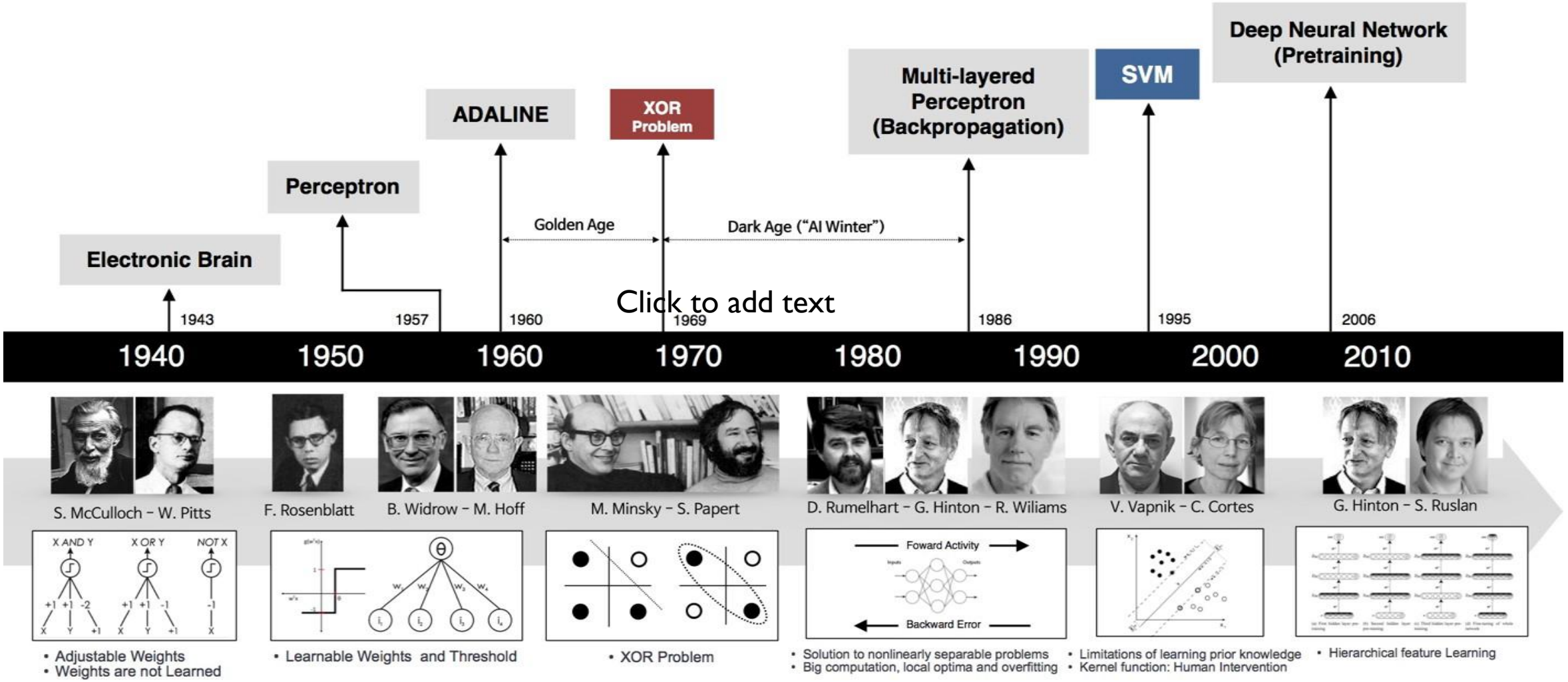
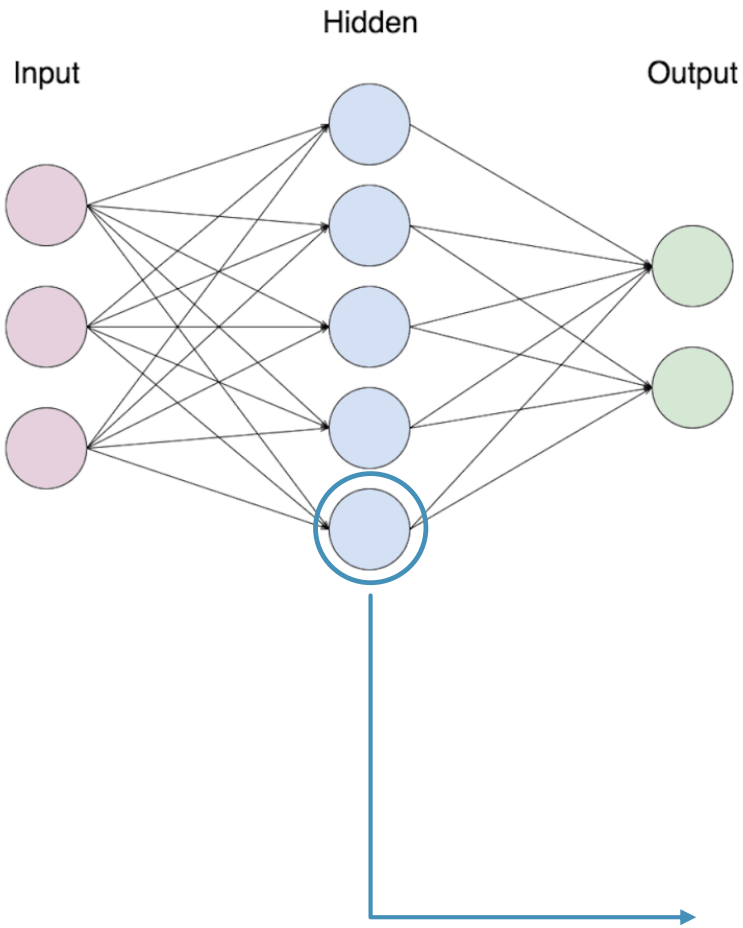
A DIVE INTO DEEP LEARNING AND APPLICATIONS

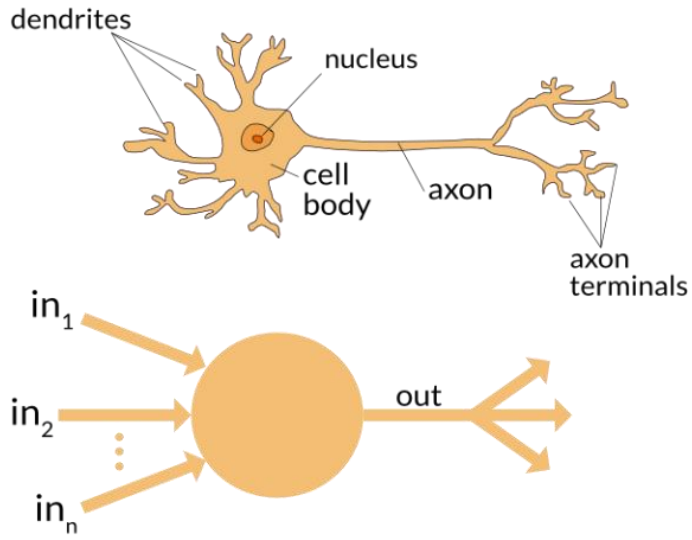
DEVELOPMENTS OF NEURAL NETS



WHAT IS A NEURAL NETWORK ?

- Base of Deep learning, a sub-field of Machine learning
- Inspired by the structure of a human brain, mimicking the way that biological neurons signal to one another

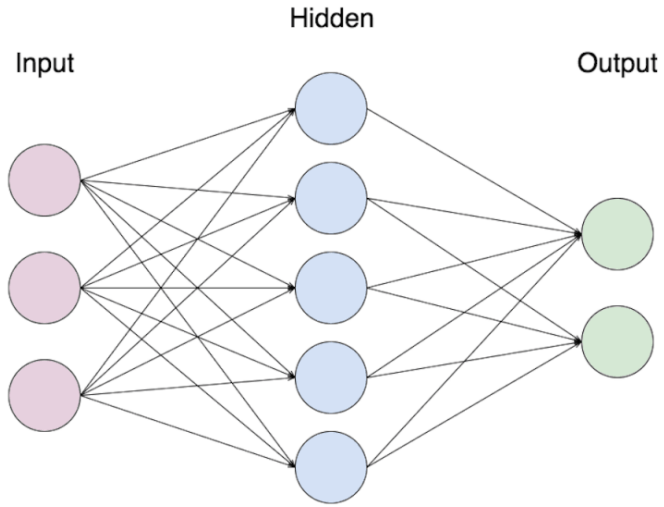


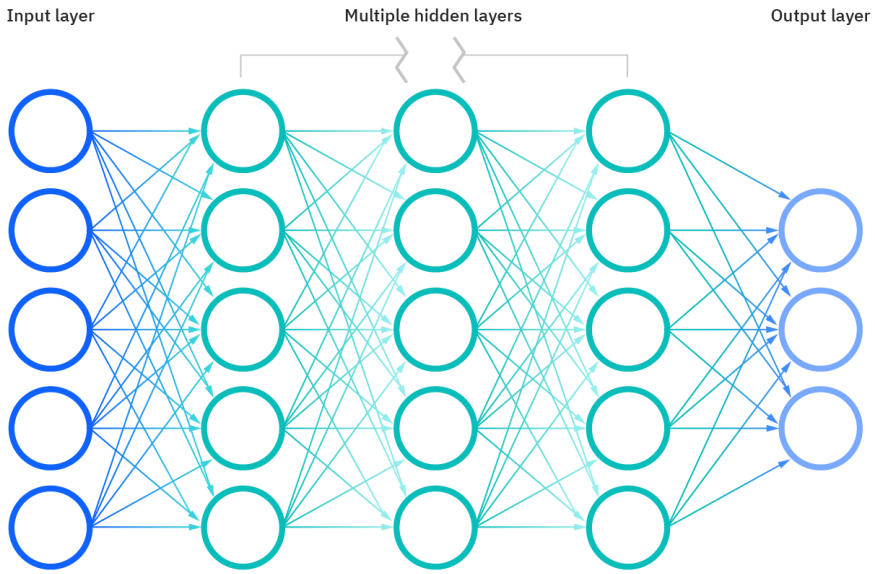


WHAT IS A NEURAL NETWORK ?

- The “deep” in Deep Learning is just referring to the depth of layers in a neural network
- Having two or more hidden layers counts as deep
- In contrast, a network with only a single hidden layer is conventionally called "shallow"

SHALLOW NEURAL NETWORK

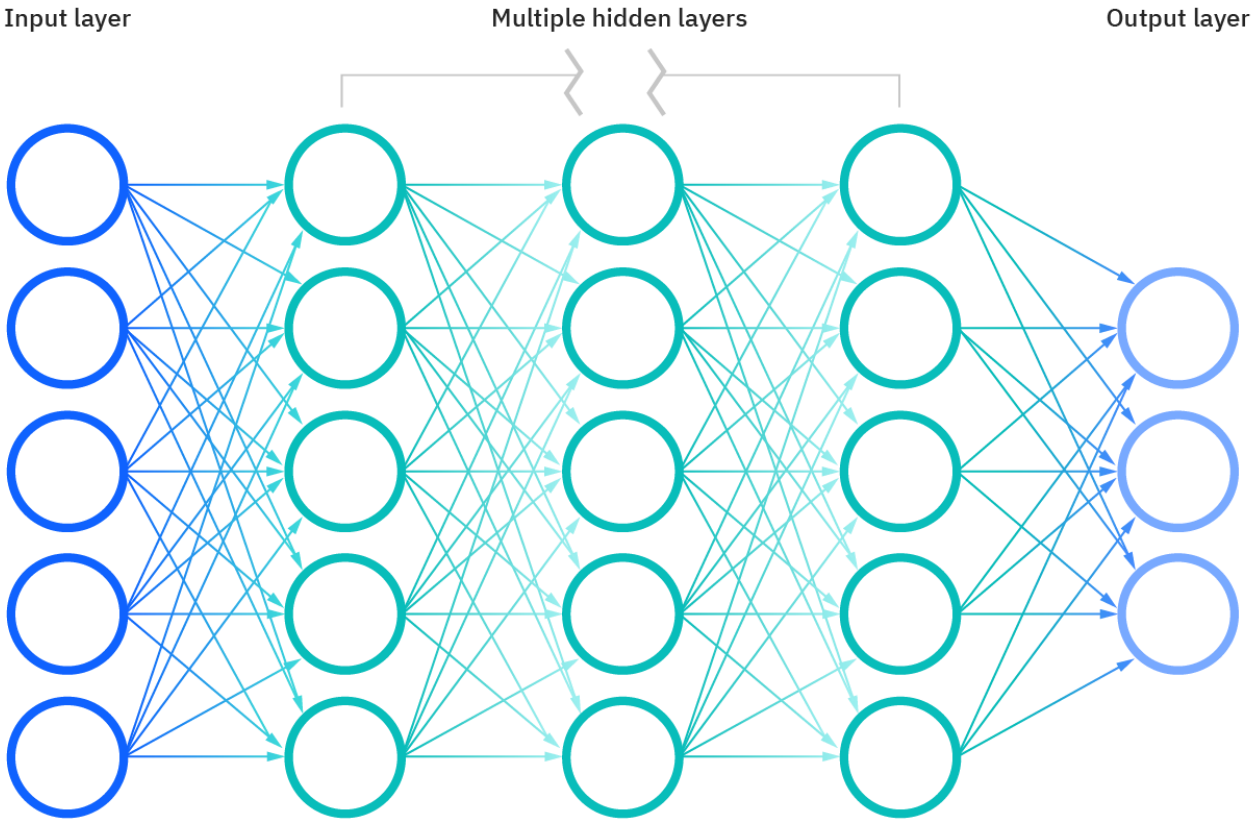




DEEP NEURAL NETWORK

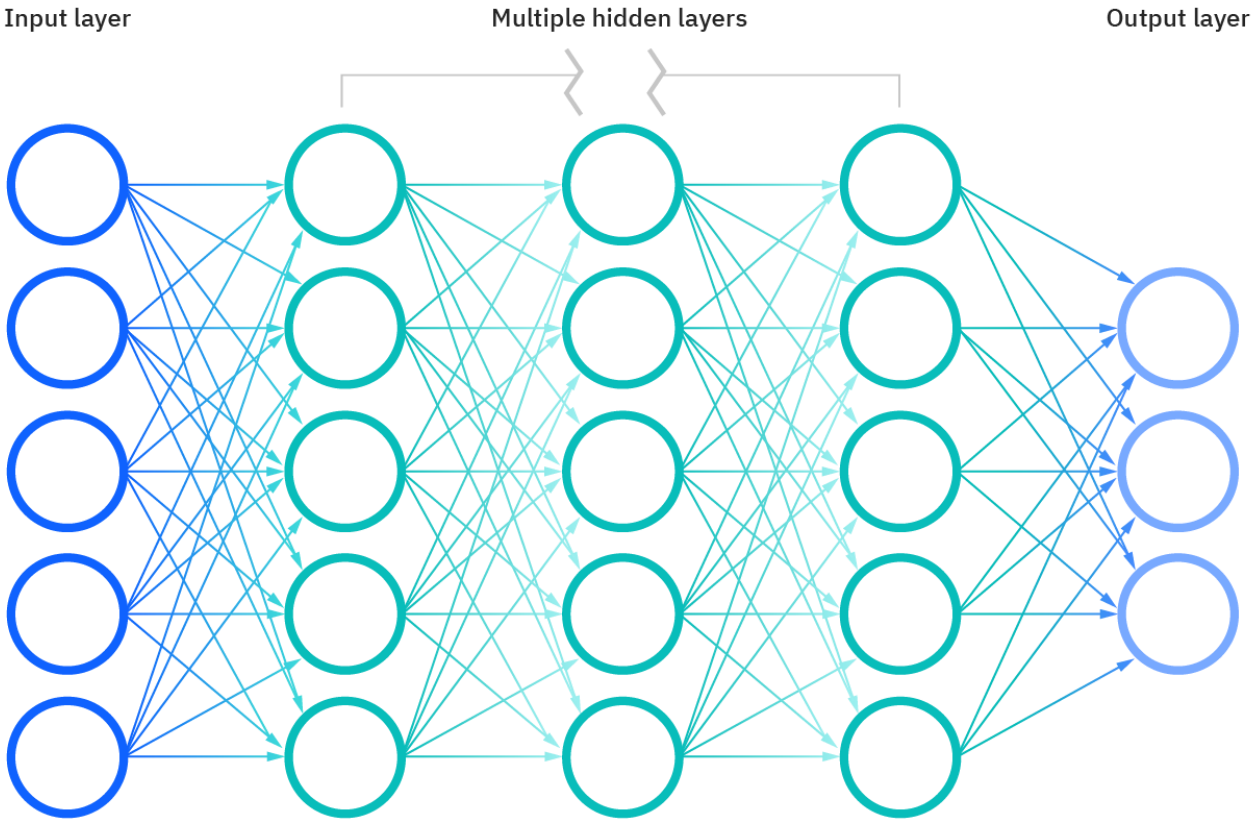
WHAT IS A NEURAL NETWORK ?

