFITTING



https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

PREVENT OVERFITTING

- Get more training data
- Reducing the network size
- Regularization
  - L1 regularization—The cost added is proportional to the absolute value of the weight coefficients
  - L2 regularization—The cost added is proportional to the square of the value of the weight coefficients
- Dropout



(a) Standard Neural Net

(b) After applying dropout.

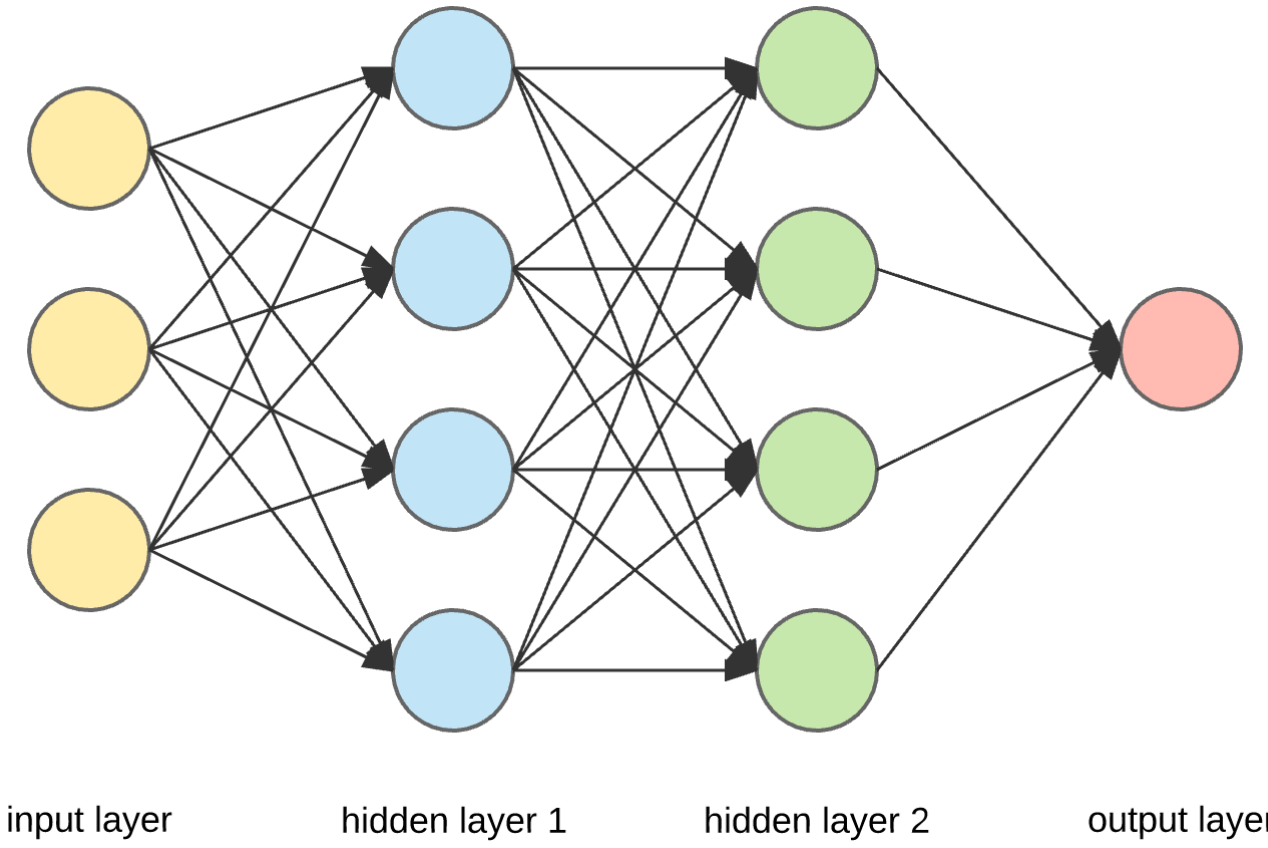
https://miro.medium.com/max/700/1*iWQzxhVlvadk6VAJjsgXgg.png

