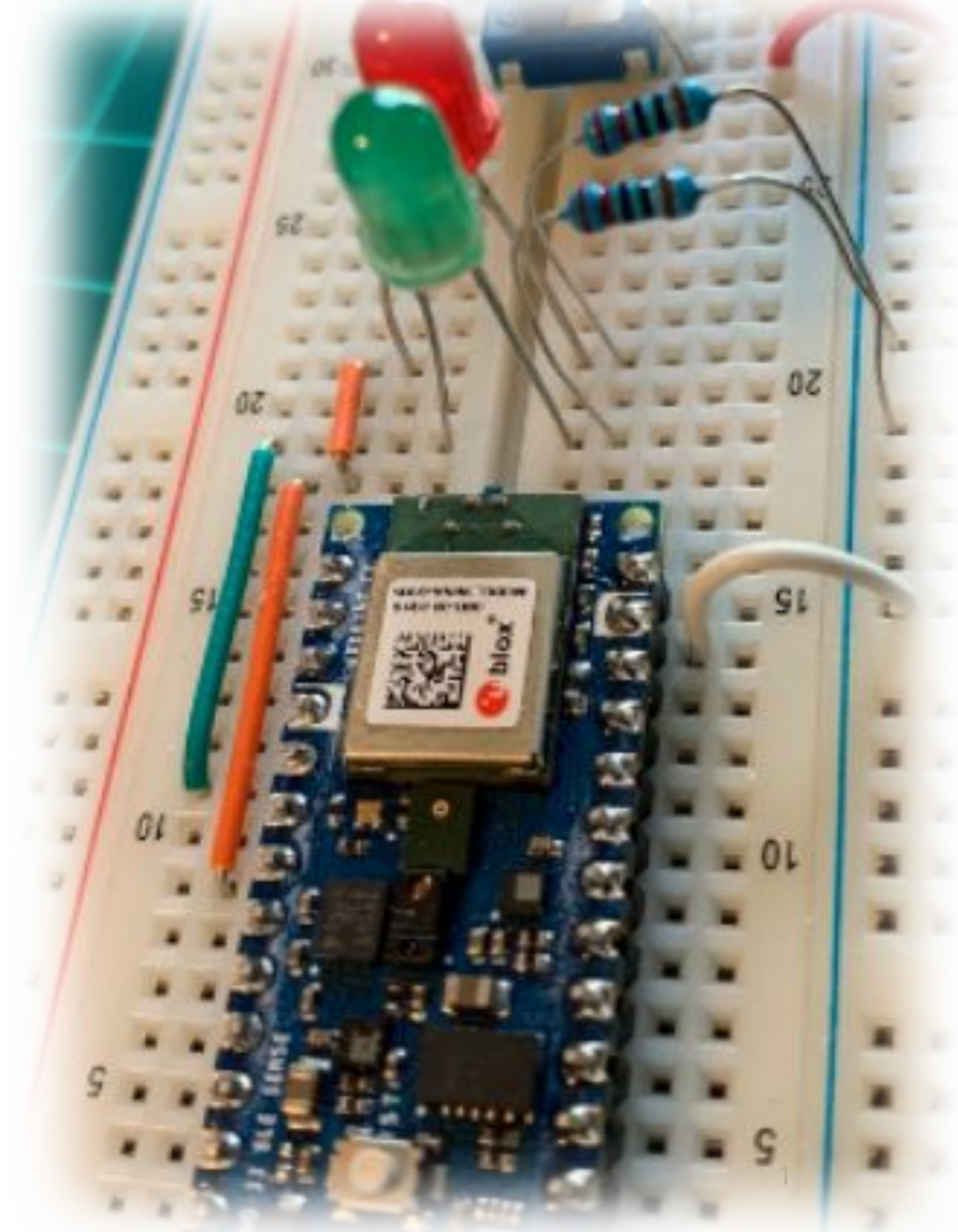
IESTI01 – TinyML

Embedded Machine Learning

6. The Building Blocks of Deep Learning – Part A
   - Regression

Prof. Marcelo Rovai

UNIFEI

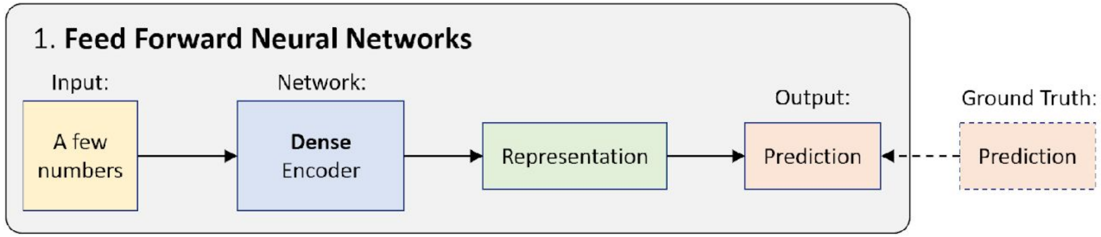


Machine Learning

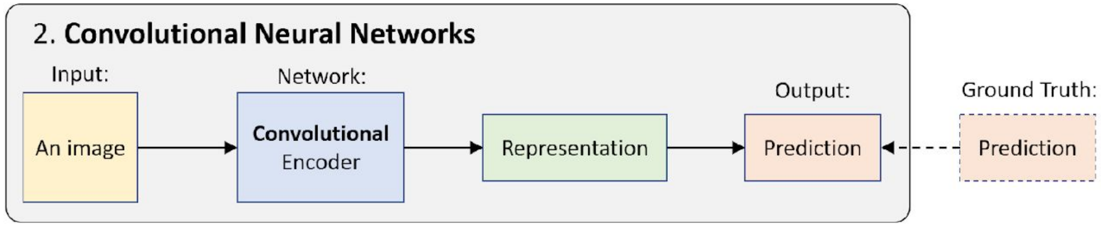
Models

Machine Learning Types and Arquitectures

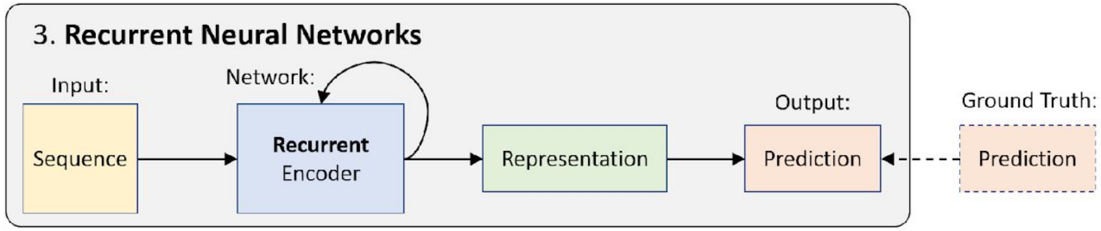
Supervised Learning


1. Feed Forward Neural Networks

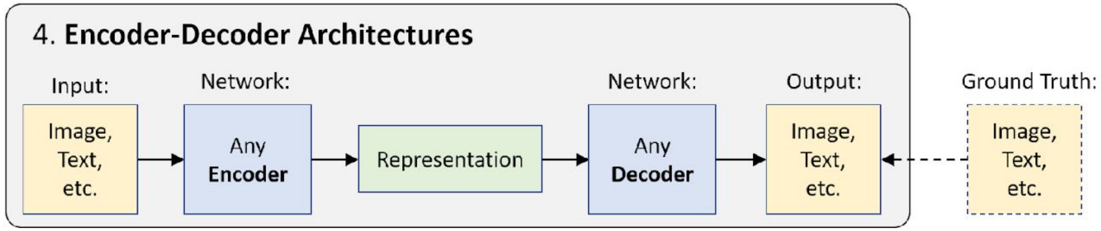
Input: A few numbers → Network: Dense Encoder → Representation → Output: Prediction ← Ground Truth: Prediction


2. Convolutional Neural Networks

Input: An image → Network: Convolutional Encoder → Representation → Output: Prediction ← Ground Truth: Prediction