- Neural networks take in data, train themselves to recognize the patterns, and then predicts the output



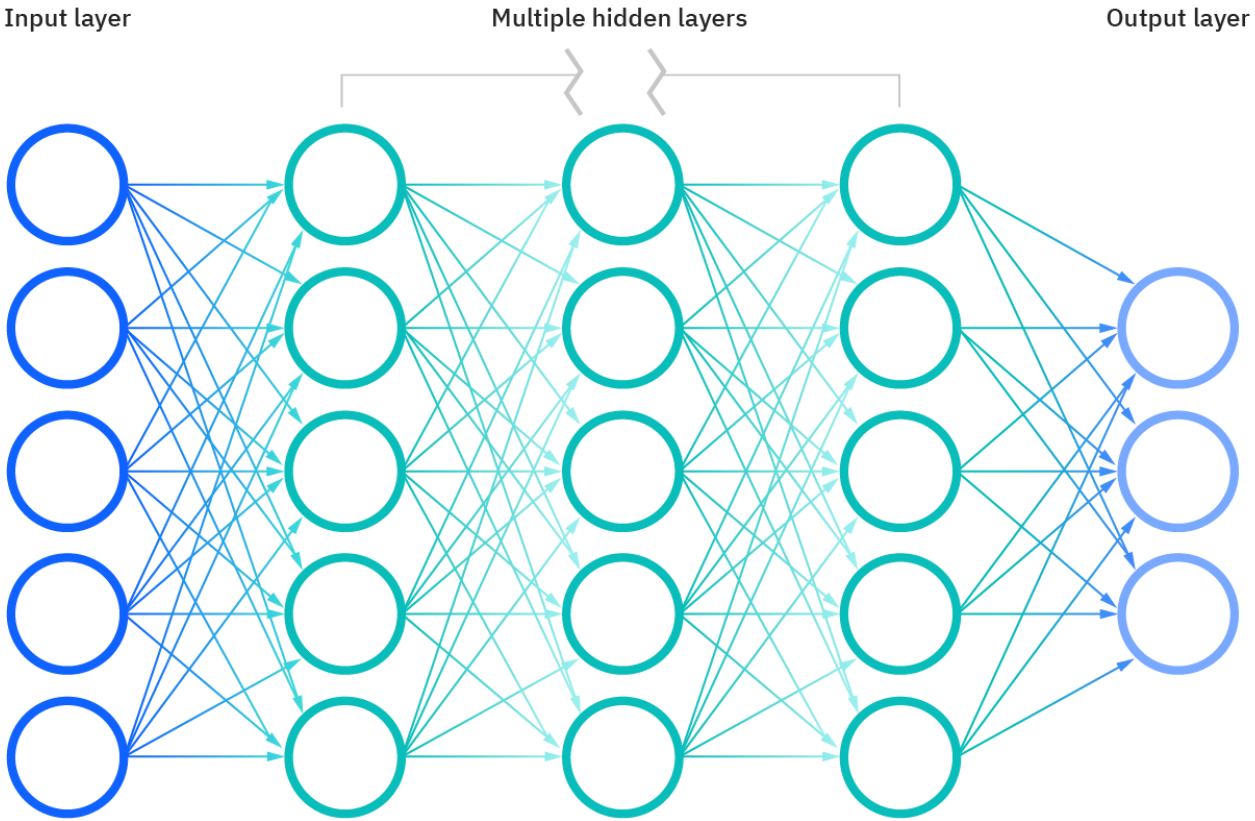
WHAT IS A NEURAL NETWORK ?

- Comprised of node layers, containing an input layer, one or more hidden layers, and an output layer
- Each node, or artificial neuron, connects to another and has an associated weight and threshold



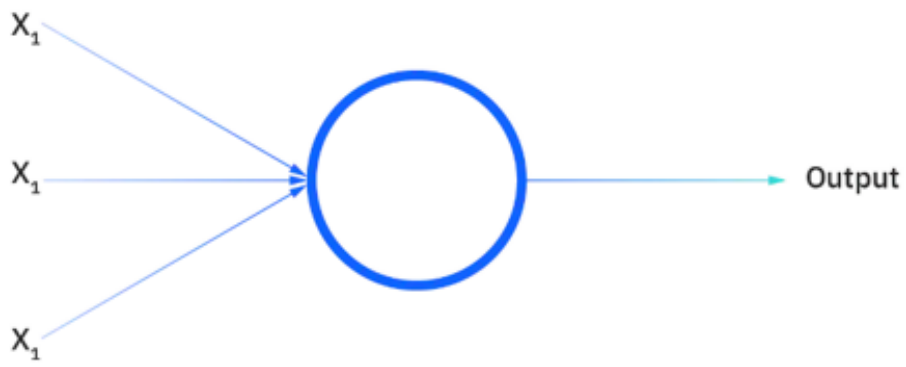
WHAT IS A NEURAL NETWORK ?

- If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network
- Otherwise, no data is passed along to the next layer of the network
- We will learn more on the next step-by-step example

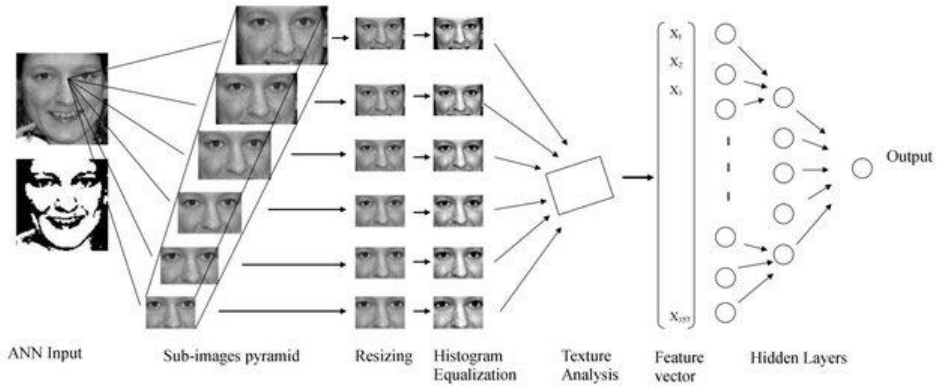


WHAT IS A NEURAL NETWORK ?

- The perceptron is the oldest neural network, created by Frank Rosenblatt in 1958
- It has a single neuron and is the simplest form of a neural network



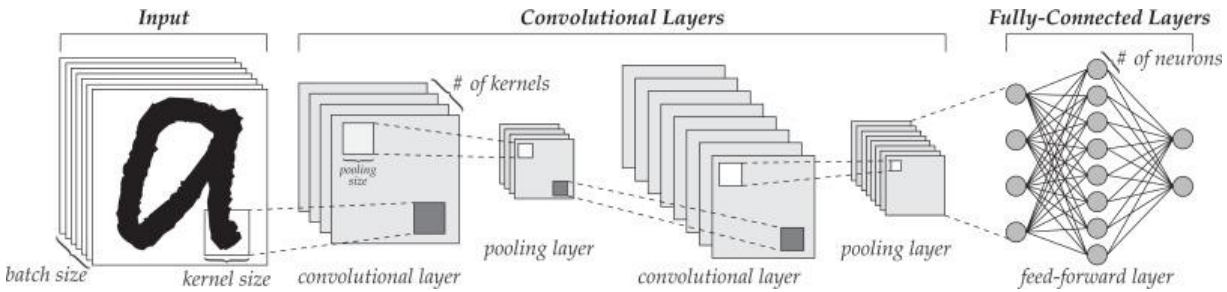
NEURAL NETWORKS APPLICATIONS



FACIAL RECOGNITION

- SPEECH RECOGNITION

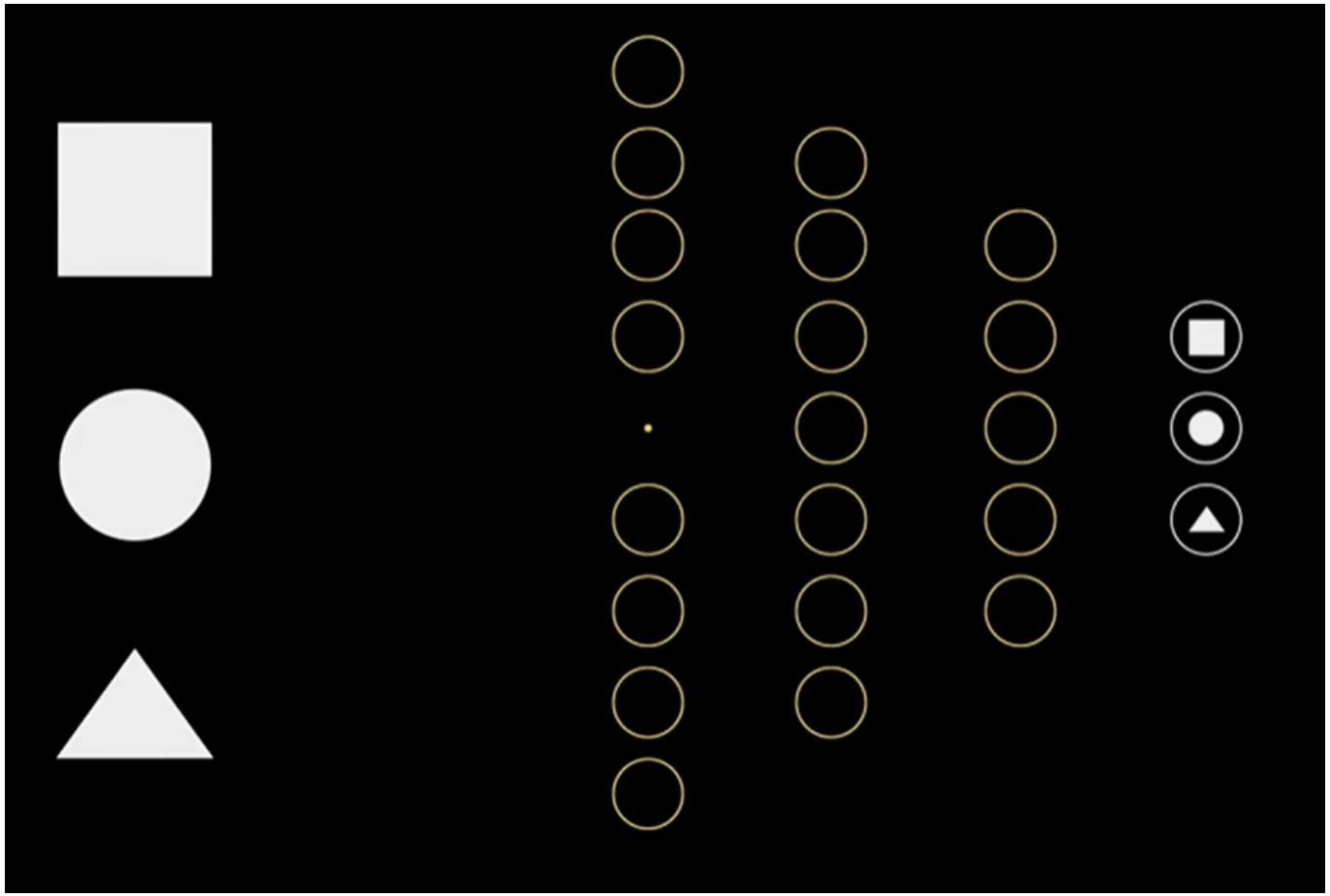
- TEXT TRANSLATION



HANDWRITING RECOGNITION

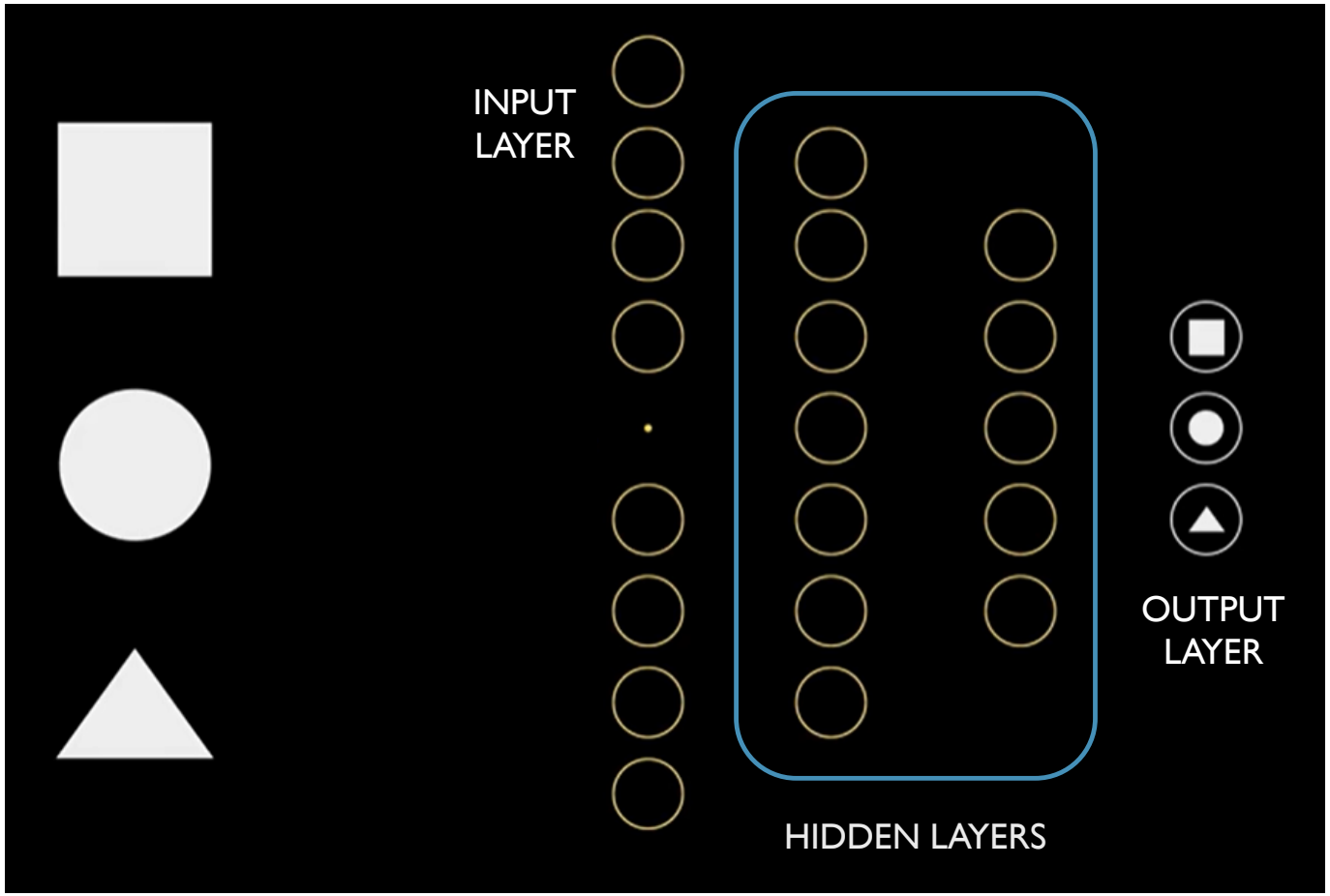
HOW DOES IT REALLY WORK?

