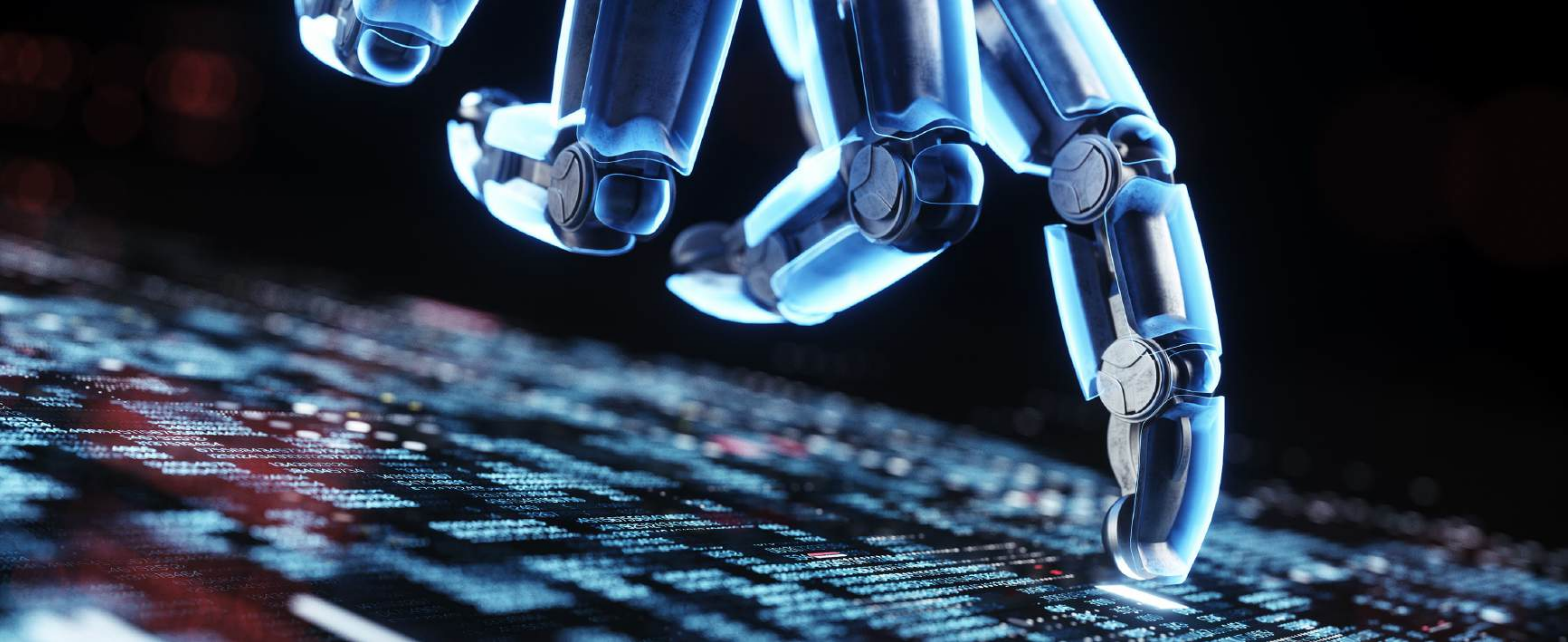


WALC 2024
Online
11 al 15 de Noviembre
27º
Taller sobre Tecnologías de Redes Internet para América Latina y el Caribe
Fundación EsLaRed
Organización General

Pontificia Universidad
JAVERIANA
Bogotá
Facultad de Ingeniería

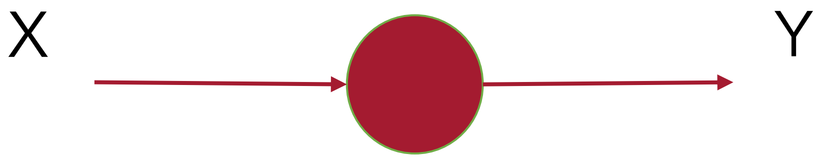
Machine Learning Fundamentals

Workshop para América Latina y el Caribe (WALC)
Track 3 – Inteligencia Artificial Aplicada
November 12, 2024

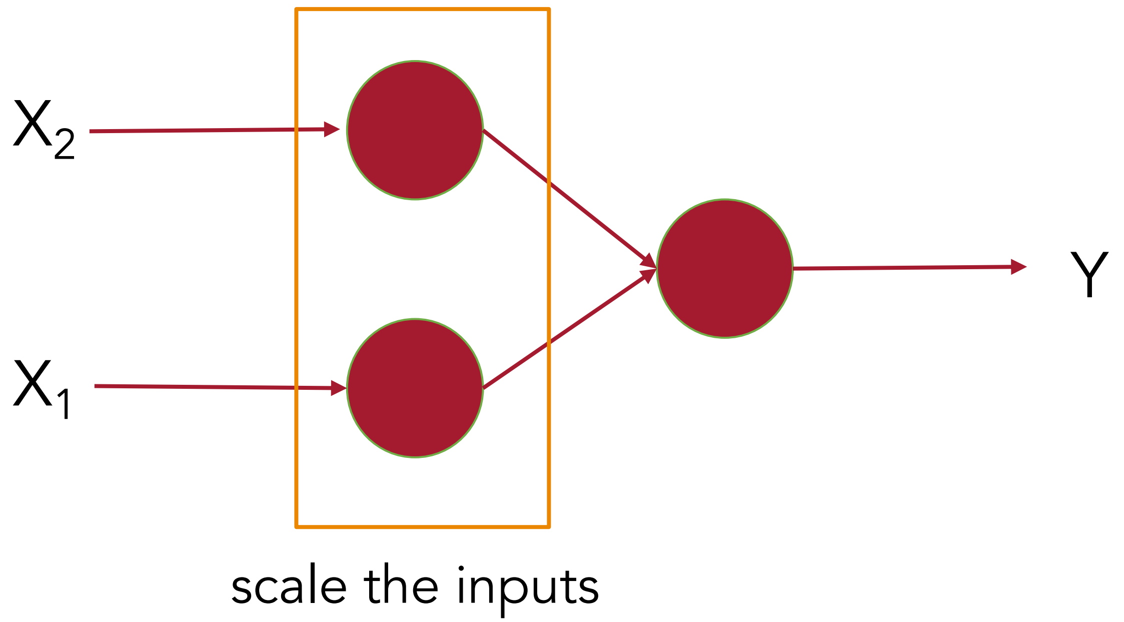
Agenda

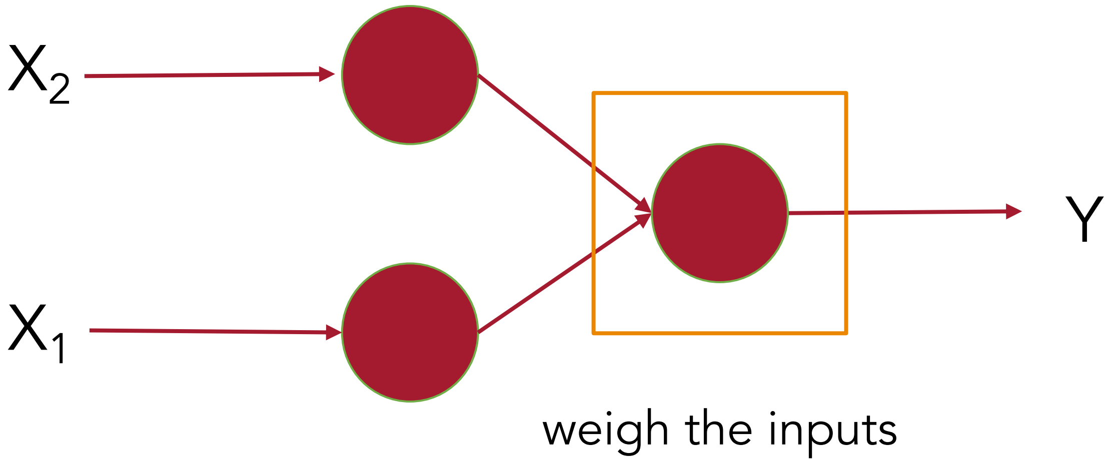
- **Parte 1:**
  - Introducción
  - El paradigma de ML
  - Exploración de la función de pérdida y costo
  - Redes neuronales artificiales
- **Parte 2:**
  - Dense Neural Networks – Regresión
  - Dense Neural Networks – Clasificación
  - Métricas de ML

…a more interesting regression application



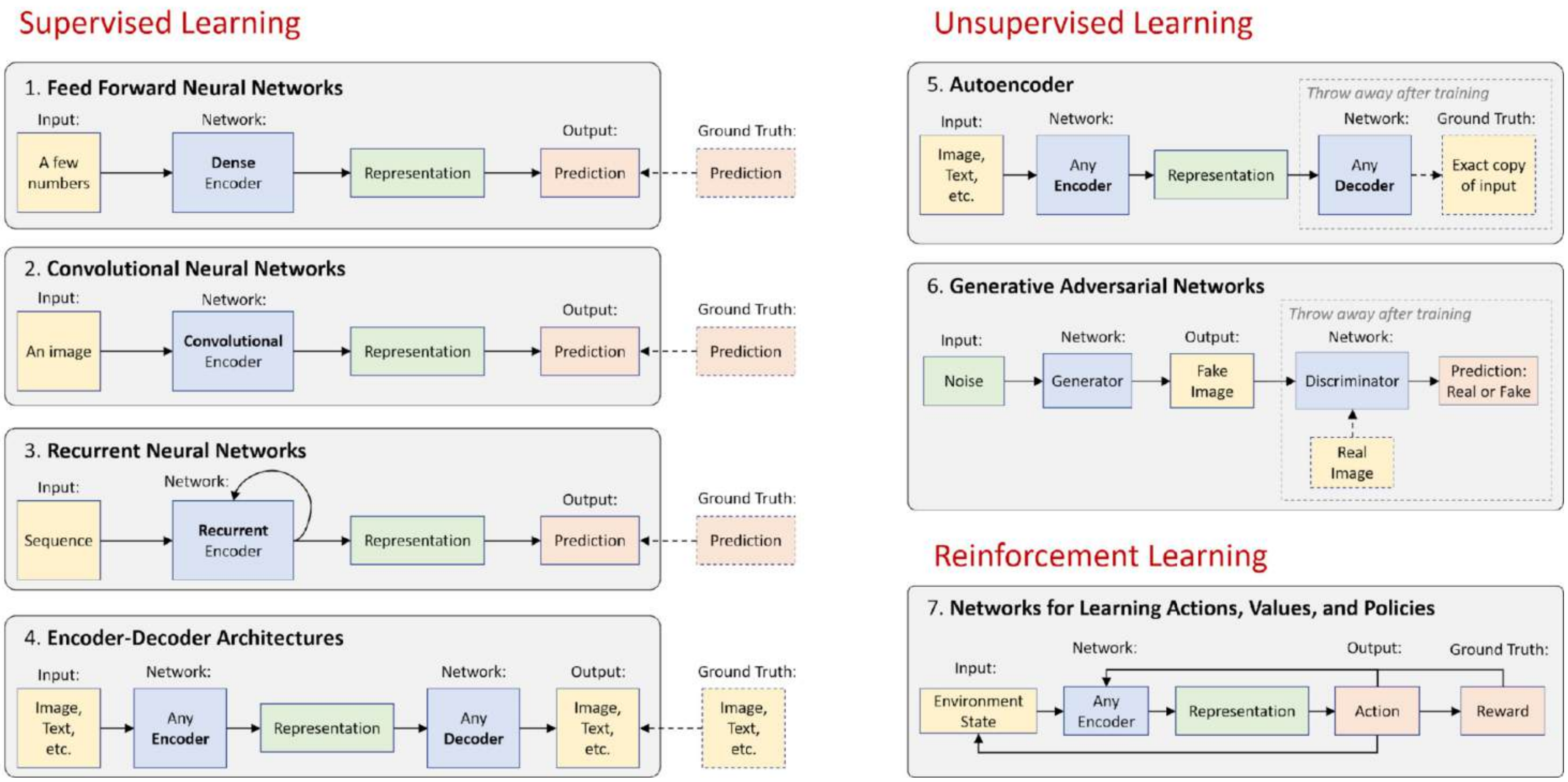
What about more than one input?