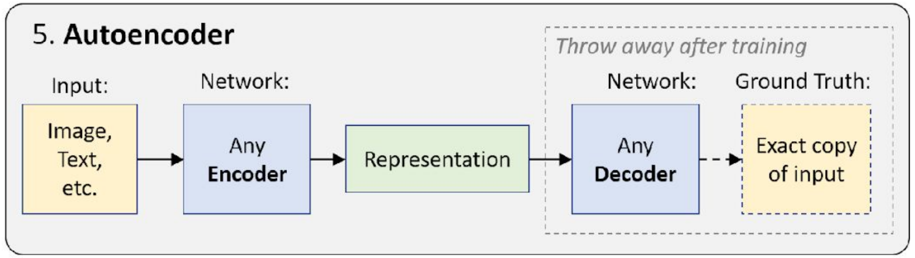

3. Recurrent Neural Networks

Input: Sequence → Network: Recurrent Encoder → Representation → Output: Prediction ← Ground Truth: Prediction


4. Encoder-Decoder Architectures

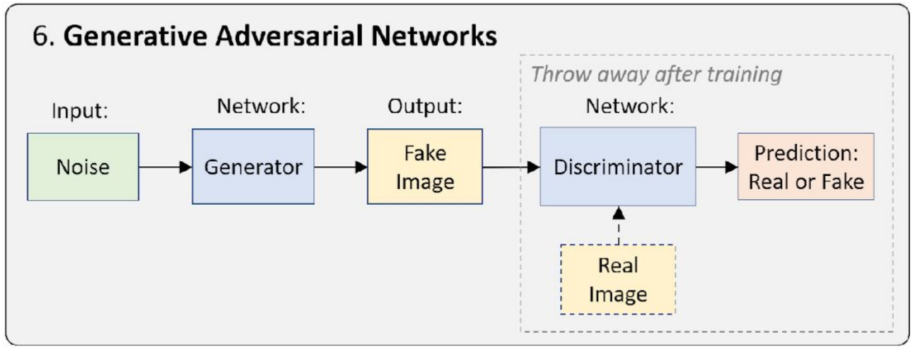
Input: Image, Text, etc. → Network: Any Encoder → Representation → Network: Any Decoder → Output: Image, Text, etc. ← Ground Truth: Image, Text, etc.

Unsupervised Learning


5. Autoencoder

Input: Image, Text, etc. → Network: Any Encoder → Representation → Network: Any Decoder → Ground Truth: Exact copy of input

Throw away after training

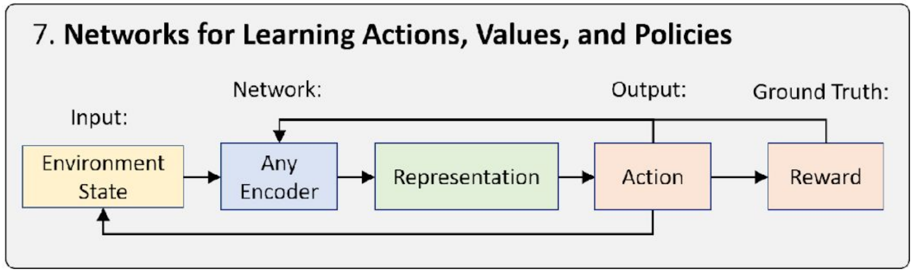

6. Generative Adversarial Networks

Input: Noise → Network: Generator → Output: Fake Image → Network: Discriminator → Prediction: Real or Fake

Real Image → Discriminator

Throw away after training

Reinforcement Learning


7. Networks for Learning Actions, Values, and Policies

Input: Environment State → Network: Any Encoder → Representation → Output: Action → Ground Truth: Reward

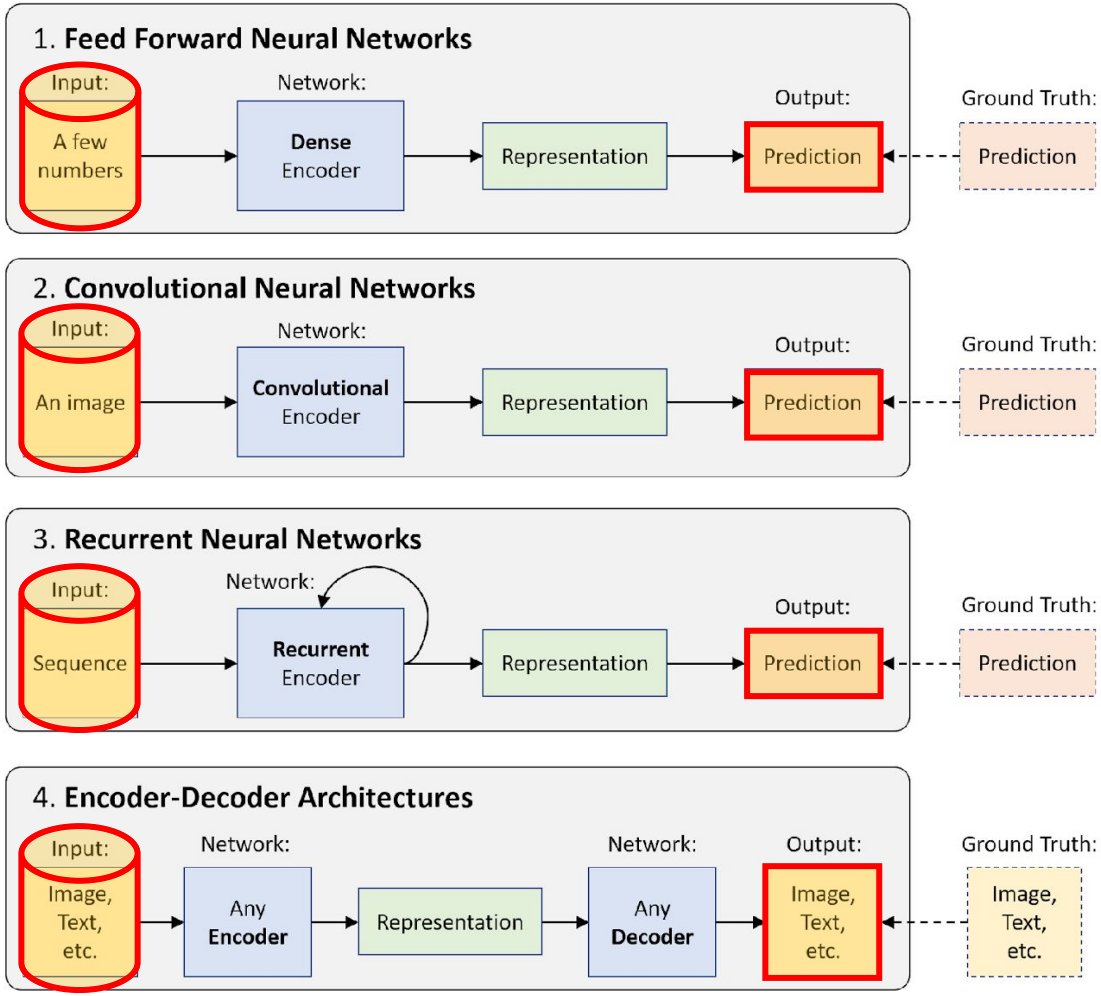
Deep Learning Basics: An introductory lecture for MIT course 6.S094 by Prof. Lex Fridman

Machine Learning

Supervised Learning

Training

Inference



1. Feed Forward Neural Networks

Input: A few numbers → Network: **Dense** Encoder → Representation → Output: Prediction ← Ground Truth: Prediction

2. Convolutional Neural Networks

Input: An image → Network: **Convolutional** Encoder → Representation → Output: Prediction ← Ground Truth: Prediction

