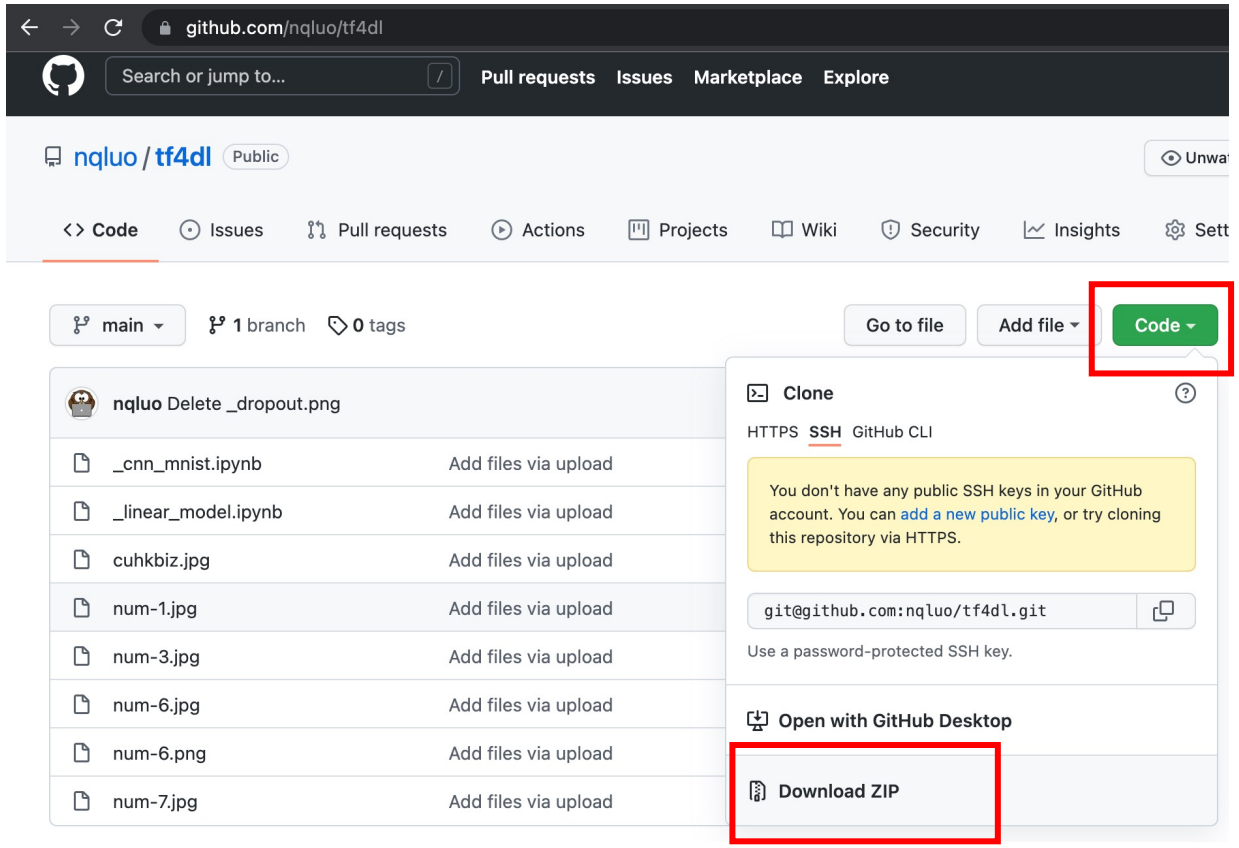
Set up Compiling Environment (use your own **gmail account**)

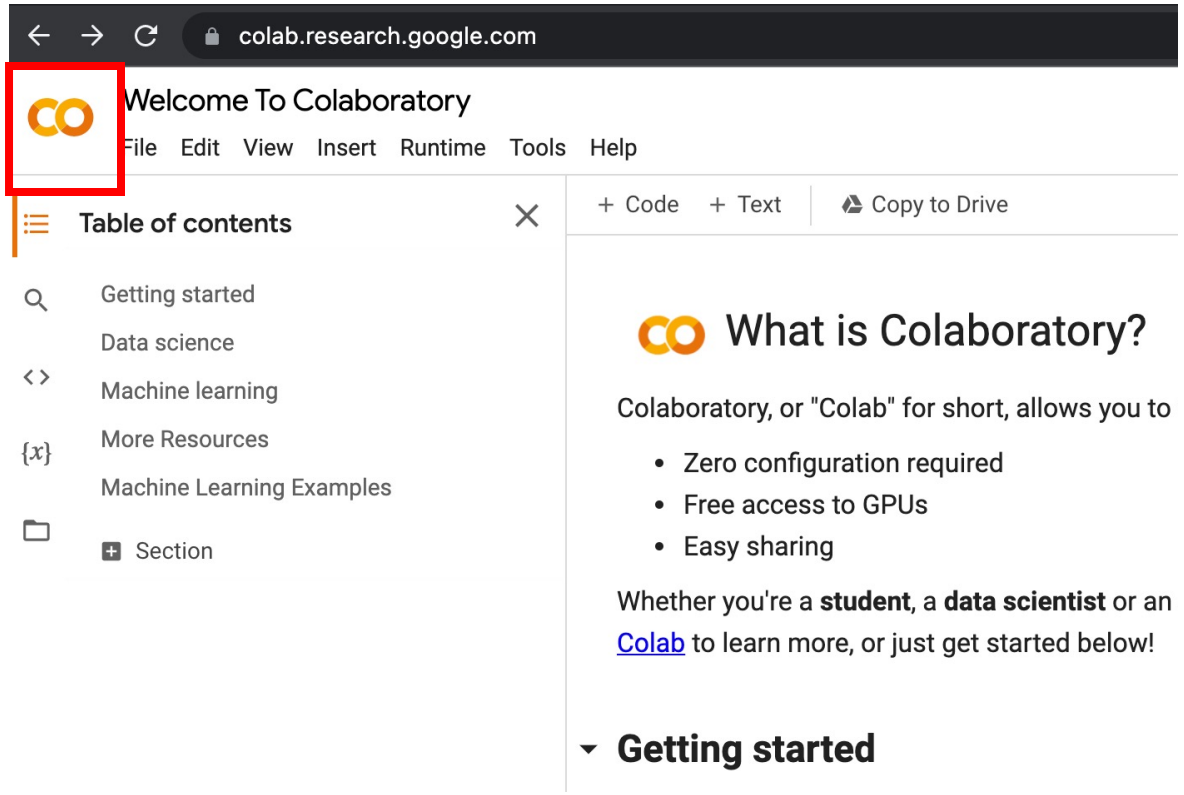
Download example code from github

https://github.com/nqluo/tf4dl



Run code on Colab

https://colab.research.google.com/



TensorFlow for Deep Learning

Ben (Ningqi) Luo, Ph.D.

nqluo@protonmail.com

Outline

What is AI, Machine Learning, Deep Learning?

What is Neuron and Neural Network?

What and How does machine learn?

How to interpret the result?

How to implement the Deep Learning in TensorFlow?

What is Machine Learning

Machine Learning is the science (and art) of programming computers so they can ***learn from data***.

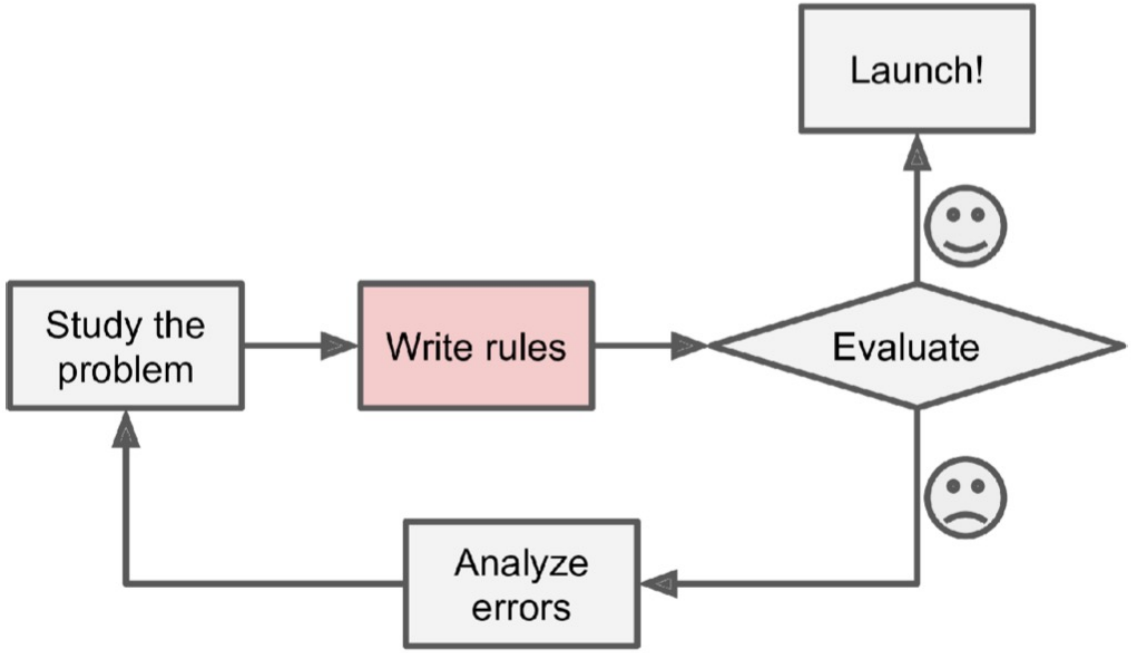


Figure 1-1. The traditional approach

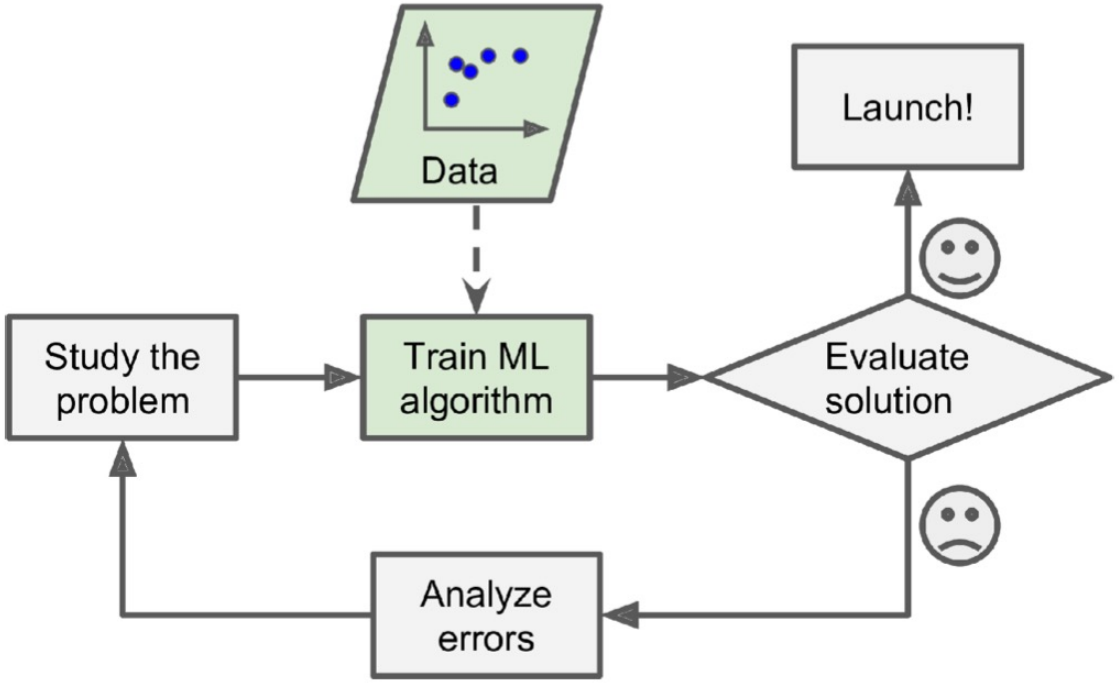


Figure 1-2. The Machine Learning approach

Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd, 2019

Example

```
0101001010100101010
1001010101001011101
0100101010010101001
0101001010100101010

Label = WALKING
```

```
1010100101001010101
0101010010010010001
0010011111010101111
1010100100111101011

Label = RUNNING
```

```
1001010011111010101
1101010111010101110
1010101111010101011
1111110001111010101

Label = BIKING
```

```
1111111111010011101
0011111010111110101
0101110101010101110
1010101010100111110

Label = GOLFING
```

Tell the machine **how** to do.

```
if(speed<4){
    status=WALKING;
} else if(speed<12){
    status=RUNNING;
} else {
    status=BIKING;
}
```

```
// ???
```

Gathering and labeling data

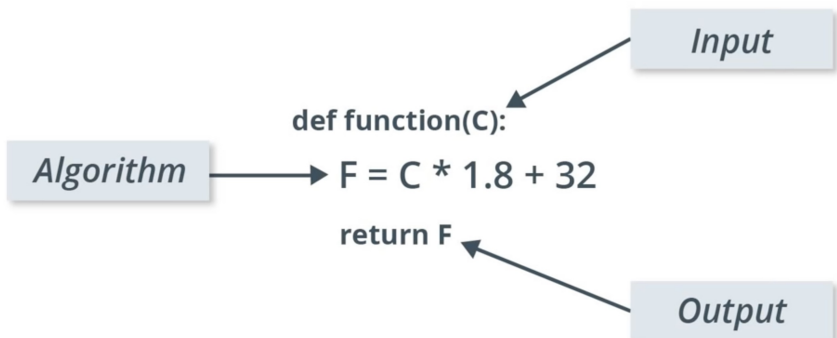
Show the machine **what** to do.