X_2
X_1
Y
scale the inputs



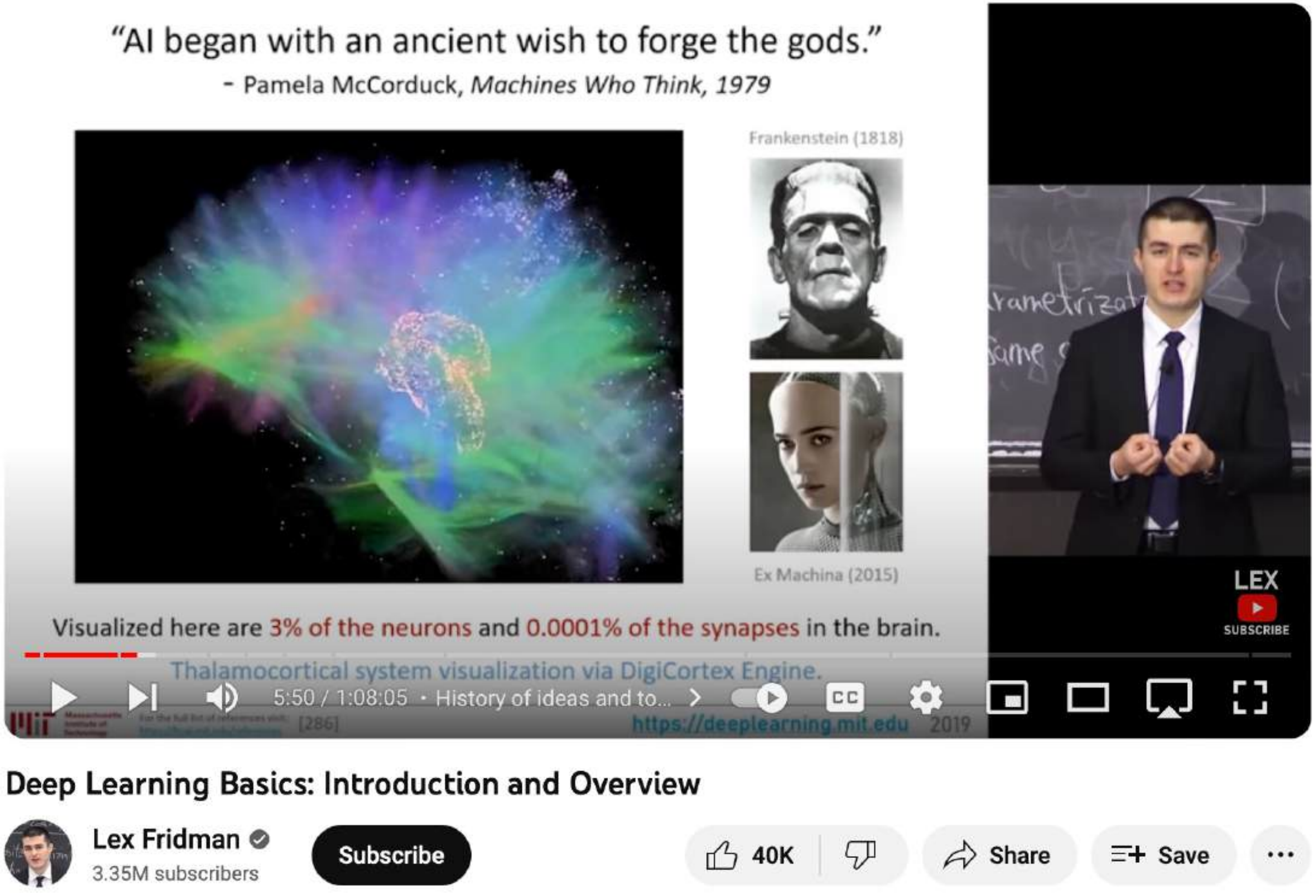
More inputs?

Machine Learning Types and Architectures



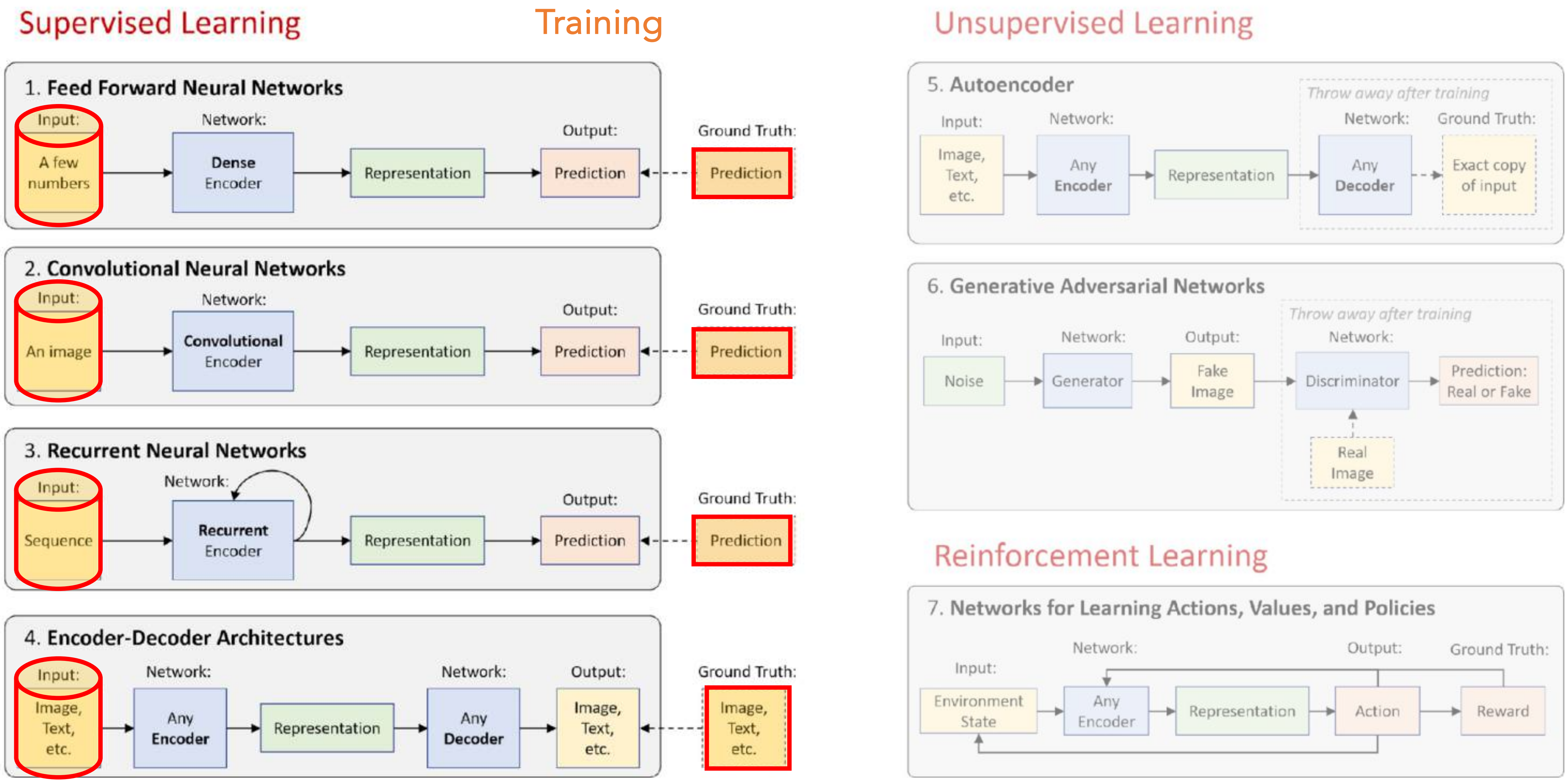
[Deep Learning Basics: An introductory lecture for MIT course 6.S094 by Prof. Lex Fridman]

Machine Learning Types and Architectures

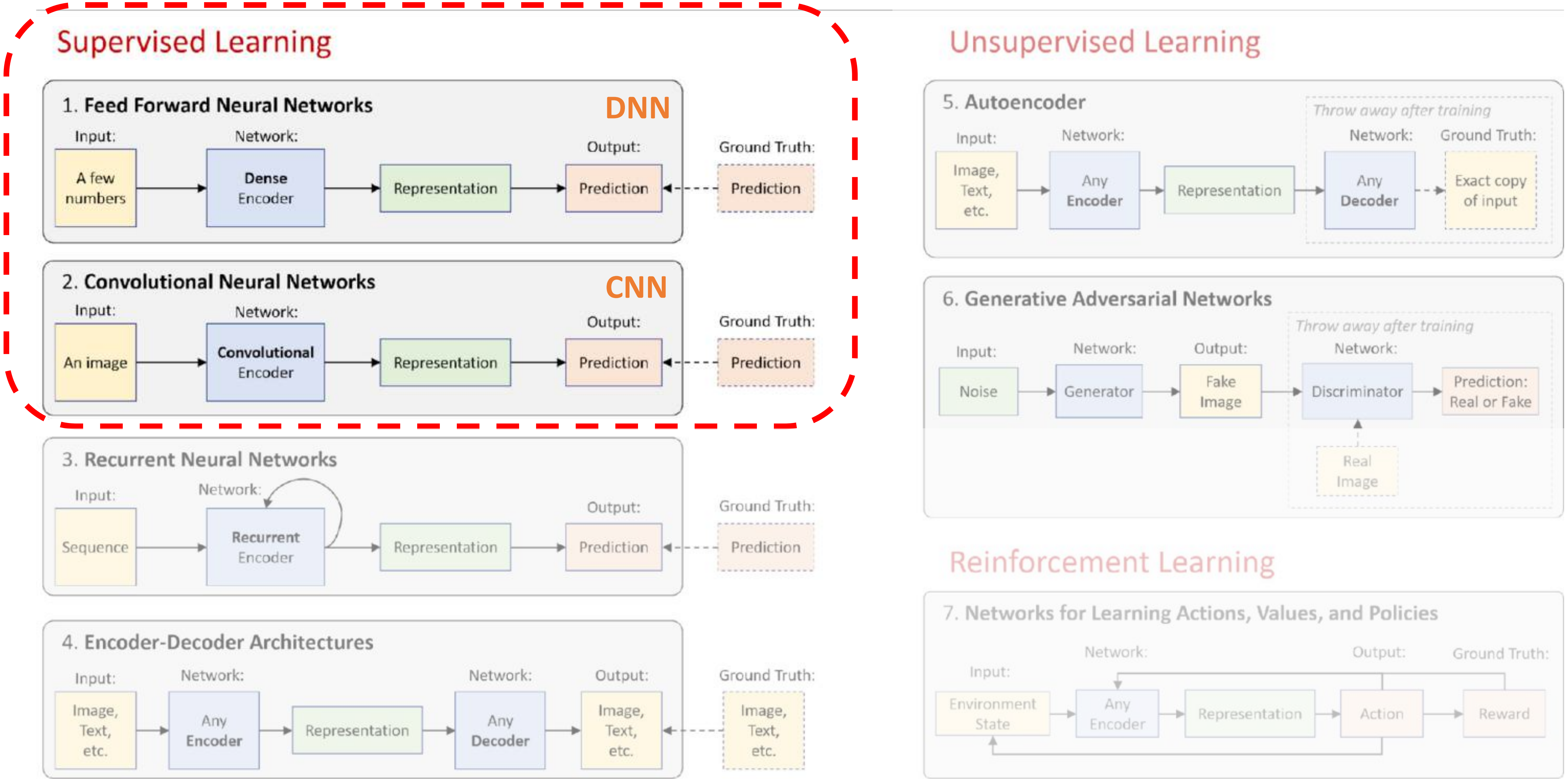


[Deep Learning Basics: An introductory lecture for MIT course 6.S094 by Prof. Lex Fridman]()