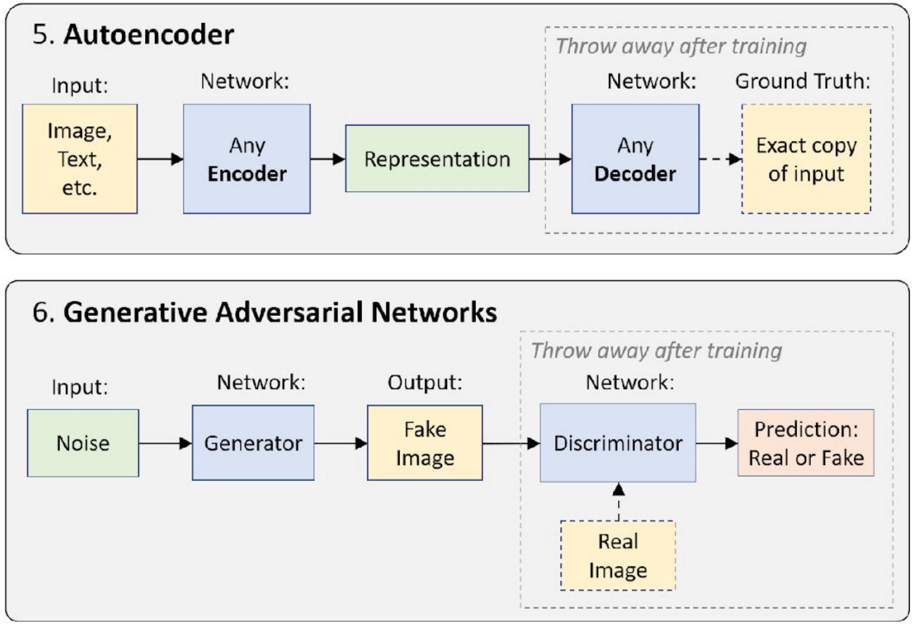
3. Recurrent Neural Networks

Input: Sequence → Network: **Recurrent** Encoder → Representation → Output: Prediction ← Ground Truth: Prediction

4. Encoder-Decoder Architectures

Input: Image, Text, etc. → Network: Any **Encoder** → Representation → Network: Any **Decoder** → Output: Image, Text, etc. ← Ground Truth: Image, Text, etc.

Unsupervised Learning



5. Autoencoder

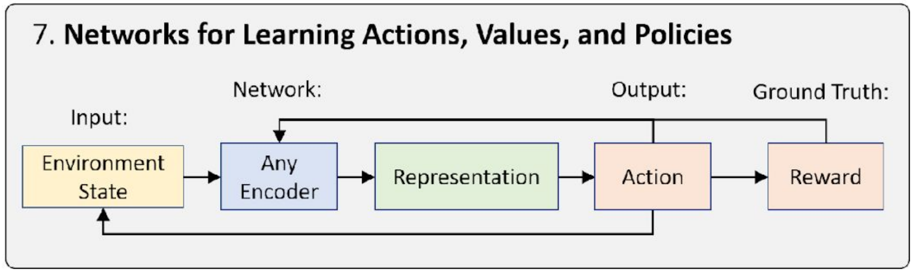
Input: Image, Text, etc. → Network: Any **Encoder** → Representation → *Throw away after training* Network: Any **Decoder** → Ground Truth: Exact copy of input

6. Generative Adversarial Networks

Input: Noise → Network: Generator → Output: Fake Image → *Throw away after training* Network: Discriminator → Prediction: Real or Fake

Real Image → Discriminator

Reinforcement Learning

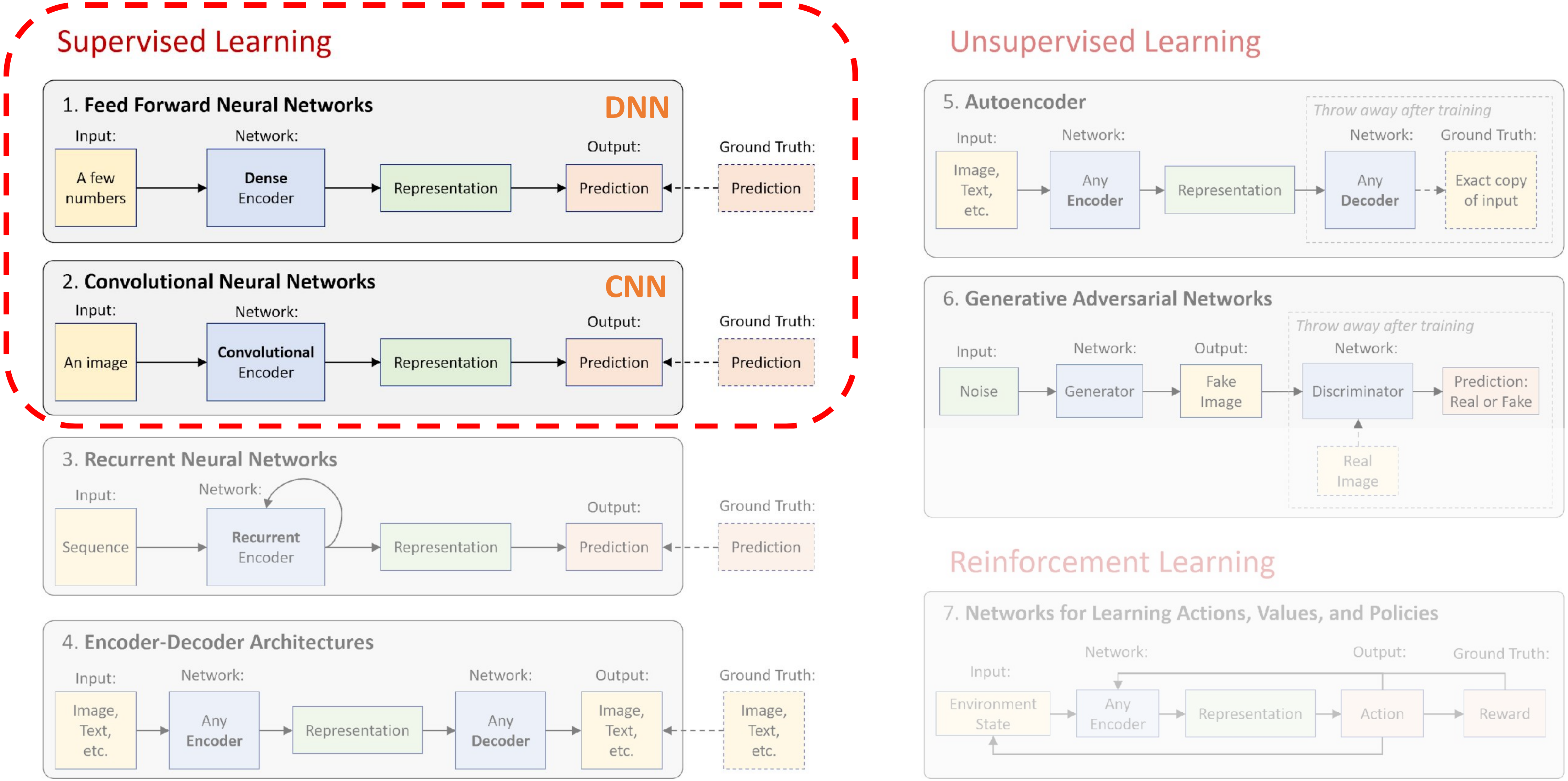


7. Networks for Learning Actions, Values, and Policies

Input: Environment State → Network: Any Encoder → Representation → Output: Action → Ground Truth: Reward

Deep Learning Basics: An introductory lecture for MIT course 6.S094 by Prof. Lex Fridman

Machine Learning



Supervised Learning

1. Feed Forward Neural Networks DNN

Input: A few numbers → Network: Dense Encoder → Representation → Output: Prediction ← Ground Truth: Prediction

2. Convolutional Neural Networks CNN

Input: An image → Network: Convolutional Encoder → Representation → Output: Prediction ← Ground Truth: Prediction

3. Recurrent Neural Networks

Input: Sequence → Network: Recurrent Encoder → Representation → Output: Prediction ← Ground Truth: Prediction

4. Encoder-Decoder Architectures

Input: Image, Text, etc. → Network: Any Encoder → Representation → Network: Any Decoder → Output: Image, Text, etc. ← Ground Truth: Image, Text, etc.

