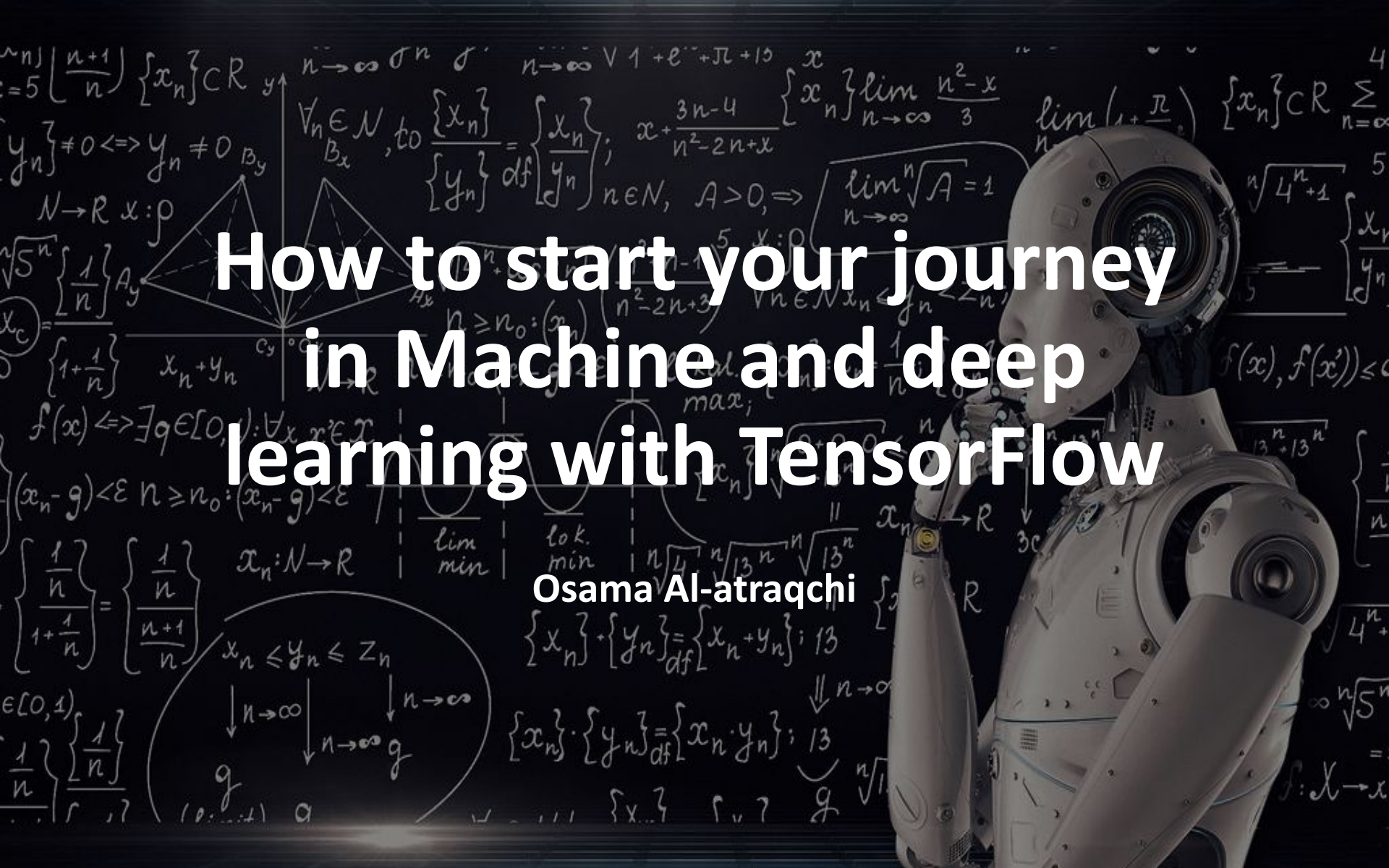
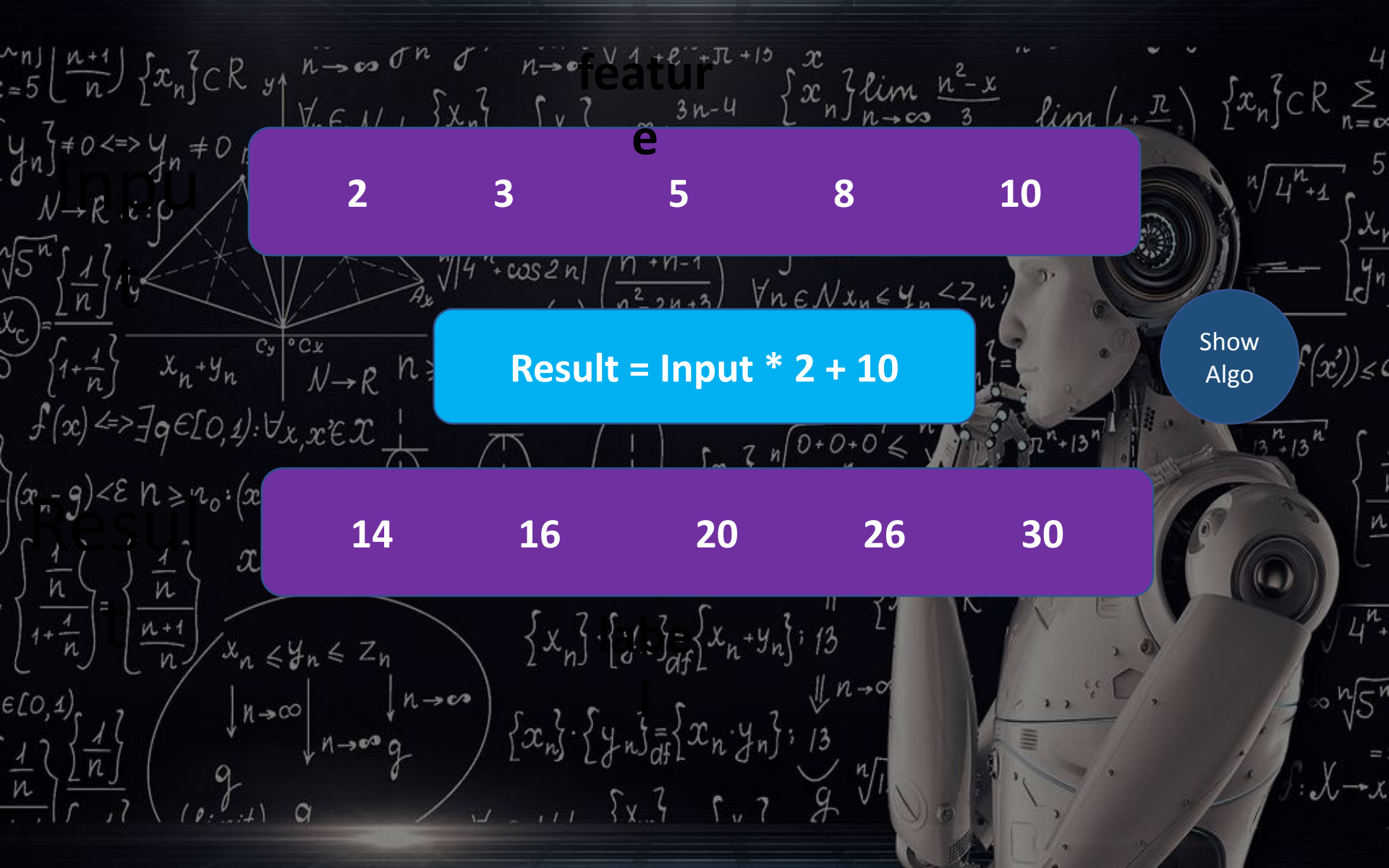