AVOID UNDERFITTING

- Increase model complexity
- Reduce the regularization parameters

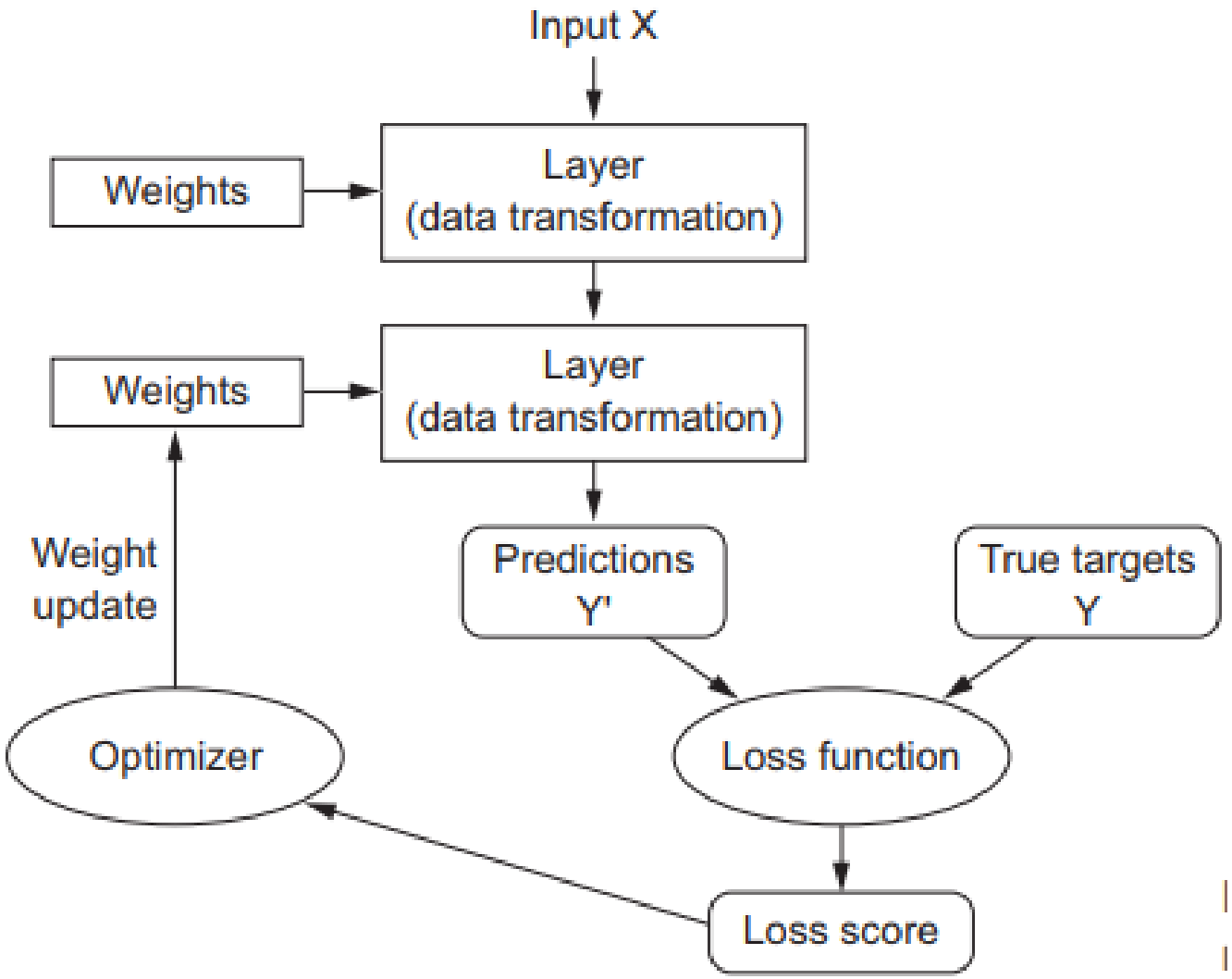
NEURAL NETWORK

- Input layer
- Hidden layer
- Output layer



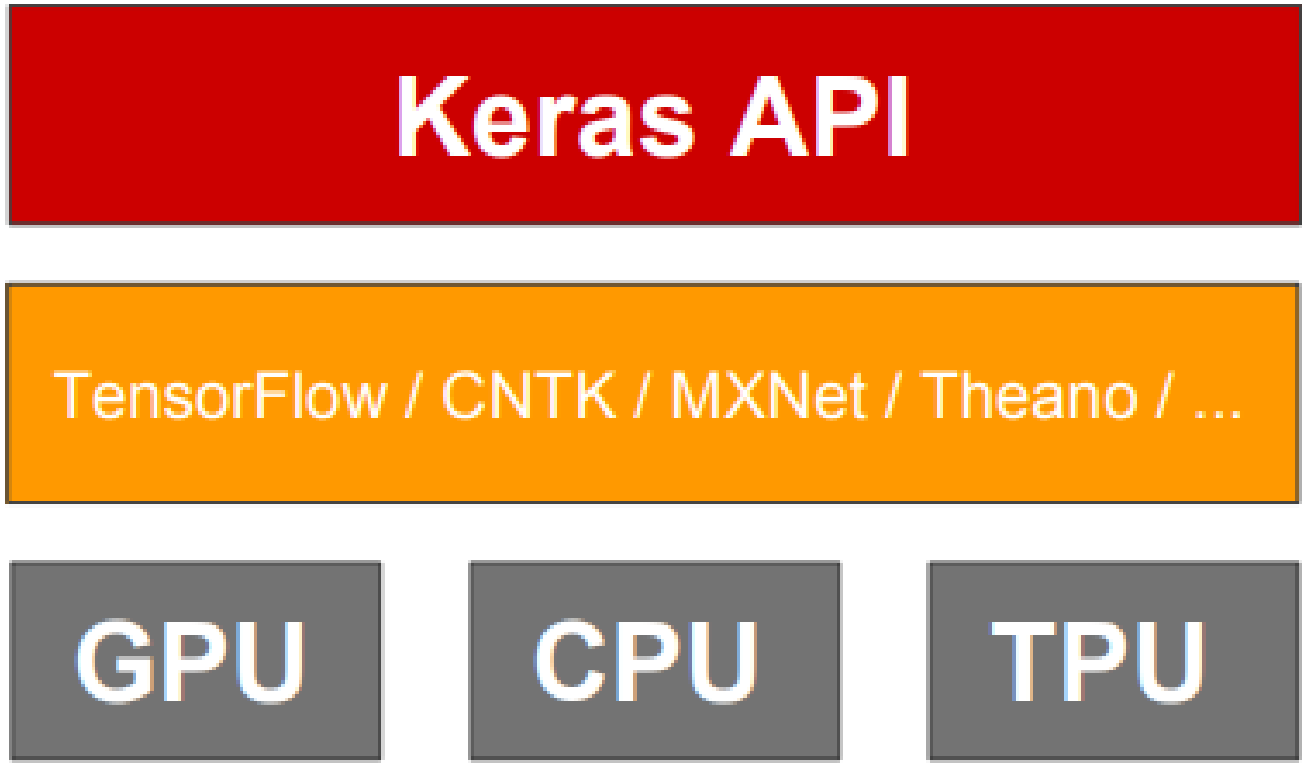
ANATOMY OF A NEURAL NETWORK

- 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