Machine Learning



Machine Learning



Tiny Machine Learning

Supervised Learning

Regression

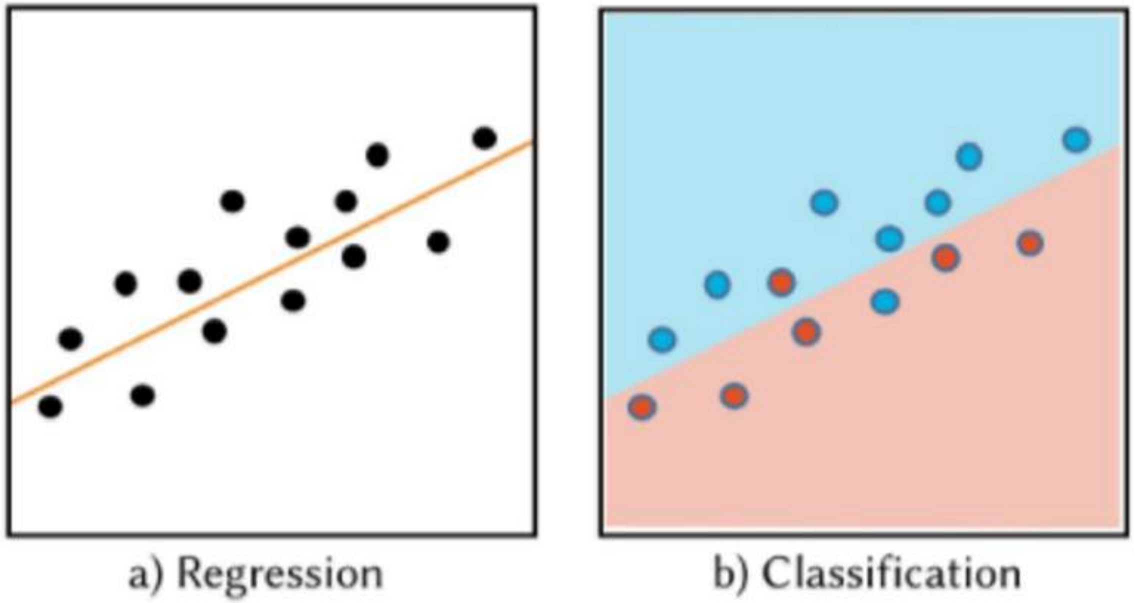
Classification

Tiny Machine Learning

Supervised Learning

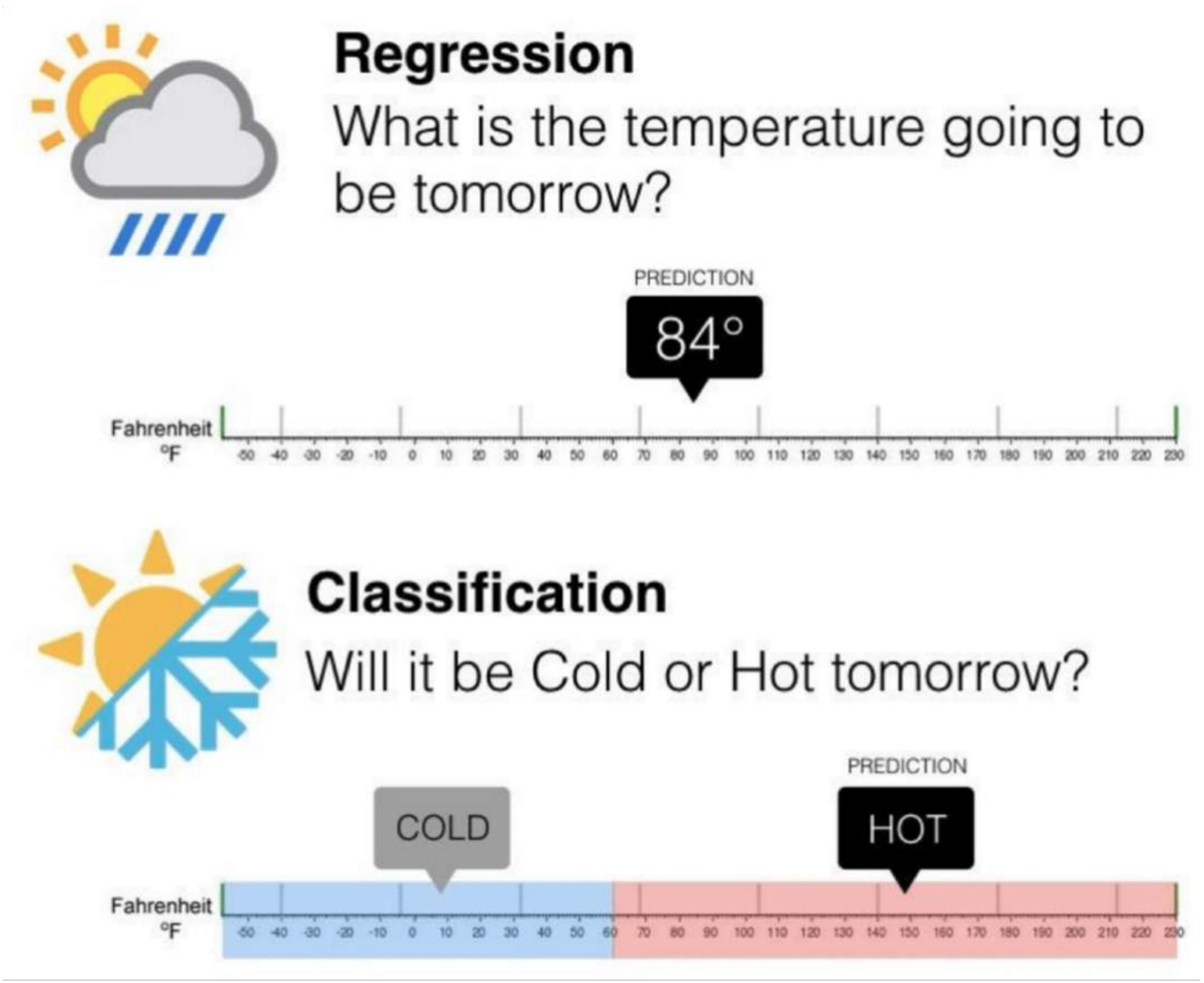
Regression

Classification



Regression

Classification

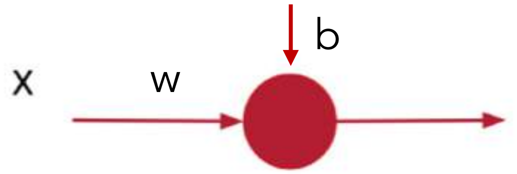


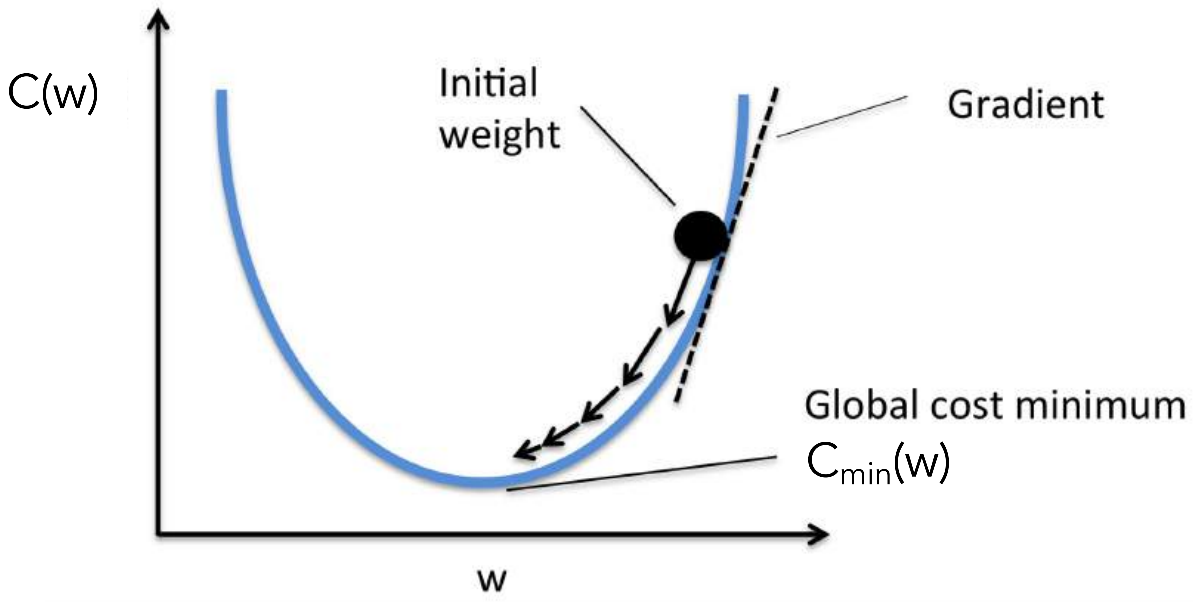
X ➡ -1, 0, 1, 2, 3, 4

Y ➡ -3, -1, 1, 3, 5, 7

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

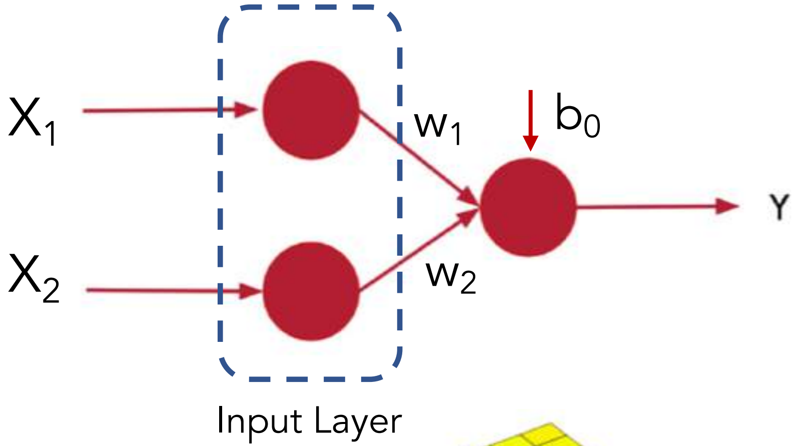
$$Y = wX + b$$

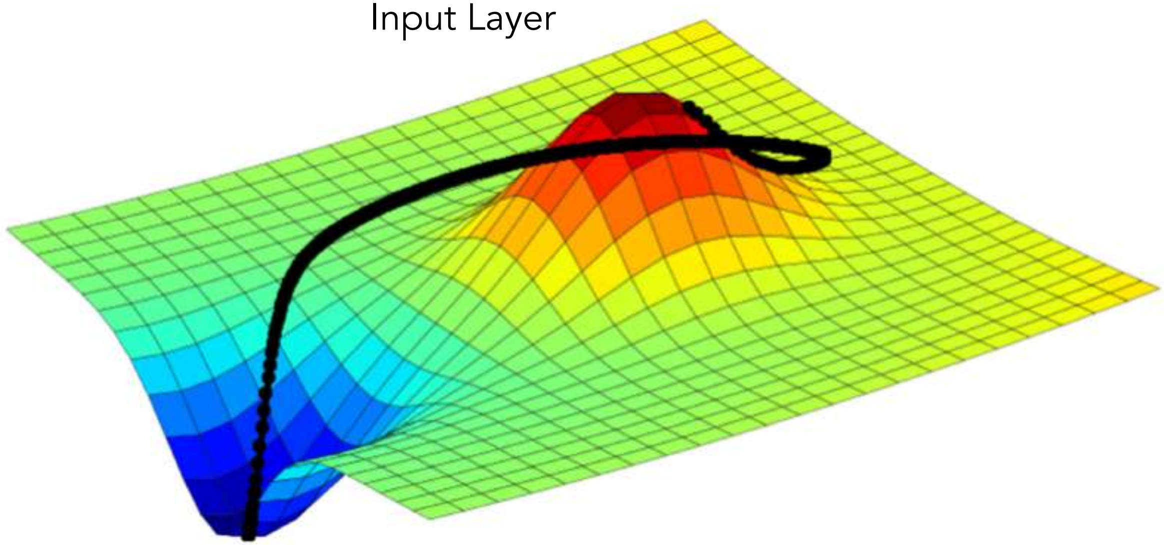




Cost Function

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

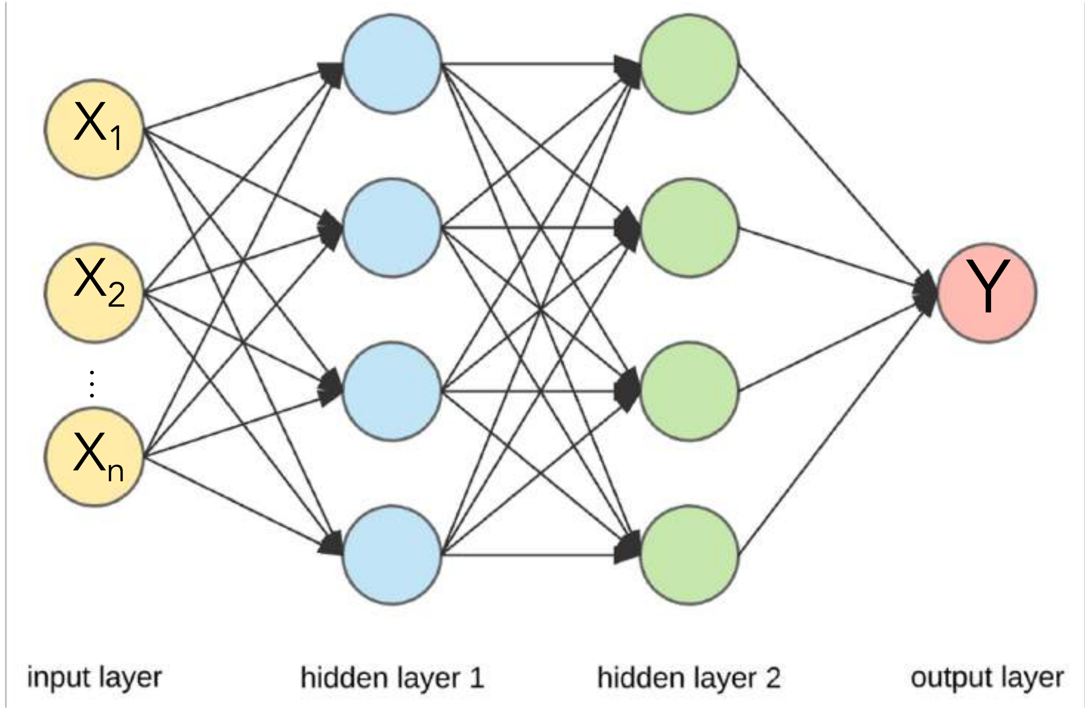




$$Y = w_1 X_1 + w_2 X_2 + b_0$$

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



$$Y = w_1 X_1 + w_2 X_2 + \dots + w_n X_n + b_0 + b_1 + \dots + b_n$$

Regression using DNN with TF2
Code Time!

Open in Colab

Machine Learning Workflow

Collect Data

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

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

- The Boston Housing dataset is taken from the StatLib library which is maintained at CMU.
- There are 506 samples (404:102), each one with 13 attributes (Features Xi from 0 to 12) of houses at different locations around the Boston suburbs in the late 1970s. The attributes themselves are defined in the StatLib website (as per capita crime rate in the area, number of rooms, distance from employment center, etc.).
- Target (Y) is the median values of the houses at a location (in USD 1,000).

Machine Learning Workflow


```python