Laurence Moroney, AI and Machine Learning for Coders, 2020

Traditional Software Development

The input and the algorithm is known, and you write a function to produce an output

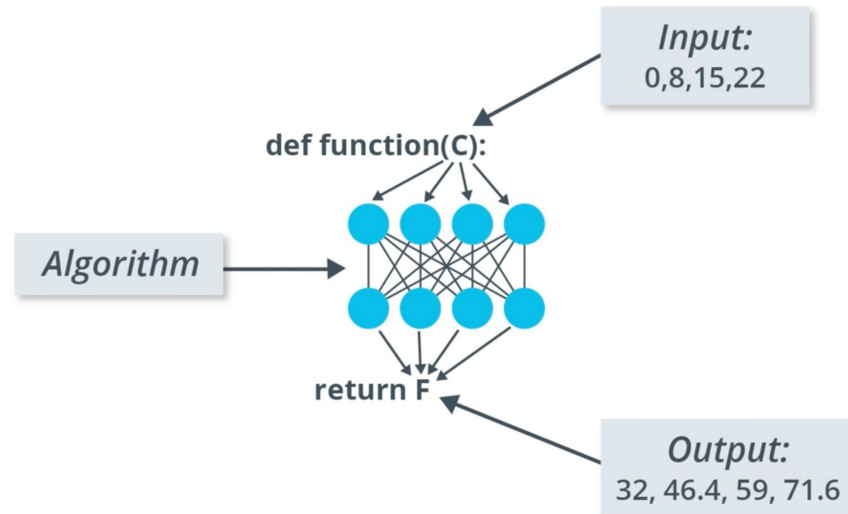
- Input data
- Apply logic to it
- Which produces a result



Machine Learning

You know the input and the output, but you don't know the algorithm that creates the output given the input

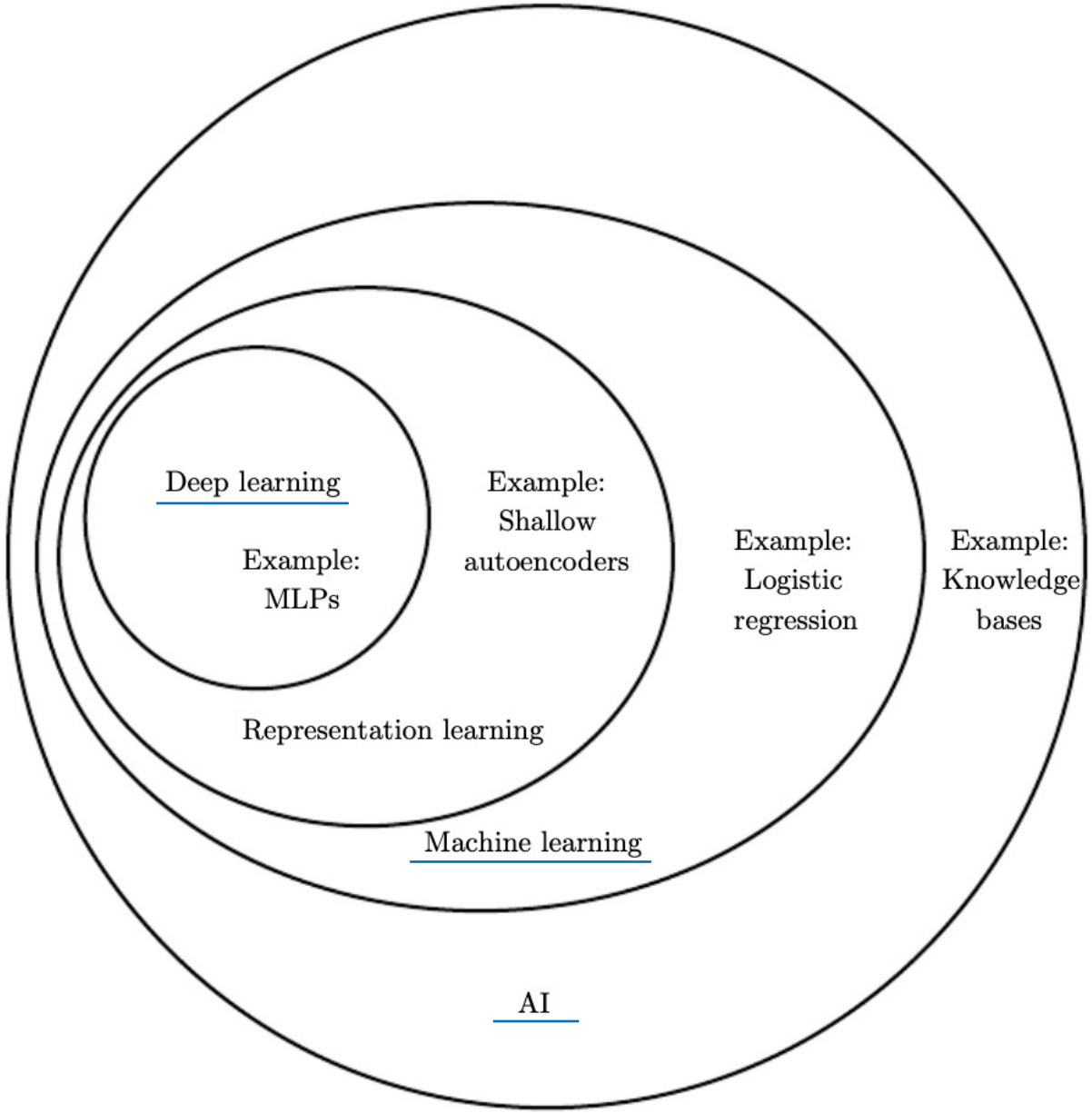
- Take pairs of input and output data
- Figure out the algorithm

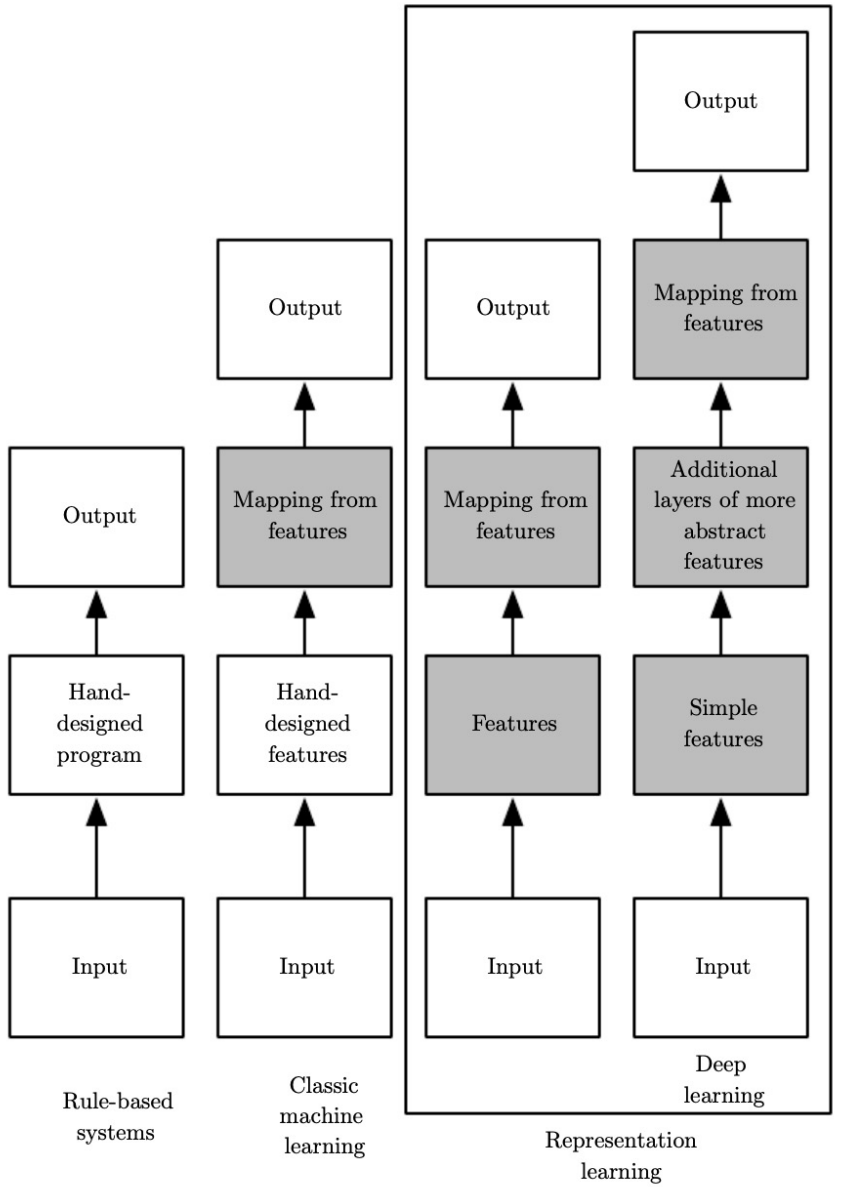


Machine Learning is great for:

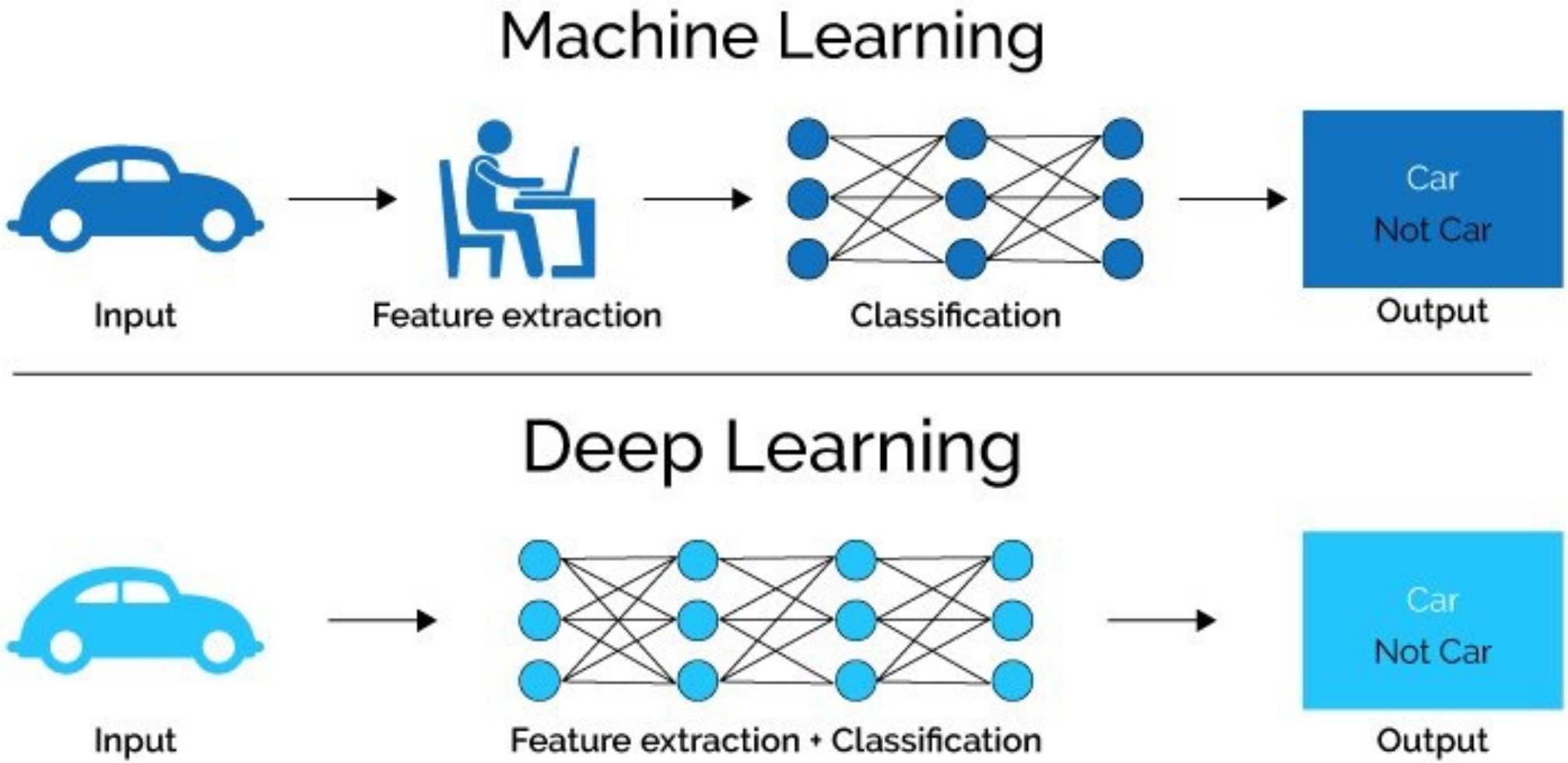
- Problems for which existing solutions require a lot of fine-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better than the traditional approach.
- Complex problems for which using a traditional approach yields no good solution: the best Machine Learning techniques can perhaps find a solution.
- Fluctuating environments: a Machine Learning system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd, 2019