- Let's take an example of classifying an object
- Our goal is to take the input, and match it to one of the 3 classes; square, circle, or triangle



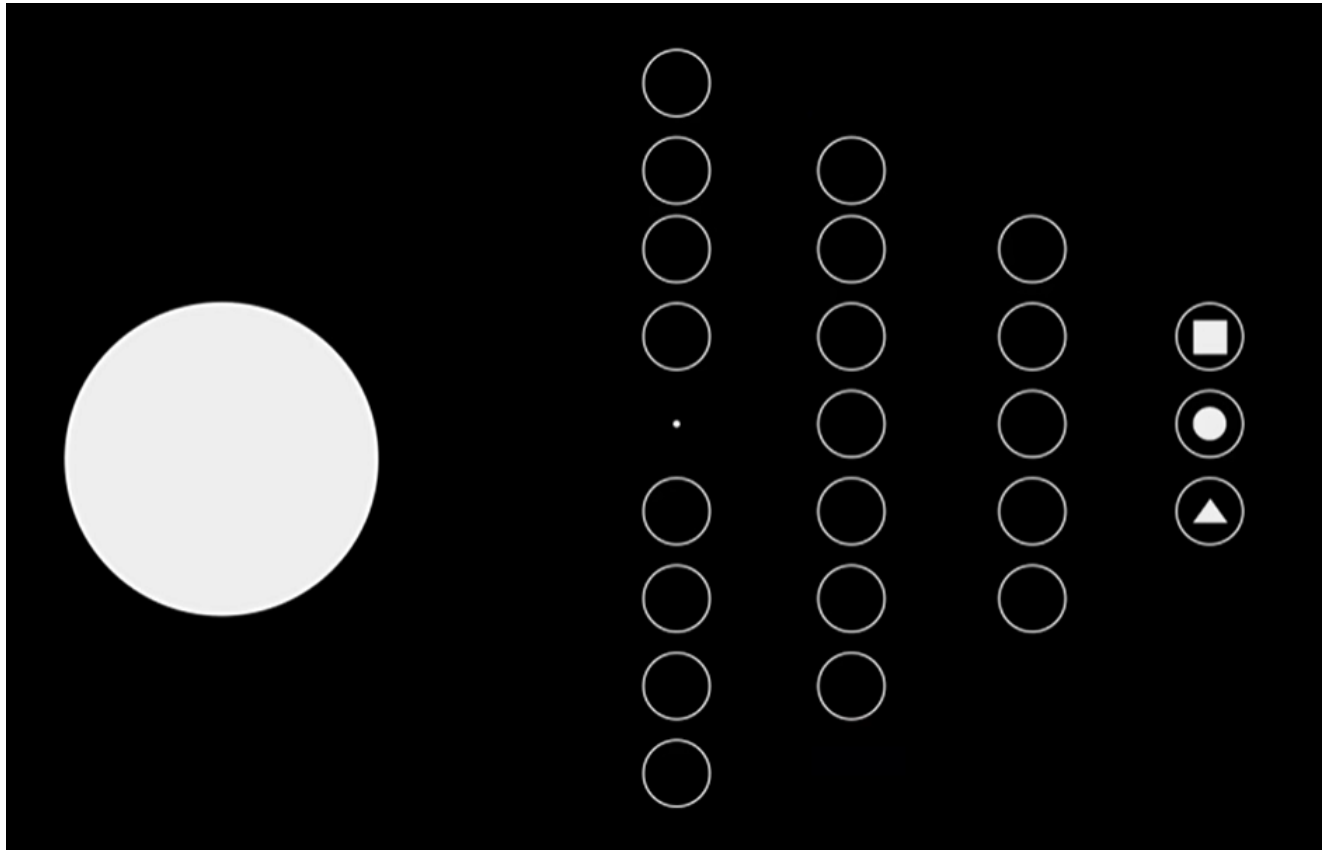
HOW DOES IT REALLY WORK?

- Hidden Layers can automatically extract features from data
- Part of the art of Neural Networks is deciding how many Hidden Layers to use and how many Nodes should be in each one



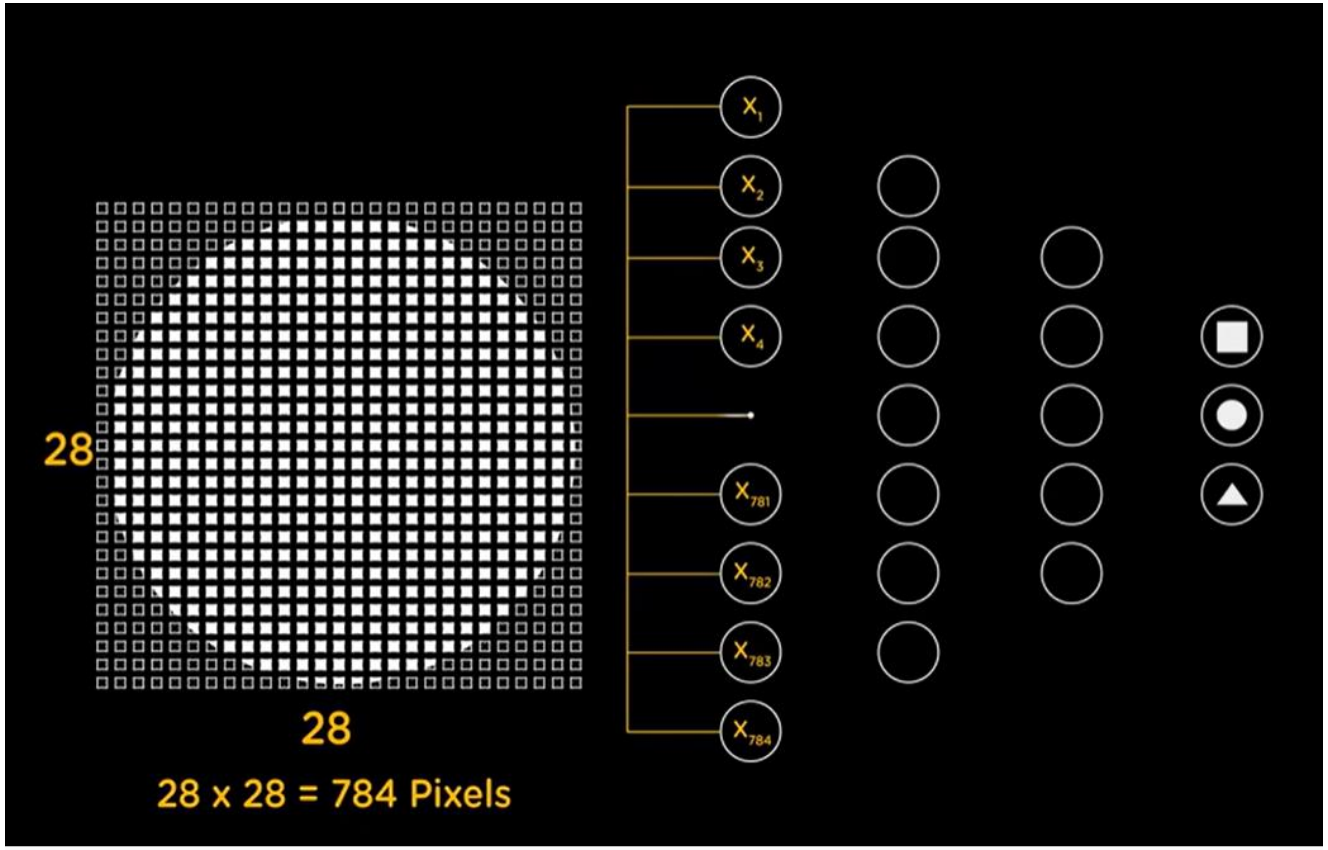
HOW DOES IT REALLY WORK?

- Assuming that we have this circle as an input that we need to classify



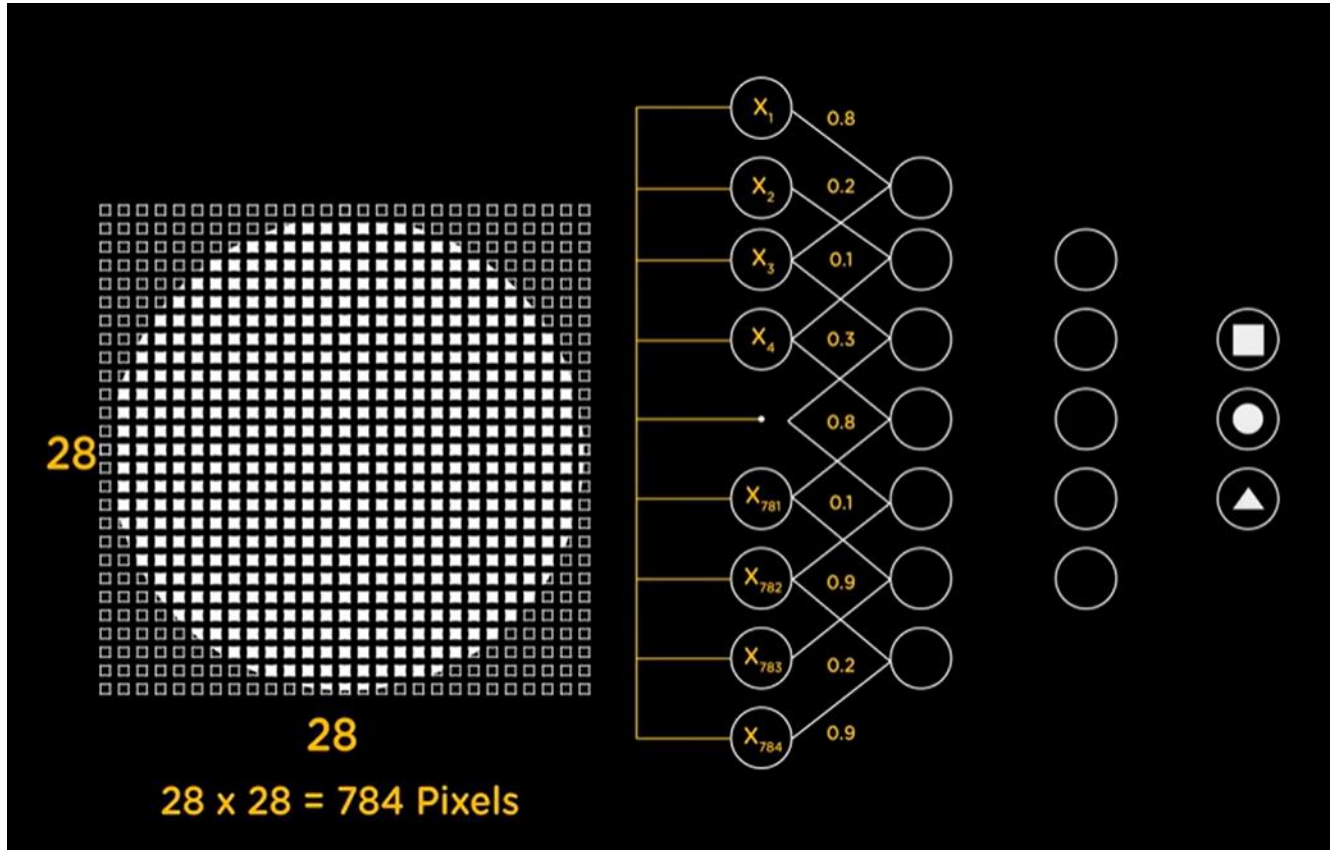
HOW DOES IT REALLY WORK?

- The circle image has a 784 pixels, and each pixel will go as an input to the first input layer



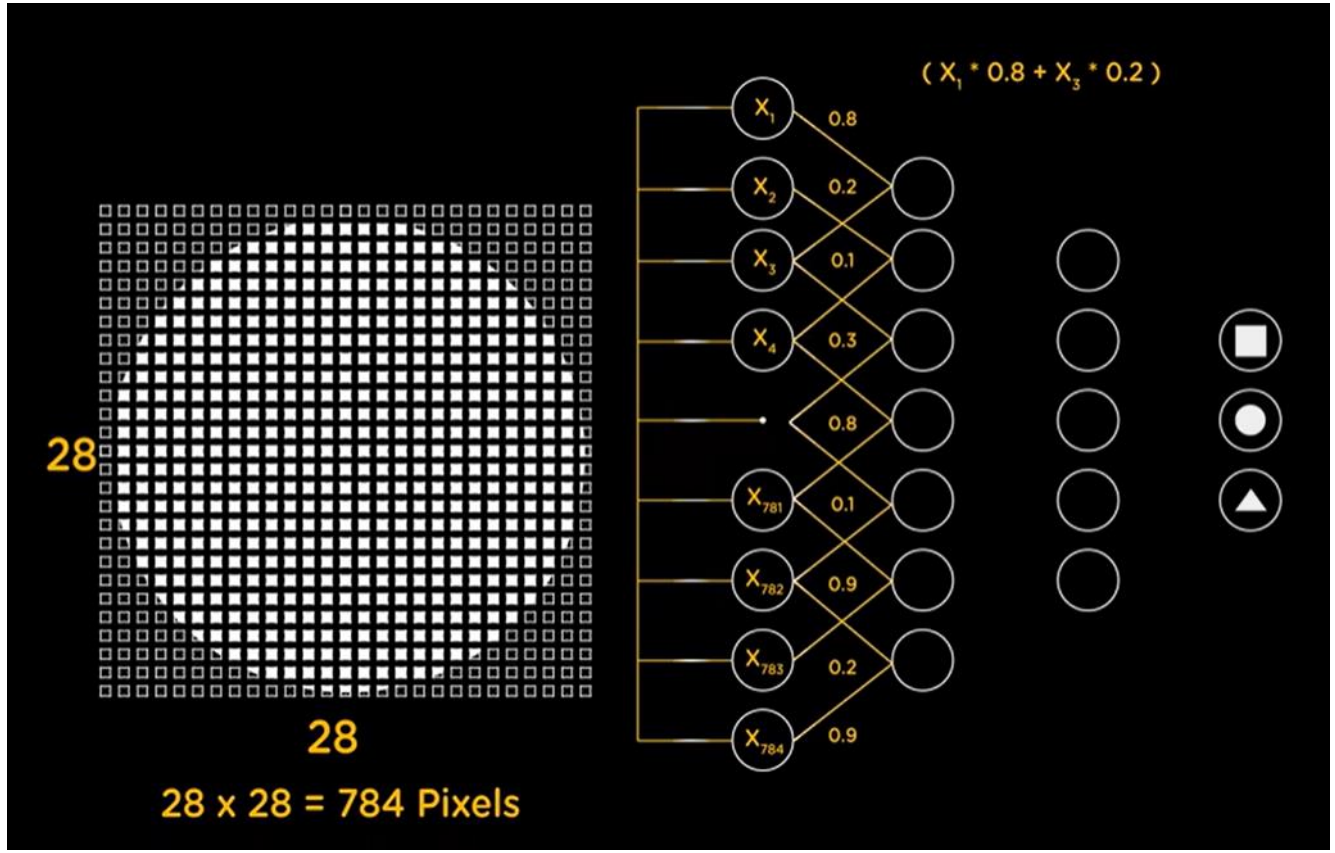
HOW DOES IT REALLY WORK?

- The nodes of each layer are connected to the next layer nodes, with Channels
- Each of these Channels is a assigned a numerical value, known as weights



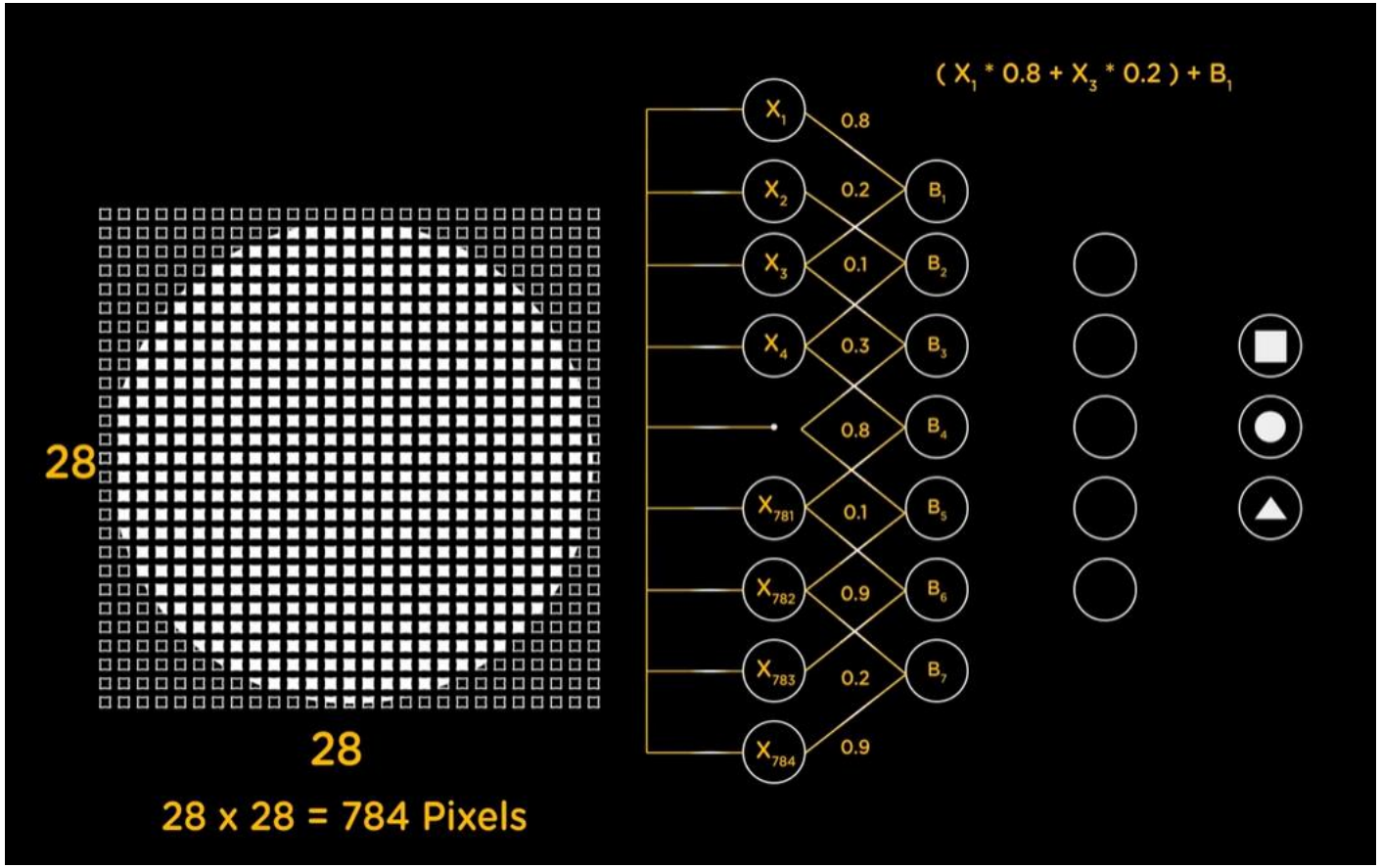
HOW DOES IT REALLY WORK?