KERAS

- Keras is multi-backend, multi-platform
- Easy productization of models

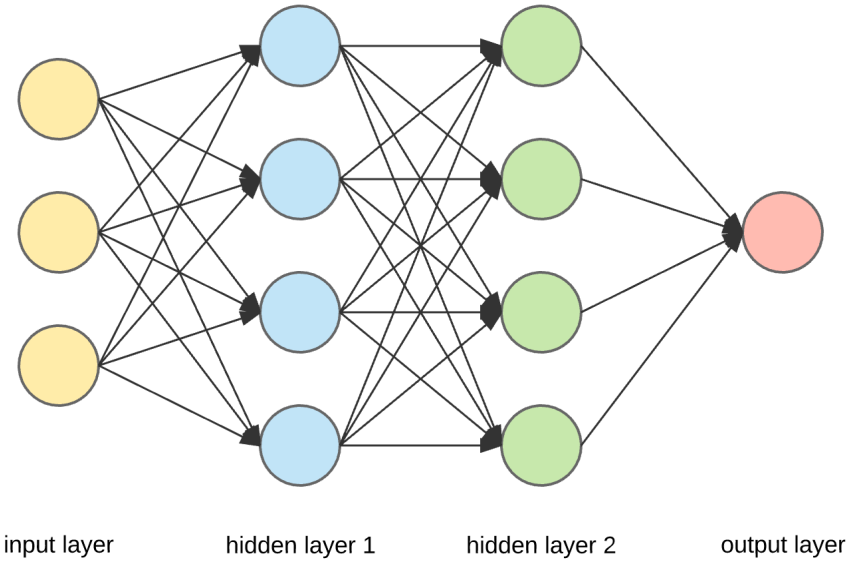


https://web.stanford.edu/class/cs20si/lectures/marchguestlecture.pdf

THREE API STYLES

- The sequential model
- The functional API
- Model sub classing

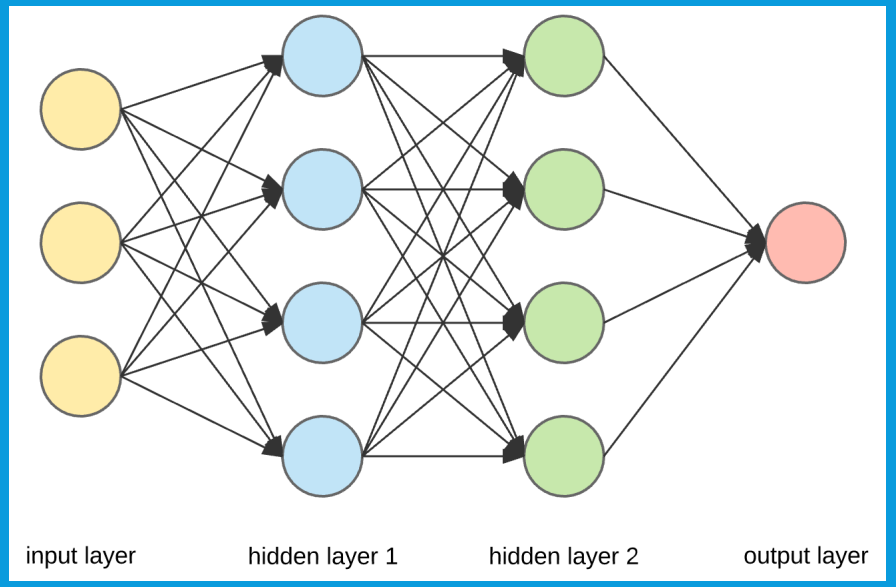
THE SEQUETIAL API




```
import keras
from keras import layers
model=keras.Sequential()
model.add(layers.Dense(4,activation='relu',input_shape=(3,)))
model.add(layers.Dense(4,activation='relu'))
model.add(layers.Dense(10,activation='softmax'))
model.fit(x,y, epochs=10, batch_size=32)
```