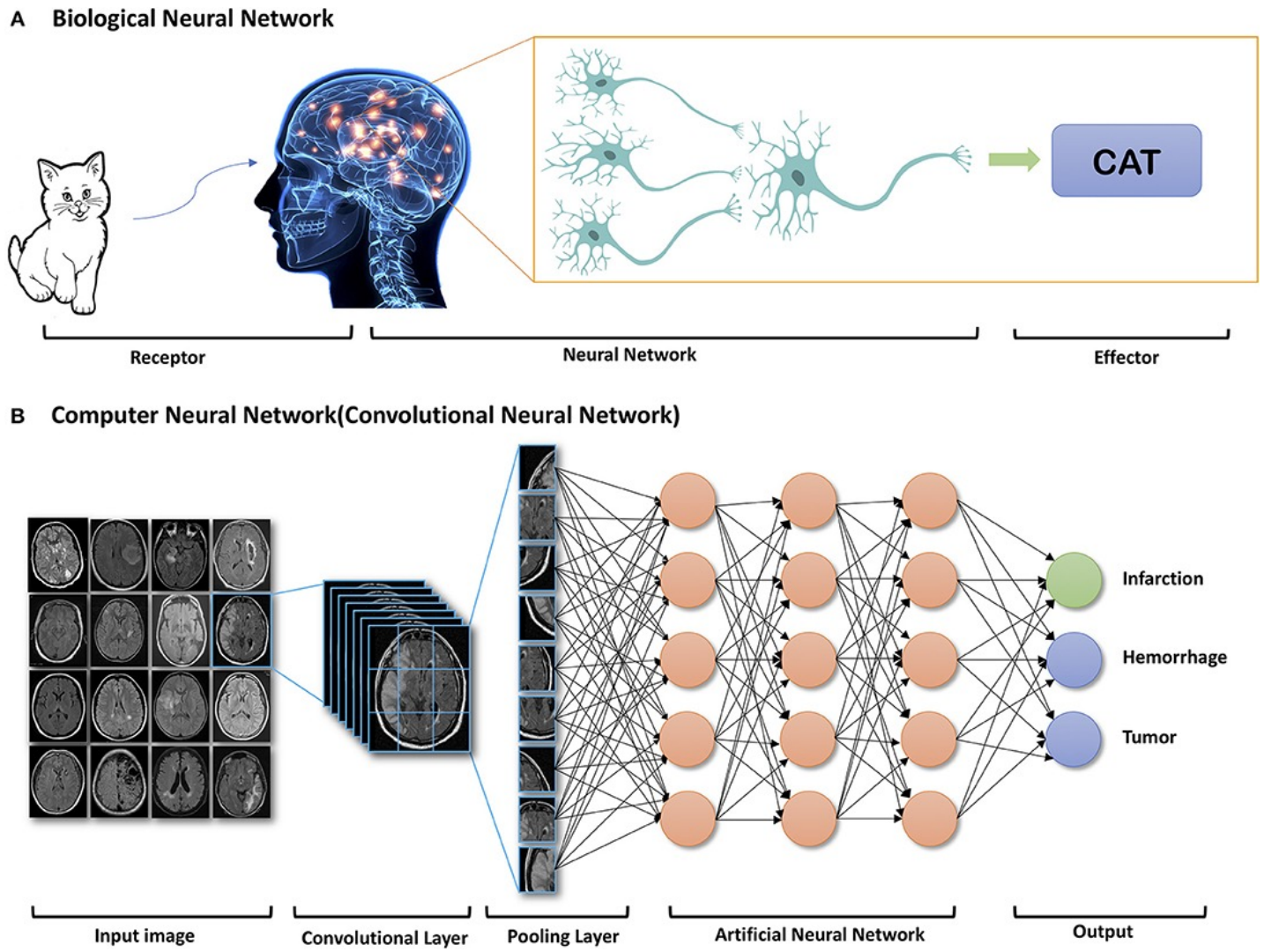




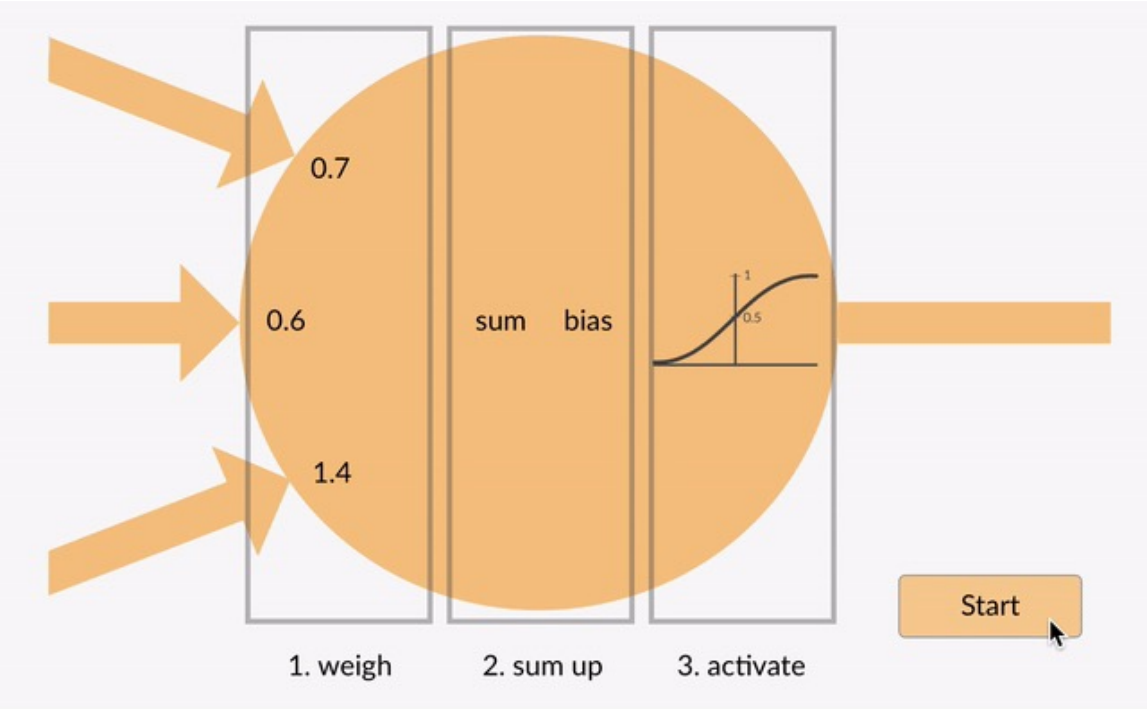
https://www.deeplearningbook.org/


Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output

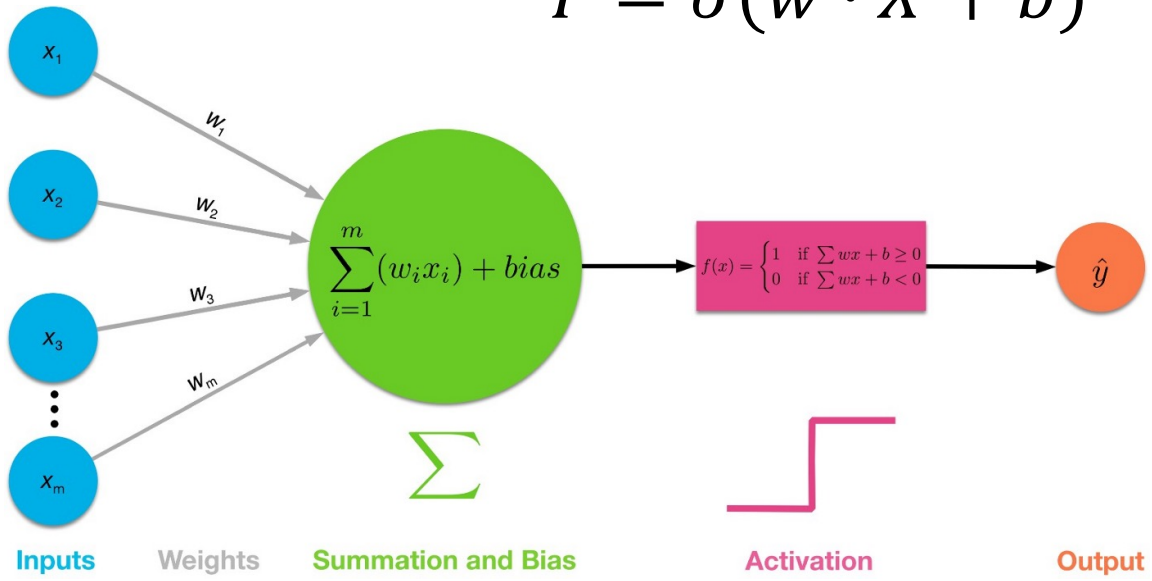
Deep Learning and Neural Network



Neuron



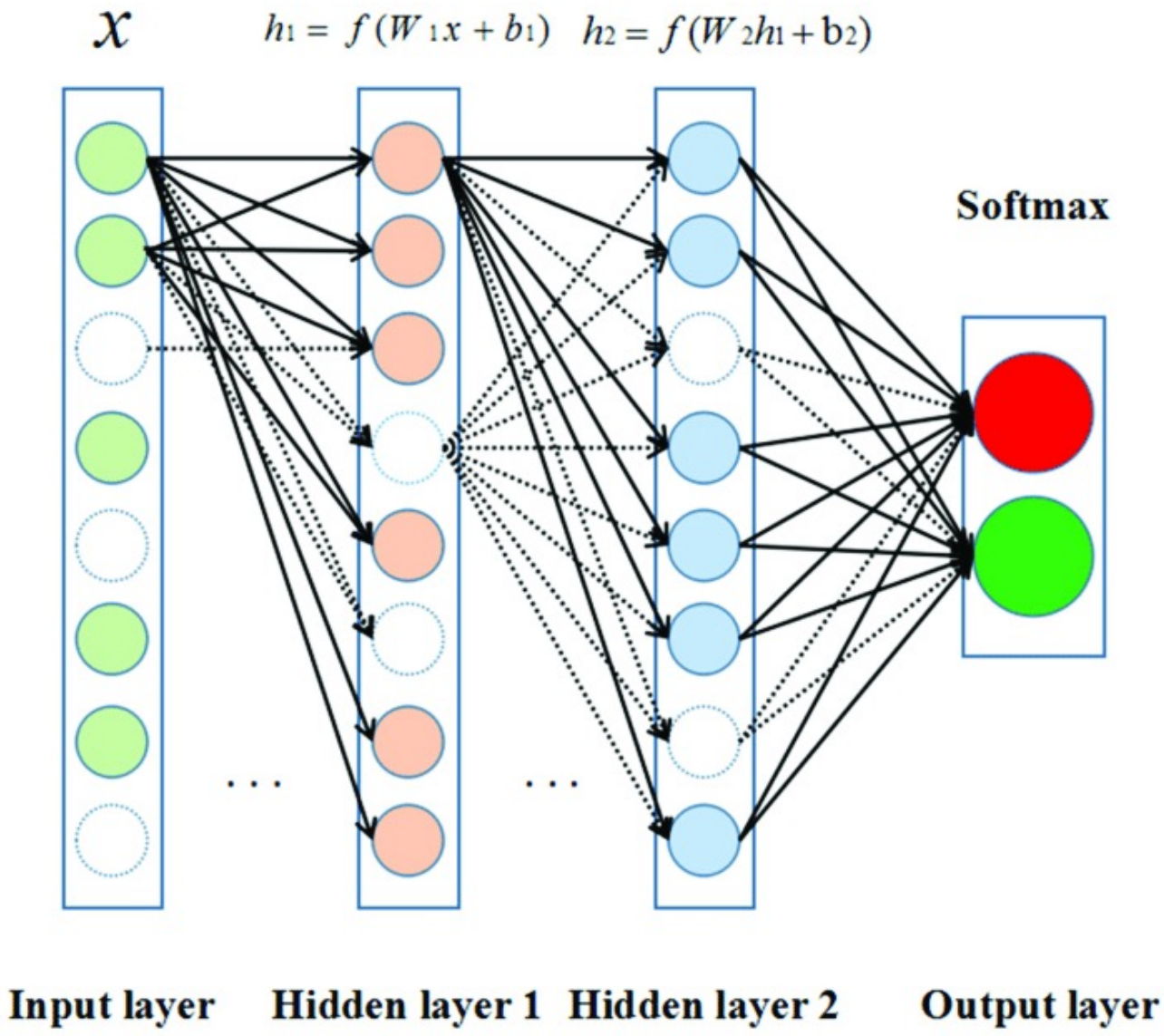
$$Y = \sigma(w \cdot X + b)$$



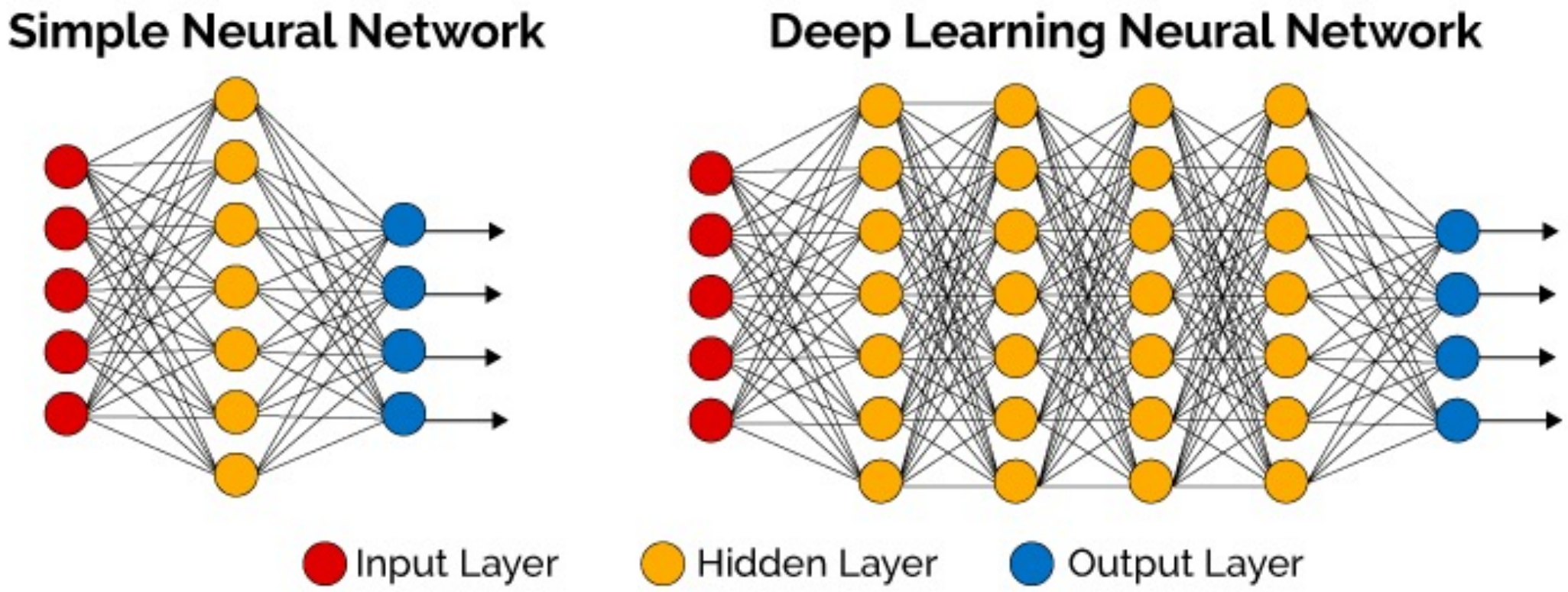
X: Input; Y: Output; w: Weight; b: Bias; σ: Activation function

Neural Network

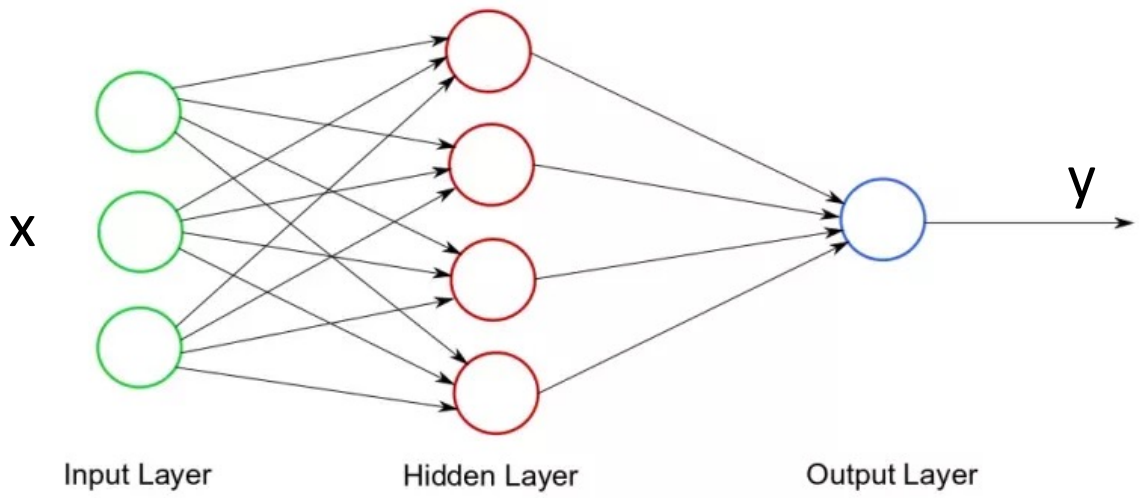
$$Y = \sigma(w \cdot X + b)$$



Deep Neural Network (DNN)



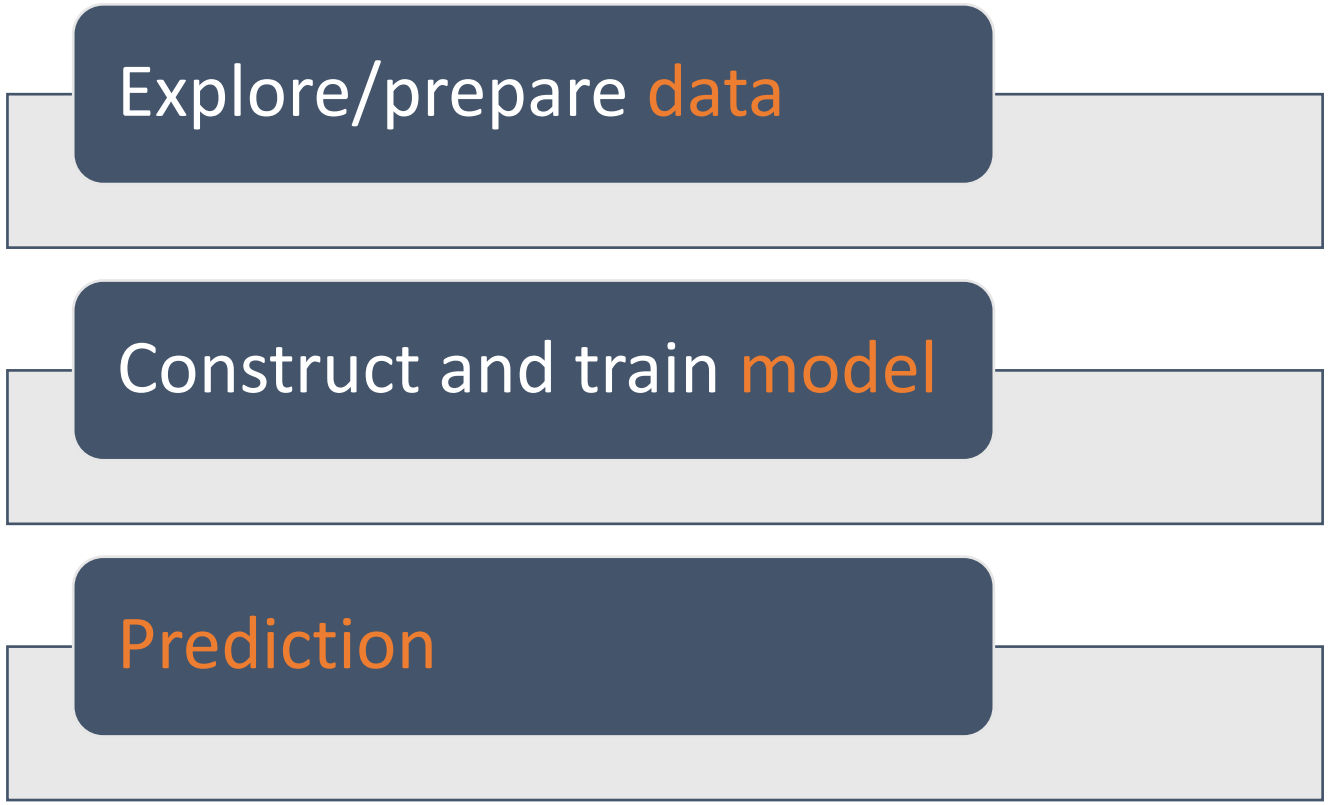
Neural Network and the Code