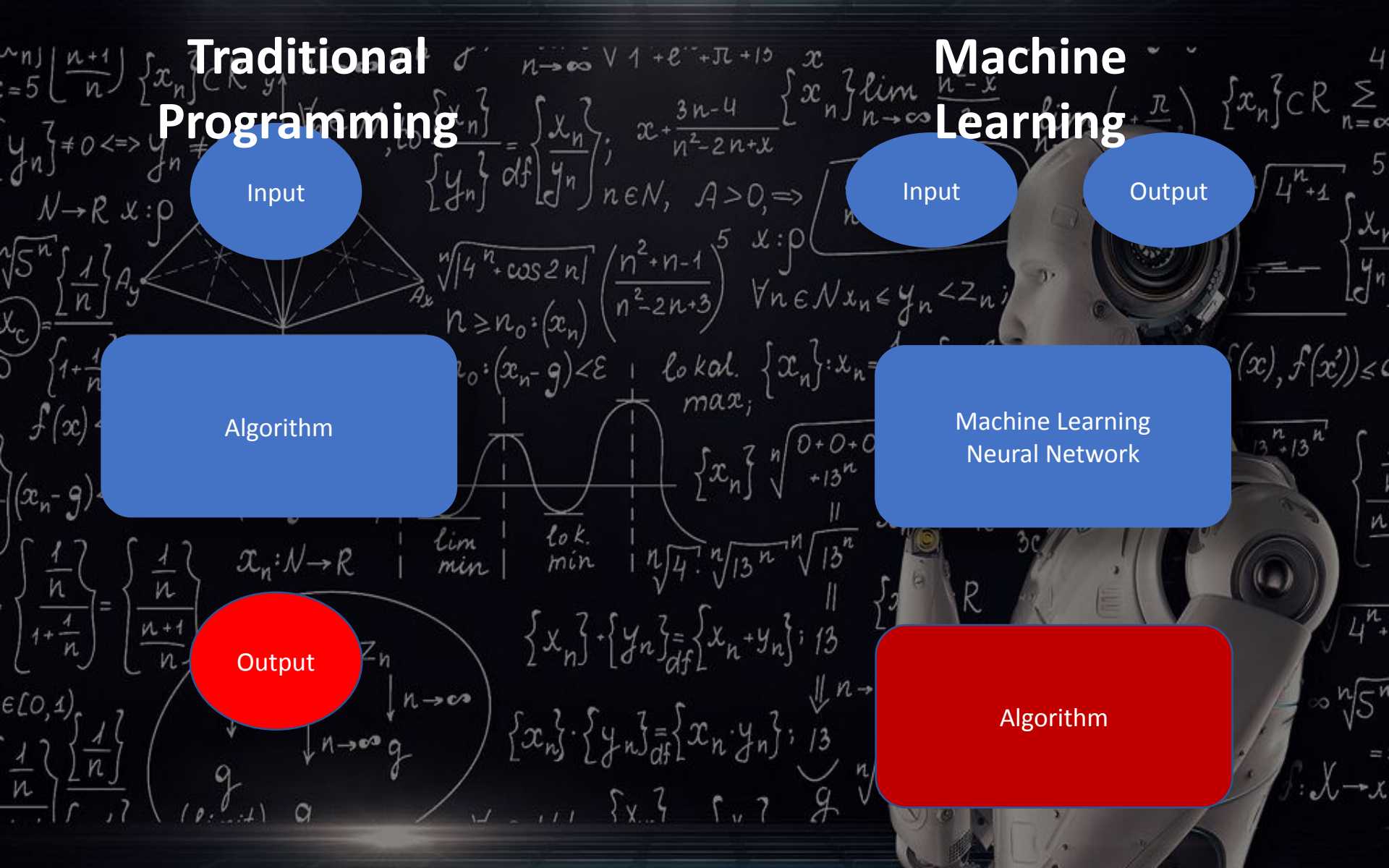
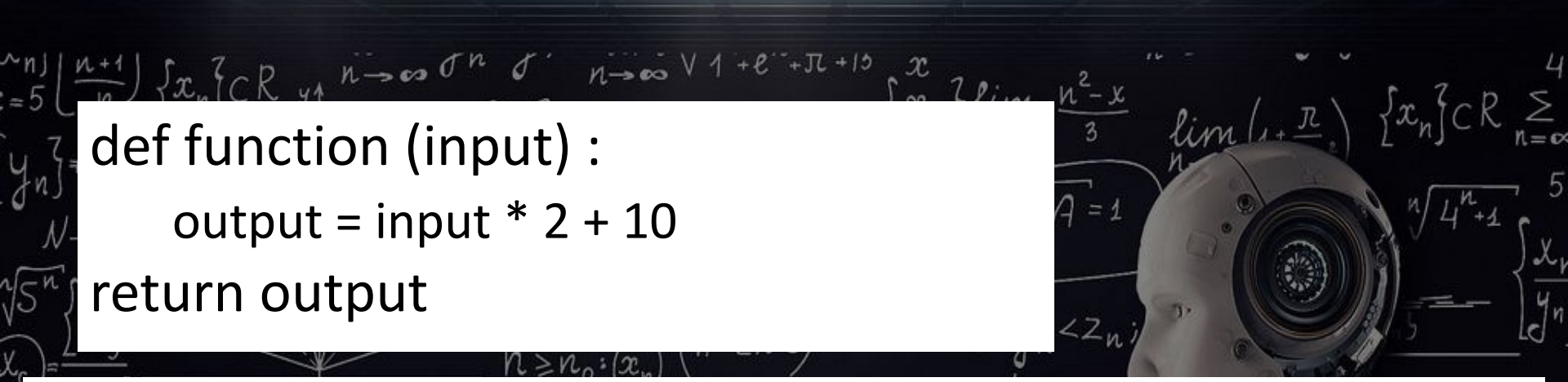