Unsupervised Learning

5. Autoencoder

Input: Image, Text, etc. → Network: Any Encoder → Representation → Network: Any Decoder → Ground Truth: Exact copy of input

Throw away after training

6. Generative Adversarial Networks

Input: Noise → Network: Generator → Output: Fake Image → Network: Discriminator → Prediction: Real or Fake

Real Image

Throw away after training

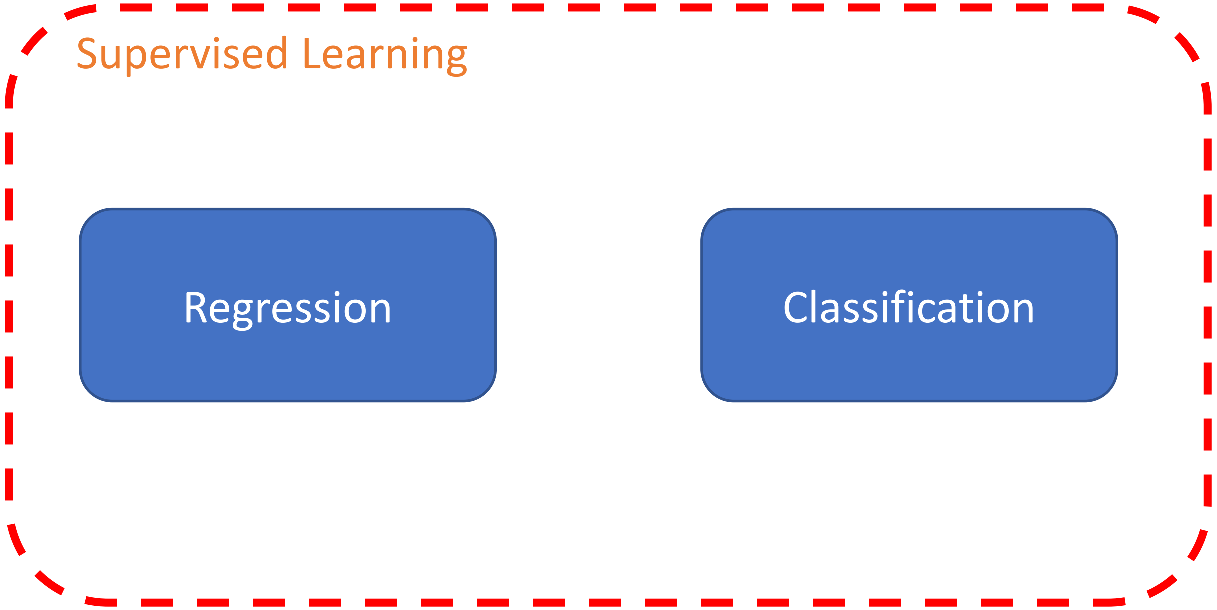
Reinforcement Learning

7. Networks for Learning Actions, Values, and Policies

Input: Environment State → Network: Any Encoder → Representation → Output: Action → Ground Truth: Reward

Deep Learning Basics: An introductory lecture for MIT course 6.S094 by Prof. Lex Fridman

Tiny Machine Learning



Supervised Learning

Regression

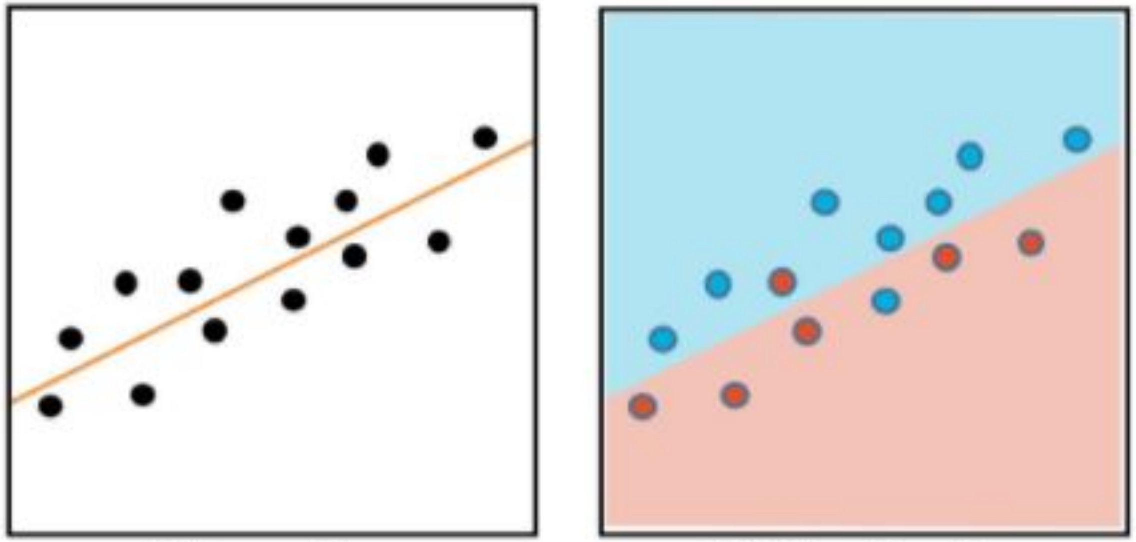
Classification

Tiny Machine Learning

Supervised Learning

Regression

Classification

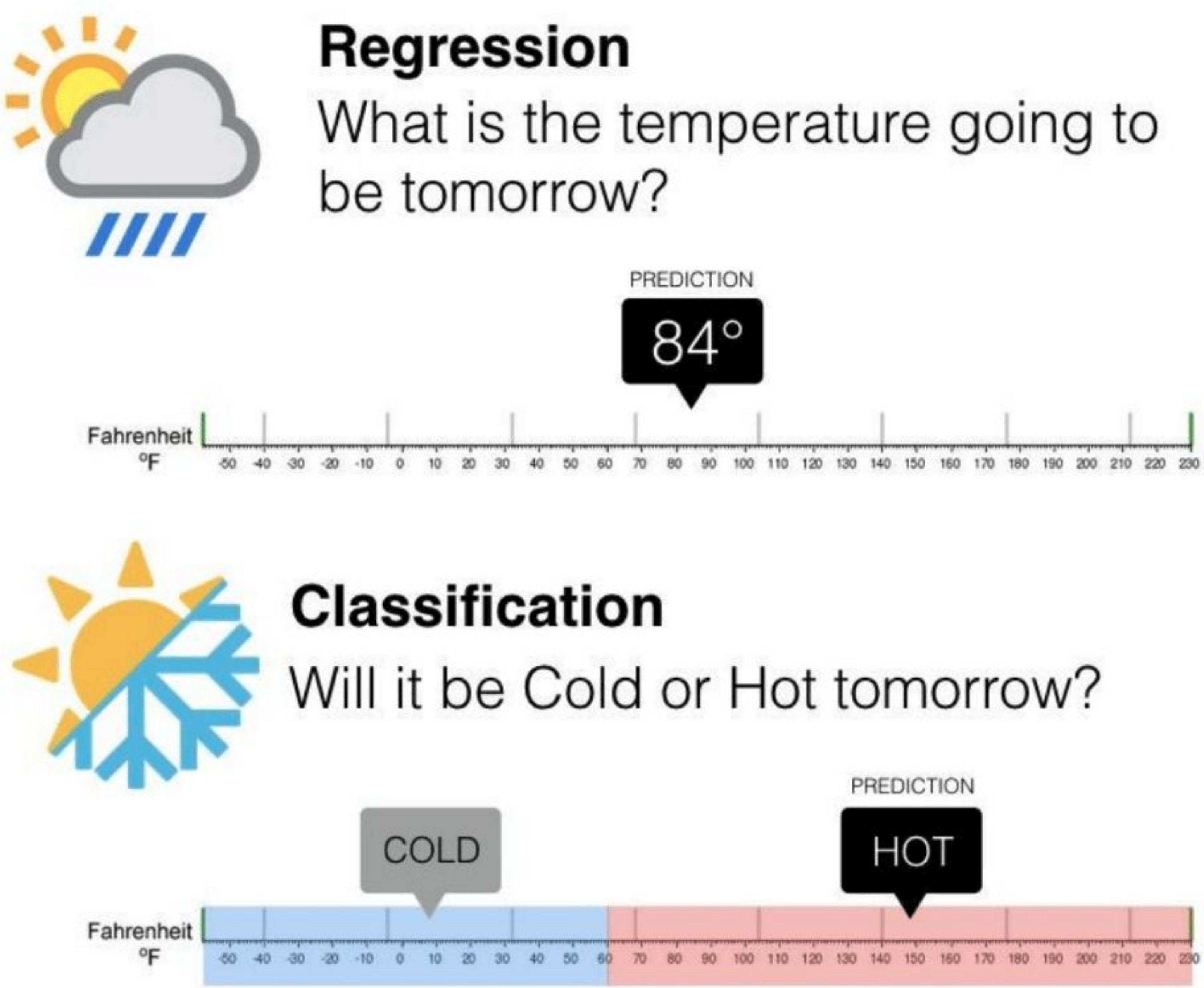


a) Regression

b) Classification

Regression

Classification



Regression

What is the temperature going to be tomorrow?