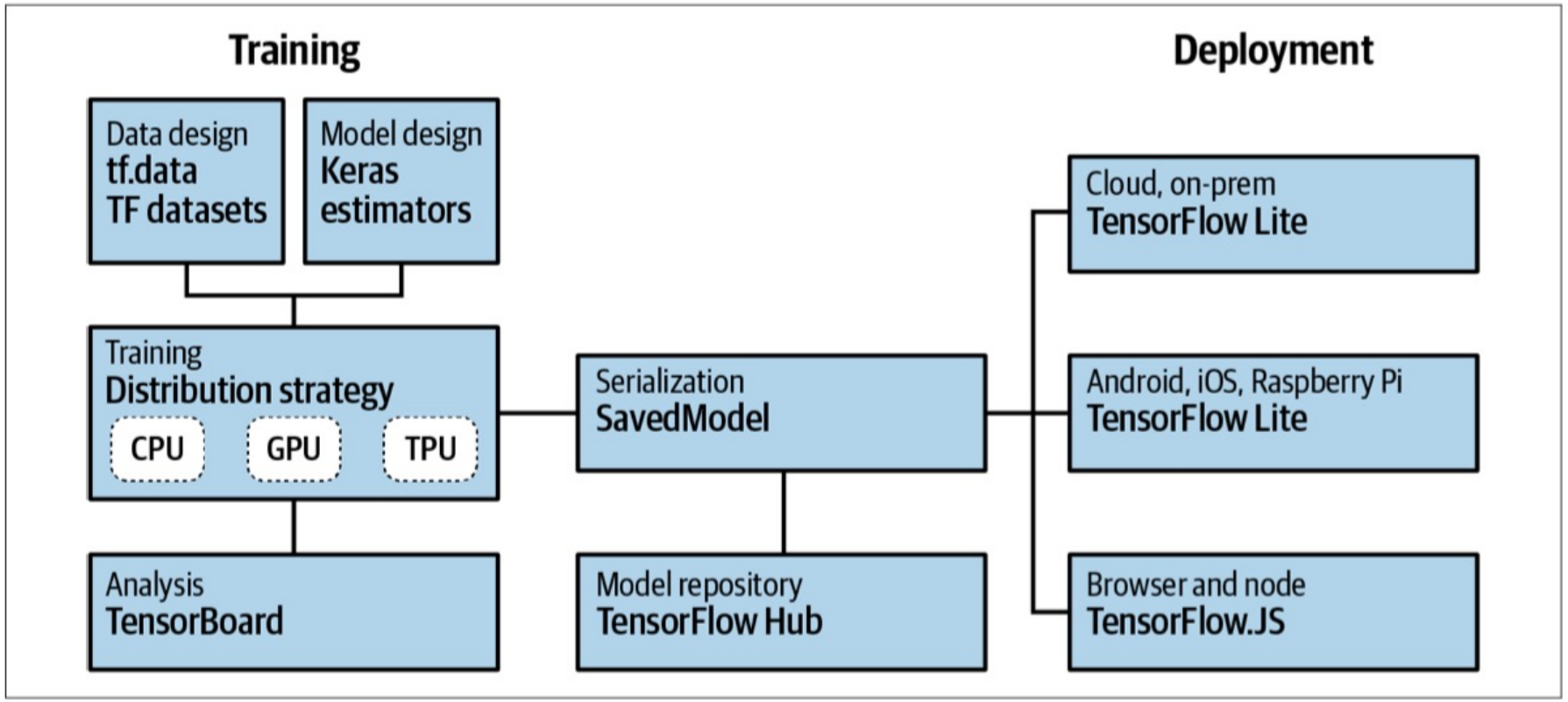

```
model = tf.keras.Sequential([
                                tf.keras.layers.Dense(units=3, input_shape=[1]),
                                tf.keras.layers.Dense(units=4),
                                tf.keras.layers.Dense(units=1)
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 6
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 16
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 5
=================================================================
Total params: 27
Trainable params: 27
Non-trainable params: 0
_________________________________________________________________
```