How to start your journey in Machine and deep learning with TensorFlow

Osama Al-atraqchi


featur
e
Inpu
t
2
3
5
8
10
Result = Input * 2 + 10
Show
Algo
Resul
t
14
16
20
26
30
Labe
l


Traditional Programming
Input
Algorithm
Output
Machine Learning
Input
Output
Machine Learning
Neural Network
Algorithm

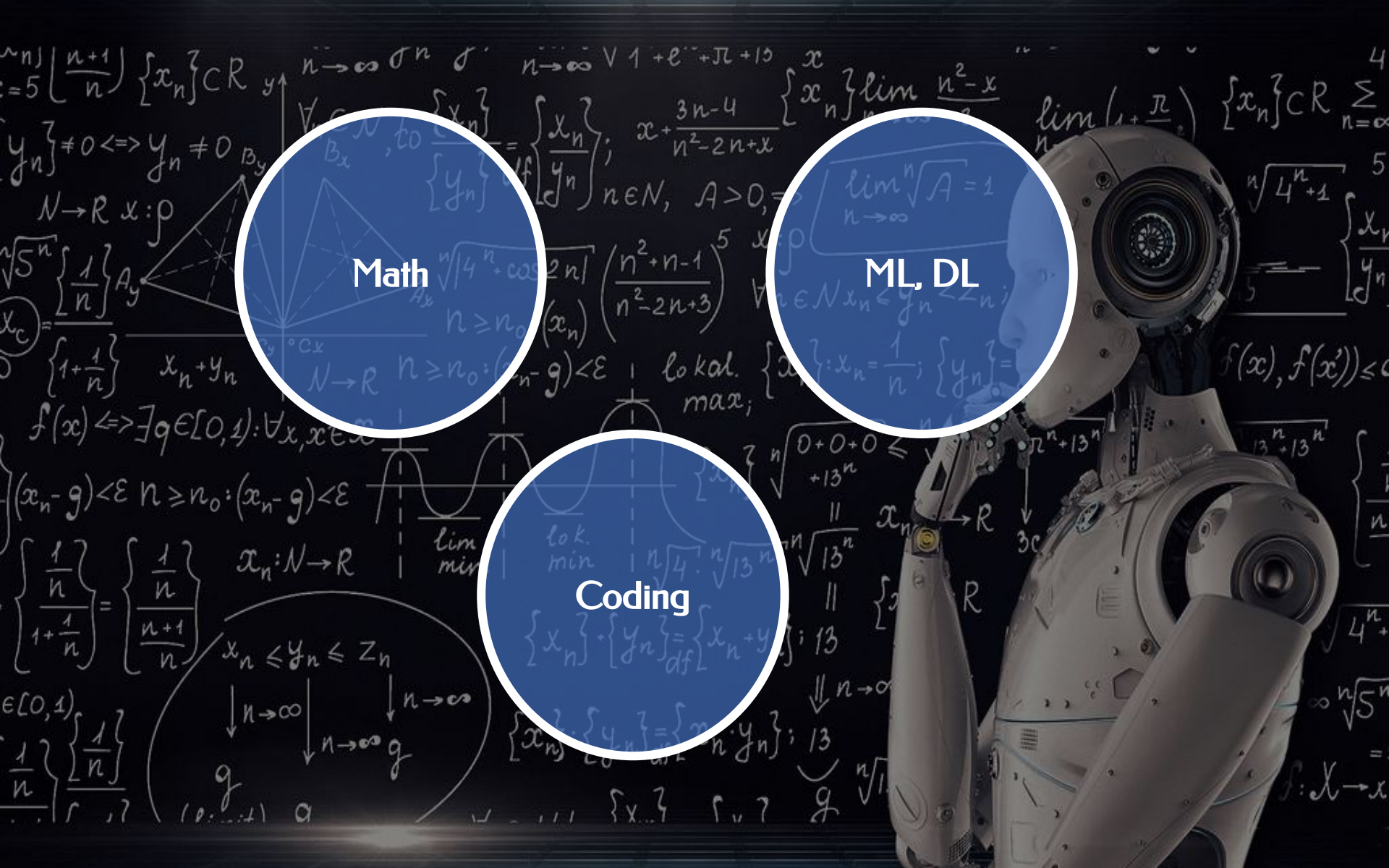


```
def function (input) :
    output = input * 2 + 10
return output
```

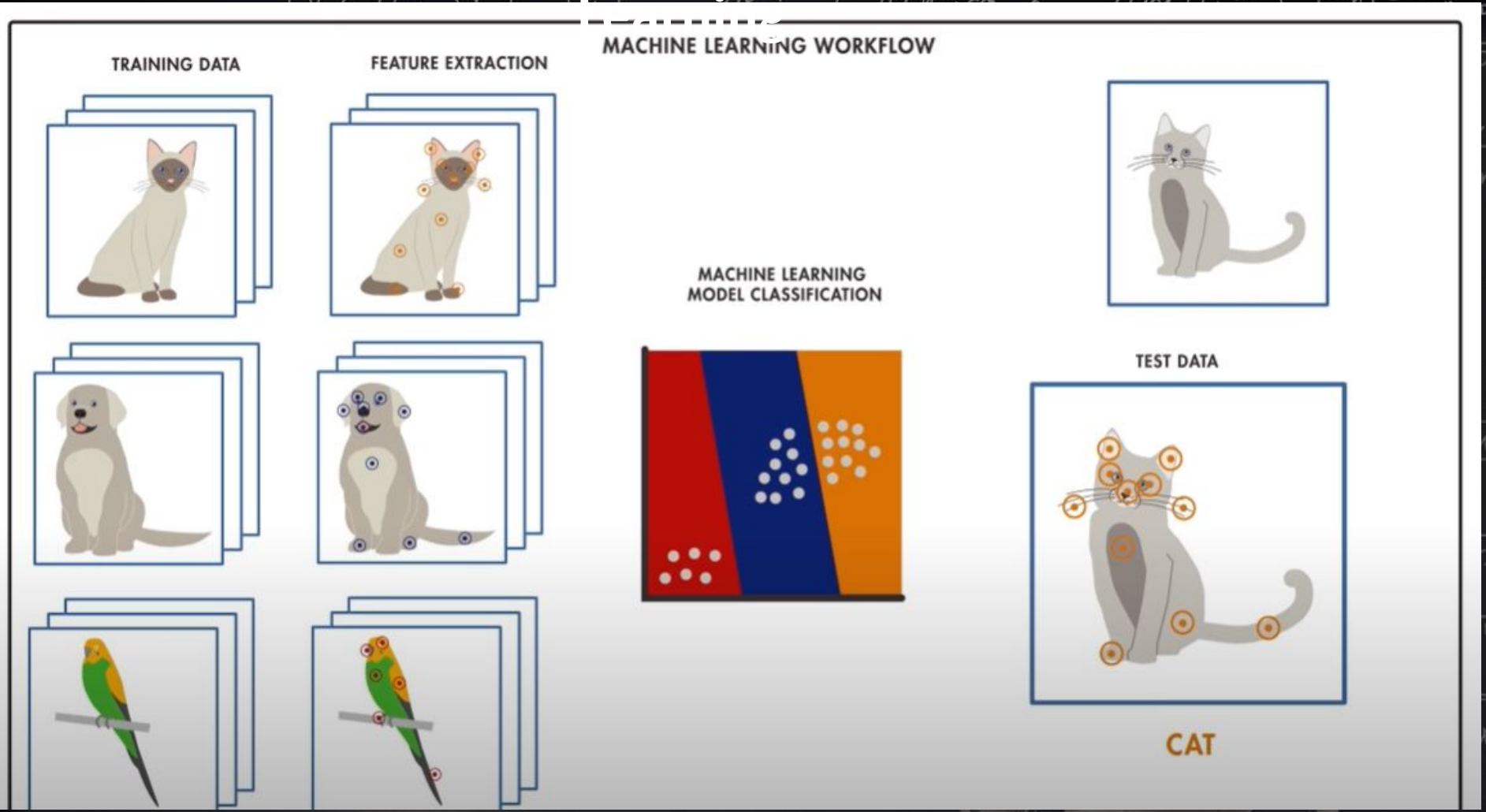
def function (input) :