Classification

Will it be Cold or Hot tomorrow?

Machine Learning

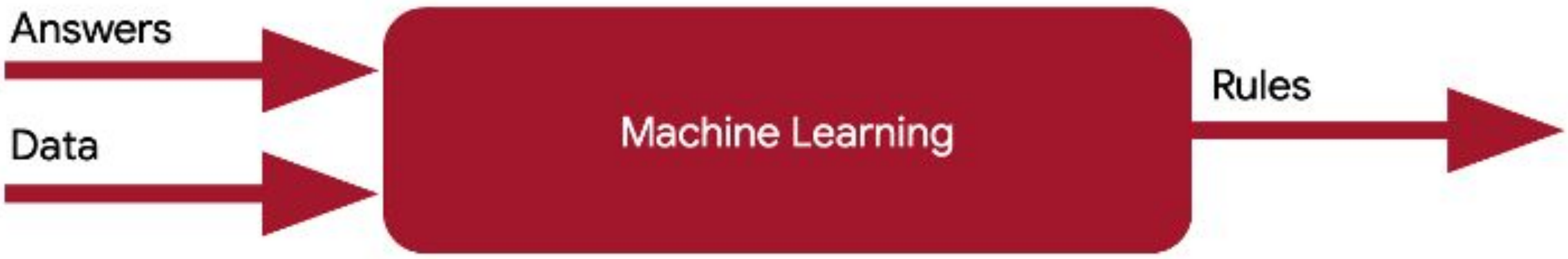
Supervised models - Regression

X ➡ -1, 0, 1, 2, 3, 4

Y ➡ -3, -1, 1, 3, 5, 7

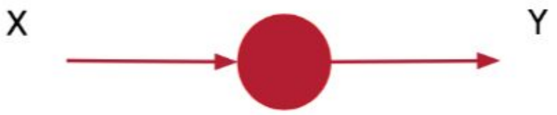
Y

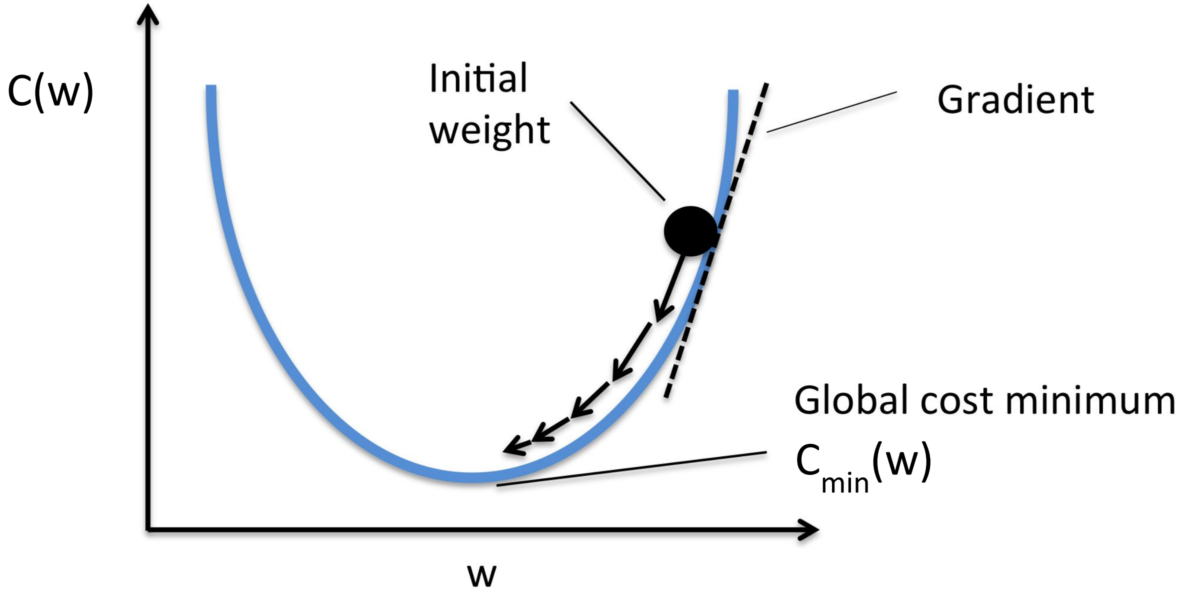
X



Model
(w, b)