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)
```

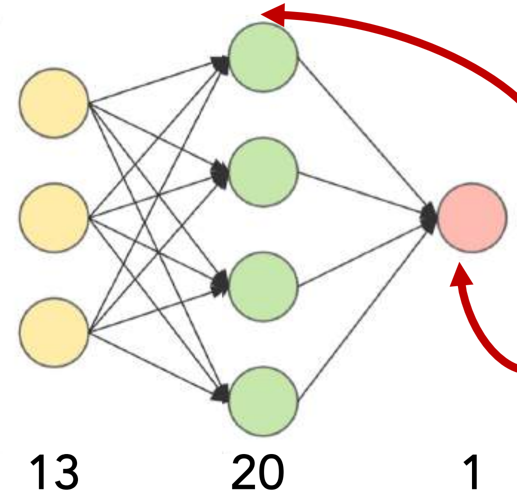
Normalizing Data: We notice that values range varies depending on the type of the feature. If we are training a neural network, for various reasons it's easier if we treat all values as between 0 and 1 (or at least with similar ranges), a process called 'normalizing'. In this case, all features will be rescaled.

Machine Learning Workflow


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

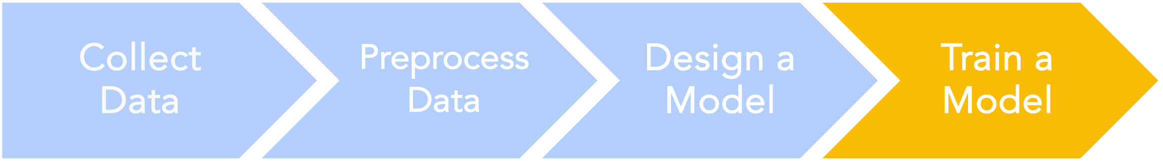
```python
model.compile(
    optimizer='adam',
    loss='mse',
    metrics=['mae']
    )
```

ADAM, a stochastic gradient descent method



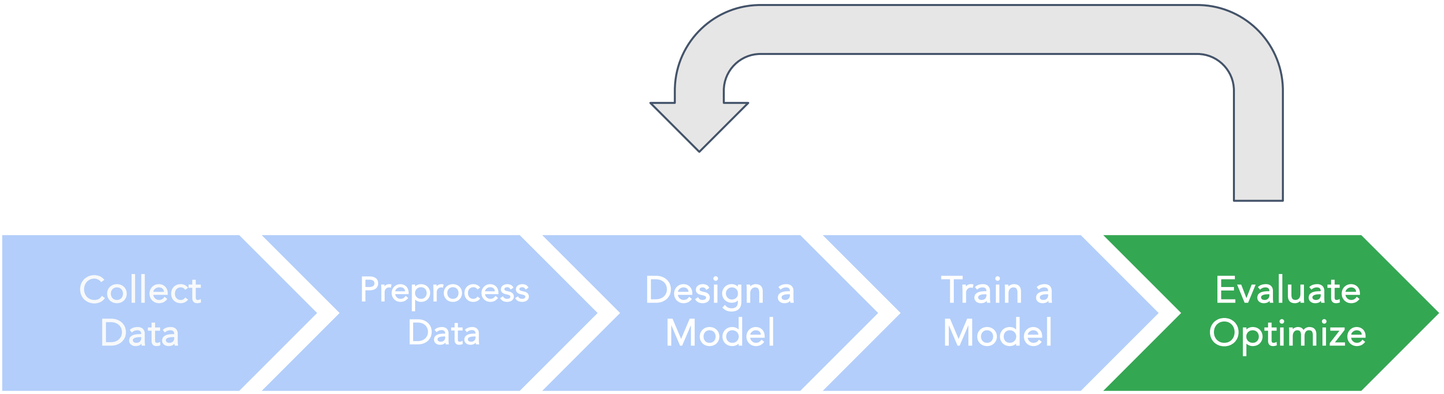
- Total parameters in the dense layer will be 13 x 20 weights (260) and 20 biases => <u>280</u>.
- The output layer will be only one neuron that has one input for each output of the neurons in the previous layer (20) and 1 bias => <u>21</u>.

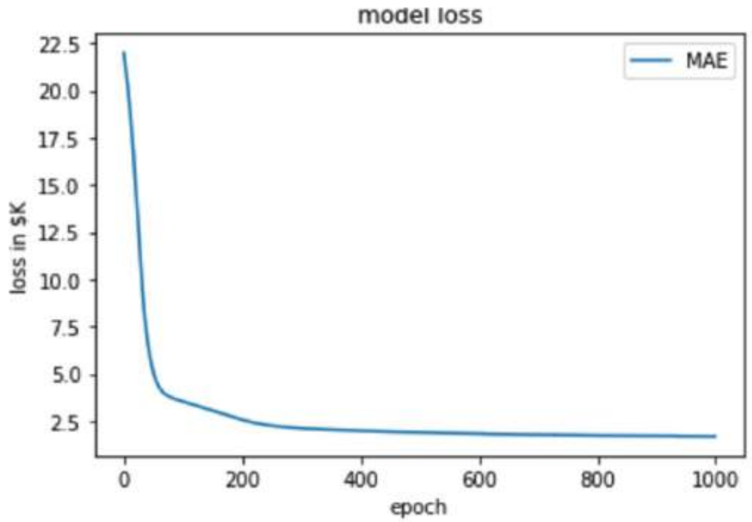
Machine Learning Workflow




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

Machine Learning Workflow






```
tuner.results_summary()
```

```
Results summary
Results in ./untitled_project
Showing 10 best trials
Objective(name="val_loss", direction="min")

Trial 1 summary
Hyperparameters:
units: 30
Score: 16.199304580688477

Trial 0 summary
Hyperparameters:
units: 20
Score: 17.700151443481445

Trial 2 summary
Hyperparameters:
units: 10
Score: 18.69426727294922
```

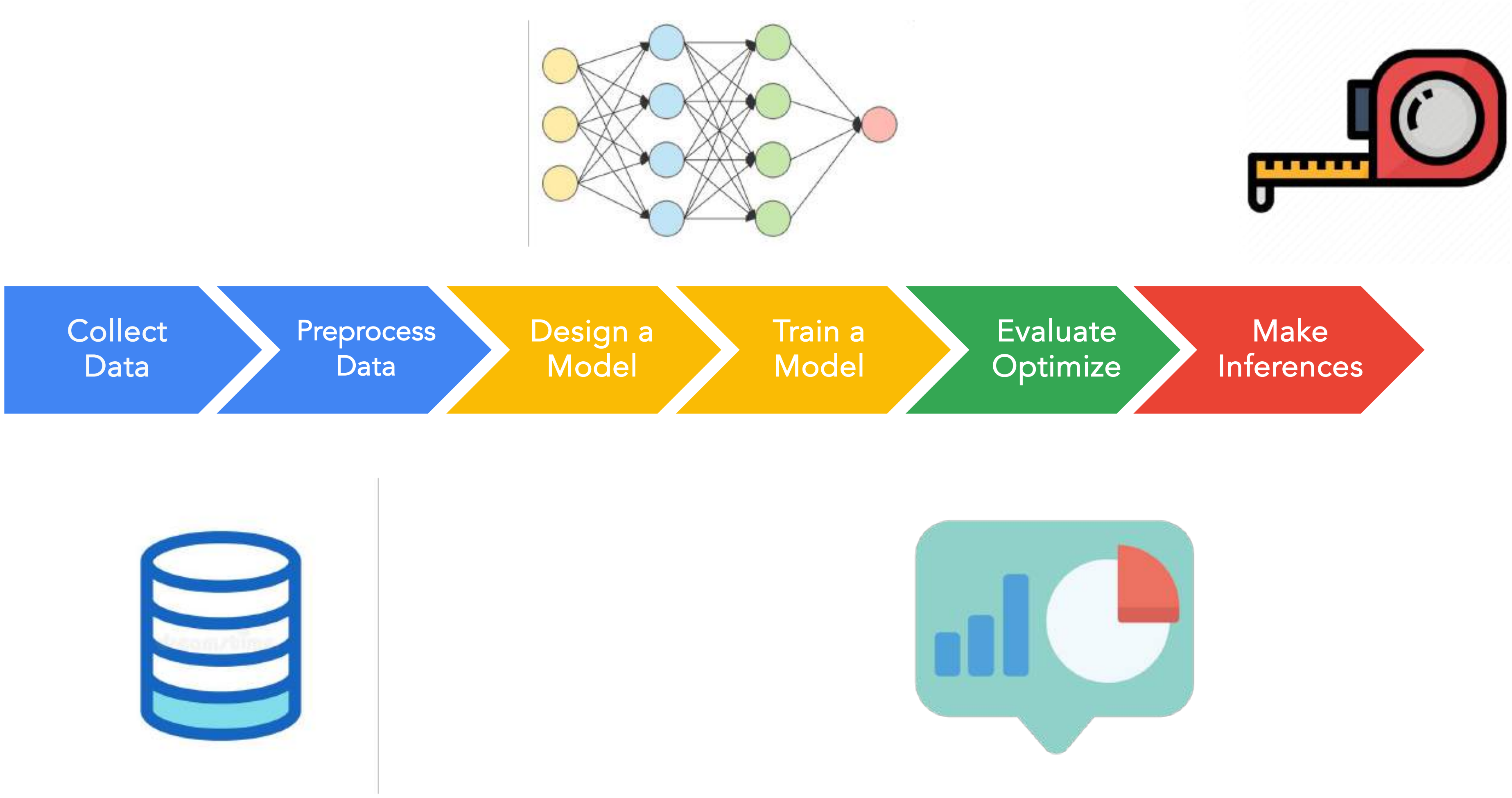
```
train_eval = model.evaluate(x_train_norm, y_train)
print ("Training data MSE: {:.2}".format(train_eval[1]))
```