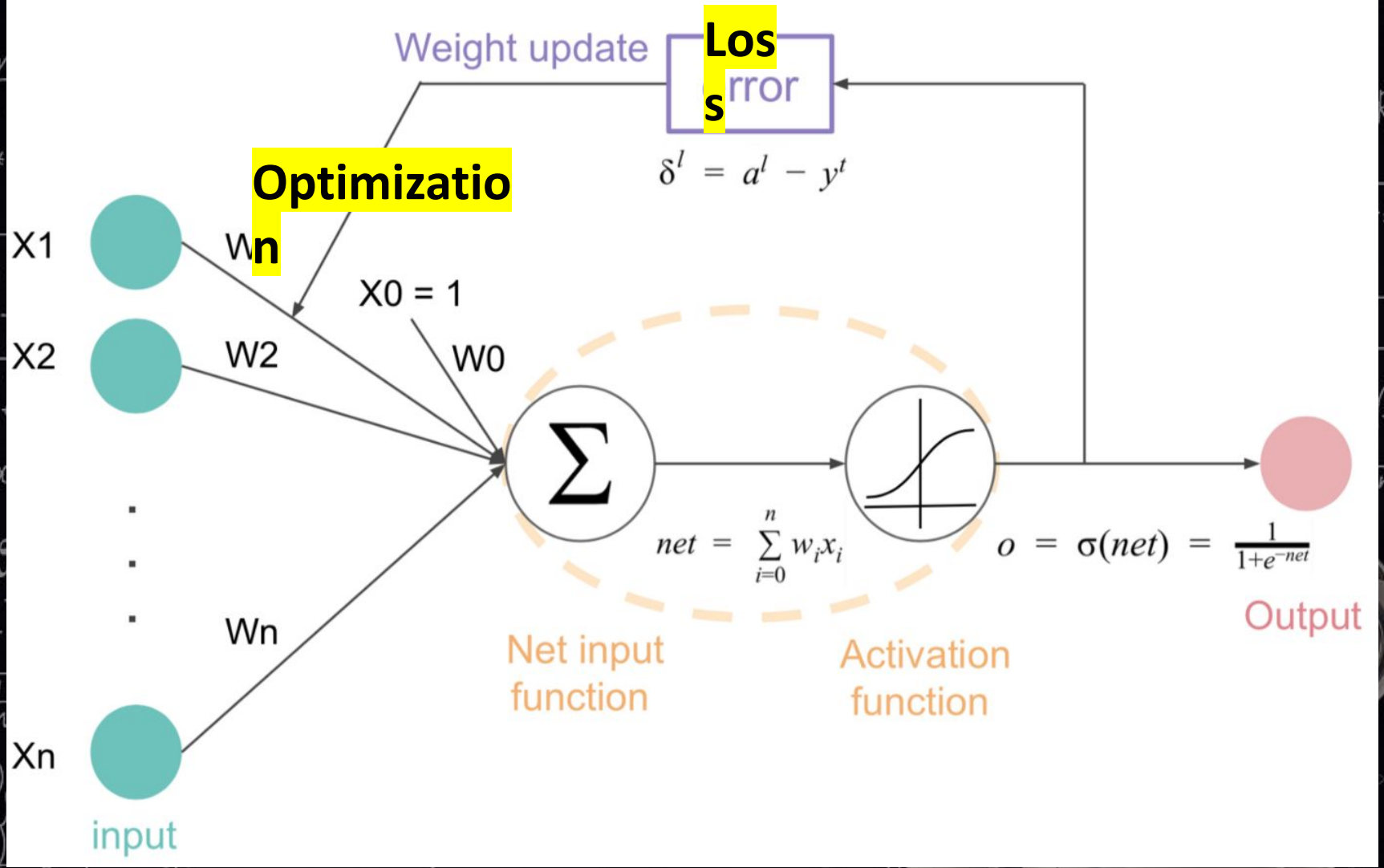
```
input= np.array([2, 3, 8, 5, 10],  dtype=int)
output = np.array([14, 16, 26, 20, 30],  dtype=int)

model = tf.keras.Sequential([
tf.keras.layers.Dense(units=1, input_shape=[1])
])
model.compile(loss='mean_squared_error', optimizer=tf.keras.optimizers.Adam(0.
1))
training = model.fit(input, output, epochs=500, verbose=False)
predict = model.predict([input])
```

Return predict


Math
ML, DL
Coding

Machine Learning VS Deep Learning




Weight update
Los
s
rror
$\delta^l = a^l - y^t$
Optimizatio
n
X1
X2
Xn
W
W2
Wn
X0 = 1
W0
Σ
$net = \sum_{i=0}^{n} w_i x_i$
$o = \sigma(net) = \frac{1}{1+e^{-net}}$
Output
Net input
function
Activation
function
input