Workflow


Explore/prepare data
Construct and train model
Prediction

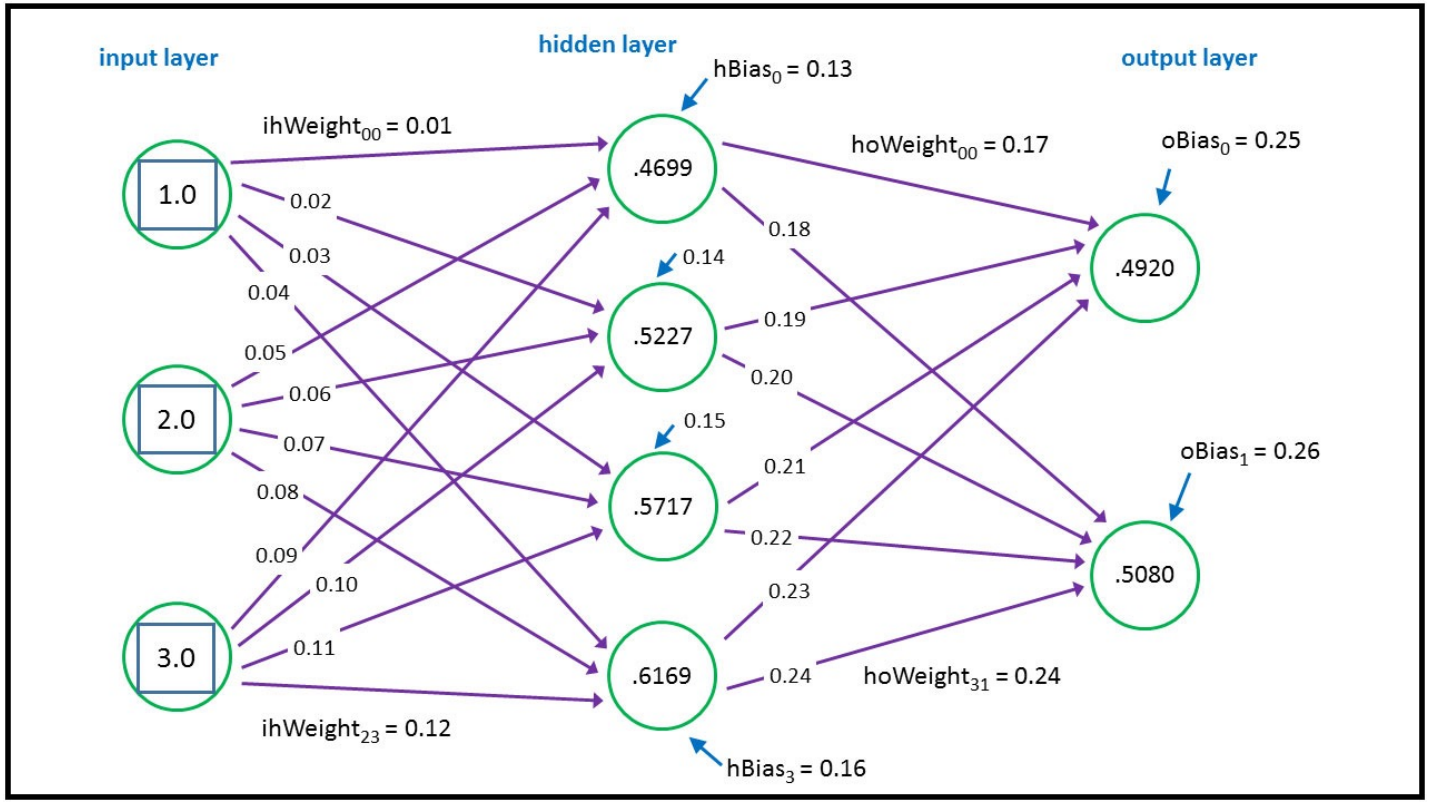
TensorFlow high-level architecture



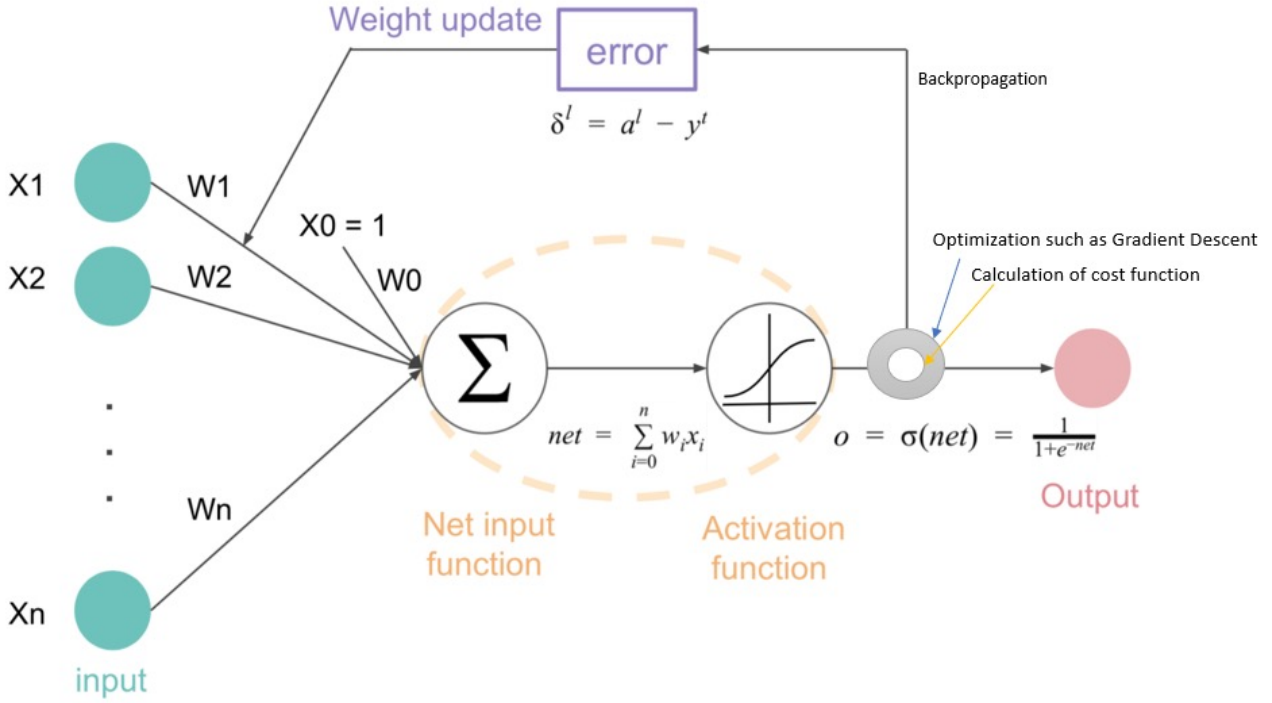
demo

Laurence Moroney, AI and Machine Learning for Coders, 2020

What does the machine learn? Weight and Bias in Neuron

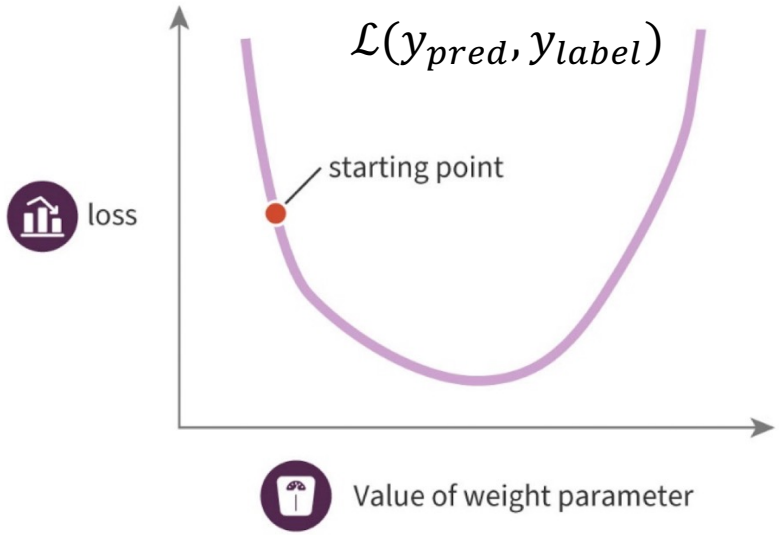


How does the machine learn?



Gradient Descent

- Optimization algorithm to find good model parameters by minimizing a loss function.

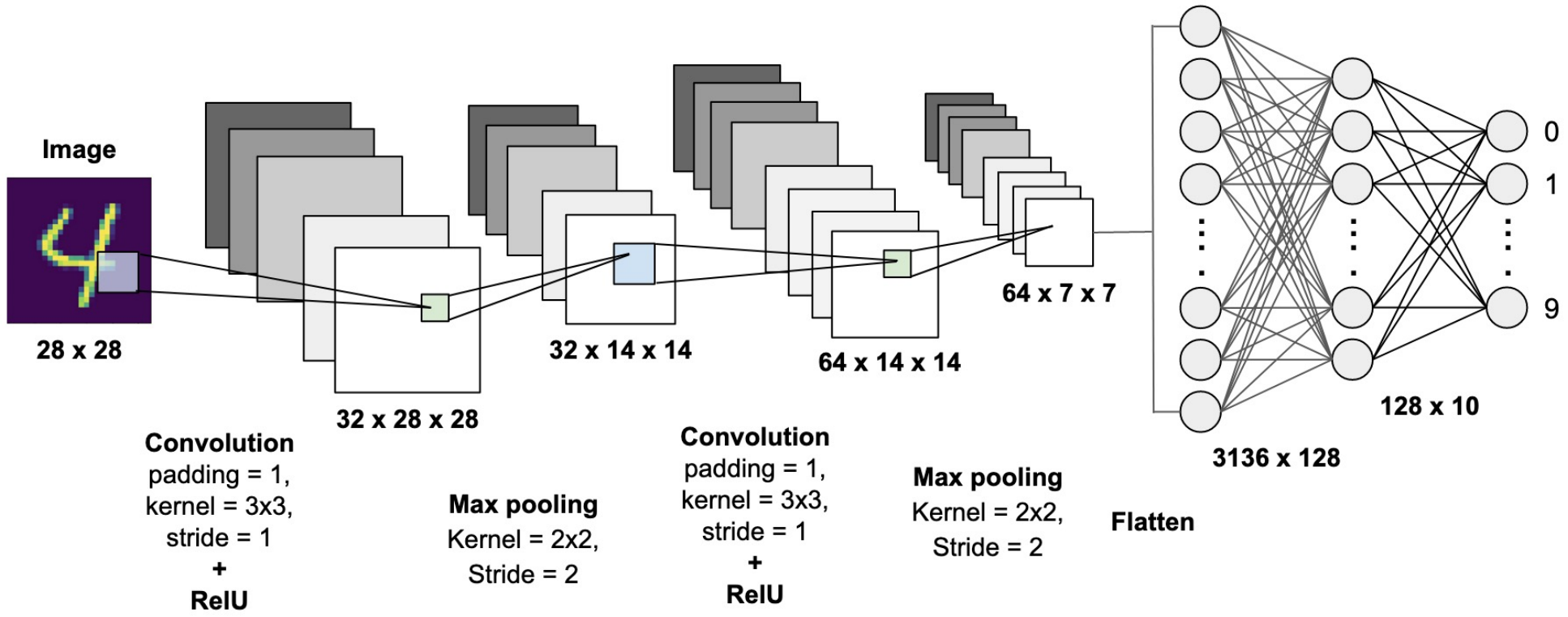


DNN

```
model = tf.keras.Sequential([
              tf.keras.layers.Dense(units=1,input_shape=[1])
              ])
```

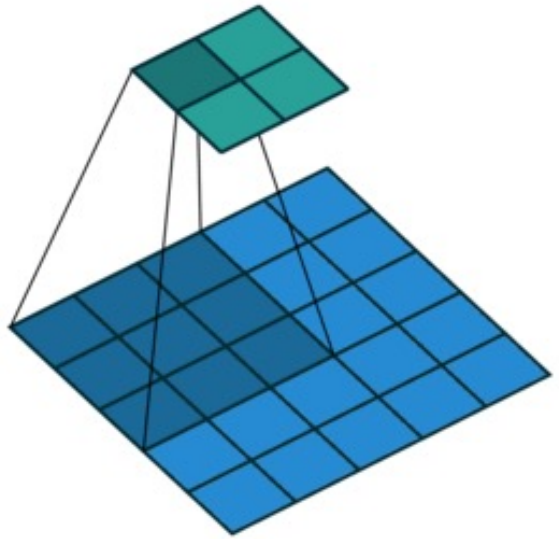
```
model = tf.keras.Sequential([
              tf.keras.layers.Dense(units=3, input_shape=[1]),
              tf.keras.layers.Dense(units=4),
              tf.keras.layers.Dense(units=1)
              ])
```

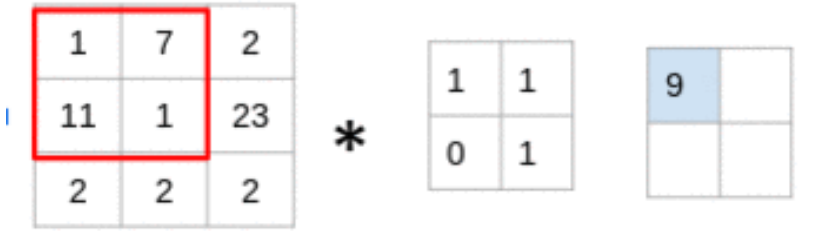
Convolution Neural Network (CNN)

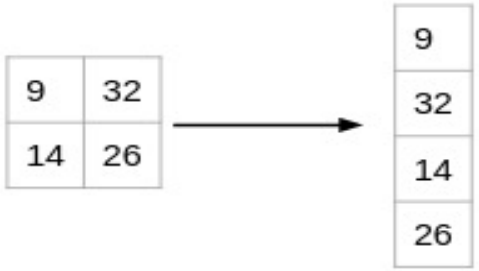


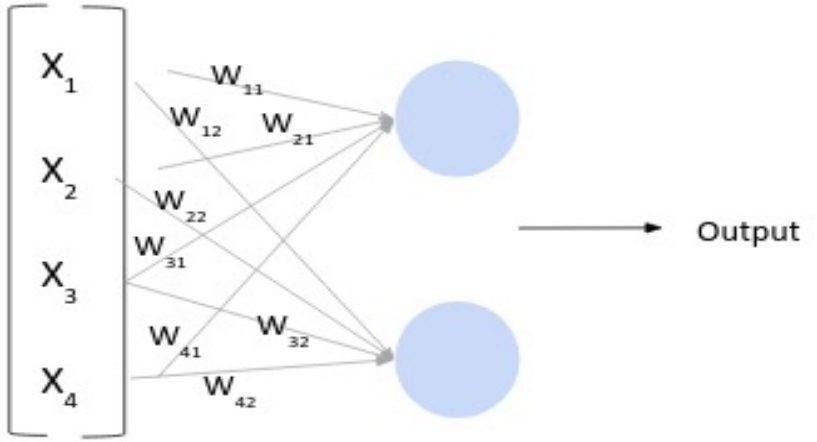
Building Blocks of CNN

- The building blocks of CNN are filters, i.e., kernels (neuron). Kernels are used to extract the relevant features from the input using the convolution operation.