X	Y
-1	-3
0	-1
1	1
2	3
3	5
4	7



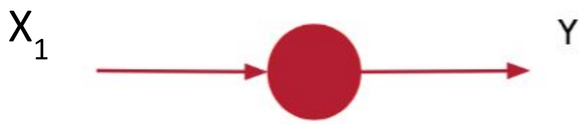


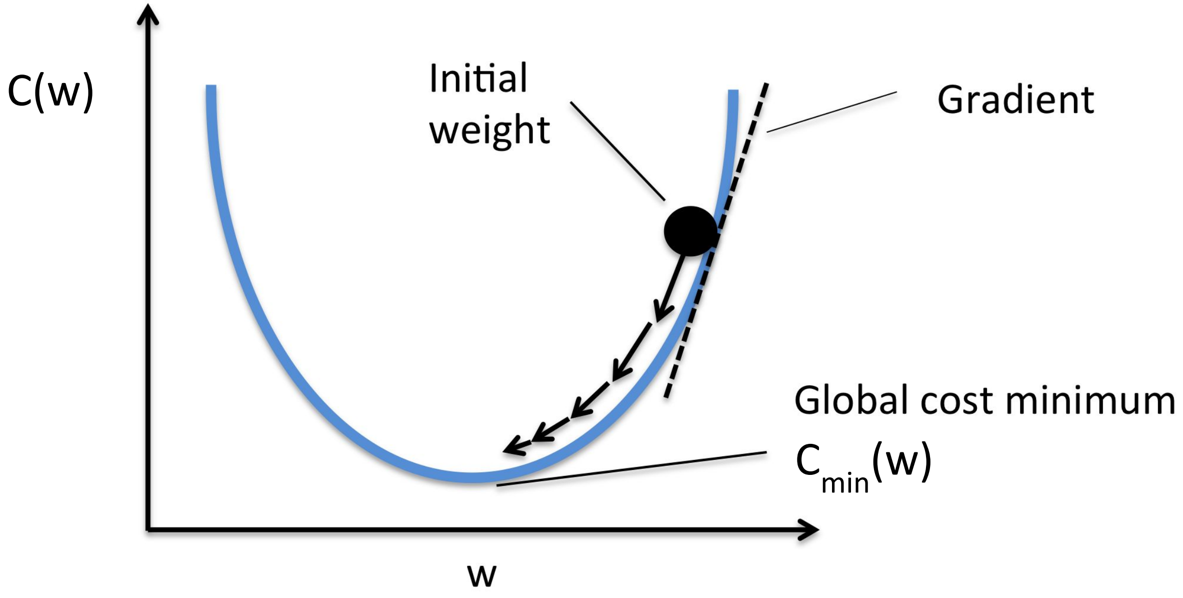
$$Y = w*X + b$$

Cost Function

X_1	Y
-1	-3
0	-1
1	1
2	3
3	5
4	7

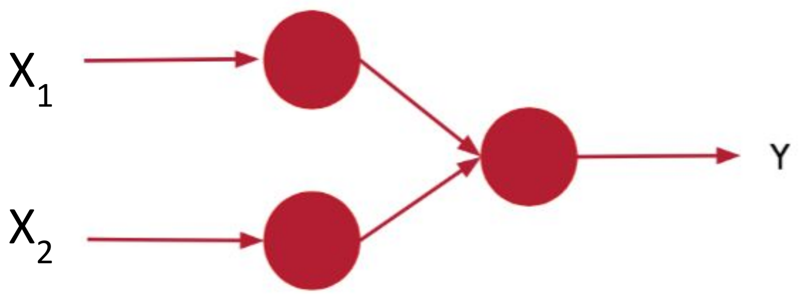
$$Y = w_1 * X_1 + b_0$$

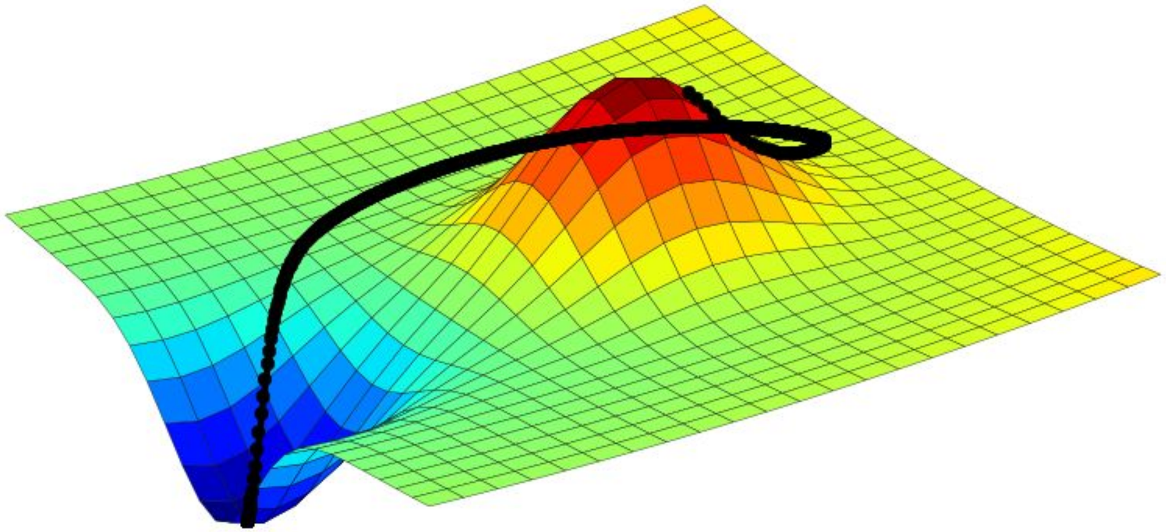




Cost Function

X_1	X_2	Y
-1	-8	-8
0	1	0
1	3	7
2	7	1
3	0	2
4	2	3

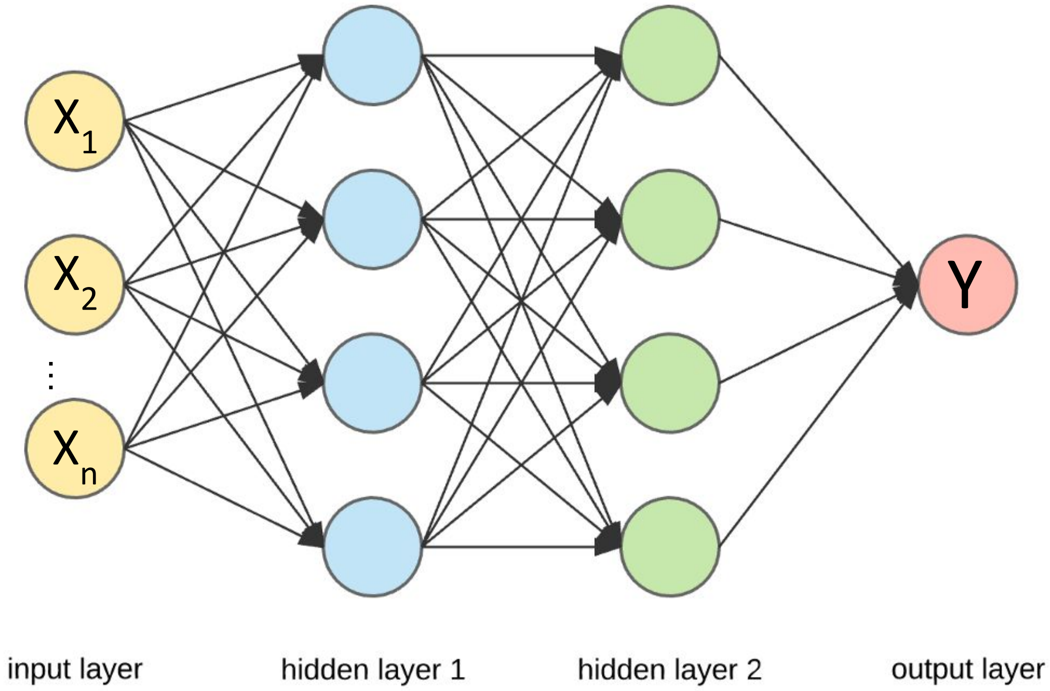




$$Y = w_1 * X_1 + w_2 * X_2 + b_{\cup}$$

Cost Function

X_1	X_2	…	X_n	Y
-1	-8		-81	-8
0	1		10	0
1	3		3	7
2	7		7	1
3	0		0	2
4	2		-7	3



$$Y = w_1 * X_1 + w_2 * X_2 + \dots + w_n * X_n + b_0$$

Regression using DNN with TF2

Code Time!