Activation Functions Types

Linear

$F(x) = x *$

Binary step

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

Sigmoid / Logistic

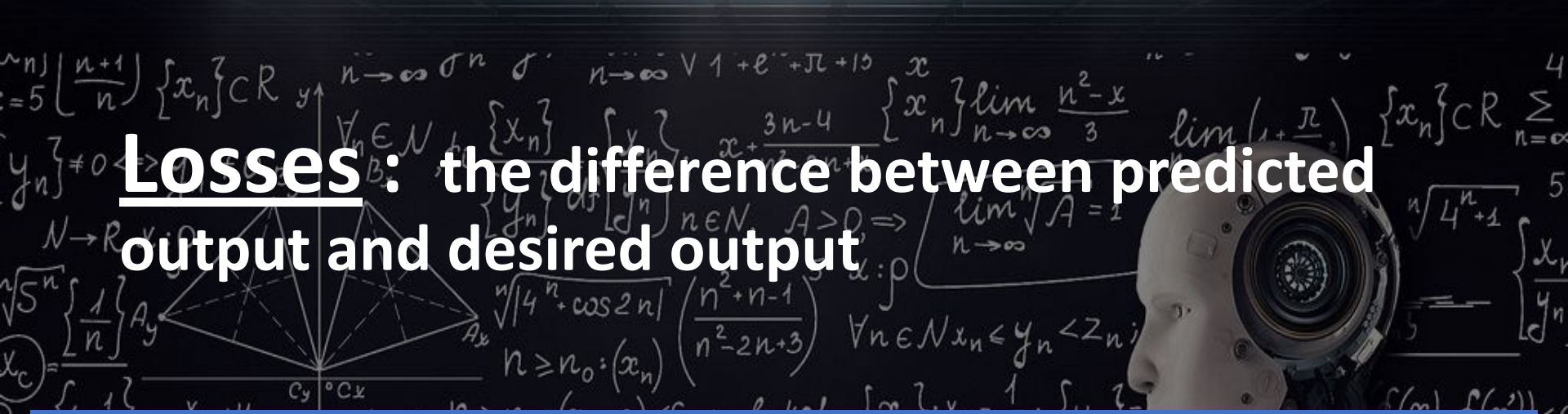
$$f(x) = \frac{1}{1 + e^{-x}}$$

Softmax

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Parametric ReLU

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



Losses : the difference between predicted output and desired output

- The mean squared error (MSE)

MSE formula = $(1/n) * \Sigma(\text{actual} - \text{forecast})^2$

- Cross-entropy:

$$H(P^* | P) = -\sum_i P^*(i) \log P(i)$$

Optimizers

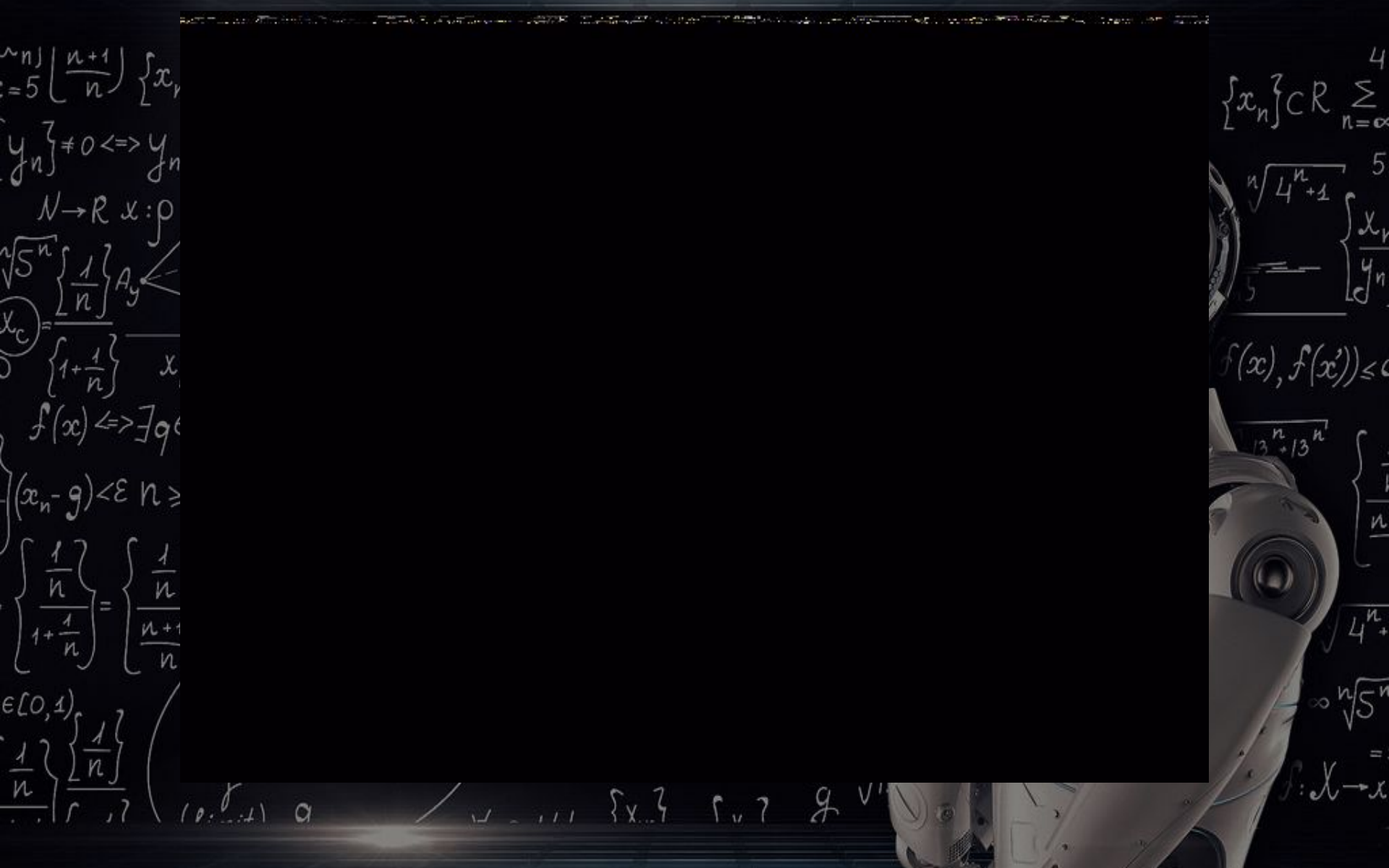
- Optimizers are mathematical functions which are dependent on model's learnable parameters i.e Weights & Biases. Optimizers help to know how to change weights to reduce the losses.
- Gradient Descent:

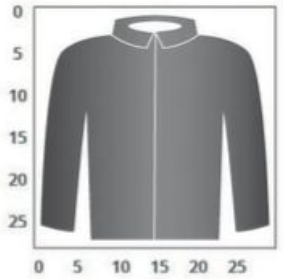
$$W_{new} = W_{old} - \alpha * \frac{\partial(Loss)}{\partial(W_{old})}$$

- Adam(Adaptive Moment Estimation) : the most popular and famous gradient descent optimization algorithms.