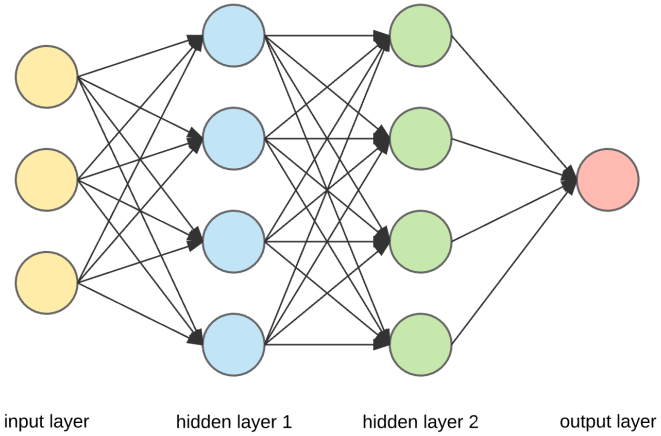
THE FUNCTIONAL API

```
import keras
from keras import layers, Input
inputs=Input(shape=(3,))
x=layers.Dense(4,activation='relu')(inputs)
x=layers.Dense(4,activation='relu')(x)
outputs=layers.Dense(10,activation='softmax')(x)
model=keras.Model(inputs,outputs)
model.fit(x,y,epochs=10,batch_size=32)
```



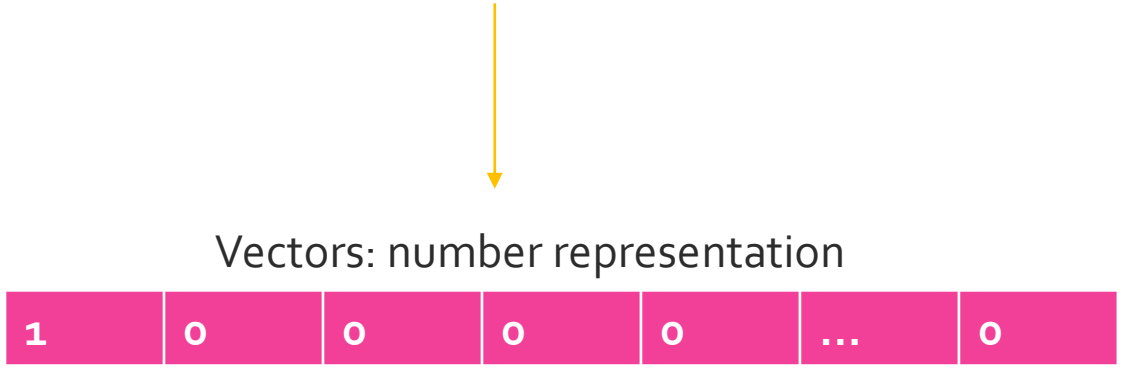
MODEL SUB CLASSING

```
import keras
from keras import layers
class MyModel(keras.Model):
        def __init__(self):
                super(MyModel, self).__init__()
                self.dense1=layers.Dense(4,activation='relu')
                self.dense2=layers.Dense(4,activation='relu')
                self.dense3=layers.Dense(10,activation='softmax')
        def call(self, inputs):
                x=self.dense1(x)
                x=self.dense2(x)
                return self.dense3(x)
model =MyModel()
model.fit(x,y, epochs=10,batch_size=32)
```