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

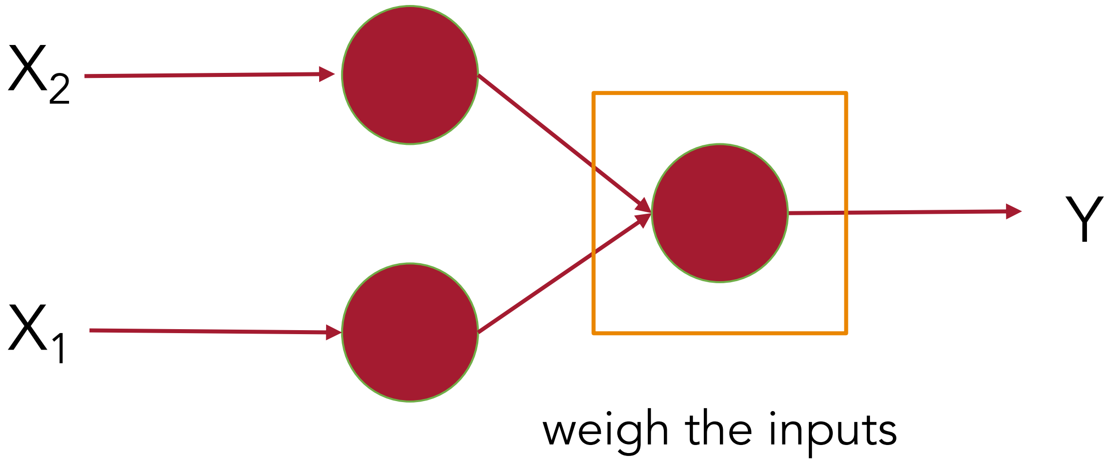
Machine Learning Workflow


```
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)
```

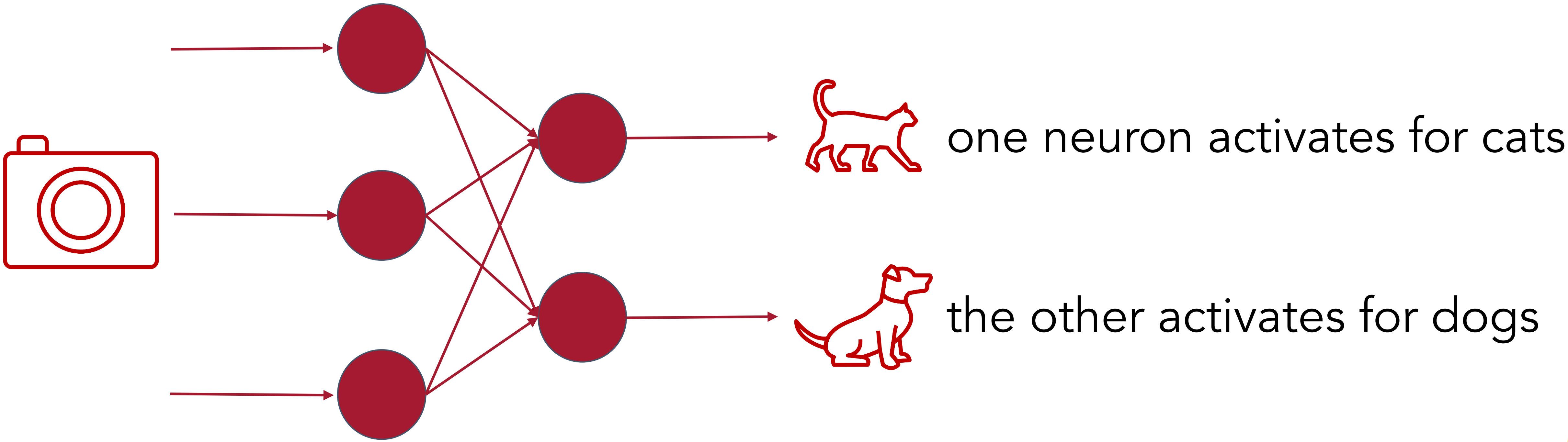
Machine Learning Workflow

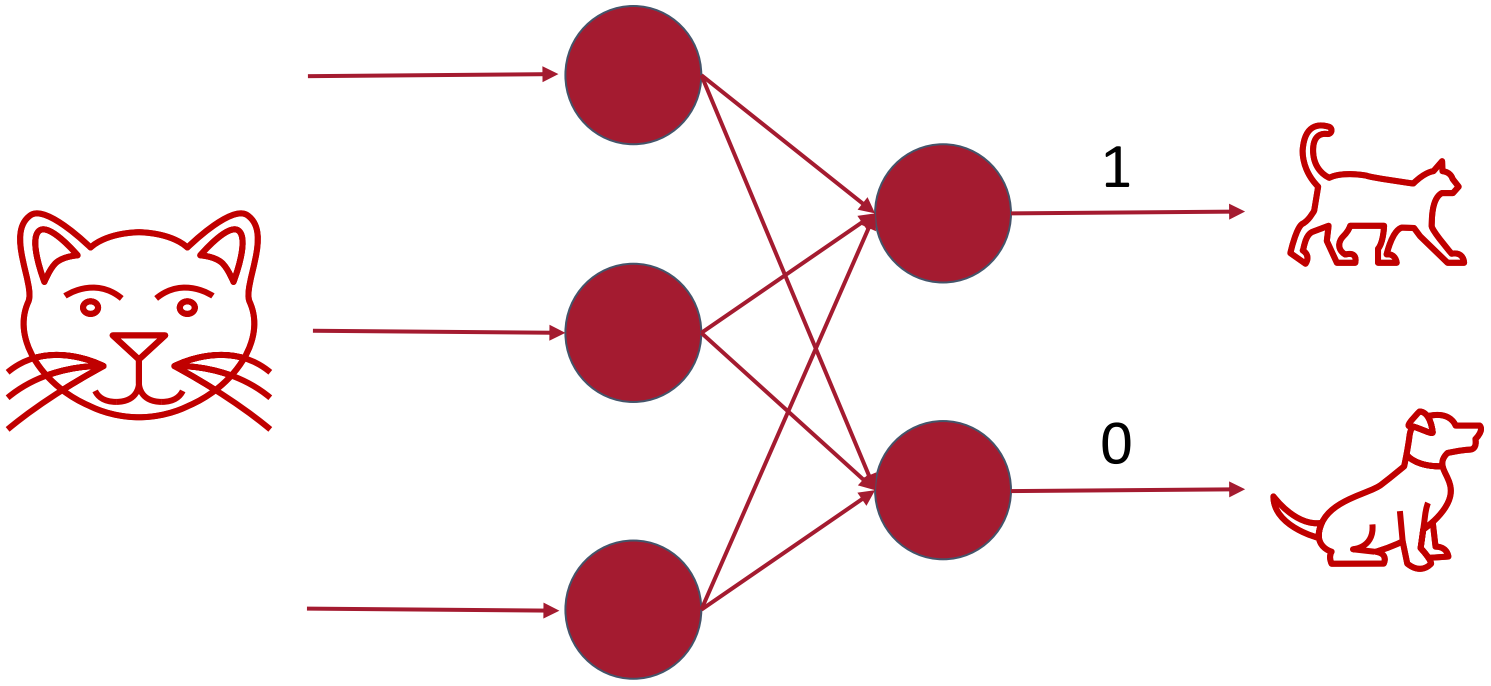


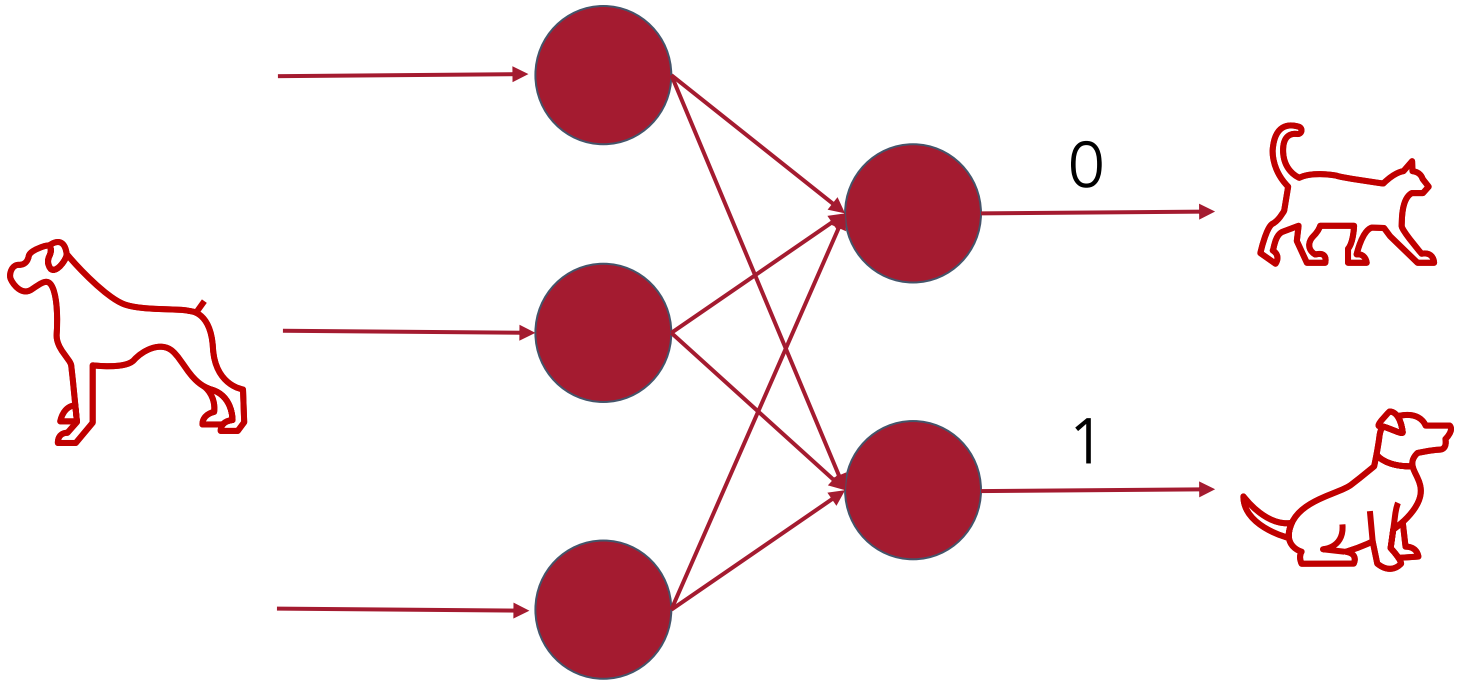
Now, Classification

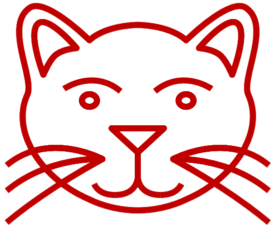
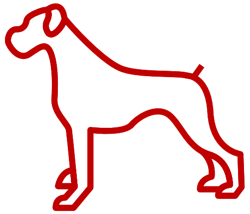


More inputs?


one neuron activates for cats
the other activates for dogs


1
0


0
1

Data	Label
	[1, 0]
	[0, 1]

We can extend this example to other domains

0	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
1	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
2	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
3	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
4	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
5	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
6	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
7	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
8	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
9	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

The MNIST (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.