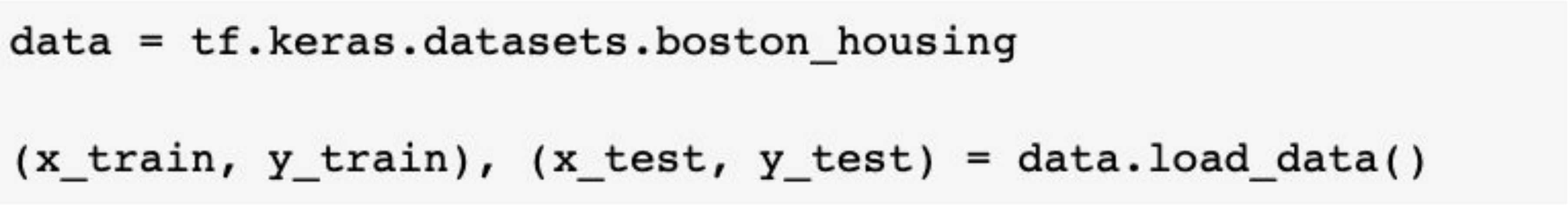
TF_Boston_Housing_Regression.ipynb

Machine Learning

Workflow

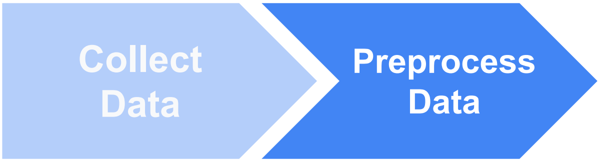
Machine Learning Workflow

Collect Data



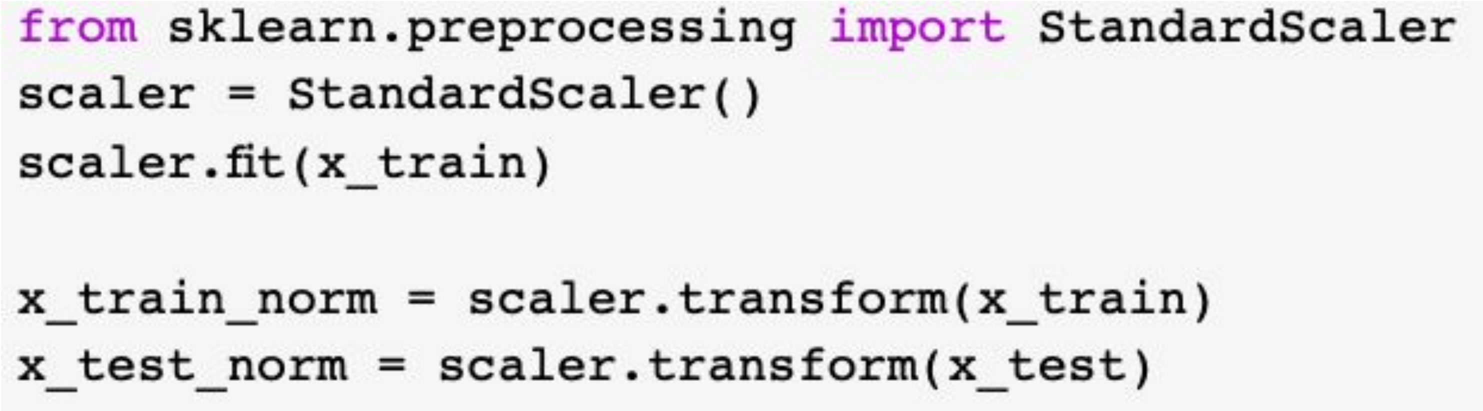
```
data = tf.keras.datasets.boston_housing

(x_train, y_train), (x_test, y_test) = data.load_data()
```

Machine Learning Workflow



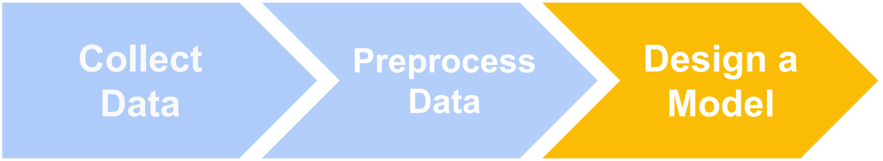
Collect Data

Preprocess Data



```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)

x_train_norm = scaler.transform(x_train)
x_test_norm = scaler.transform(x_test)
```

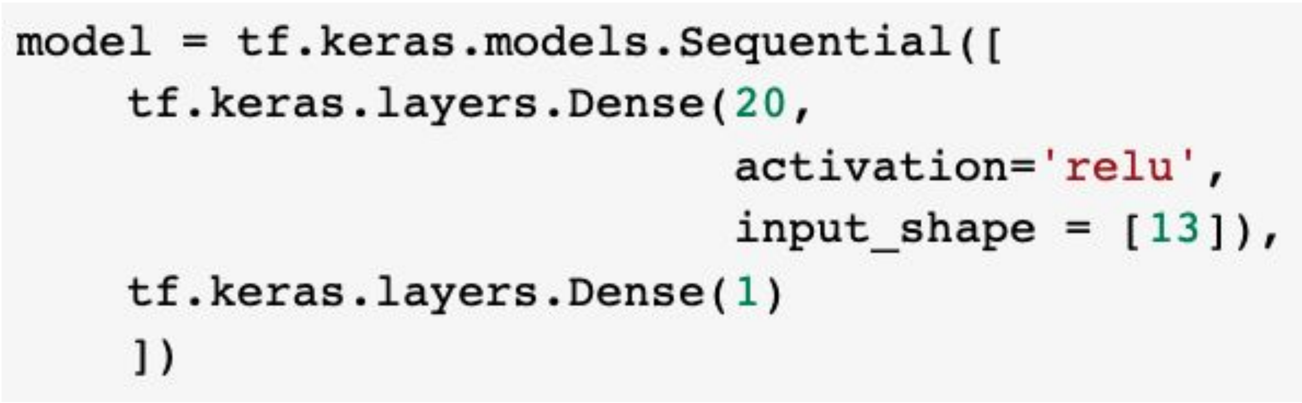
Machine Learning Workflow



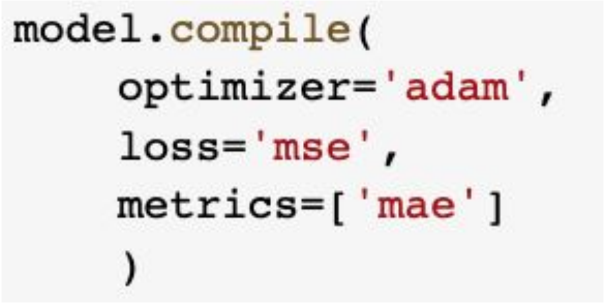
Collect Data

Preprocess Data

Design a Model

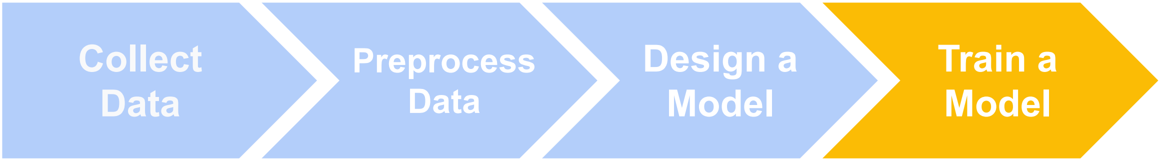


```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(20,
                          activation='relu',
                          input_shape = [13]),
    tf.keras.layers.Dense(1)
    ])
```



```
model.compile(
    optimizer='adam',
    loss='mse',
    metrics=['mae']
    )
```