Kernel (filter)

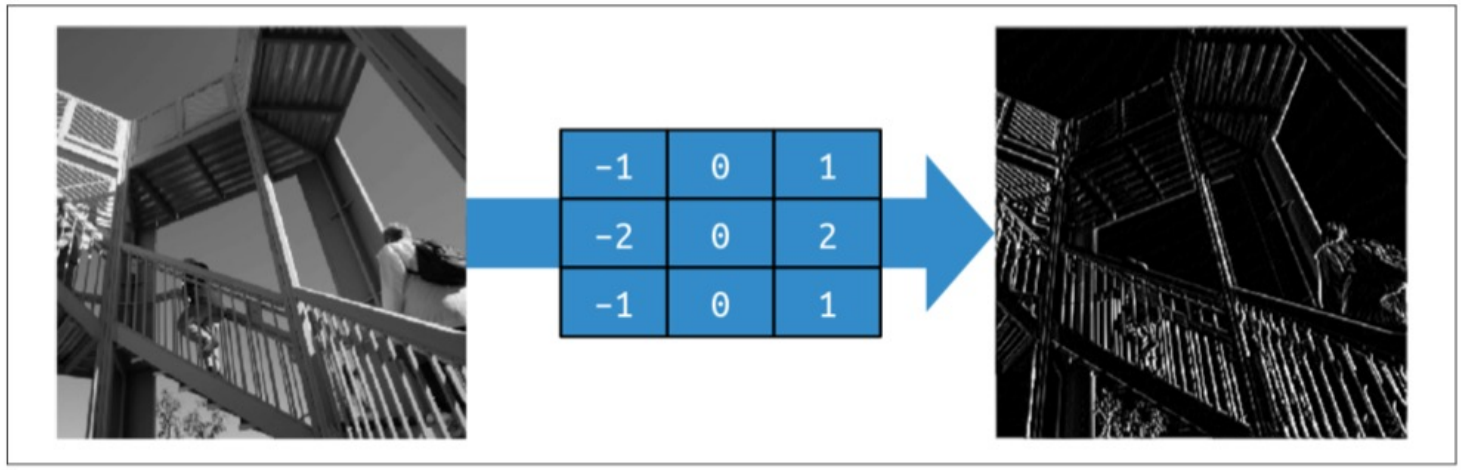


Figure 3-2. Using a filter to get vertical lines

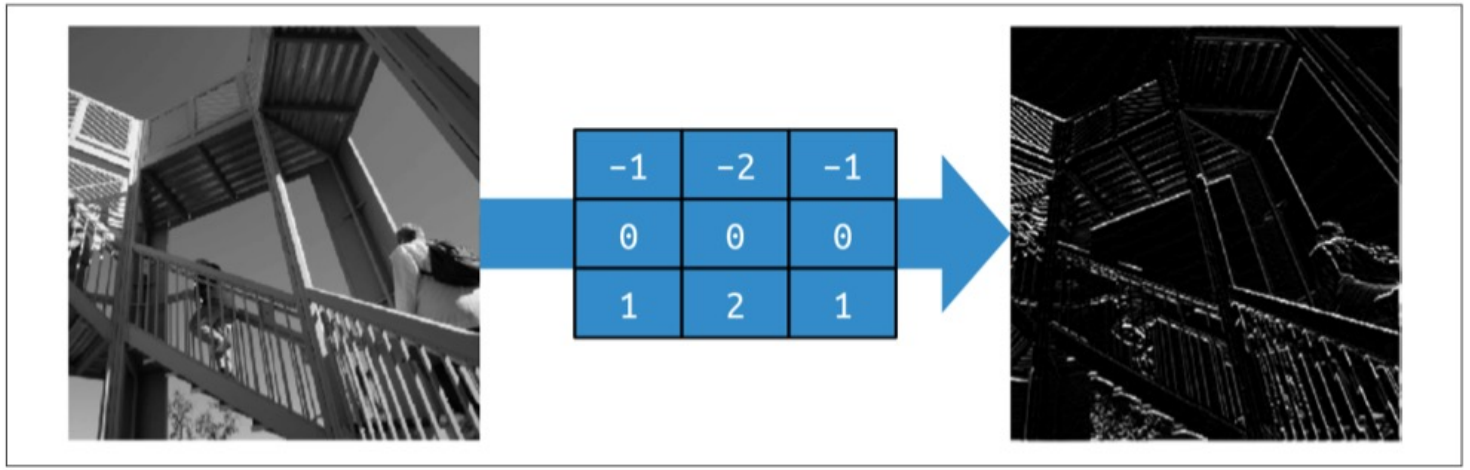
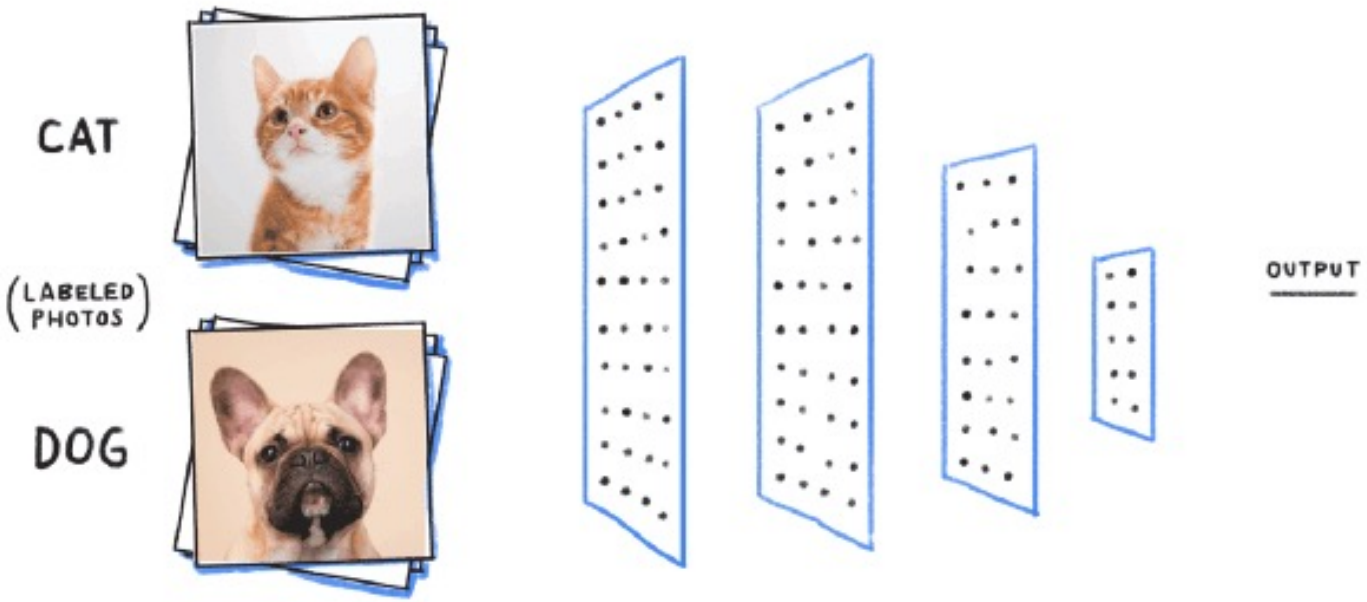


Figure 3-3. Using a filter to get horizontal lines

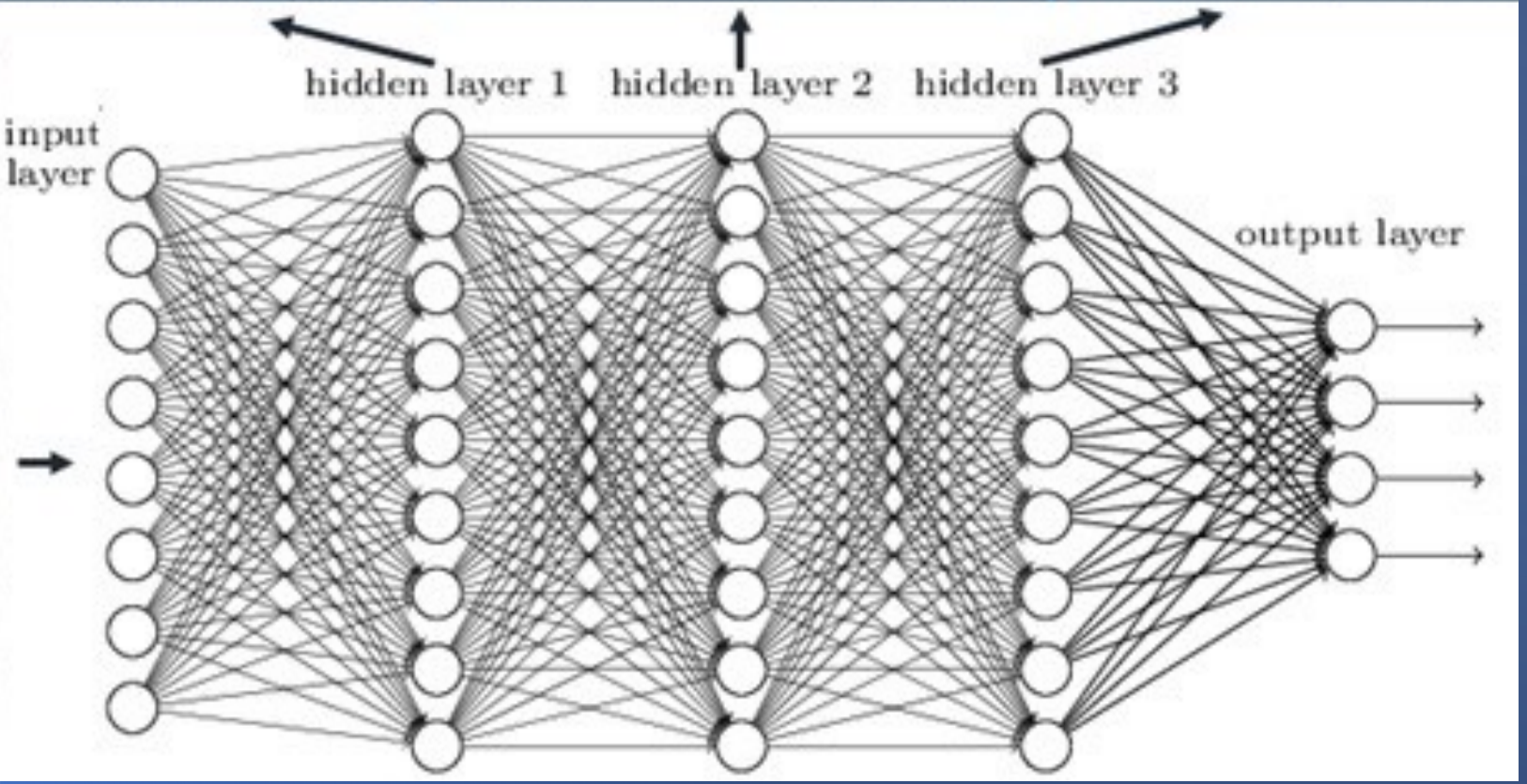
Laurence Moroney, AI and Machine Learning for Coders, 2020

Advantages of CNN

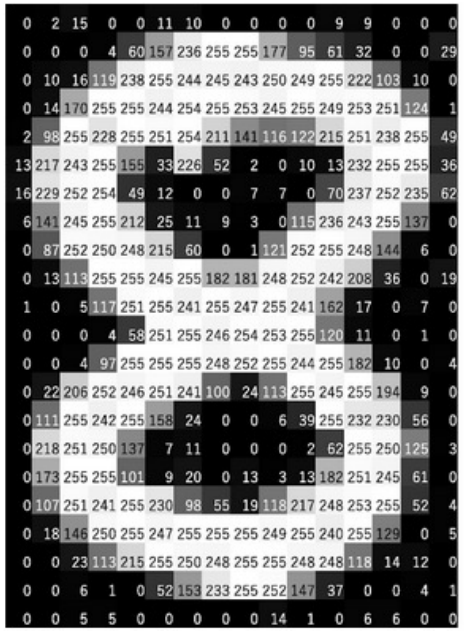
- CNN learns the filters automatically without mentioning it explicitly. These filters help in extracting the right and relevant features from the input data
- CNN captures the **spatial features** from an image. Spatial features refer to the arrangement of pixels and the relationship between them in an image. They help us in identifying the object accurately, the location of an object, as well as its relation with other objects in an image.
- CNN also follows the concept of parameter sharing. A single filter is applied across different parts of an input to produce a feature map.