0 [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
1 [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
2 [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
3 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
4 [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
5 [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
6 [0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
7 [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
8 [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
9 [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

60,000 Labelled Training Examples
10.000 Labelled Validation Examples

Open in Colab

```python
import tensorflow as tf

data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images  = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
     [tf.keras.layers.Flatten(input_shape=(28,28)),
      tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

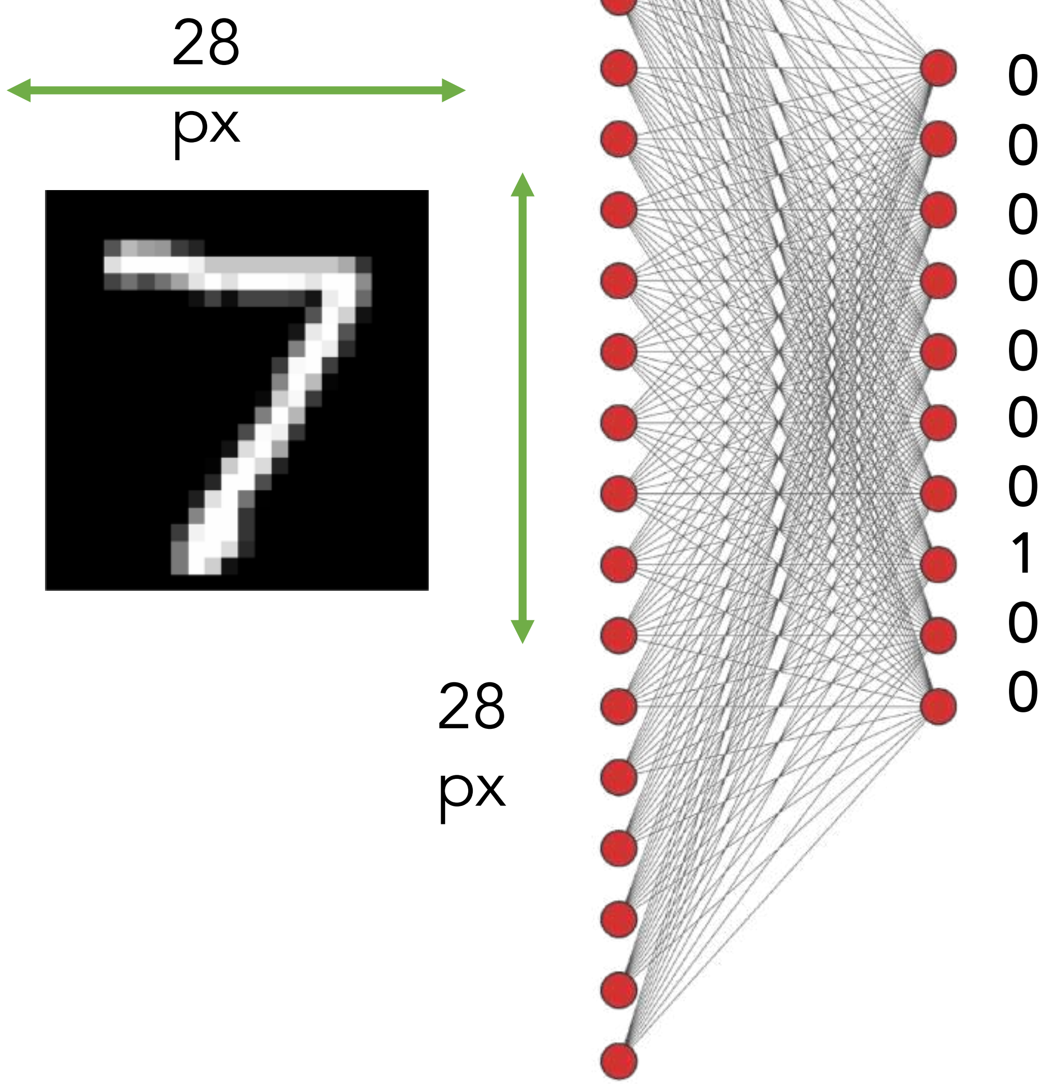
Collect Data

```python
import tensorflow as tf


data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()

training_images  = training_images / 255.0
val_images = val_images / 255.0

model = tf.keras.models.Sequential(
     [tf.keras.layers.Flatten(input_shape=(28,28)),
      tf.keras.layers.Dense(20, activation=tf.nn.relu),
      tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

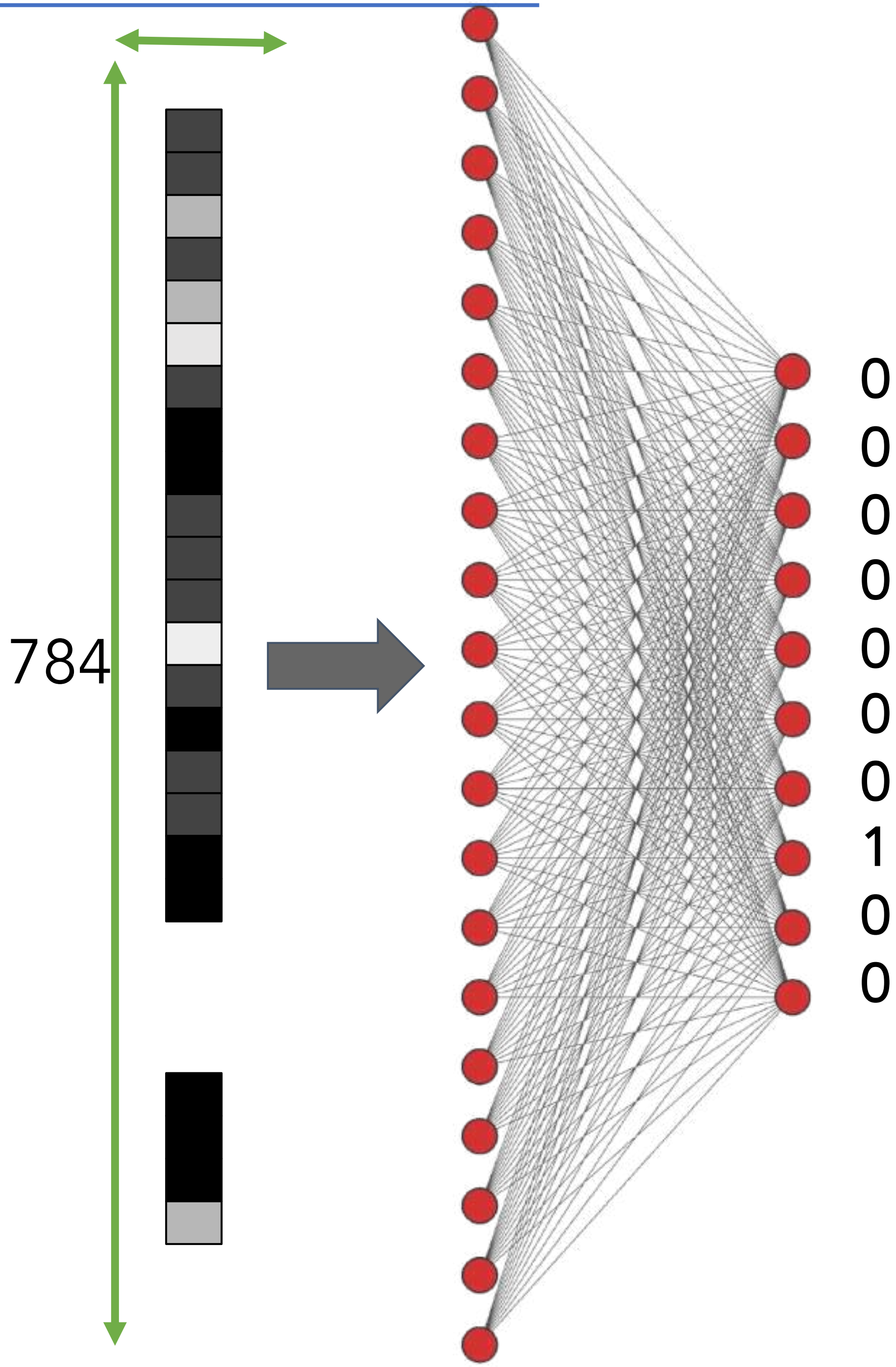
Collect Data

```python
import tensorflow as tf


data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()


training_images  = training_images / 255.0
val_images = val_images / 255.0


model = tf.keras.models.Sequential(
     [tf.keras.layers.Flatten(input_shape=(28,28)),
      tf.keras.layers.Dense(20, activation=tf.nn.relu),
      tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

Design a Model



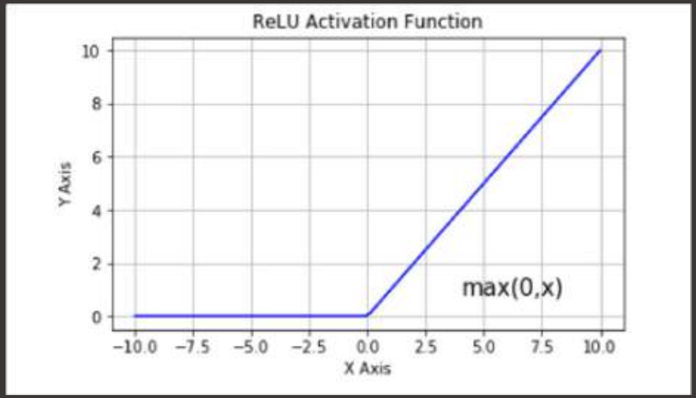
28
px
28
px
0
0
0
0
0
0
0
1
0
0


784
0
0
0
0
0
0
0
1
0
0

```python
import tensorflow as tf


data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()


training_images  = training_images / 255.0
val_images = val_images / 255.0


model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

Design a Model

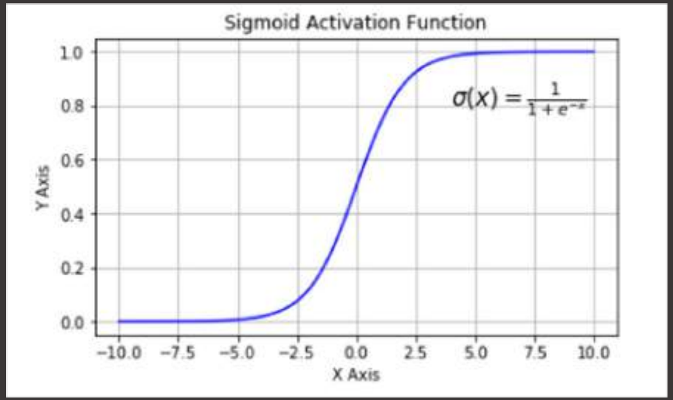
```python
import tensorflow as tf


data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()


training_images  = training_images / 255.0
val_images = val_images / 255.0


model = tf.keras.models.Sequential(
     [tf.keras.layers.Flatten(input_shape=(28,28)),
      tf.keras.layers.Dense(20, activation=tf.nn.relu),
      tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```



ReLU applies much-needed non-linearity into the model. Non-linearity is necessary to produce non-linear decision boundaries, so that the output cannot be written as a linear combination of the inputs.

Design a Model

```python
import tensorflow as tf


data = tf.keras.datasets.mnist
(training_images, training_labels), (val_images, val_labels) = data.load_data()


training_images  = training_images / 255.0
val_images = val_images / 255.0


model = tf.keras.models.Sequential(
     [tf.keras.layers.Flatten(input_shape=(28,28)),
      tf.keras.layers.Dense(20, activation=tf.nn.relu),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```



SOFTMAX: Generalization of the logistic function (or Sigmoid) to multiple dimensions. A softmax operation serves a key purpose: making sure the Neural Network (in this case, a DNN) outputs sum to 1. Because of this, softmax operations are useful to scale model outputs into probabilities.