- The inputs are now multiplied with the corresponding weights, and their sum is sent as input to the Hidden Layer nodes



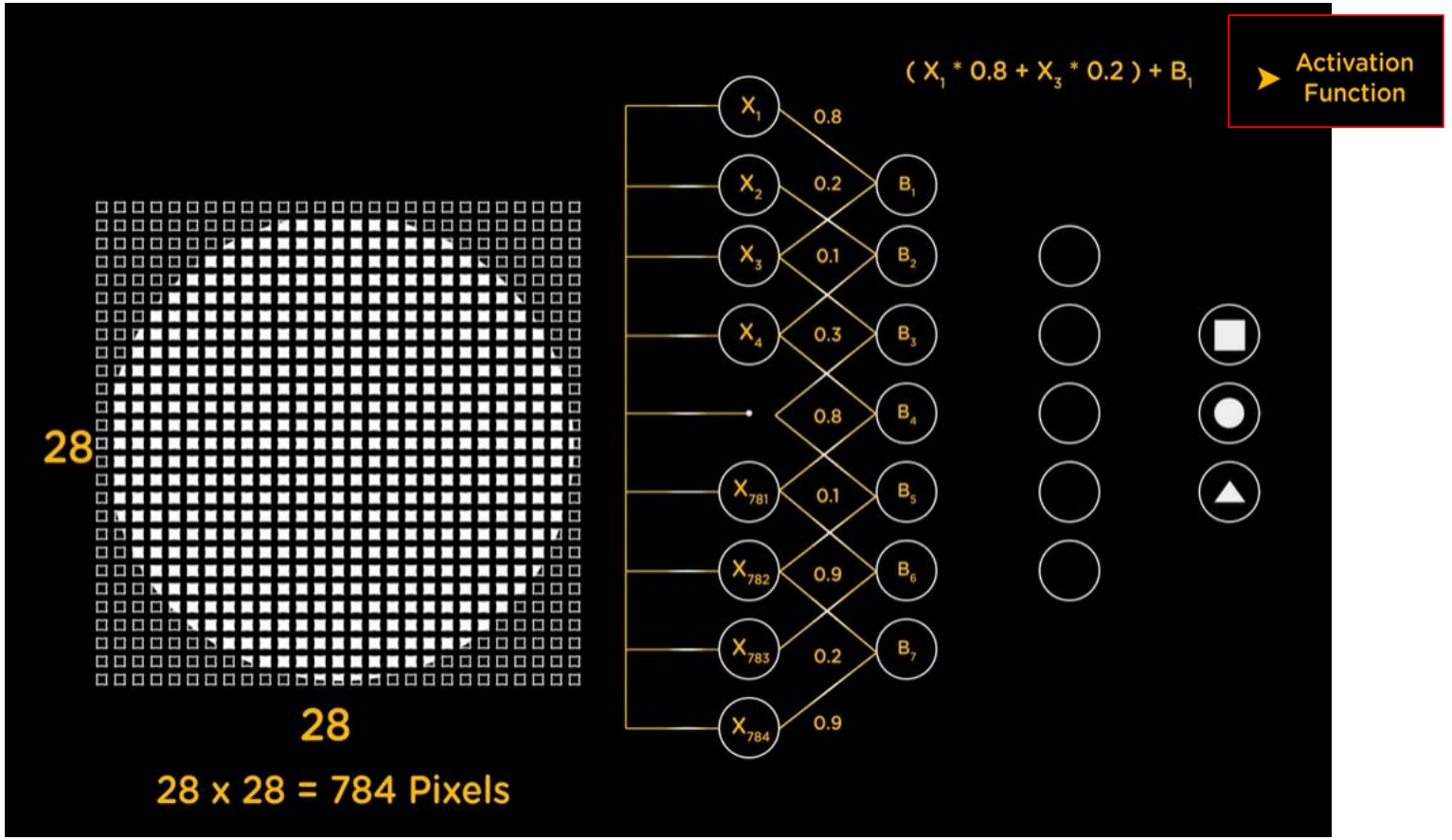
HOW DOES IT REALLY WORK?

- Each of the Hidden Layers' nodes is associated with a numerical value, called the Bias
- The Bias will be added to the input sum



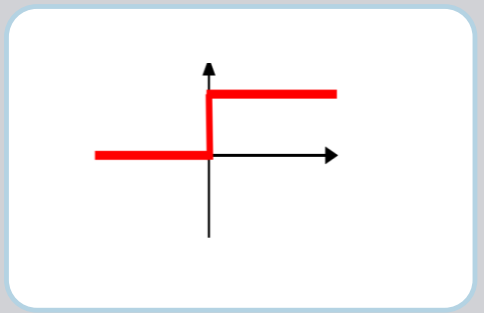
HOW DOES IT REALLY WORK?

- The input sum is then passed to a threshold function called the Activation function
- Activation function is a function that helps the network learn complex patterns in the data and introduce non-linearity



ACTIVATION FUNCTION

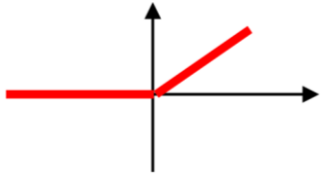
- Several types of activation functions are used, below are the most common ones:



Step function

One of the simplest kind of activation functions. In this, we consider a threshold value and if the value of net input say y is greater than the threshold then the neuron is activated

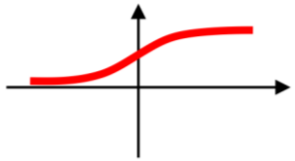
$$f(x) = 1, if\ x \geq 0$$
$$f(x) = 0, if\ x\ < 0$$



Rectified Linear Unit (ReLU)

The most widely used activation function. It is defined as:

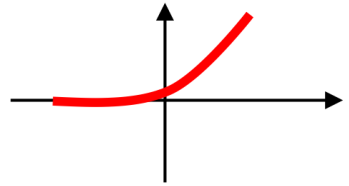
$$f(x) = \max(0, x)$$



Sigmoid

Sigmoid function is a widely used activation function. It is defined as:

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



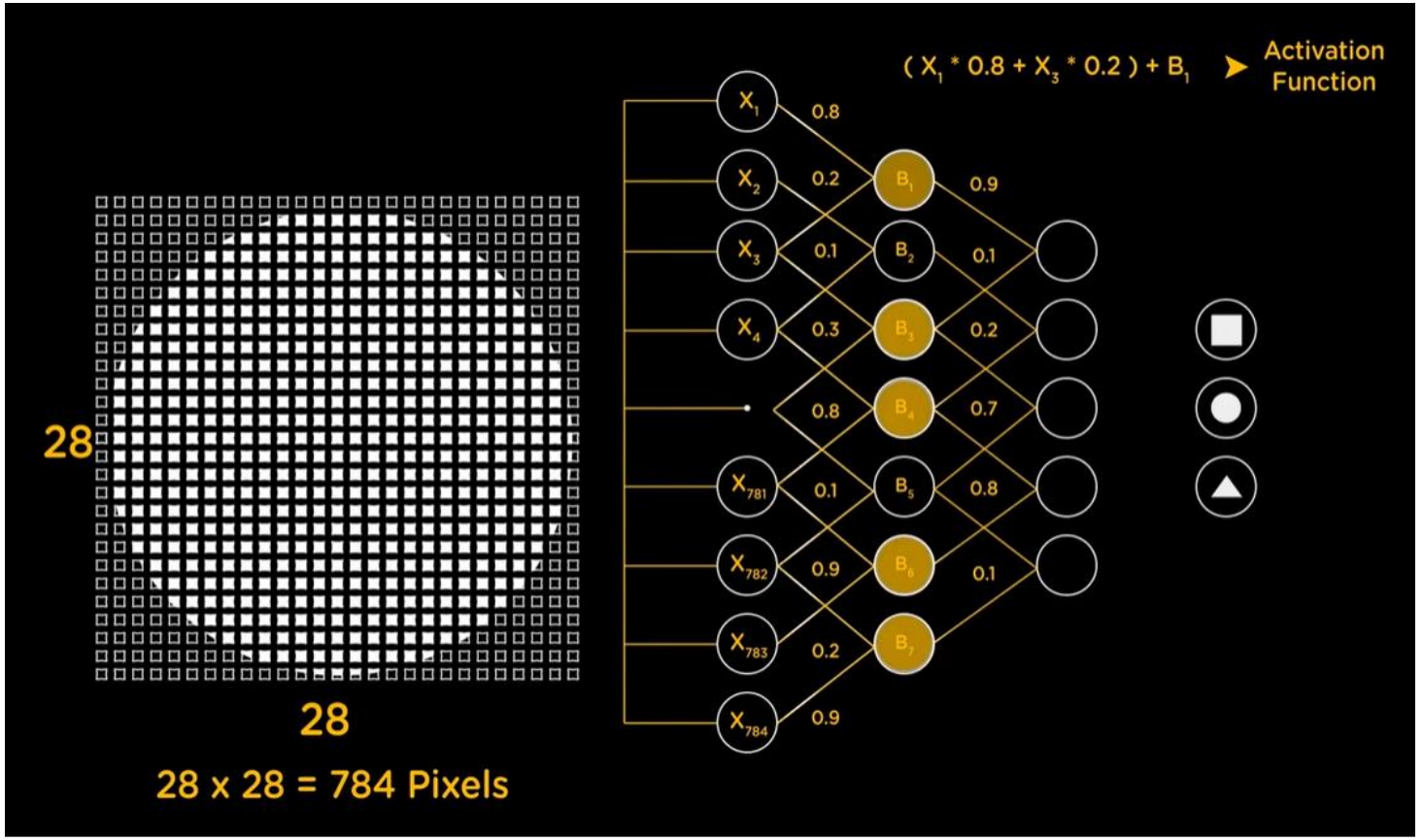
SoftPlus

modified form of the ReLU Activation Function. The main difference is that instead of the line being bent at 0, we get a curve. It is defined as:

$$f(x) = \log(1 + e^{x})$$

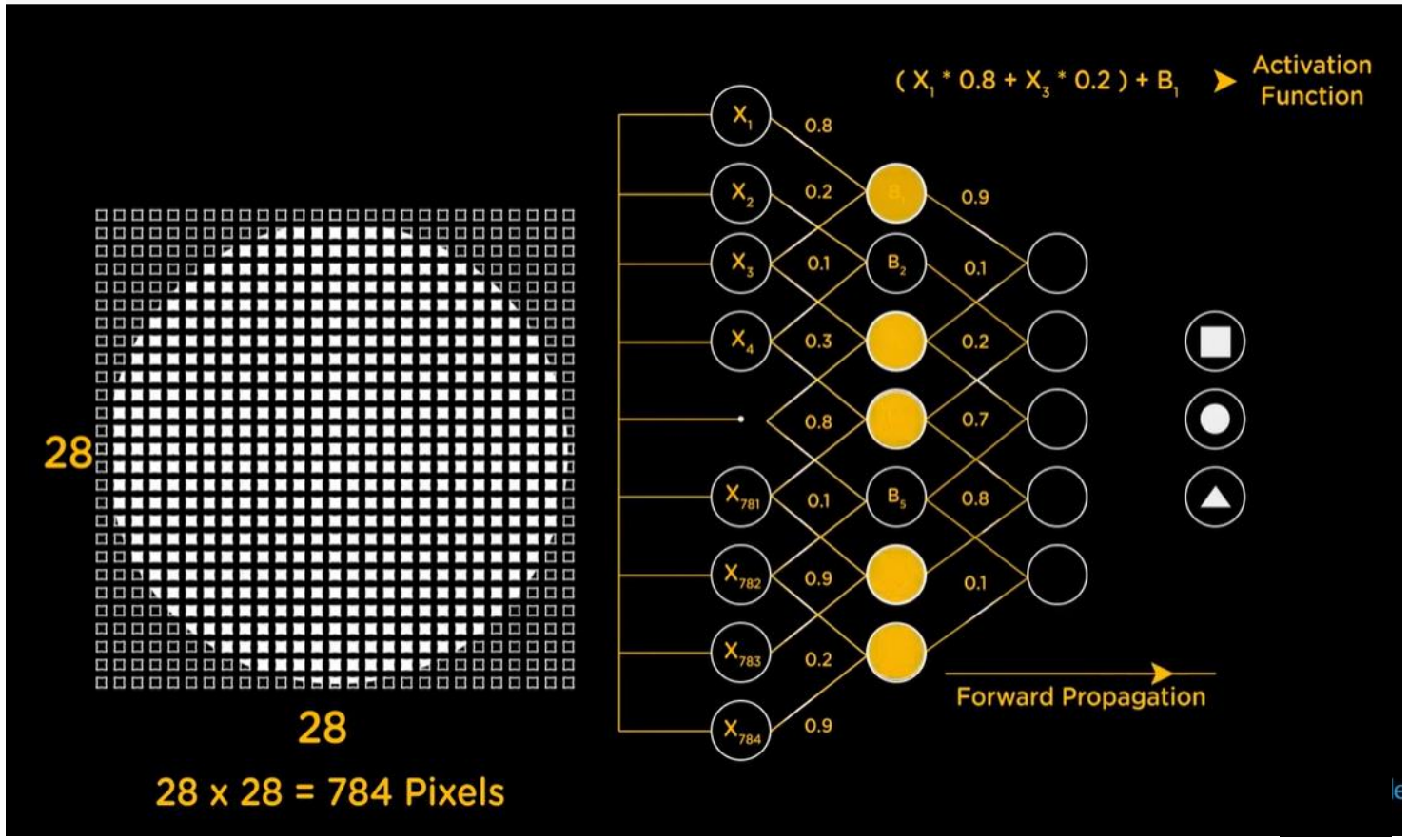
HOW DOES IT REALLY WORK?

- The result of the Activation function determines if the particular node will be activated or not
- An activated node will transmit data to the node of the next layer, over channels



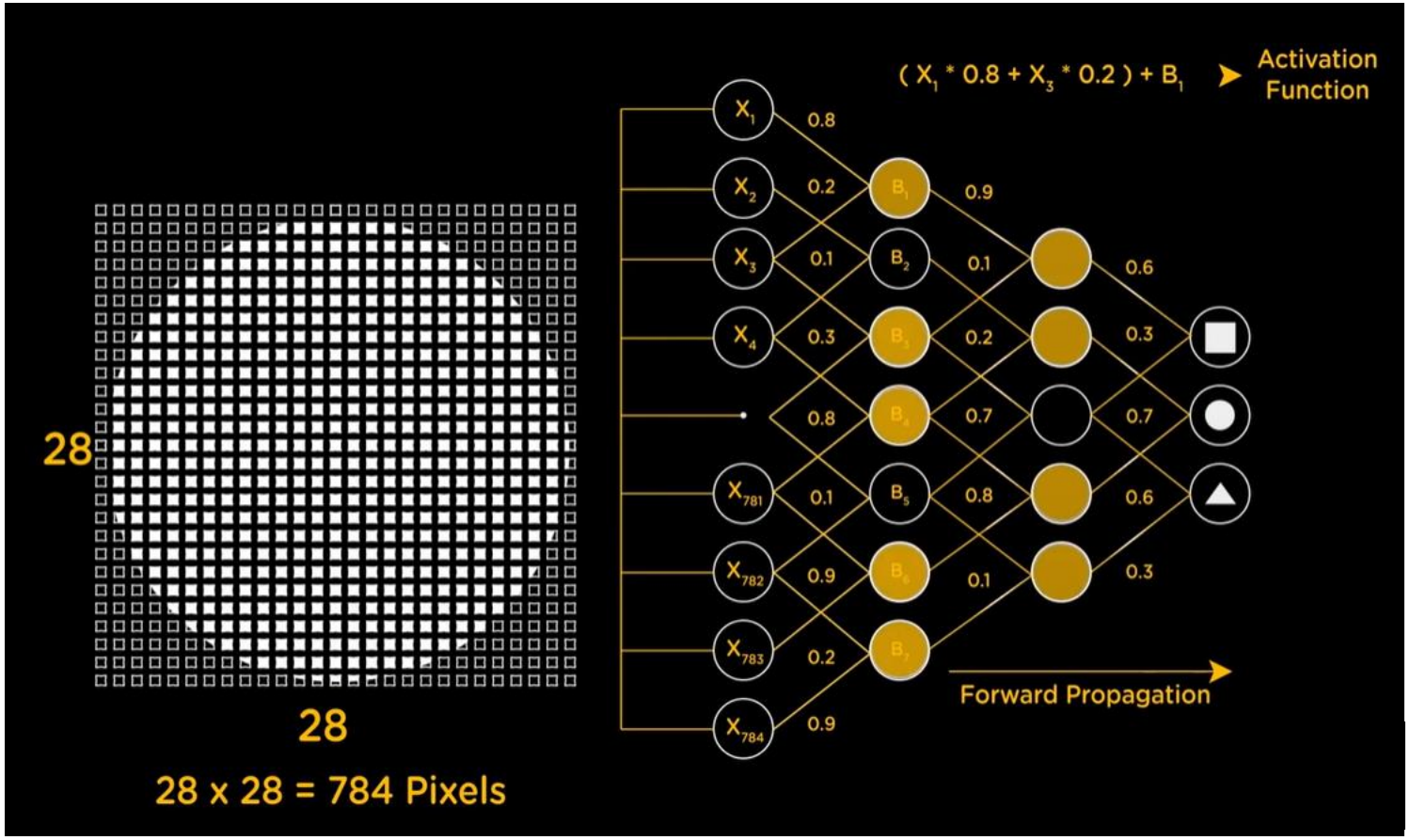
HOW DOES IT REALLY WORK?