Machine Learning Workflow

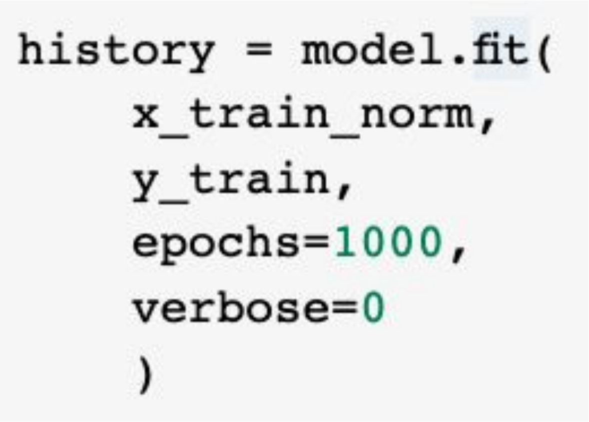


Collect Data

Preprocess Data

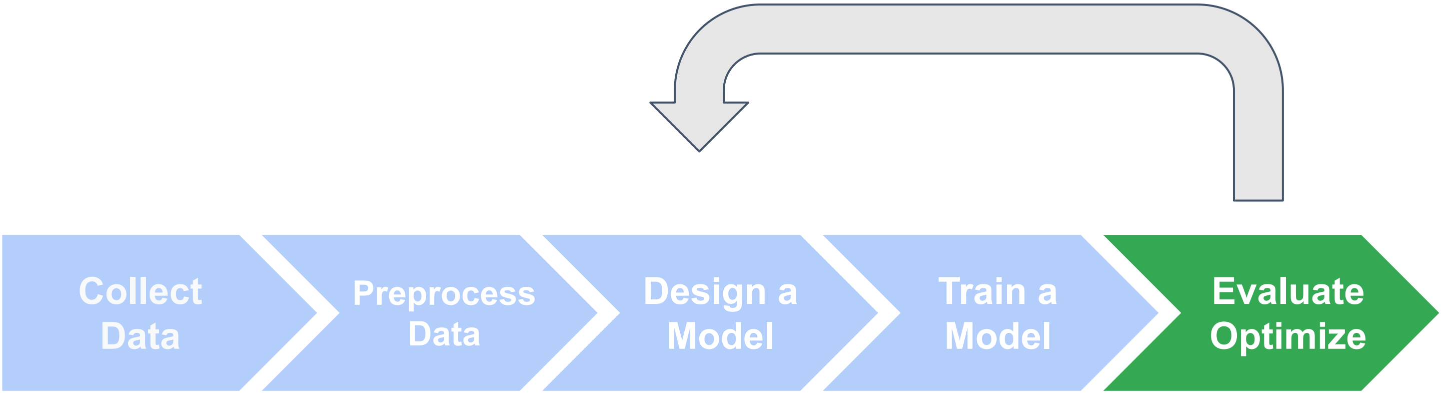
Design a Model

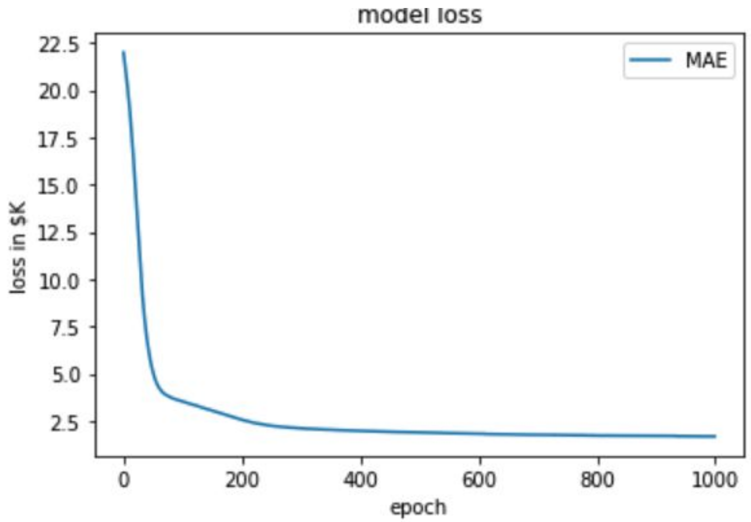
Train a Model

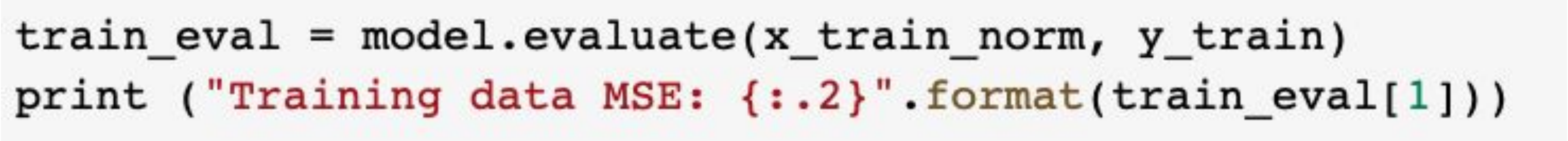


```
history = model.fit(
      x_train_norm,
      y_train,
      epochs=1000,
      verbose=0
      )
```

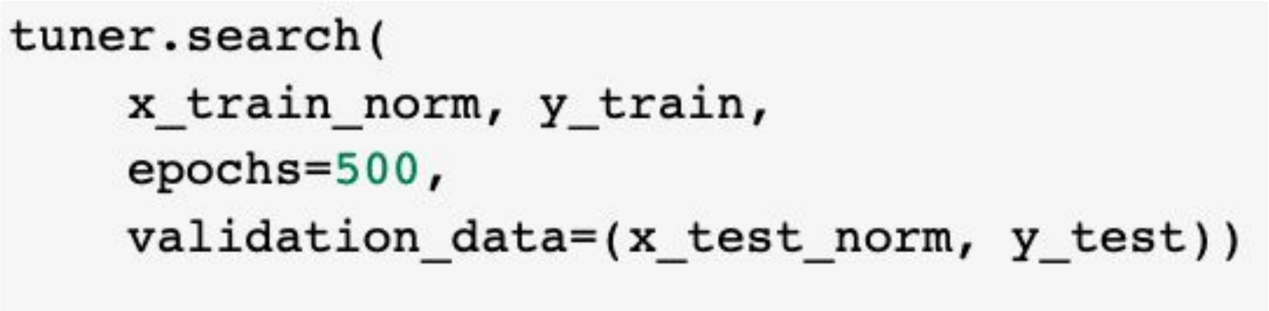
Machine Learning Workflow





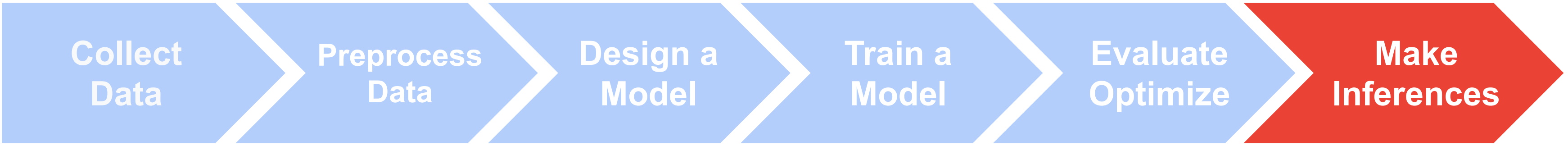


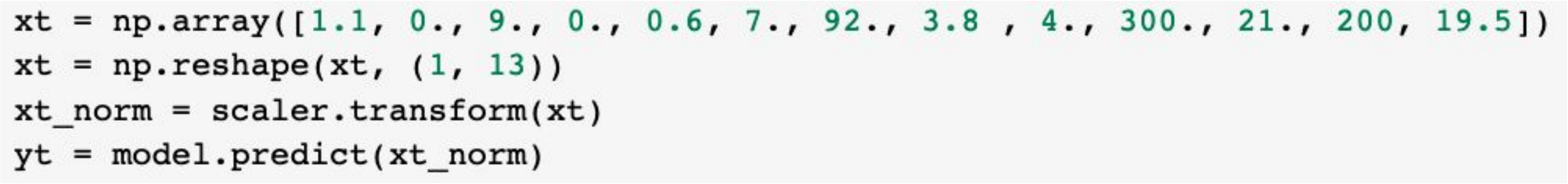
```
train_eval = model.evaluate(x_train_norm, y_train)
print ("Training data MSE: {:.2}".format(train_eval[1]))
```



```
tuner.search(
    x_train_norm, y_train,
    epochs=500,
    validation_data=(x_test_norm, y_test))
```

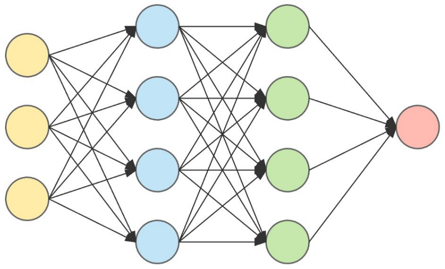
Machine Learning Workflow

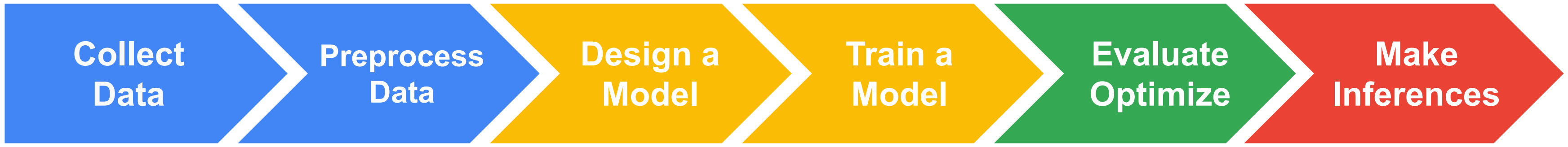




```
xt = np.array([1.1, 0., 9., 0., 0.6, 7., 92., 3.8 , 4., 300., 21., 200, 19.5])
xt = np.reshape(xt, (1, 13))
xt_norm = scaler.transform(xt)
yt = model.predict(xt_norm)
```

Machine Learning Workflow





Collect Data

Preprocess Data

Design a Model

Train a Model

Evaluate Optimize

Make Inferences

Reading Material

Main references

- Harvard School of Engineering and Applied Sciences - CS249r: Tiny Machine Learning
- Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
- Introduction to Embedded Machine Learning (Coursera)
- Text Book: "TinyML" by Pete Warden, Daniel Situnayake

I want to thank Shawn Hymel and Edge Impulse, Pete Warden and Laurence Moroney from Google, and especially Harvard professor Vijay Janapa Reddi, Ph.D. student Brian Plancher and their staff for preparing the excellent material on TinyML that is the basis of this course at UNIFEI.

The IESTI01 course is part of the TinyML4D, an initiative to make TinyML education available to everyone globally.

Thanks

And stay safe!

UNIFEI