Design a Model

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

model.fit(training_images, training_labels, epochs=20)
```

Design a
Model

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


model.fit(training_images, training_labels, epochs=20)
```

Mean Squared Error

Prediction

$$MSE = \frac{1}{N}\sum(t_i - s_i)^2$$

Ground Truth

Cross Entropy Loss

Classes

Prediction

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

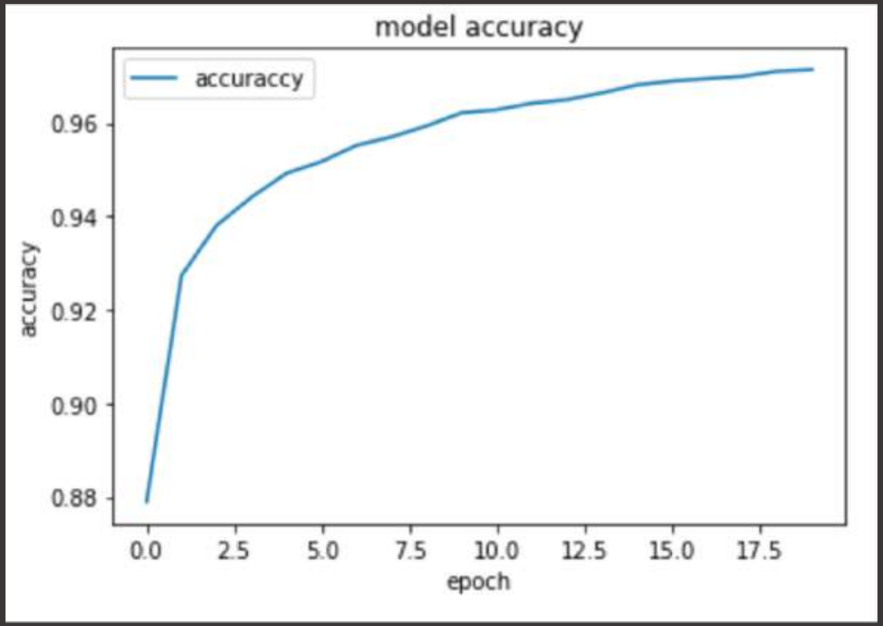
Ground Truth {0,1}

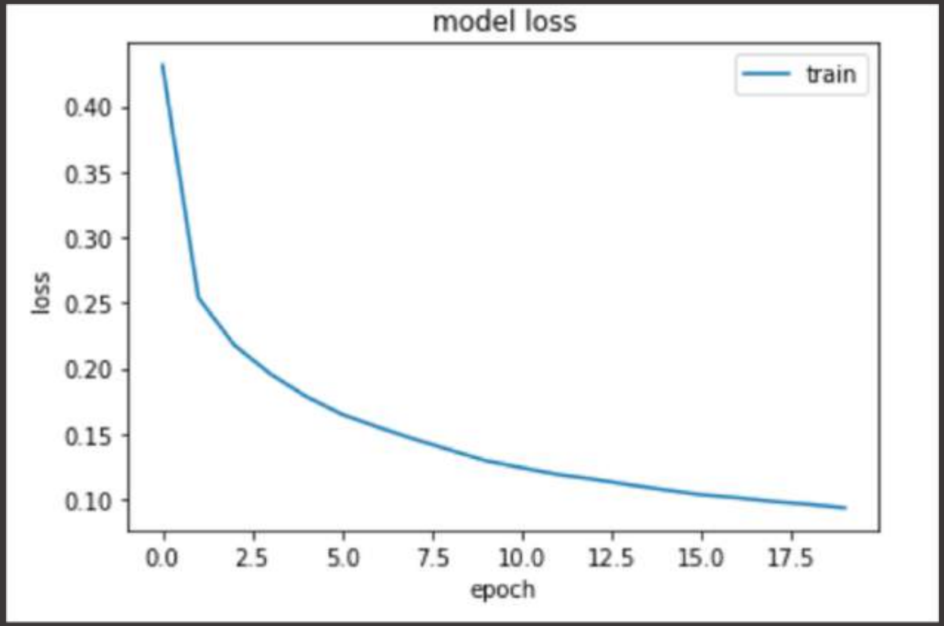
Design a Model

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

model.fit(training_images, training_labels, epochs=20)
```

Train a Model

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

model.fit(training_images, training_labels, epochs=20)
```





Evaluate
Optimize

```
classifications = model.predict(val_images)

print(classifications[0])

print(test_labels[0])
```

```
[2.4921512e-09 1.3765138e-10 8.8281205e-08 1.0477231e-03 2.8455029e-12
4.0820678e-06 2.0070659e-16 9.9894780e-01 1.0296049e-07 2.9972372e-07]
7
```

Make Inferences

a NN to classify the MNIST DB

colab.research.google.com

MNIST_NN.ipynb

```
import tensorflow as tf
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (val_images, val_labels) = mnist.load_data()
training_images=training_images / 255.0
val_images=val_images / 255.0
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(20, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(training_images, training_labels, validation_data=(val_images, val_labels), epochs=20)
```

a NN to classify the MNIST DB

colab.research.google.com

MNIST_NN.ipynb

```
Epoch 9/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3555 - accuracy: 0.8724 - val_loss: 0.4090 - val_accuracy: 0.8516
Epoch 10/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3509 - accuracy: 0.8752 - val_loss: 0.4061 - val_accuracy: 0.8537
Epoch 11/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3452 - accuracy: 0.8768 - val_loss: 0.3980 - val_accuracy: 0.8580
Epoch 12/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3398 - accuracy: 0.8783 - val_loss: 0.4052 - val_accuracy: 0.8586
Epoch 13/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3355 - accuracy: 0.8798 - val_loss: 0.4160 - val_accuracy: 0.8533
Epoch 14/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3332 - accuracy: 0.8812 - val_loss: 0.3913 - val_accuracy: 0.8609
Epoch 15/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3279 - accuracy: 0.8818 - val_loss: 0.3971 - val_accuracy: 0.8588
Epoch 16/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3250 - accuracy: 0.8839 - val_loss: 0.3945 - val_accuracy: 0.8597
Epoch 17/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3221 - accuracy: 0.8839 - val_loss: 0.3985 - val_accuracy: 0.8578
Epoch 18/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3184 - accuracy: 0.8853 - val_loss: 0.3988 - val_accuracy: 0.8595
Epoch 19/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3158 - accuracy: 0.8857 - val_loss: 0.3984 - val_accuracy: 0.8578
Epoch 20/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3140 - accuracy: 0.8856 - val_loss: 0.4069 - val_accuracy: 0.8567
<keras.callbacks.History at 0x7fe50180b750>
```