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{S_{dw_t} - \varepsilon}} * V_{dw_t}$$

$$b_t = b_{t-1} - \frac{\eta}{\sqrt{S_{db_t} - \varepsilon}} * V_{db_t}$$

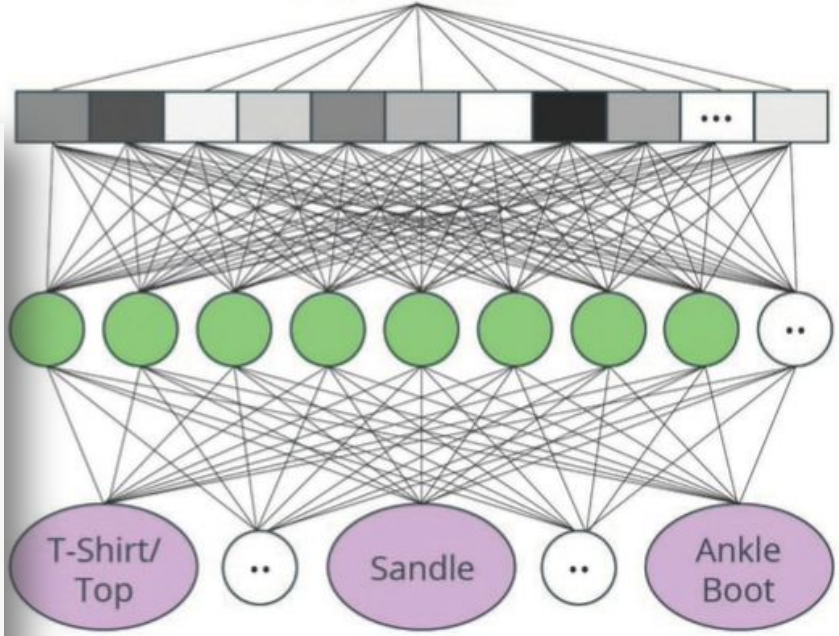




Input image (28x28 = 784 pixels)

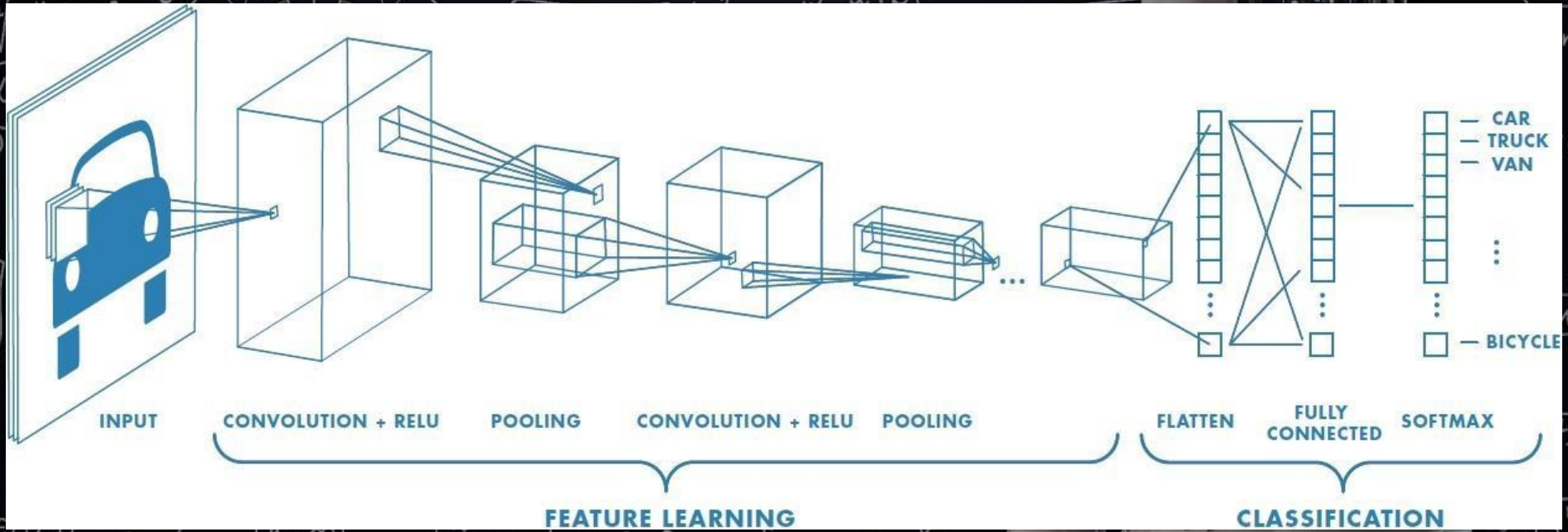
```
tf.keras.layers.Flatten(input_shape=(28, 28, 1))
```