- In this manner, the data are propagated through the network, referred to as Forward Propagation



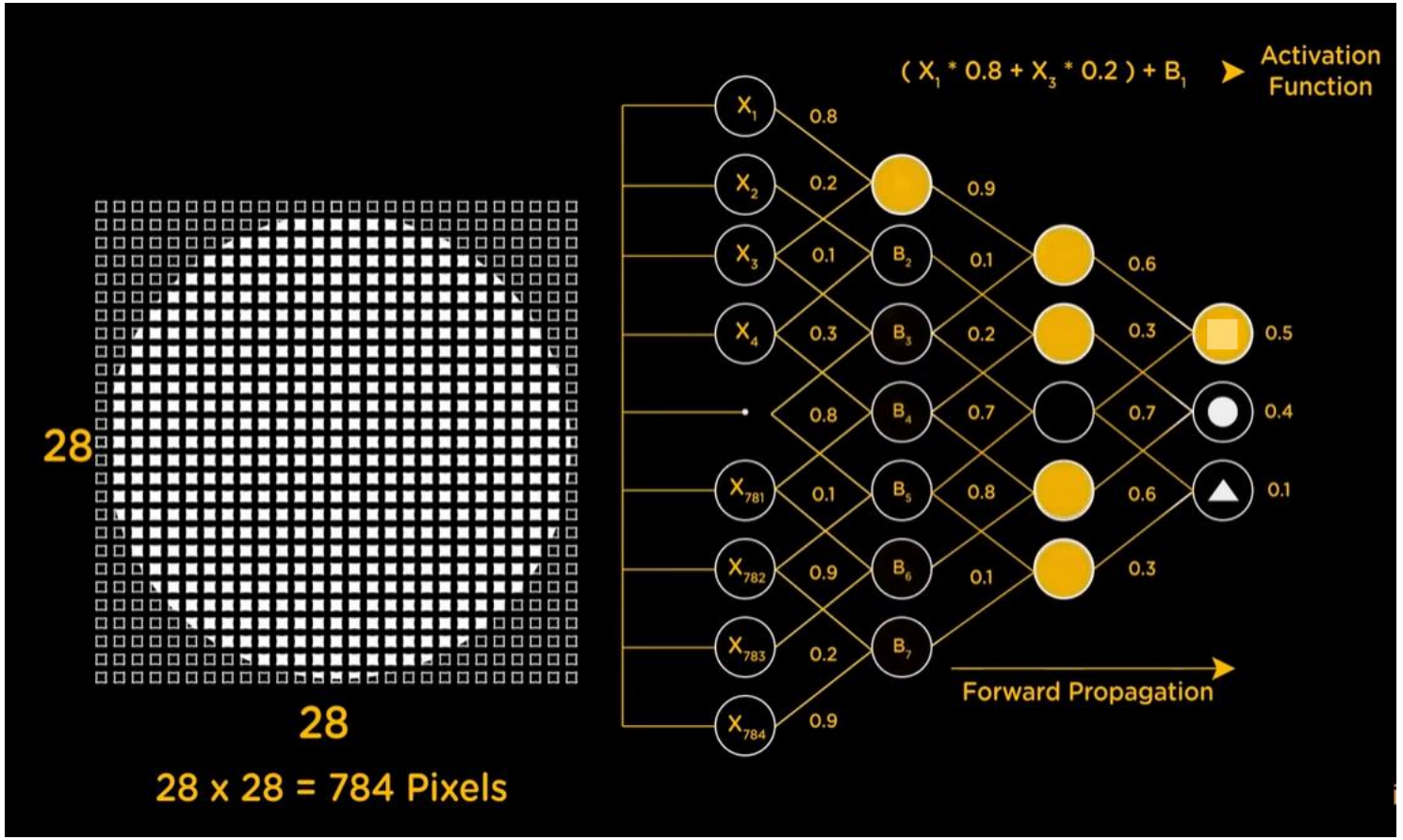
HOW DOES IT REALLY WORK?

- The same process will take place in the next layer



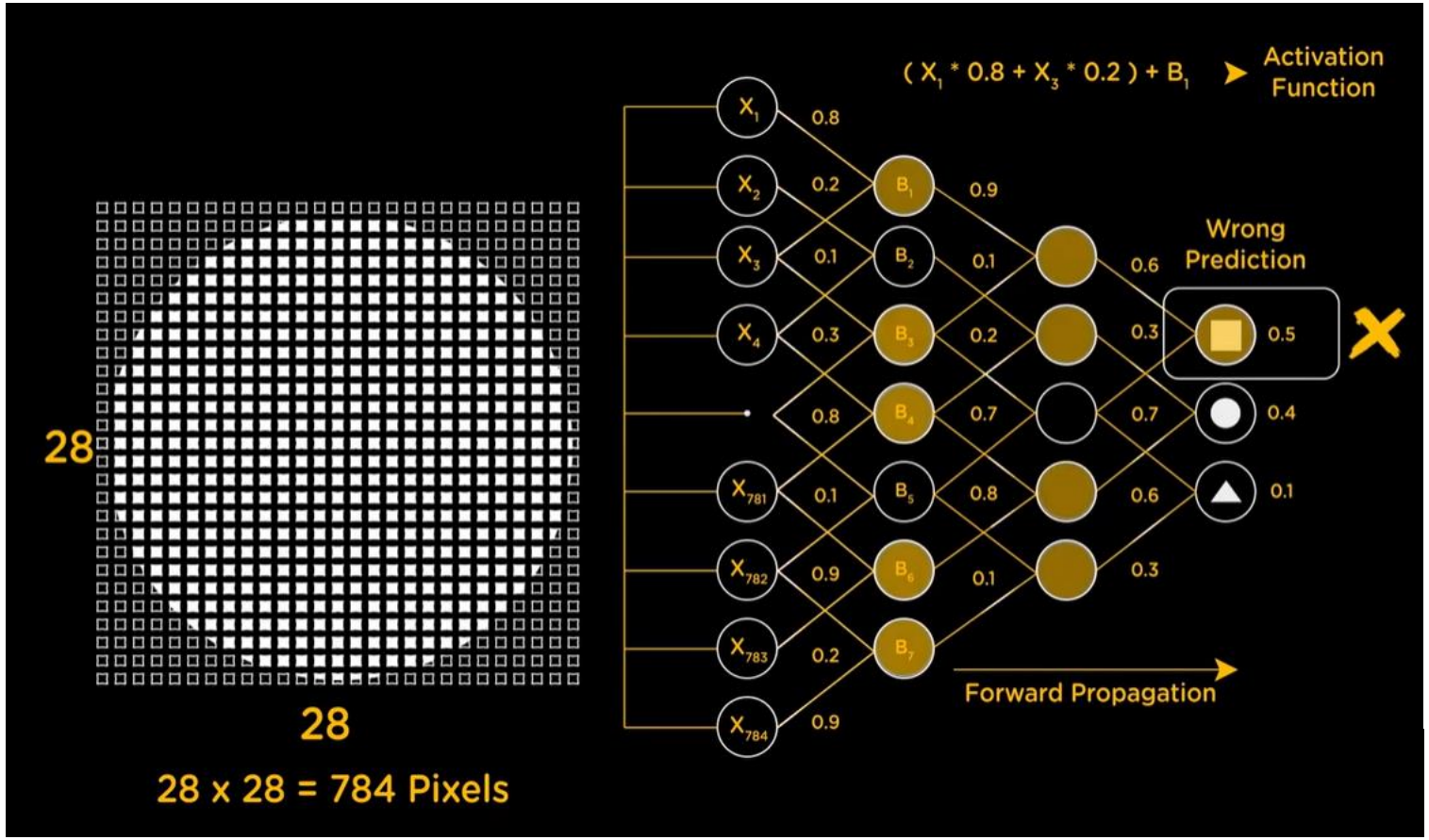
HOW DOES IT REALLY WORK?

- In the output layer, the nodes values are treated as probability, and we can see that the square has the highest probability
- The Neural Network predicted the square to be the class



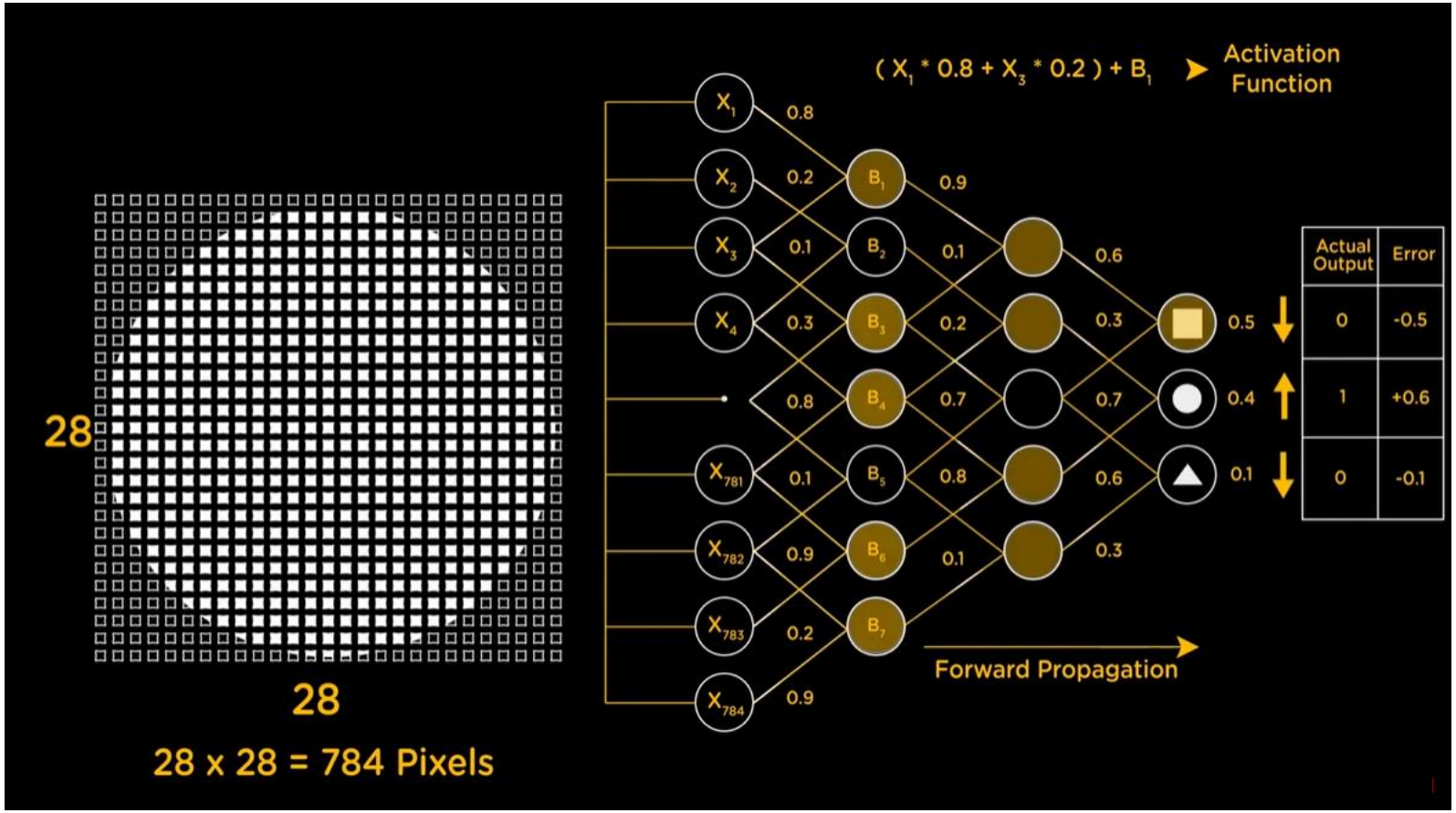
HOW DOES IT REALLY WORK?

- Obviously, this is a wrong prediction



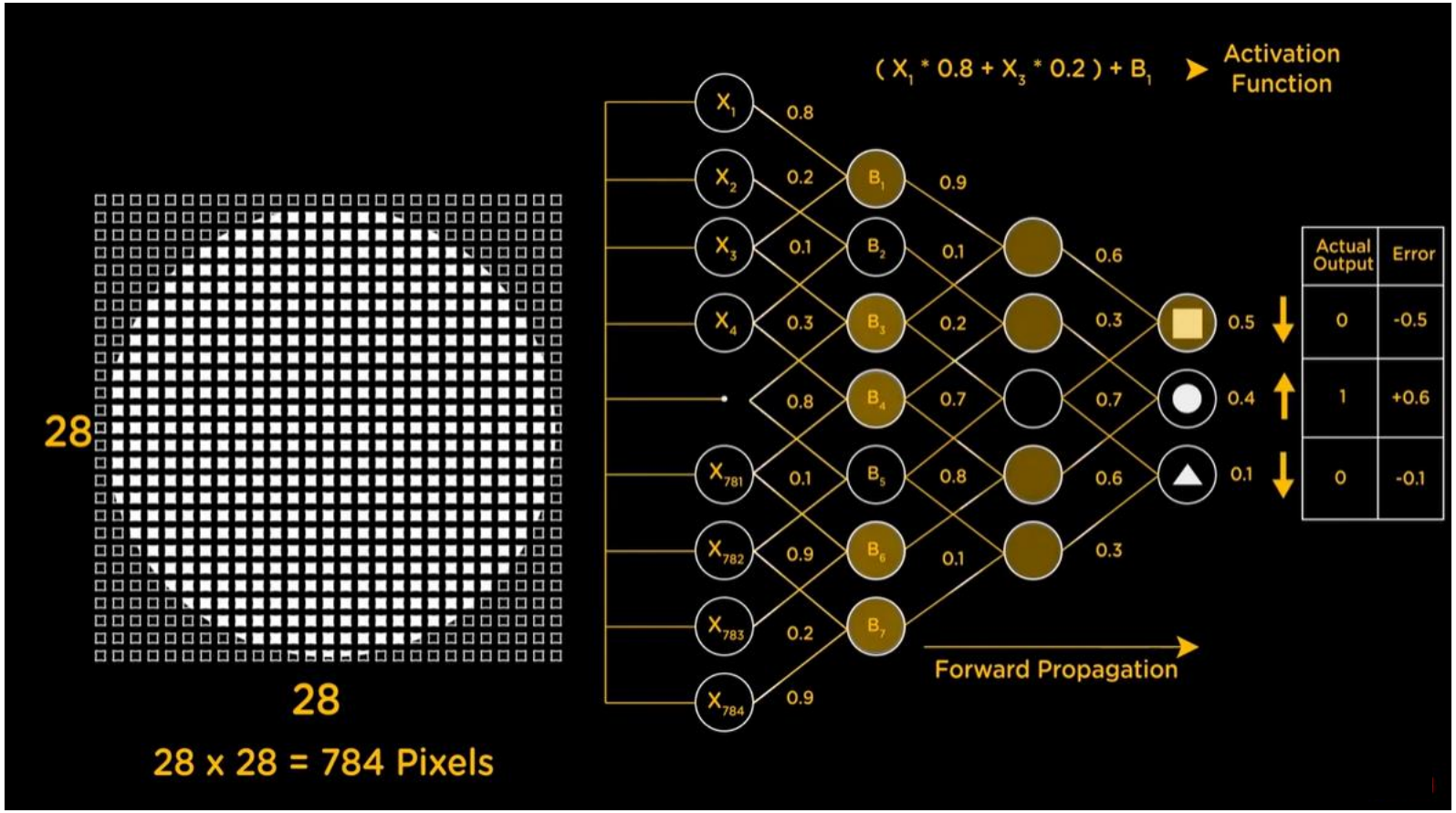
HOW DOES IT REALLY WORK?