Deep neural networks learn hierarchical feature representations



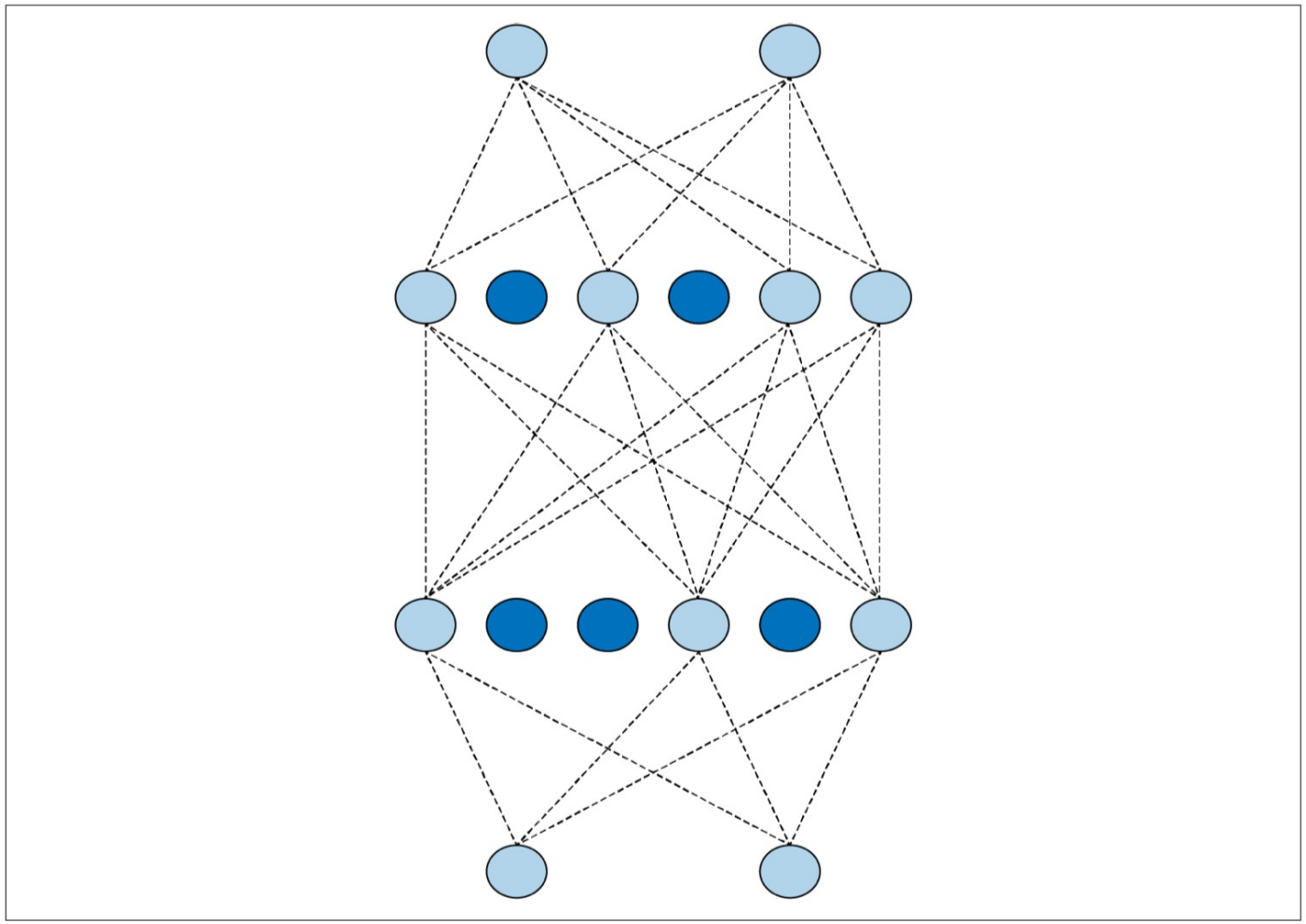
MNIST



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

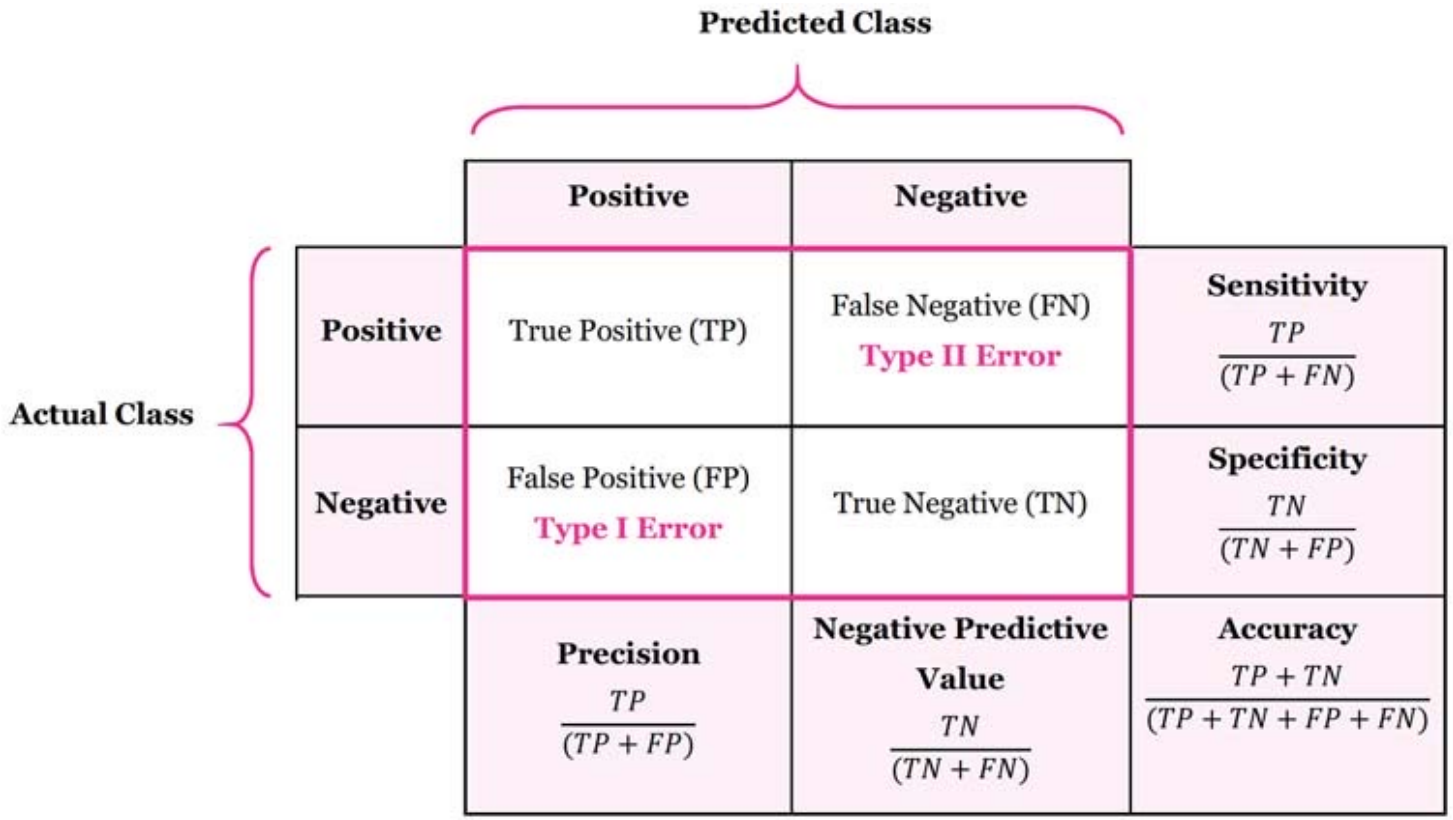
0 2 15 0 0 11 10 0 0 0 0 9 9 0 0 0
0 0 0 4 60 157 236 255 255 177 95 61 32 0 0 29
0 10 16 119 238 255 244 245 243 250 249 255 222 103 10 0
0 14 170 255 255 244 254 255 253 245 255 249 253 251 124 1
2 98 255 228 255 251 254 211 141 116 122 215 251 238 255 49
13 217 243 255 155 33 226 52 2 0 10 13 232 255 255 36
16 229 252 254 49 12 0 0 7 7 0 70 237 252 235 62
6 141 245 255 212 25 11 9 3 0 115 236 243 255 137 0
0 87 252 250 248 215 60 0 1 121 252 255 248 144 6 0
0 13 113 255 255 245 255 182 181 248 252 242 208 36 0 19
1 0 5 117 251 255 241 255 247 255 241 162 17 0 7 0
0 0 0 4 58 251 255 246 254 253 255 120 11 0 1 0
0 0 4 97 255 255 255 248 252 255 244 255 182 10 0 4
0 22 206 252 246 251 241 100 24 113 255 245 255 194 9 0
0 111 255 242 255 158 24 0 0 6 39 255 232 230 56 0
0 218 251 250 137 7 11 0 0 0 2 62 255 250 125 3
0 173 255 255 101 9 20 0 13 3 13 182 251 245 61 0
0 107 251 241 255 230 98 55 19 118 217 248 253 255 52 4
0 18 146 250 255 247 255 255 255 249 255 240 255 129 0 5
0 0 23 113 215 255 250 248 255 255 248 248 118 14 12 0
0 0 6 1 0 52 153 233 255 252 147 37 0 0 4 1
0 0 5 5 0 0 0 0 0 14 1 0 6 6 0 0

Dropout



demo

How to evaluate the model?

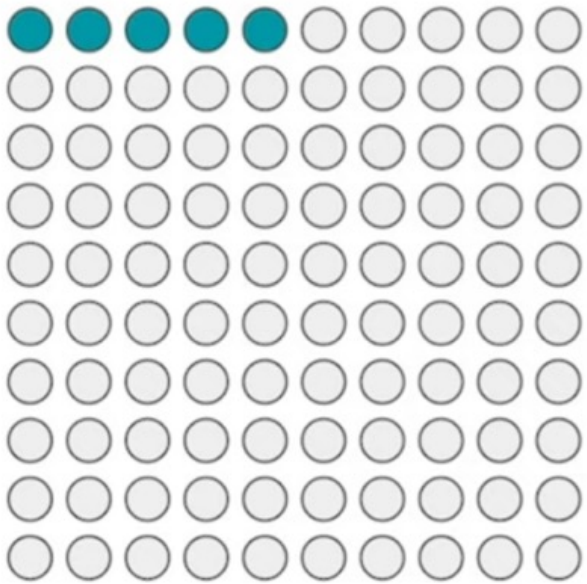


*Sensitivity is also referred to true positive rate or recall.

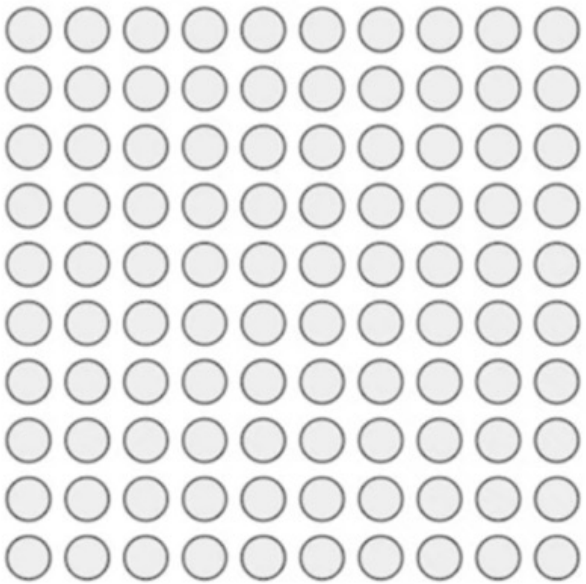
A **confusion matrix** is a table that is often used to describe the performance of a classification model.

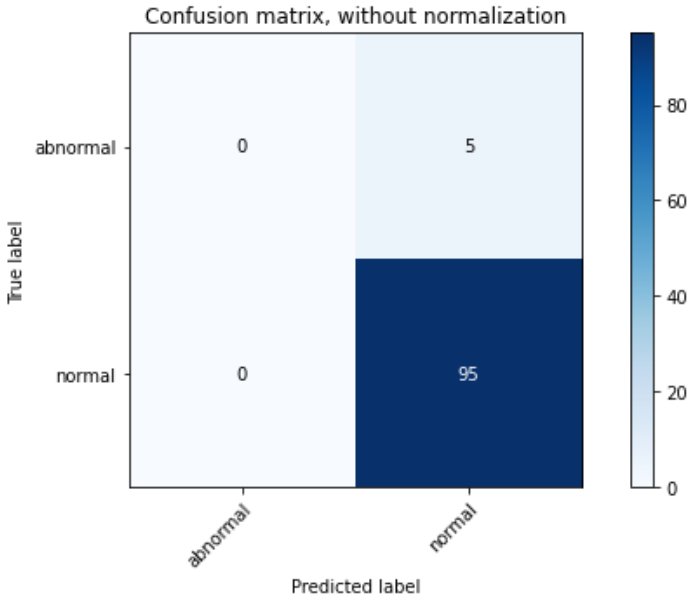
Accuracy

True Label



Algorithm Prediction





5 abnormal cases
95 normal cases

Algorithm predicts 100 normal cases

$$Accuracy = \frac{TP + TN}{All\ cases} = \frac{95}{100} = 95\%$$

Accuracy is not a good measure for unbalance datasets.

Sensitivity and Specificity

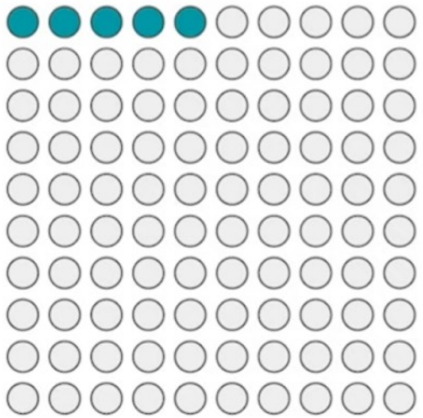
$$Sensitivity\ (recall) = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

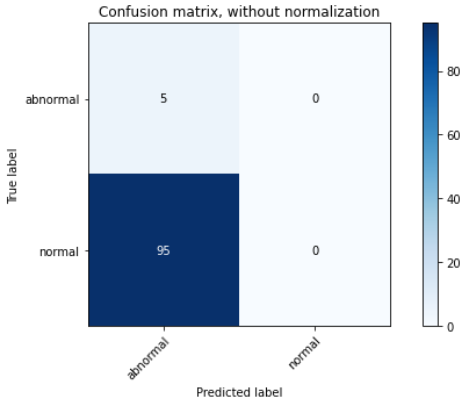
Sensitivity is also referred as True Positive Rate. It's the proportion of accurately-identified positive cases. It don't take FP into account.