DEEP LEARNING FOR TEXT AND SEQUENCES

The cat sat on the mat




```
from keras.preprocessing.text import Tokenizer
samples = ['The cat sat on the mat.', 'The dog ate my homework.']
tokenizer = Tokenizer(num_words=1000)
tokenizer.fit_on_texts(samples)
sequences = tokenizer.texts_to_sequences(samples)
one_hot_results = tokenizer.texts_to_matrix(samples, mode='binary')
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

USING WORD EMBEDDING

- One-hot word vectors
  - Sparse
  - High-dimensional
  - Hardcoded
  - N: total number of words (1M, 2M .....)
- Word embeddings
  - Dense
  - Lower-dimensional
  - Learned from data
  - K: dimension (say 100,200,..)

One-hot word vectors

	1	2	3	4	5	…	N
Man	1	0	0	0	0	…	0
Girl	0	1	0	0	0	…	0
Boy	0	0	1	0	0	…	0
….							

Word embeddings

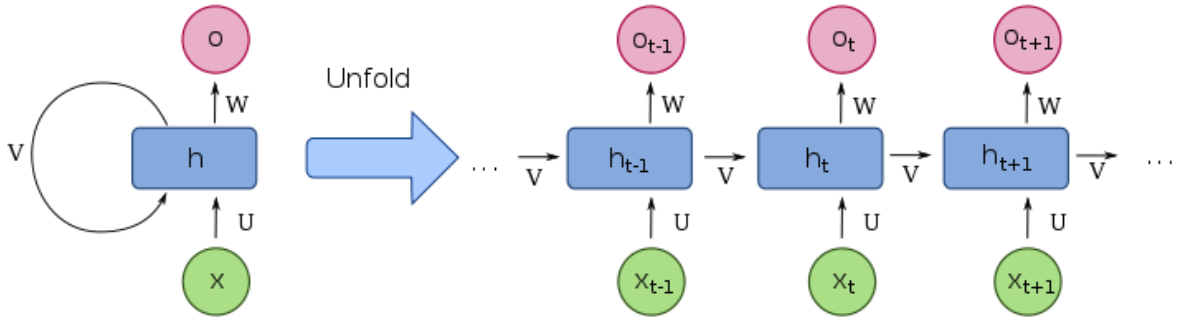
	1	2	3	4	5	…	K
Man	0.86	0.02	0.05	0.02	0.10	…	0.05
Girl	0.23	0.65	0.03	0.04	0.22	…	0.03
Boy	0.03	0.08	0.95	0.01	0.02	…	0.04
….							

LEARNING WORD EMBEDDING WITH THE EMBEDDING LAYER


```
from keras.layers import Embedding
embedding_layer = Embedding(1000, 64)
```

RECURRENT NEURAL NETWORK

- It processes sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far.



SIMPLE RNN IN KERAS