- The network needs to be trained, and the predicted outputs must be compared to the actual output to calculate the error
- The arrows indicate if the errors are higher or lower than they should be



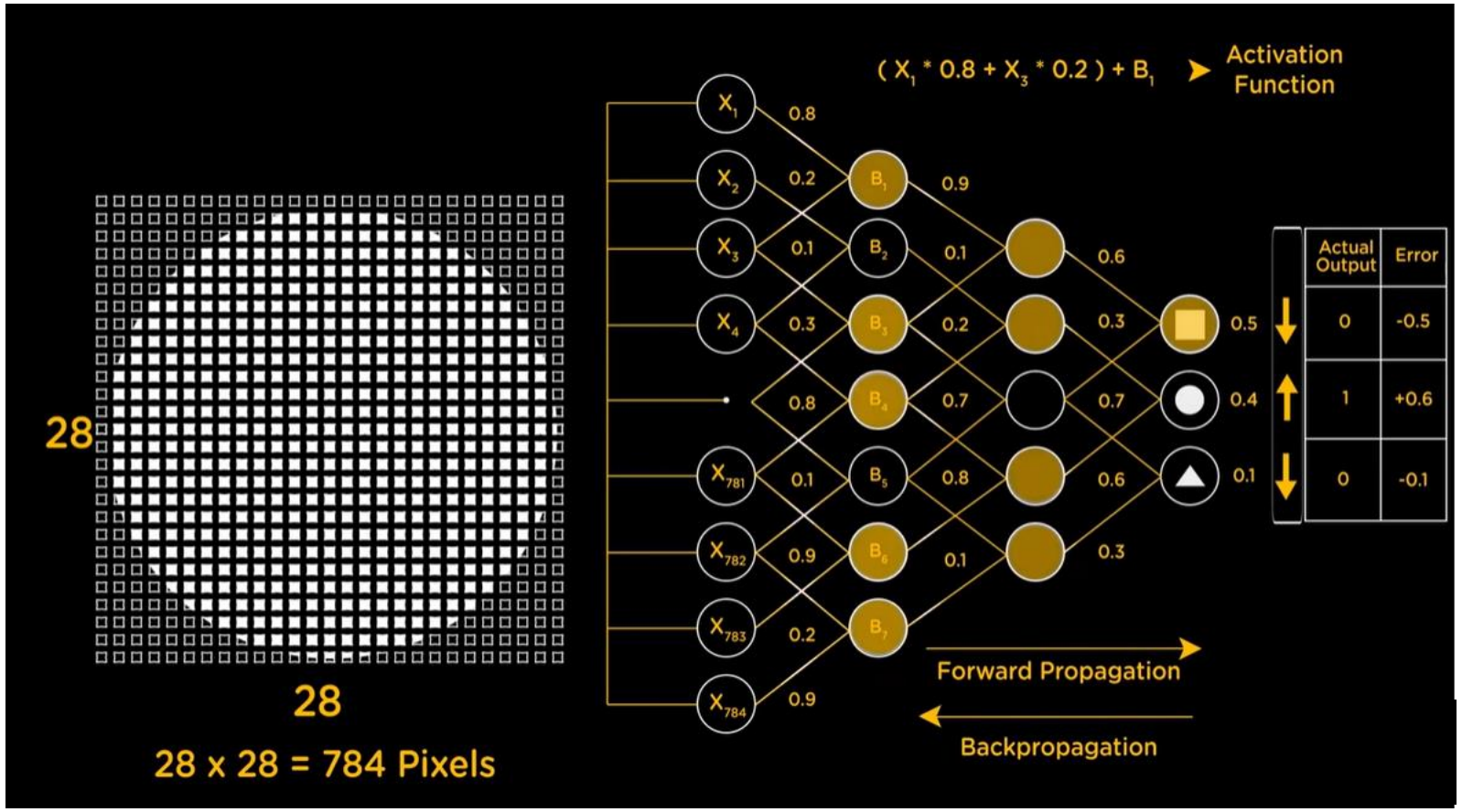
HOW DOES IT REALLY WORK?

- The calculated error give an indication of direction and magnitude of change needed to improve our network
- Remember that our main goal is to reduce the error and train the network to make the right prediction



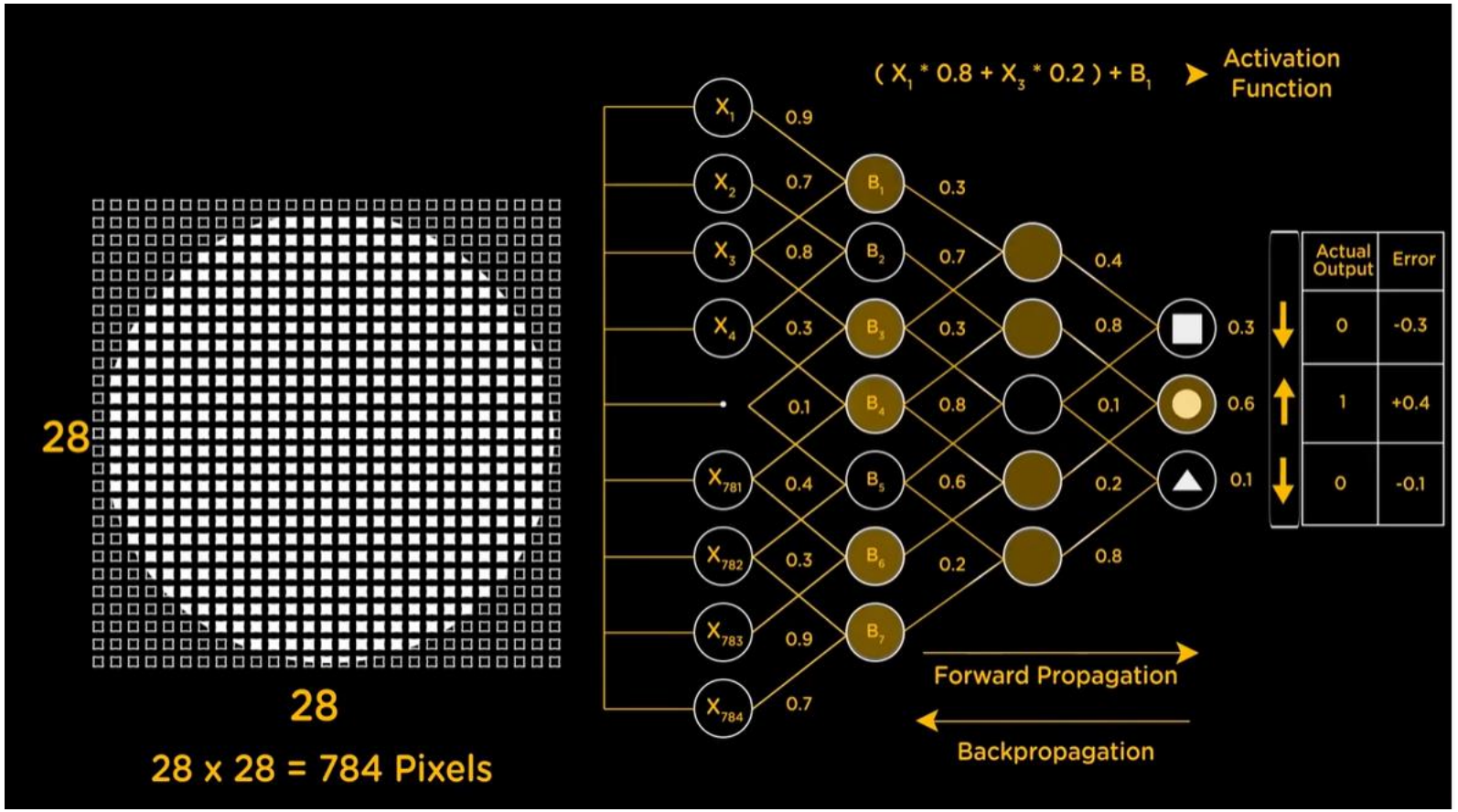
HOW DOES IT REALLY WORK?

- The error information is transferred backward to the network, known as Backpropagation



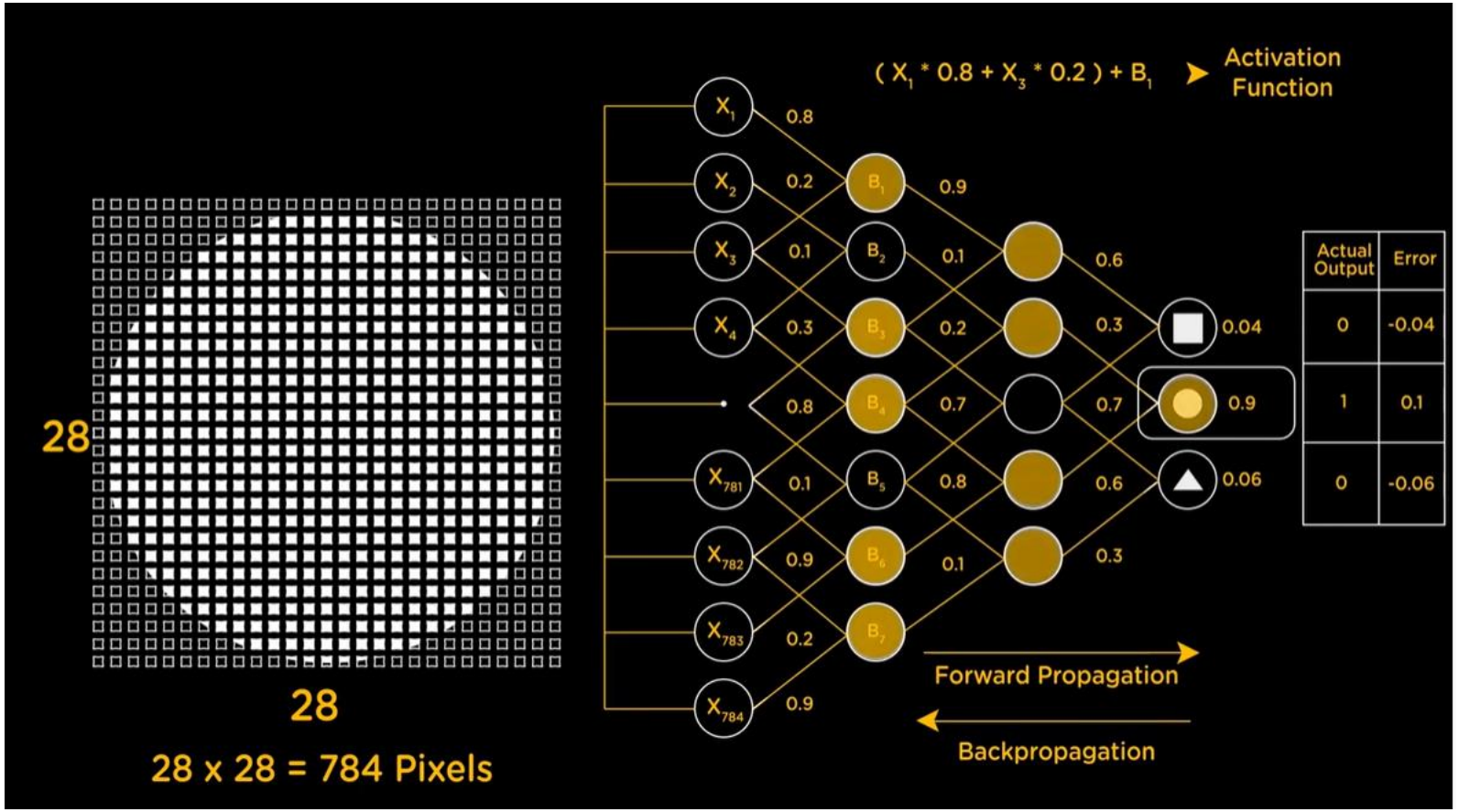
HOW DOES IT REALLY WORK?

- Based on those information, the weights are adjusted



HOW DOES IT REALLY WORK?

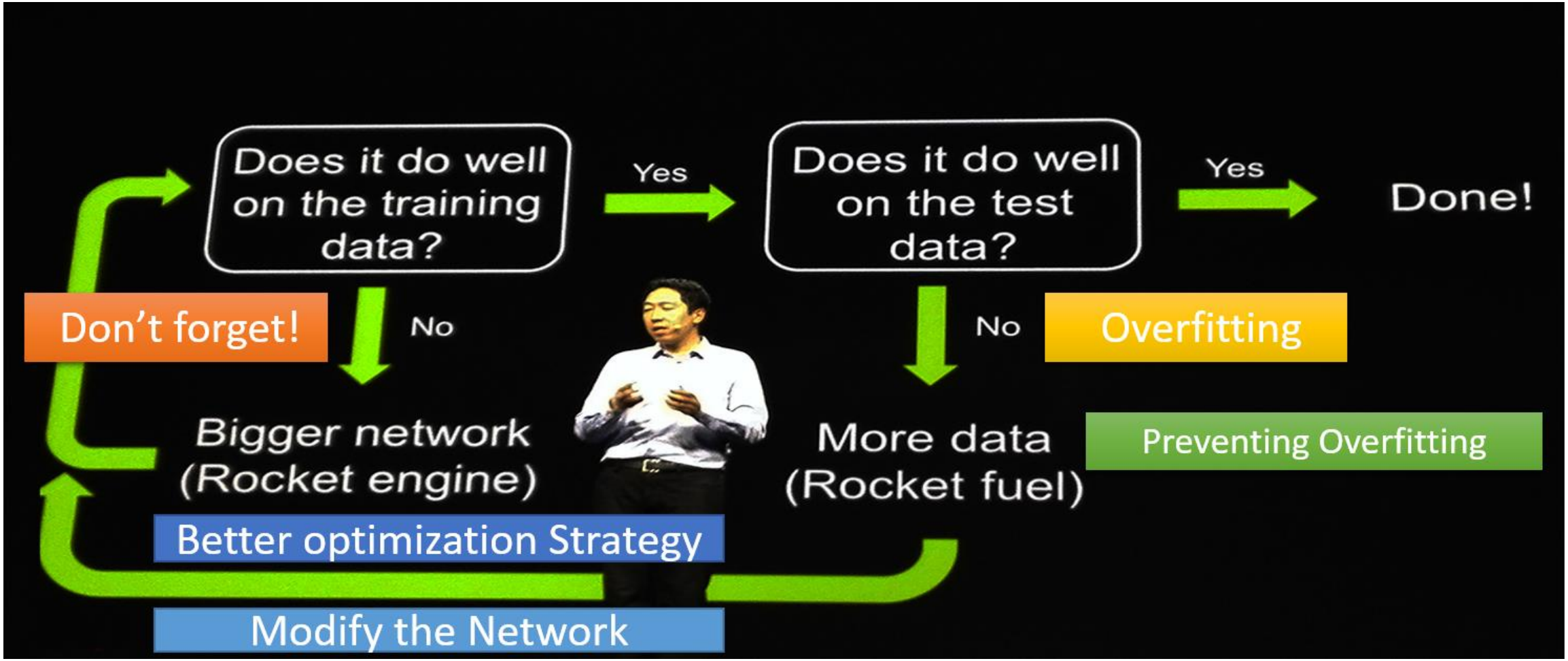
- The network keeps on training and adjusting until the network can predict the shapes correctly in most of the cases
- The training process can take hours or even months to train, based on the problem itself



BACKPROPAGATION

- Backpropagation is the essence of neural network training
- It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration)
- Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization
- Backpropagation in neural network is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks
- This method helps calculate the gradient of a loss function with respect to all the weights in the network

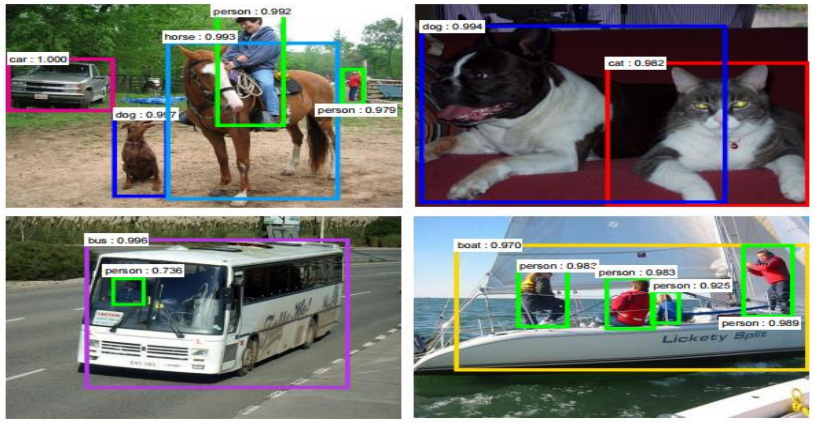
RECIPE FOR LEARNING



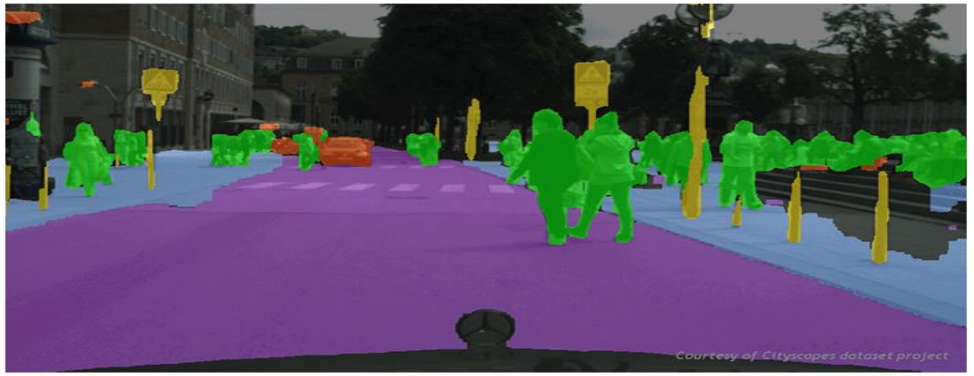
APPLICATIONS: DEEP LEARNING IN COMPUTER VISION

[Krizhevsky 2012]

[Ciresan et al. 2013]

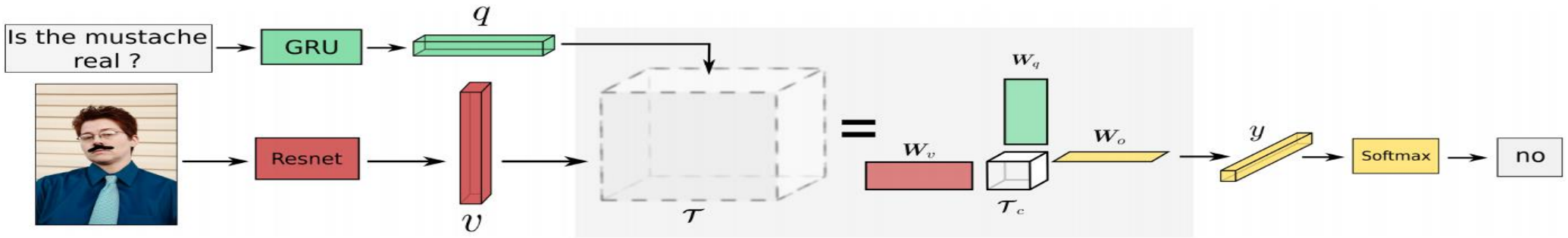


[Faster R-CNN - Ren 2015]



[NVIDIA dev blog]

APPLICATIONS: ACTION RECOGNITION



[VQA - Mutan 2017]

"man in black shirt is playing guitar."