Specificity is the proportion of accurately-identified positive cases. It don't take TP into account.

True Label



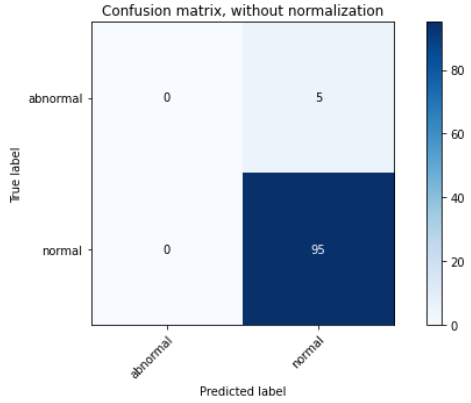
Predict all cases as Positive.



$$Sen = \frac{5}{5 + 0} = 1$$

$$Spe = \frac{0}{0 + 95} = 0$$

Predict all cases as Negative.



$$Sen = \frac{0}{0 + 5} = 0$$

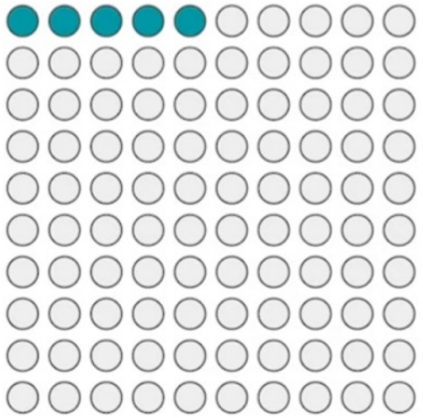
$$Spe = \frac{95}{95 + 0} = 1$$

Precision

$$Precision = \frac{TP}{TP + FP}$$

Take into account all the cases identified as positive by the algorithm.

True Label



Predict all cases as Positive.

$$Sen = \frac{0}{0+5} = 0$$

$$Spe = \frac{95}{95+0} = 1$$

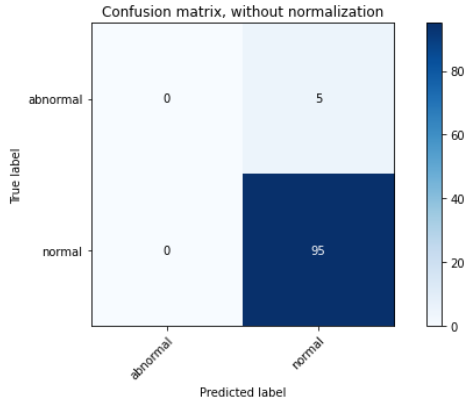
$$Pre = \frac{0}{0} = N/A$$

$$Sen = \frac{5}{5+0} = 1$$

$$Spe = \frac{0}{0+95} = 0$$

$$Pre = \frac{5}{5+95} = 5\%$$

Predict all cases as Negative.



$$Sen = \frac{0}{0+5} = 0$$

$$Spe = \frac{95}{95+0} = 1$$

$$Pre = \frac{0}{0} = N/A$$

Clinical Relevance

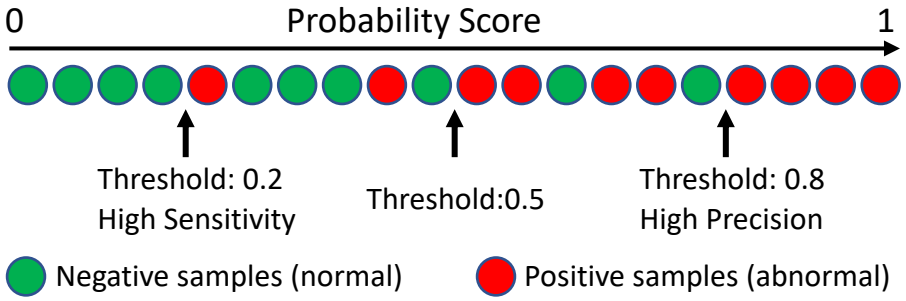
$$Precision = \frac{TP}{TP + FP}$$

- High Precision = more confidence in a positive result (FP => minimum)
- Does not take False Negatives into account.
- Application: Confirming a diagnosis

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN}$$

- High Sensitivity (Recall) = most confident when the test is negative. (FN => minimum)
- Does not take False Positive into account.
- Application: Screening studies, worklist prioritization

For example, CNN models output a probability ranging from 0-1 that indicates how likely the image belongs to a class. We will need a cut-off value called threshold to assist in making the decision if the probability is high enough to belong to one class. By changing the threshold, Recall and precision are tuned for different clinical application.

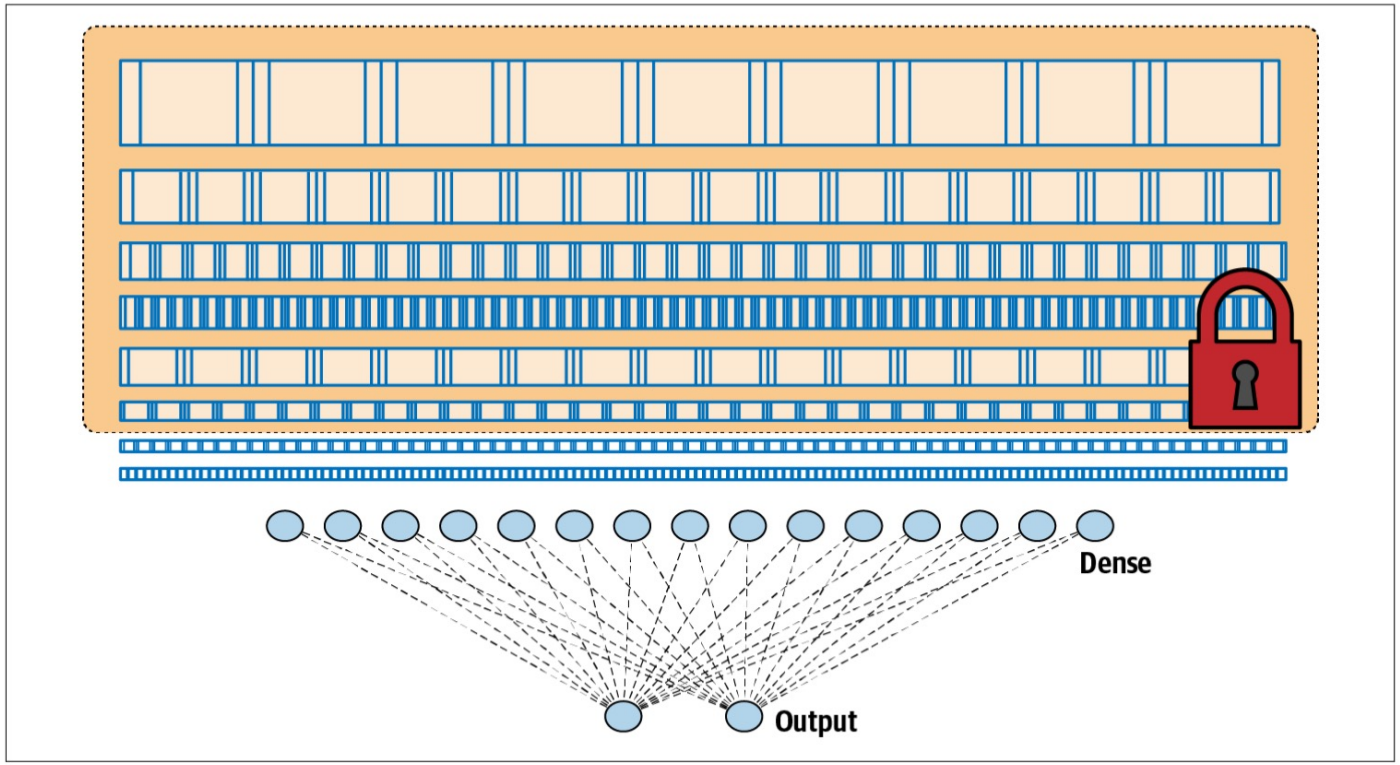


Threshold	TP	FN	FP	TN	Sen	Spe	Pre
0.2	10	0	6	4	1	0.4	0.625
0.5	8	2	2	8	0.8	0.8	0.8
0.8	4	6	0	10	0.4	1	1

CNN

```
model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
                                           input_shape=(28, 28, 1)),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation=tf.nn.relu),
        tf.keras.layers.Dense(10, activation=tf.nn.softmax)
        ])
```

Transfer learning



demo

```python
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') > 0.99):
            print("\nReached 99% accuracy so cancelling training!")
            self.model.stop_training = True


callbacks = myCallback()

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(training_images, training_labels, validation_data=[test_images,test_labels],
                    epochs=50, callbacks=[callbacks])

```

[77] ✓ 30m 13.9s

```
Epoch 1/50
1875/1875 [==============================] - 152s 80ms/step - loss: 0.1087 - accuracy: 0.9658 - val_loss: 0.0620 - val_accuracy: 0.9777
Epoch 2/50
1875/1875 [==============================] - 151s 81ms/step - loss: 0.0647 - accuracy: 0.9797 - val_loss: 0.0438 - val_accuracy: 0.9870
Epoch 3/50
1875/1875 [==============================] - 161s 86ms/step - loss: 0.0522 - accuracy: 0.9837 - val_loss: 0.0475 - val_accuracy: 0.9855
Epoch 4/50
1875/1875 [==============================] - 162s 87ms/step - loss: 0.0488 - accuracy: 0.9852 - val_loss: 0.0910 - val_accuracy: 0.9718
Epoch 5/50
1875/1875 [==============================] - 525s 280ms/step - loss: 0.0397 - accuracy: 0.9879 - val_loss: 0.0484 - val_accuracy: 0.9856
Epoch 6/50
1875/1875 [==============================] - 169s 90ms/step - loss: 0.0359 - accuracy: 0.9888 - val_loss: 0.0934 - val_accuracy: 0.9741
Epoch 7/50
1875/1875 [==============================] - 168s 89ms/step - loss: 0.0318 - accuracy: 0.9898 - val_loss: 0.0484 - val_accuracy: 0.9871
Epoch 8/50
1875/1875 [==============================] - 162s 86ms/step - loss: 0.0341 - accuracy: 0.9899 - val_loss: 0.0659 - val_accuracy: 0.9853
Epoch 9/50
1875/1875 [==============================] - ETA: 0s - loss: 0.0275 - accuracy: 0.9915
Reached 99% accuracy so cancelling training!
1875/1875 [==============================] - 164s 88ms/step - loss: 0.0275 - accuracy: 0.9915 - val_loss: 0.0466 - val_accuracy: 0.9882
```

Transfer learning

```
# Import model (weights)
weights_file = "inception_v3.h5"
pre_trained_model = InceptionV3(input_shape=(75, 75, 3),
                                include_top=False, weights=None)
pre_trained_model.load_weights(weights_file)
# Set weights untrainable
for layer in pre_trained_model.layers:
       layer.trainable = False
# Define output layer from pretrained model
last_layer = pre_trained_model.get_layer('mixed7’)
last_output = last_layer.output

# Construct new model by adopting weights from pretrained model
x = layers.Flatten()(last_output)
x = layers.Dense(1024, activation=tf.nn.relu)(x)
x = layers.Dense(10, activation=tf.nn.softmax)(x)
model = Model(pre_trained_model.input, x)
```

Summary

- DNN, CNN
  - data, model, prediction
  - Data format (dimension)
- Transfer Learning
- Use existing model (directly)

Model Construction

```
model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(28, 28, 1)),
        tf.keras.layers.MaxPooling2D(2, 2),
        ...,
        tf.keras.layers.Flatten(),
        ...,
        tf.keras.layers.Dense(10, activation='softmax')
])
        tf.keras.layers.Dense(1, activation='sigmoid')
```

1. Define Input
2. Construct Model (layer)
3. Define Output

```
# Import model (weights)
...
# Construct new model by adopting weights from pretrained model
x = layers.Flatten()(last_output)
x = ...(x)
x = layers.Dense(10, activation='softmax')(x)

model = Model(pre_trained_model.input, x)
```

Model Training

```
model.compile(loss='mean_squared_error', optimizer=tf.keras.optimizers.Adam(0.1))
history = model.fit(times, vt, epochs=100, verbose=False)
```

```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit(training_images, training_labels, epochs=20, callbacks=[callbacks])
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
       metrics=['accuracy'])
history = model.fit(training_images, training_labels,
       validation_data=[test_images,test_labels], epochs=50, callbacks=[callbacks])
```

Reference

- Books:
  - Laurence Moroney, AI and Machine Learning for Coders, 2020 (code examples available: https://github.com/lmoroney/tfbook)
  - Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd, 2019 (code examples available: https://github.com/ageron/handson-ml2)
- Online courses:
  - Laurence Moroney, DeepLearning.AI TensorFlow Developer Professional Certificate, Coursera.org (TensorFlow practice)
  - Andrew Ng, Deep Learning Specialization, Coursera.org (Deep learning theory)
- Suggest practices:
  - Titanic survival prediction: https://www.kaggle.com/c/titanic/data
  - Cat vs. Dog