```
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN

model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32))
model.summary()
```

CLASSIFICATION OF IMDB MOVIE REVIEW

- from keras.datasets import imdb
- from keras.preprocessing import sequence
- max_features = 10000
- maxlen = 500
- batch_size = 32
- (input_train, y_train), (input_test, y_test) = imdb.load_data(num_words=max_features)
- input_train = sequence.pad_sequences(input_train, maxlen=maxlen)
- input_test = sequence.pad_sequences(input_test, maxlen=maxlen)

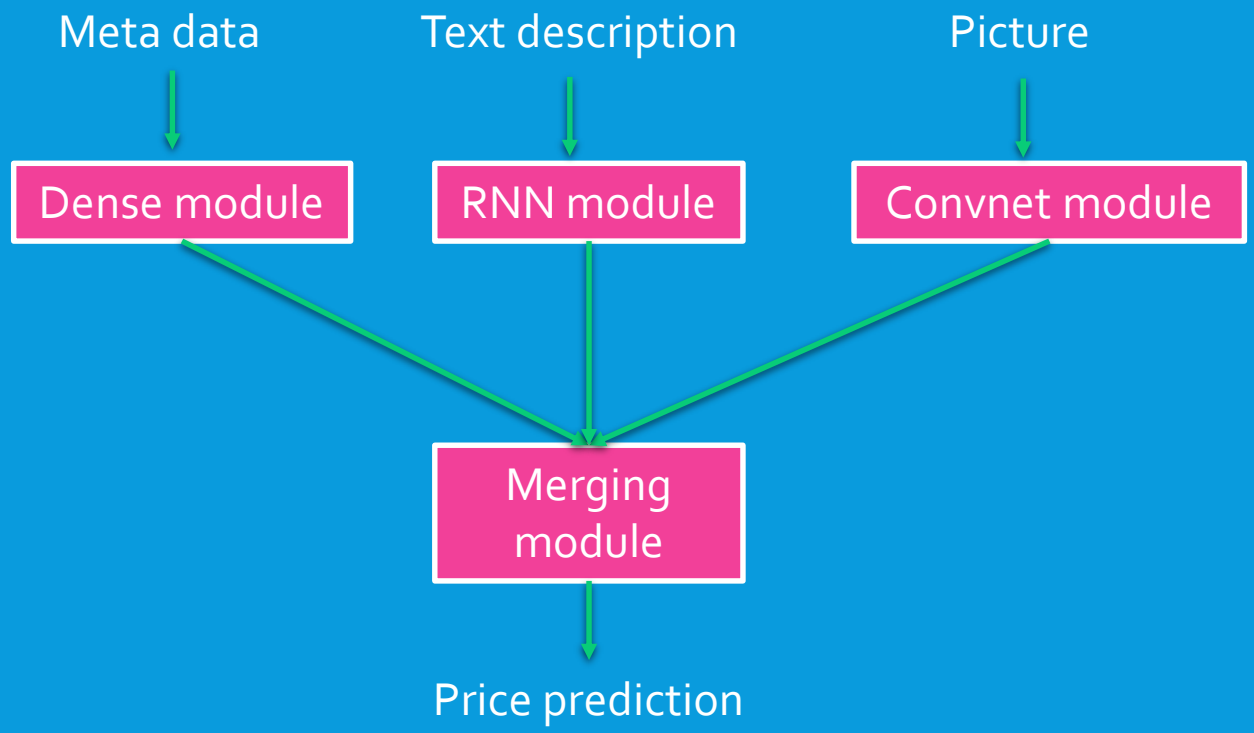
TRAINING RNN

- from keras.layers import Dense
- model = Sequential()
- model.add(Embedding(max_features, 32))
- model.add(SimpleRNN(32))
- model.add(Dense(1, activation='sigmoid'))
- model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
- history = model.fit(input_train, y_train, epochs=10, batch_size=128, validation_split=0.2)

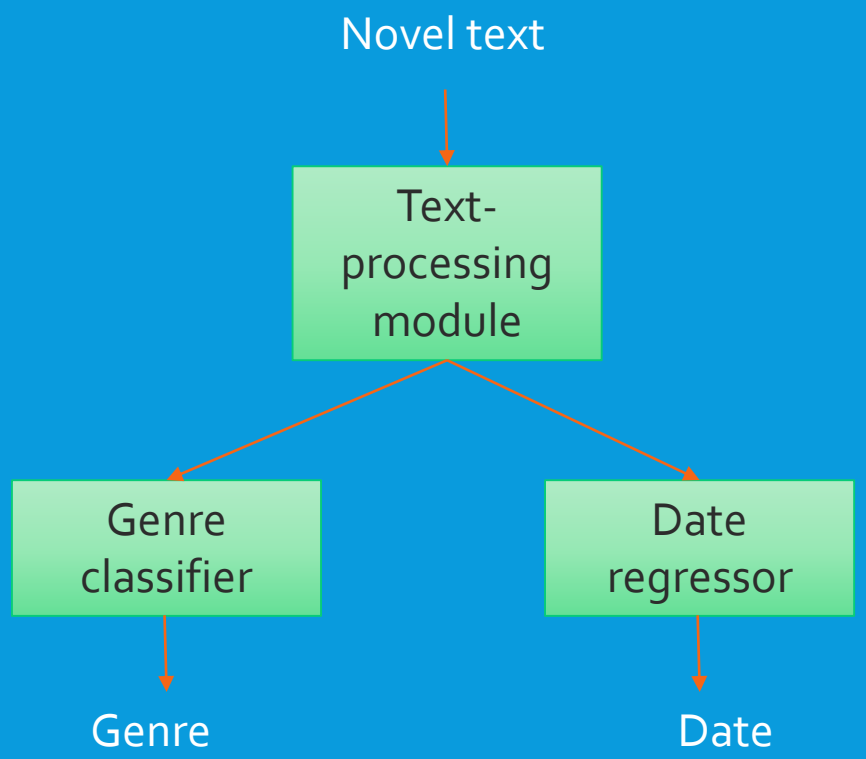
ADVANCED DEEP LEARNING PRACTICES

- Multi-input model
- Multi-output model

MULTI-INPUT MODEL



MULTIPLE OUTPUT MODEL



REFERENCES

- Francois Chollet. 2017. Deep Learning with Python (1st ed.). Manning Publications Co., Greenwich, CT, USA.
- https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.html
- https://keras.io/
- https://www.tensorflow.org/guide/keras

THANKS