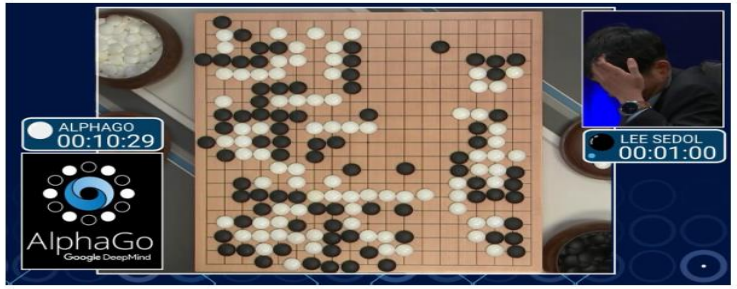
"construction worker in orange safety vest is working on road."

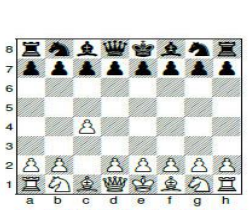
"two young girls are playing with lego toy."

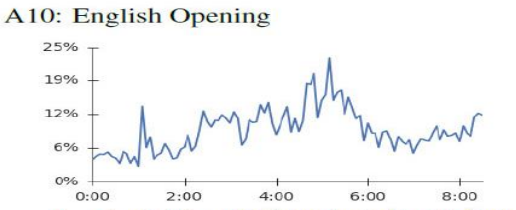
"boy is doing backflip on wakeboard."

[Karpathy 2015]

APPLICATIONS: DEEP LEARNING GENERATIVE MODELS



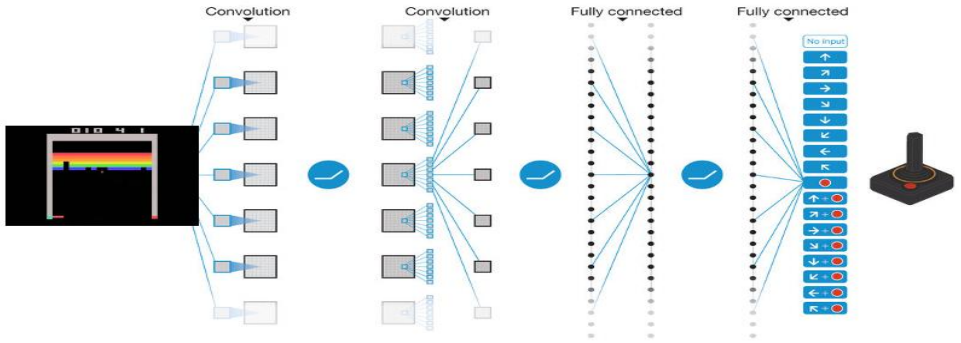




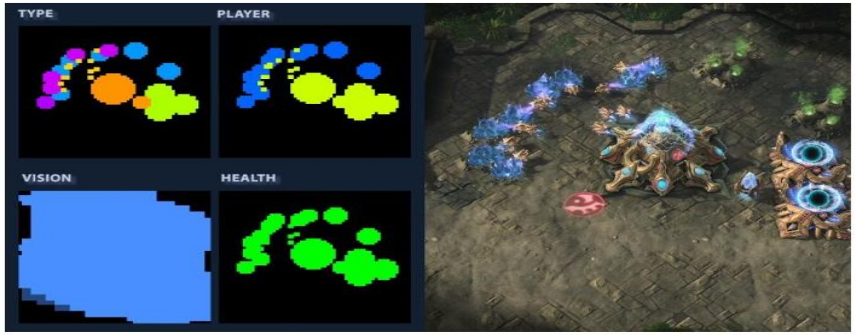
w 20/30/0, b 8/40/2

1...e5 g3 d5 cxd5 ♘f6 ♗g2 ♘xd5 ♘f3

[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]



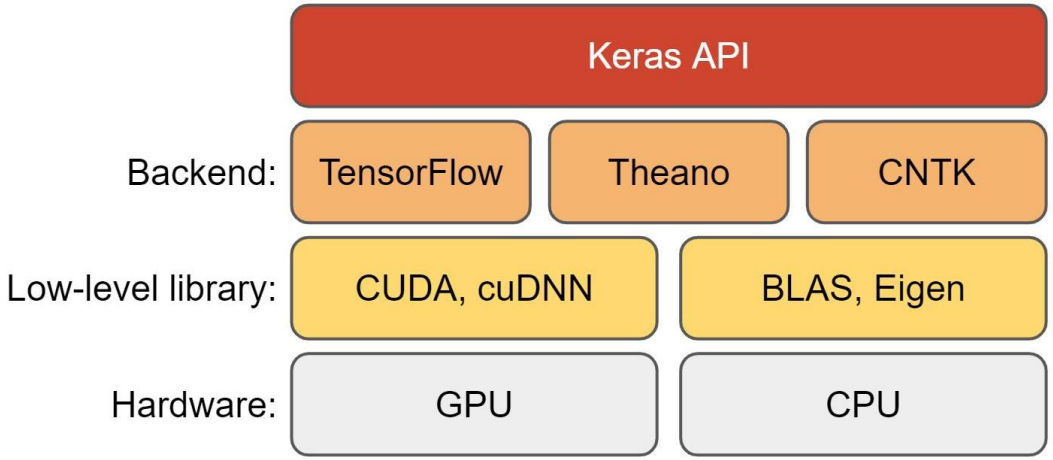
[Starcraft 2 for AI research]

INTRODUCTION TO Keras

- *"The framework was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research."* - Francois Chollet
- Francois Chollet, Google AI Developer/Researcher developed Keras in 2015 to facilitate his own research and experiments
- With the explosion of deep learning popularity, many developers, programmers, and machine learning practitioners flocked to Keras due to its easy-to-use API
- Keras is open-source framework written in Python
- Contains Neural Networks building blocks like layers, optimizers, and activation functions

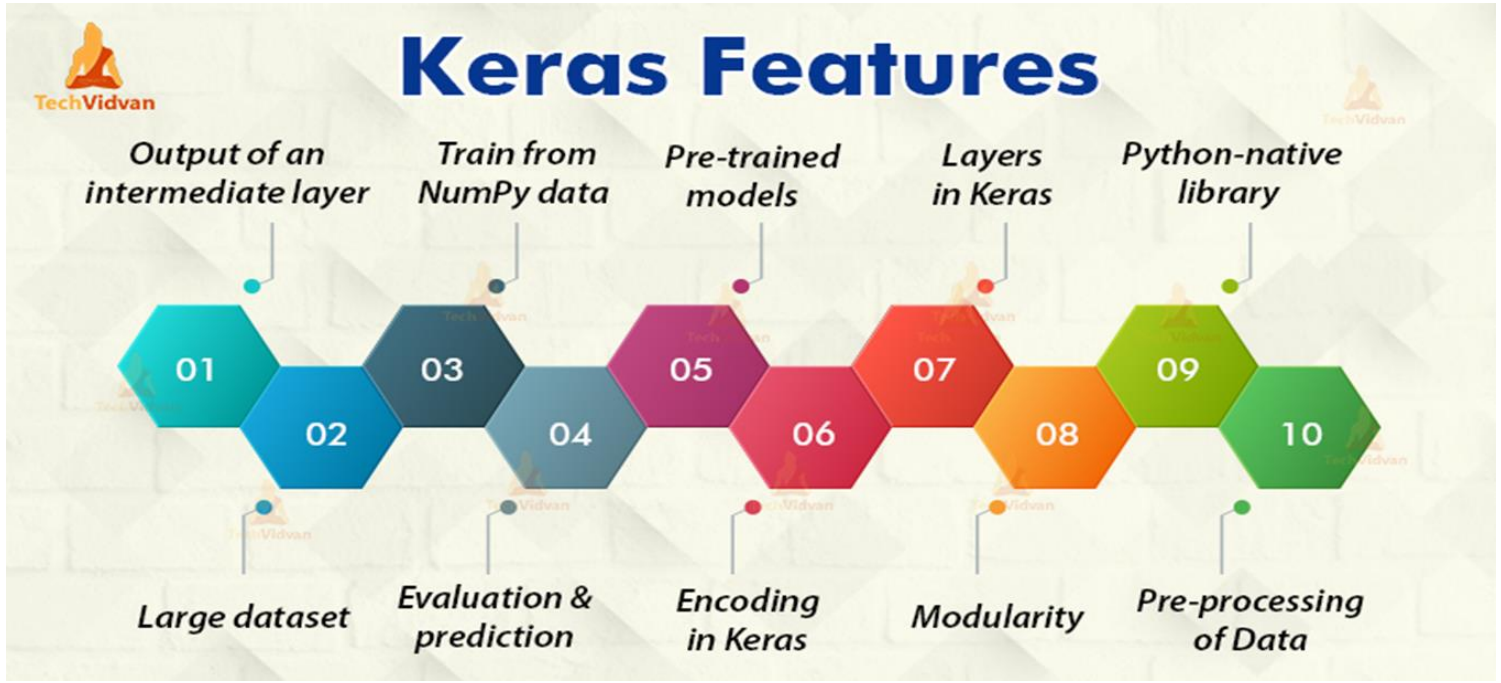
INTRODUCTION TO Keras

- Keras Features:
  - Keras is a high-level interface and uses Theano or TensorFlow for its backend
  - Runs smoothly in both CPU and GPU

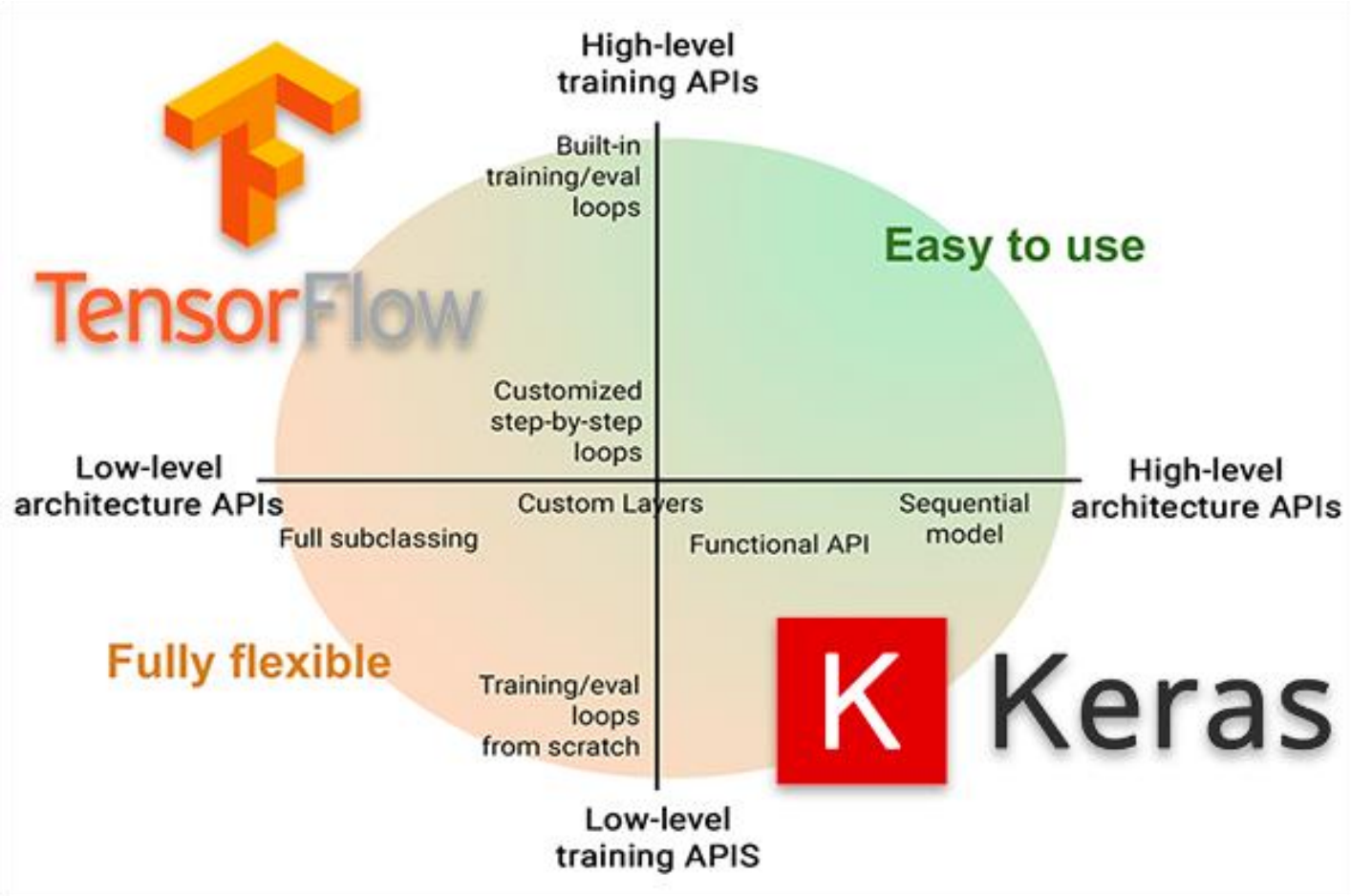


INTRODUCTION TO Keras

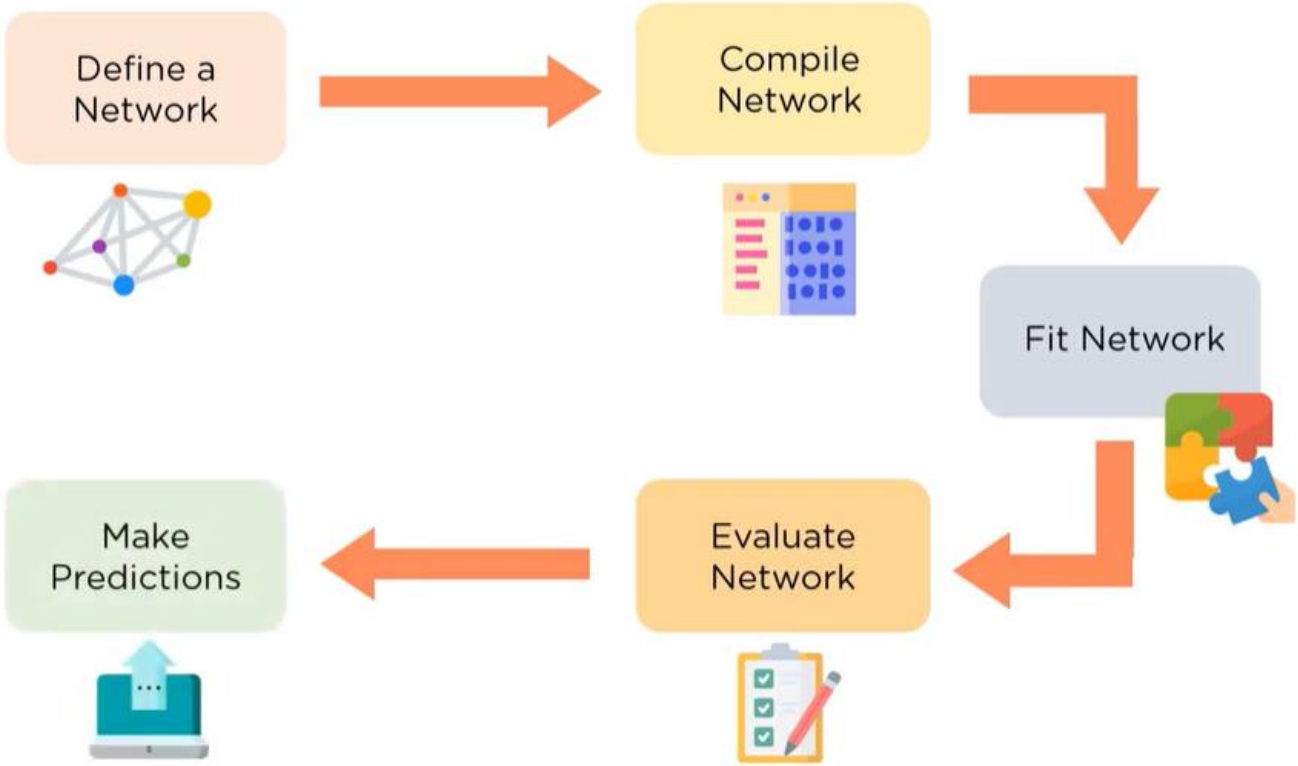
- Keras Features:
  - Rapid prototyping
    - Build neural network with minimal lines of code
    - Build simple or complex neural networks within a few minutes
  - Flexibility
    - Sometimes it is desired to define own metrics, layers, a cost function. Keras provide freedom to do so