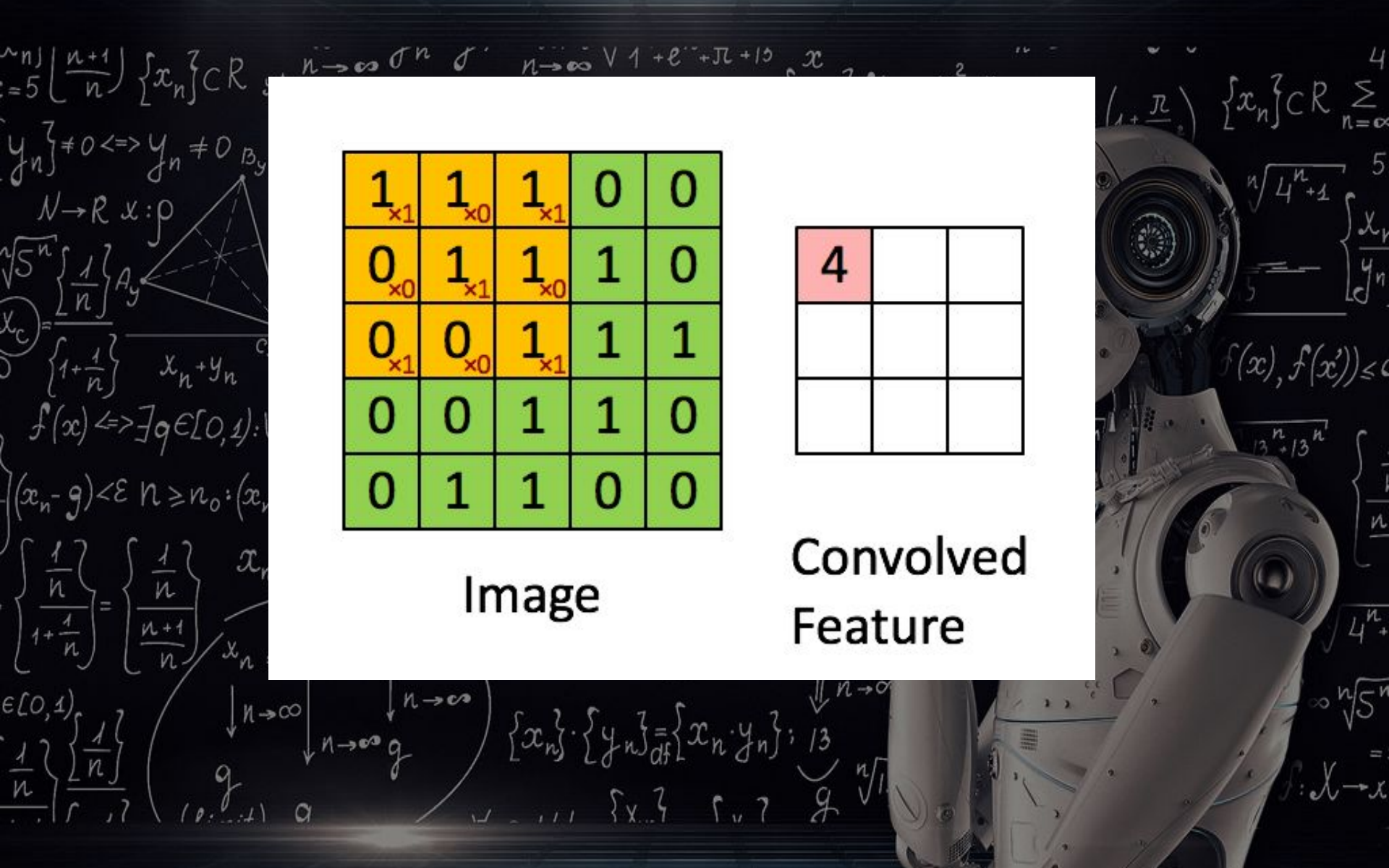
```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28, 1)),
    tf.keras.layers.Dense(500, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```



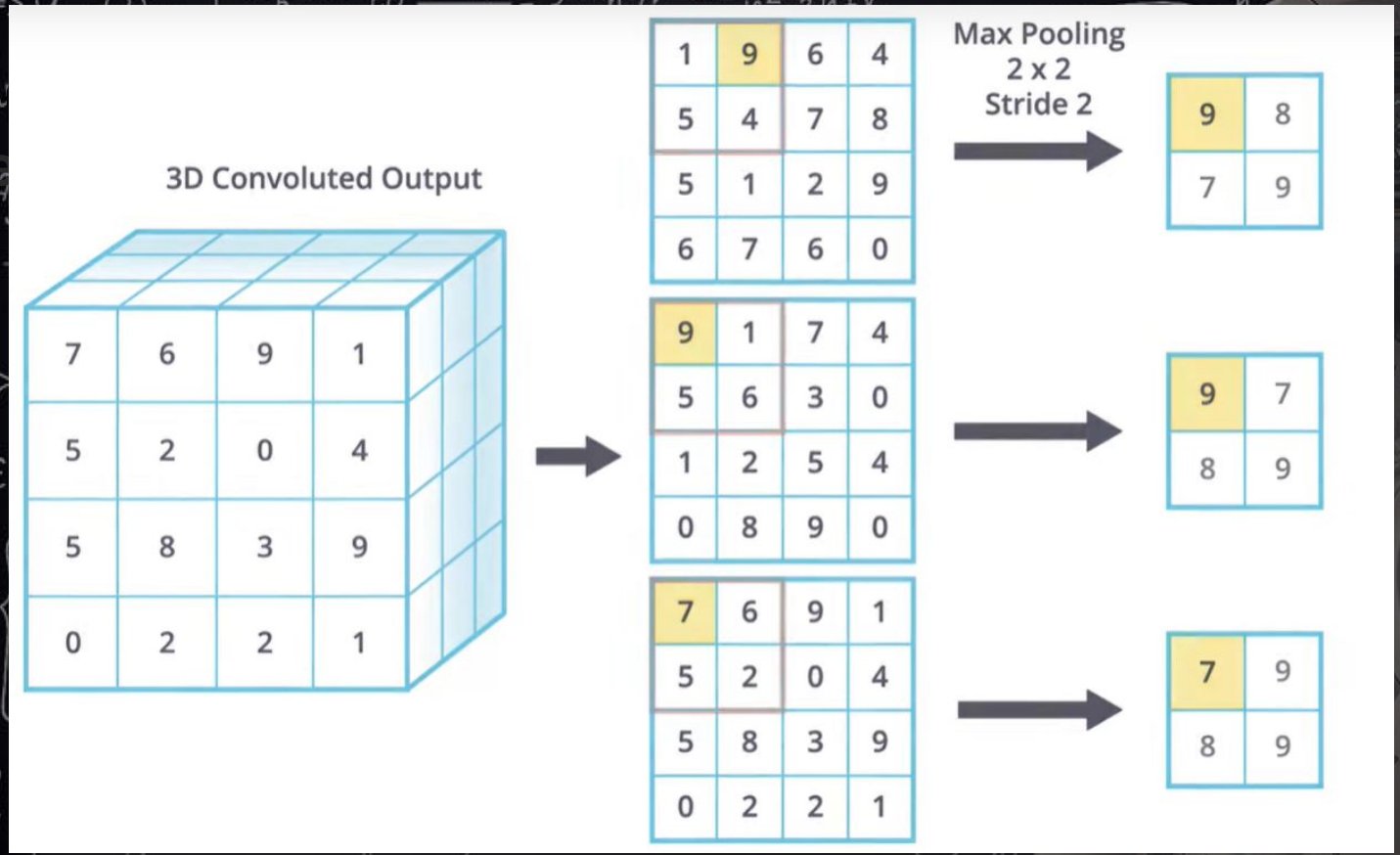
Probability of each class
Sum of all values == 1 (100%)

CNN

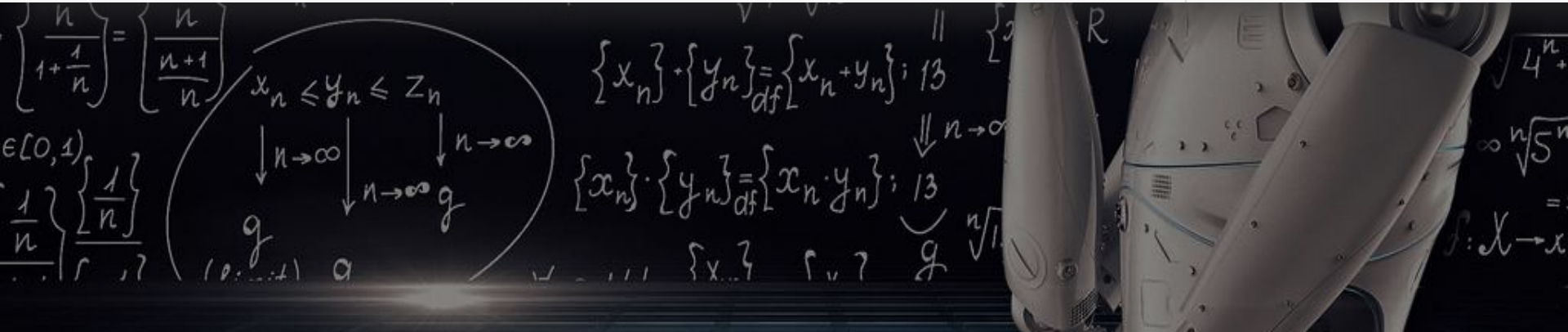

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
×1 ×0 ×1
×0 ×1 ×0
×1 ×0 ×1
4
Image
Convolved Feature

Pooling Layer

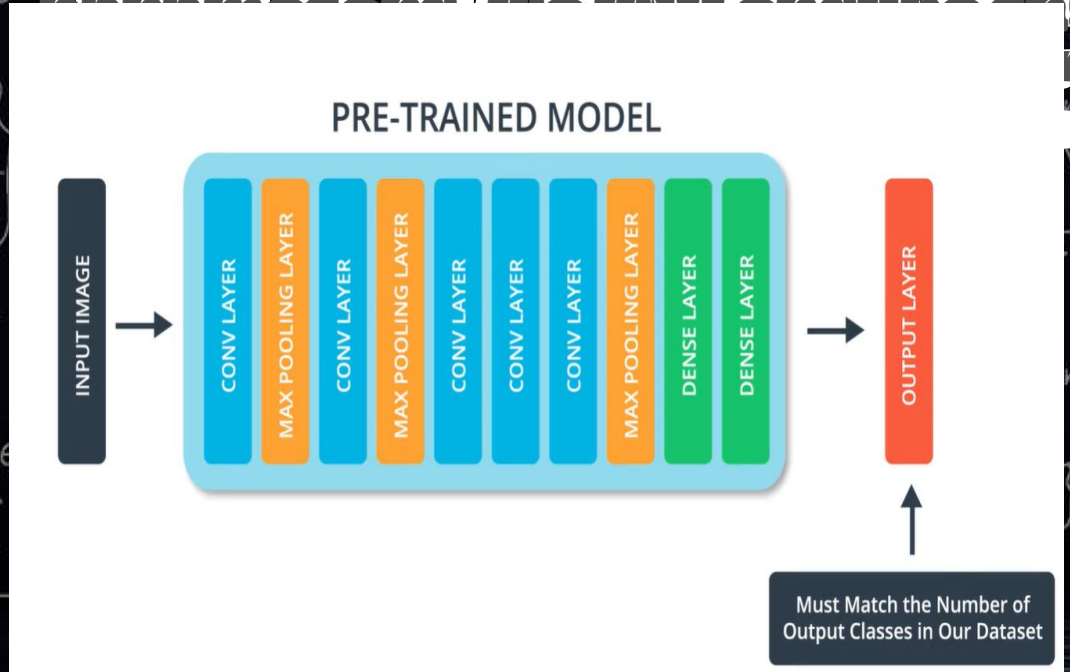


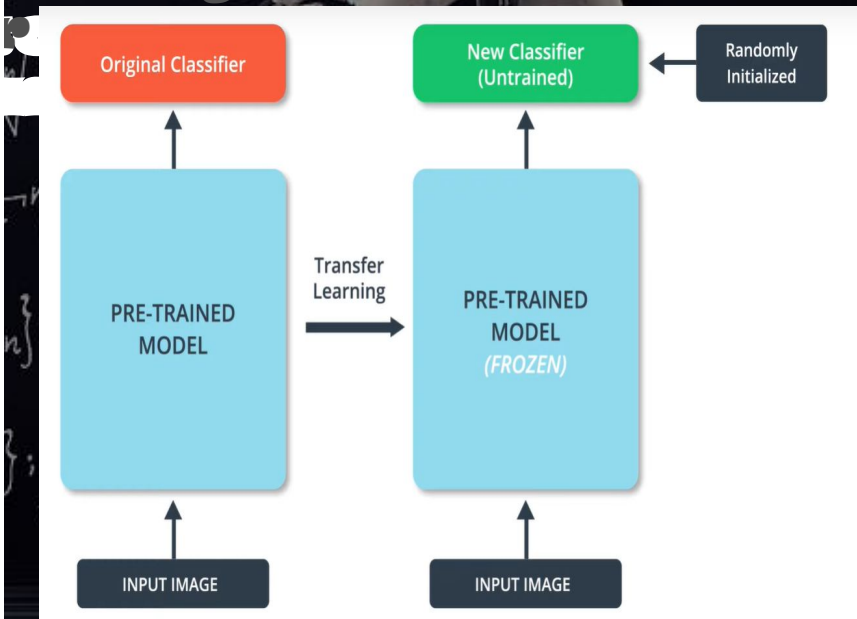
```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), padding="same" , activation=tf.nn.relu, input_shape=(28, 28, 1) ),
    tf.keras.layers.MaxPooling2D((2,2), strides=2),

    tf.keras.layers.Conv2D(64, (3,3), padding="same", activation=tf.nn.relu),
    tf.keras.layers.MaxPooling2D((2,2), strides=2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation=tf.nn.relu),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
```



Transfer Learning

A technique that reuses a model that was created by machine learning experts and that has already been trained on a large dataset. When performing transfer learning, we set the variables of a pre-trained model to non-trainable. By freezing the [illegible] only the variables [illegible]





```
URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/2"
feature_extractor = hub.KerasLayer(URL,
                                   input_shape=(IMAGE_RES, IMAGE_RES,3))
```

```
feature_extractor.trainable = False

model = tf.keras.Sequential([
  feature_extractor,
  layers.Dense(2)
])

model.summary()

# train the model with our new inputs
model.compile(
  optimizer='adam',
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
```