KERAS vs. TENSORFLOW



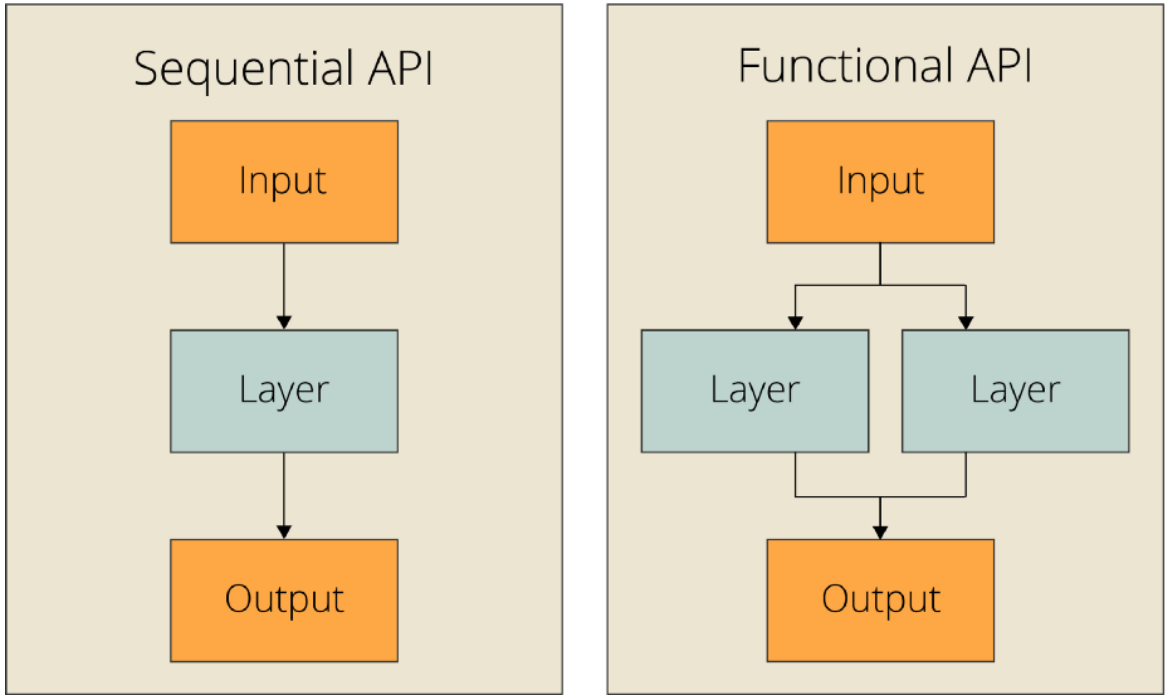
BUILDING A MODEL IN KERAS



BUILDING A MODEL IN KERAS

- We can build a Keras model in 2 ways:

1. The Sequential API:
   Best method when you are trying to build a simple model with a single input, output, and layer branch
2. The Functional API:
   The most popular method to build Keras models. It can do everything that the Sequential API can do. Also, it allows multiple inputs, multiple outputs, branching, and layer sharing



BUILDING A MODEL IN KERAS

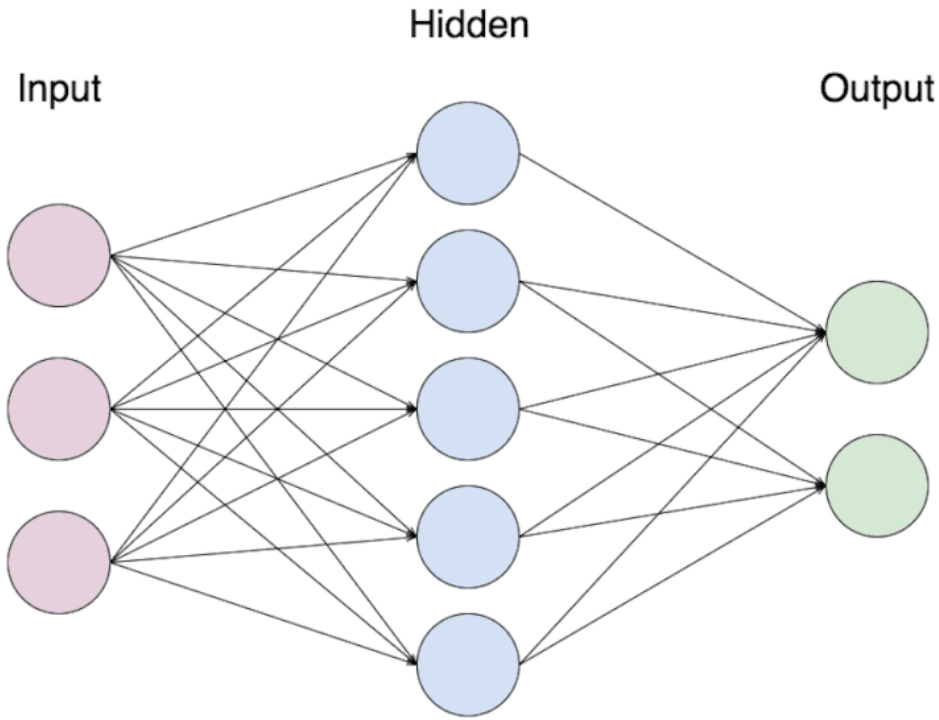
- Example on how to Define a Keras model:

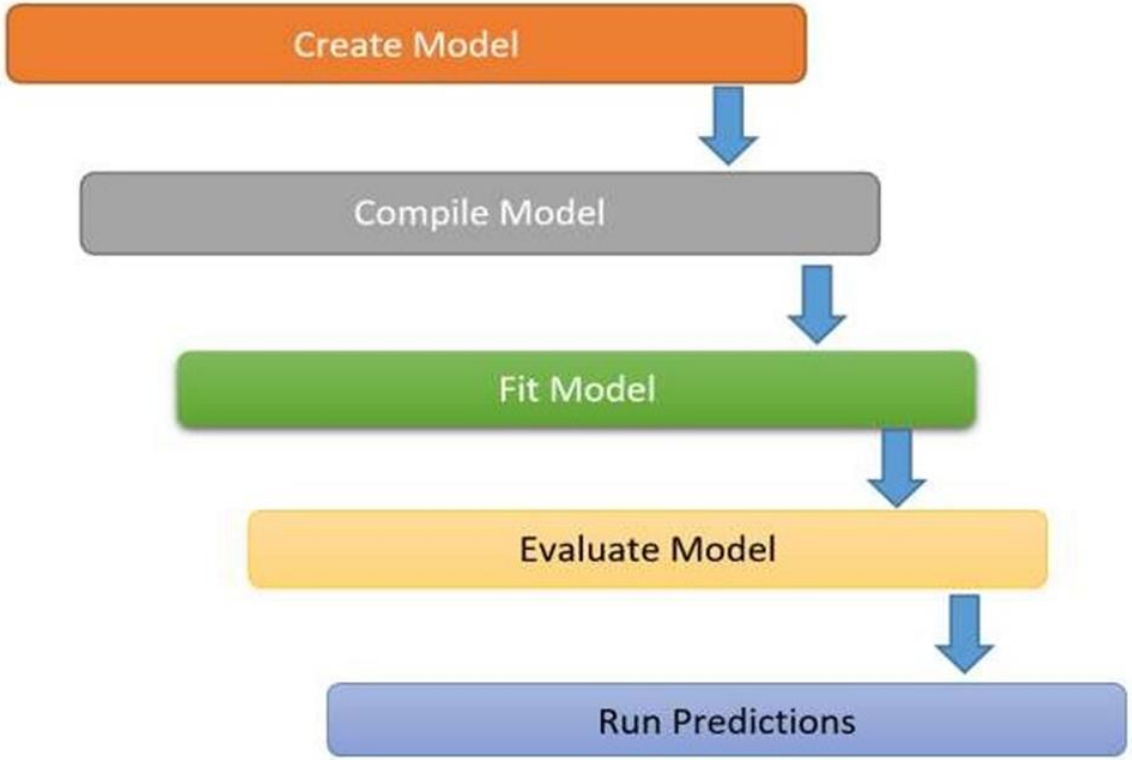
```
...
# define the keras model
model = Sequential()
model.add(Dense(12, input_shape=(8,), activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
...
```

- The model expects rows of data with 8 variables (the input_shape=(8,) argument)
- The **first hidden layer** has 12 nodes and uses the relu activation function.
- The **second hidden layer** has 8 nodes and uses the relu activation function
- The **output layer** has one node and uses the sigmoid activation function

KERAS LAYERS

- Dense Layer: a fully connected Neural Network layer



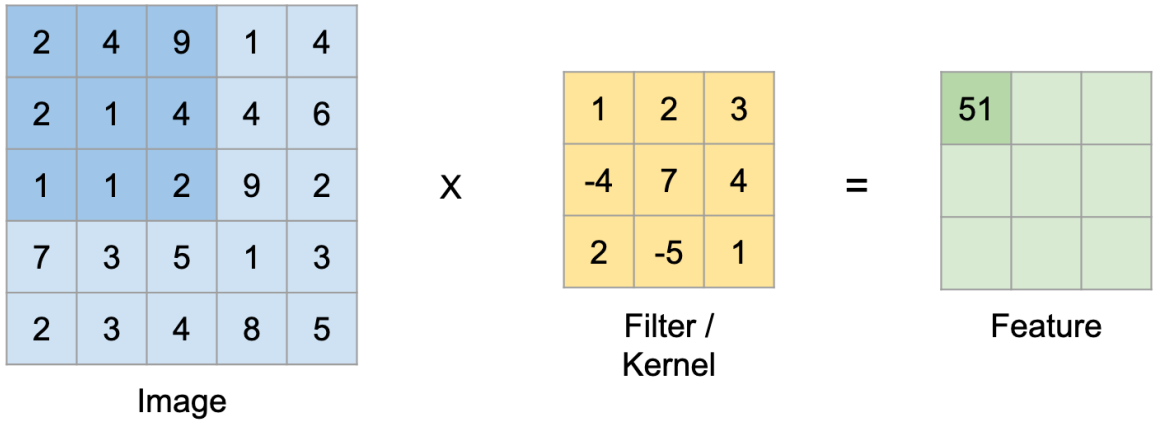


KERAS LAYERS

- Convolutional layer:

  Mostly used in computer vision

  It extract features from the input image by applying convolution filters to create feature maps



KERAS LAYERS

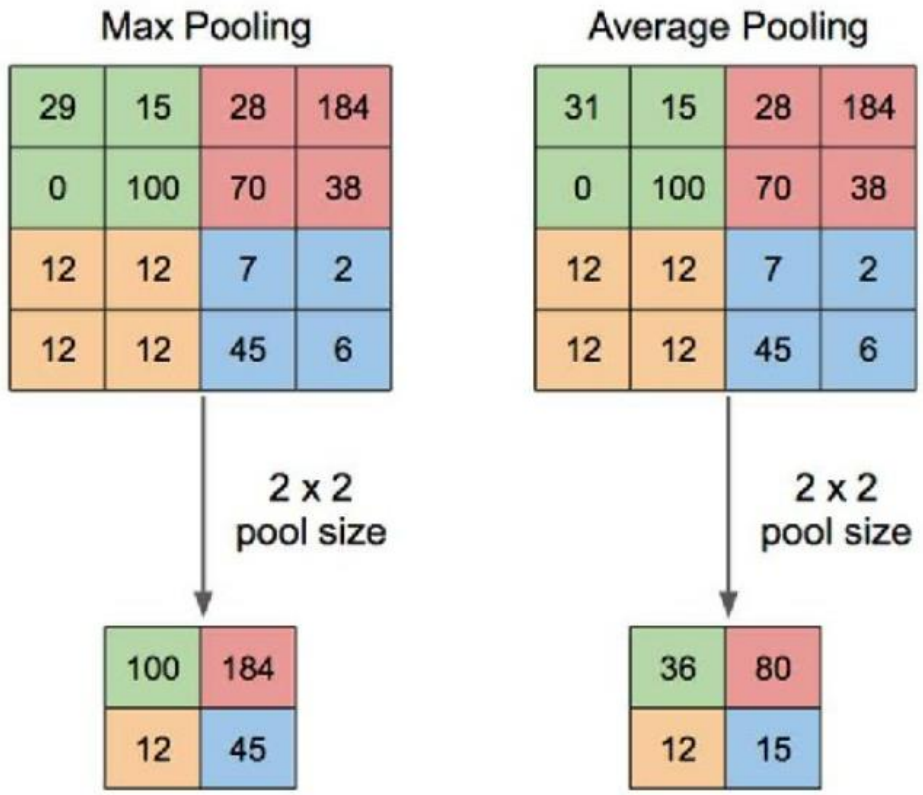
- Pooling layer:

  Also called as subsampling or down-sampling layer

  Pooling reduces the dimensionality of feature map

  Retains the most important information

  There are many types of pooling layers such as MaxPooling and AveragePooling



BUILDING A MODEL IN KERAS

- **Compile()** is used to configure the model, and it uses the following parameters:

1. Optimizer:

   This optimizer to use in back-propagation algorithm

   SGD, Adadelta, Adagrad, Adam, Nadam optimizer and many others

2. Loss:

   It track losses or the drift from the function during training the model

   For regression: mean squared error, mean absolute error etc.

   For classification: Categorical cross-entropy, binary cross entropy

BUILDING A MODEL IN KERAS

3. Metrics:

   It specifies the list of metrics that model evaluate during training and testing

   The commonly used metric is the accuracy metric

```
...
# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
...
```

BUILDING A MODEL IN KERAS

- We can train or fit our model on our loaded data by calling the **Fit()**
- Training occurs over epochs and each epoch is split into batches

1. Epoch:

   One pass through all of the rows in the training dataset

2. Batch:

   One or more samples considered by the model within an epoch before weights are updated

BUILDING A MODEL IN KERAS

- One epoch is comprised of one or more batches, based on the chosen batch size and the model is fit for many epochs
- The training process will run for a fixed number of iterations through the dataset called epochs, that we must specify using the epochs argument
- We must also set the number of dataset rows that are considered before the model weights are updated within each epoch, called the batch size and set using the batch_size argument

```
...
# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)
...
```

KERAS DATASETS

- Keras contains various datasets that are used to build neural networks. The datasets are described below:

1. Boston House Pricing dataset:
   - It contains 13 attributes of houses of Boston suburbs in the late 1970s
2. CIFAR10:
   - This dataset contains 50,000 32×32 color images
   - Images are labelled over 10 categories
   - 10,000 test images
3. CIFAR100:
   - Same as CIFAR10 but it has 100 categories

KERAS DATASETS

4. MNIST:
   - This dataset contains 60,000 28×28 grayscale images of 10 digits
   - includes 10,000 test images
5. Fashion-MNIST:
   - contains 60,000 28×28 grayscale images of 10 categories
6. IMDB movie reviews data:
   - Dataset of 25,000 movies reviews from IMDB labeled by sentiment (positive/negative).
7. Reuters newswire topics classification:
   - Dataset of 11,228 newswires from Reuters, labeled over 46 topics

BUILDING A MODEL IN KERAS

Example #1:
Designing a simple Neural Network for handwritten digit recognition problem (MNIST digit data)

Example #2:
Designing a simple Neural Network for image classification problem (Fashion-MNIST)