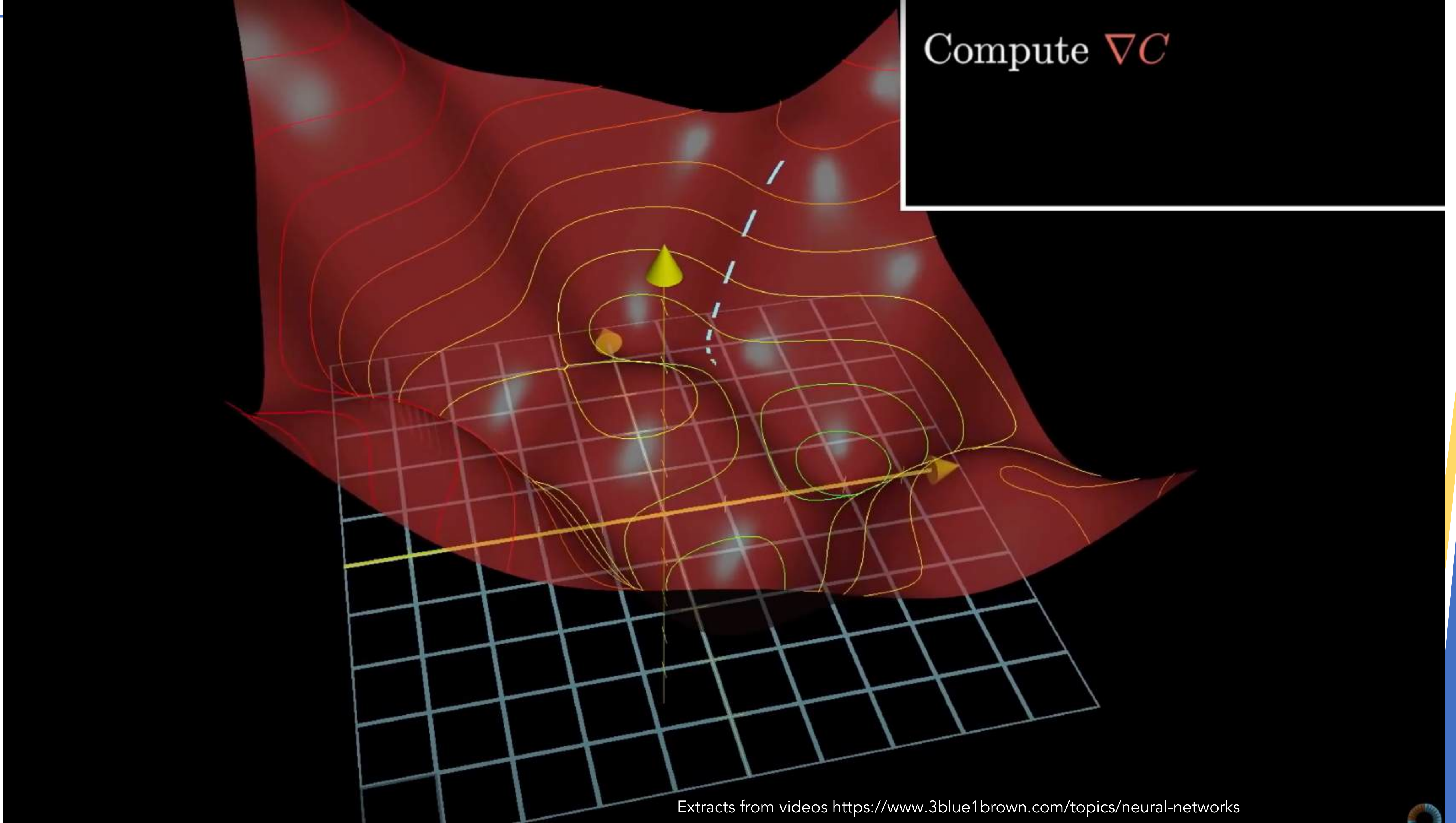
a NN to classify the MNIST DB

colab.research.google.com
MNIST_NN.ipynb

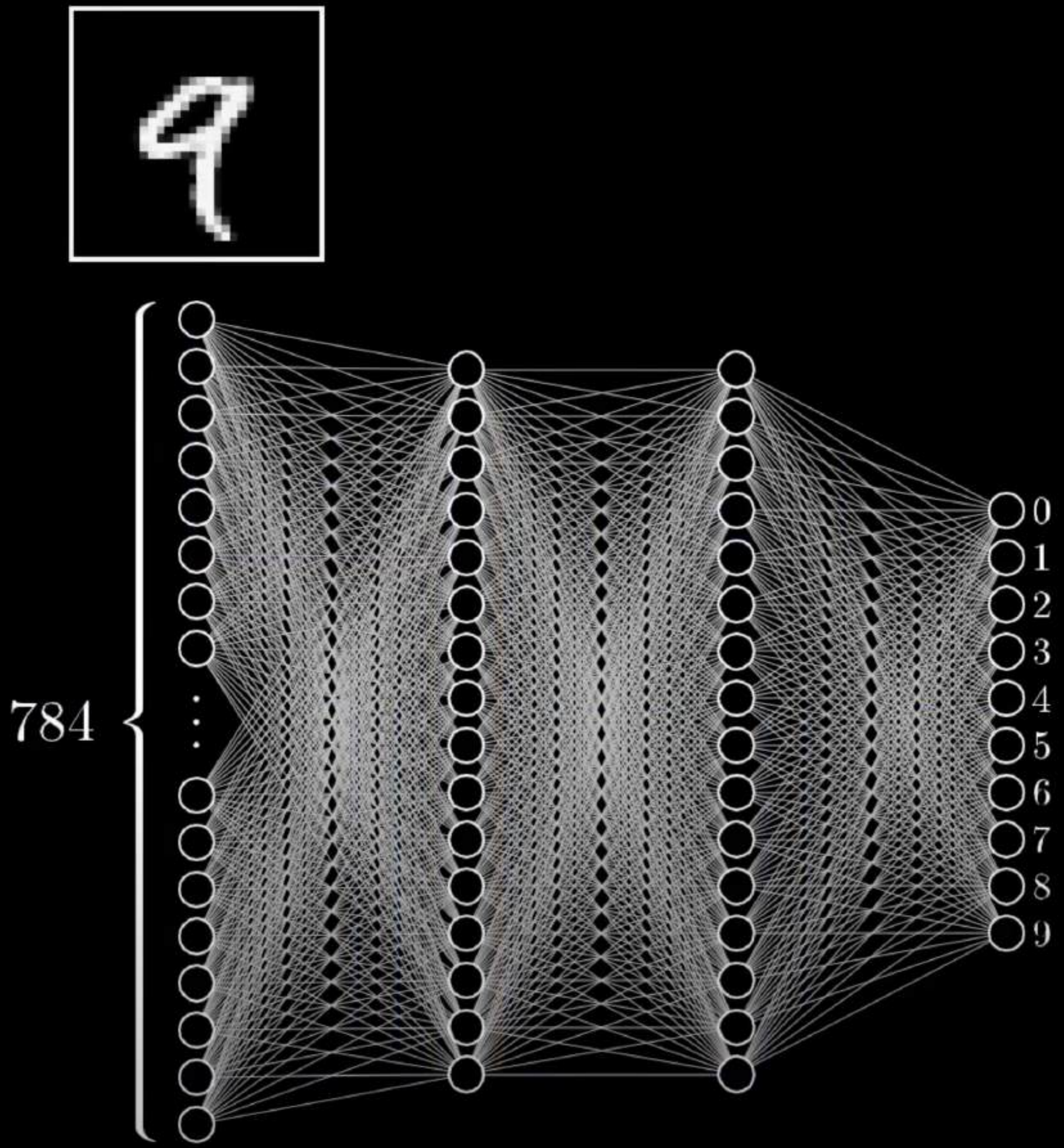
```
Epoch 19/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3022 - accuracy: 0.8914 - val_loss: 0.3834 - val_accuracy: 0.8659
Epoch 20/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2996 - accuracy: 0.8910 - val_loss: 0.3911 - val_accuracy: 0.8642
<keras.callbacks.History at 0x7f033e5f5bd0>
```

```
model.evaluate(val_images, val_labels)

classifications = model.predict(val_images)
print(classifications[0])
print(val_labels[0])
```

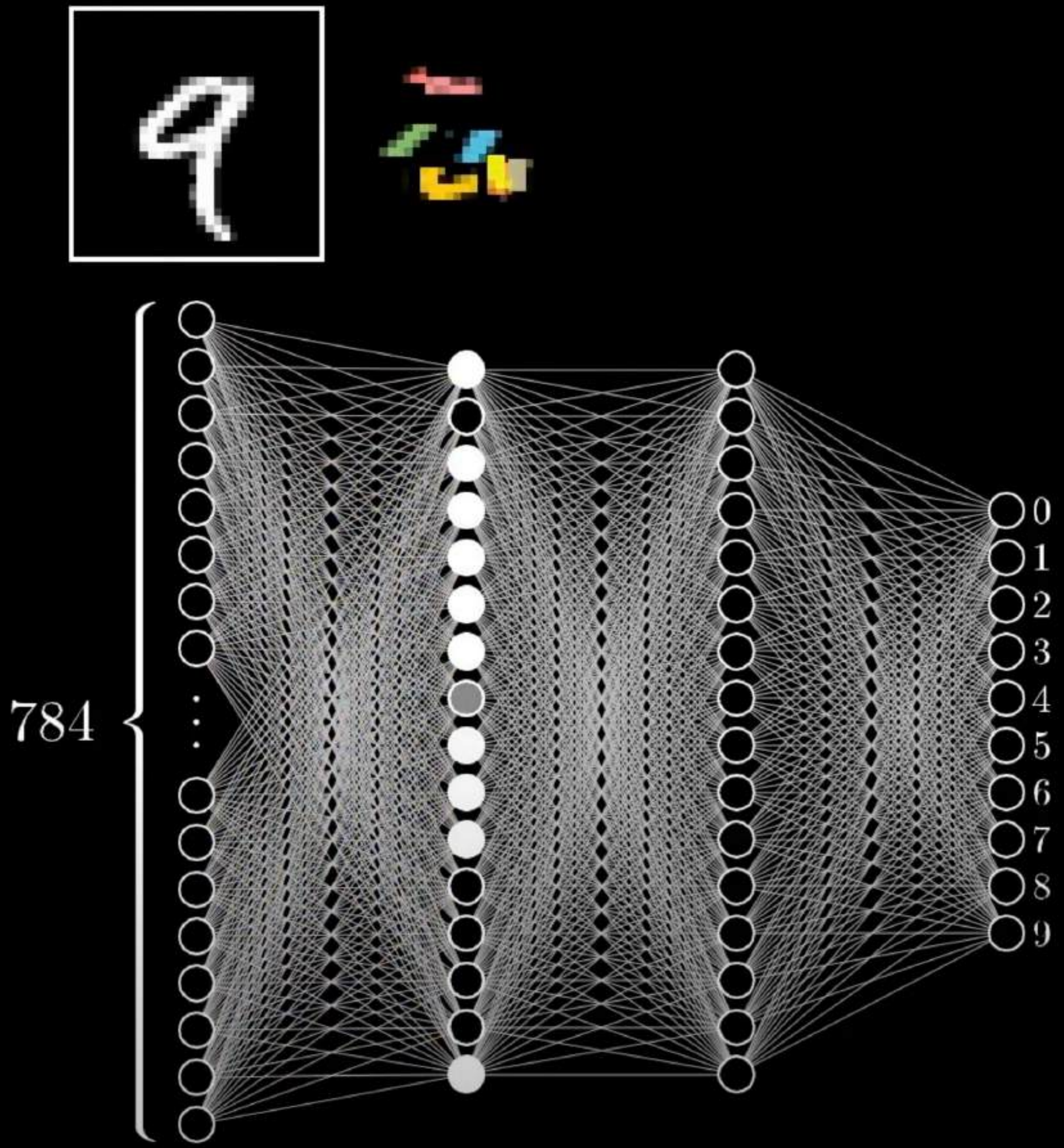
```
313/313 [==============================] - 0s 1ms/step - loss: 0.3911 - accuracy: 0.8642
[5.2699960e-09 4.4460235e-10 2.9260536e-07 1.1081011e-04 1.4583268e-08
 8.1817927e-03 5.3513944e-09 5.8446459e-02 2.9248906e-05 9.3323141e-01]
9
```



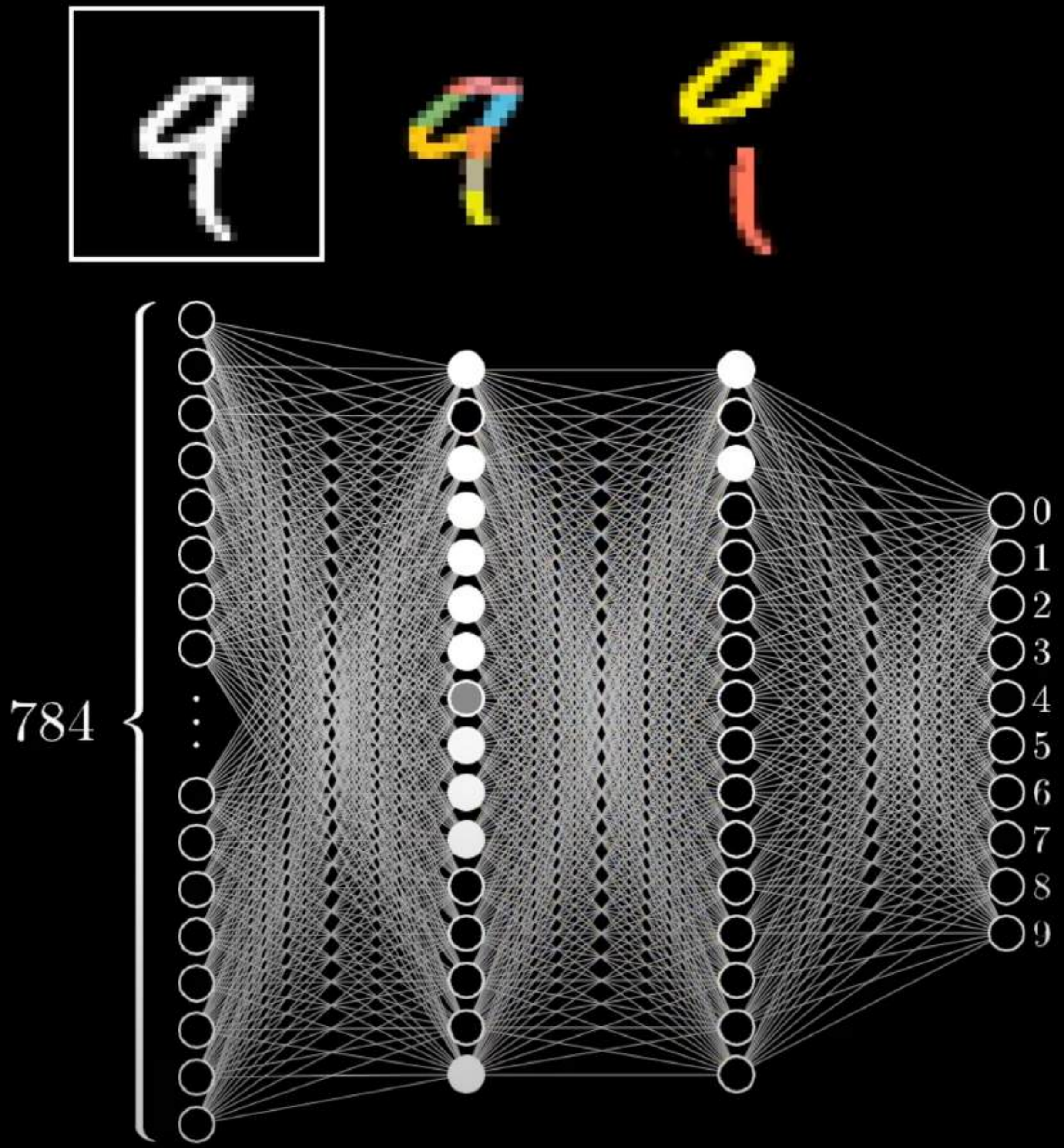
Extracts from videos https://www.3blue1brown.com/topics/neural-networks



Extracts from videos https://www.3blue1brown.com/topics/neural-networks

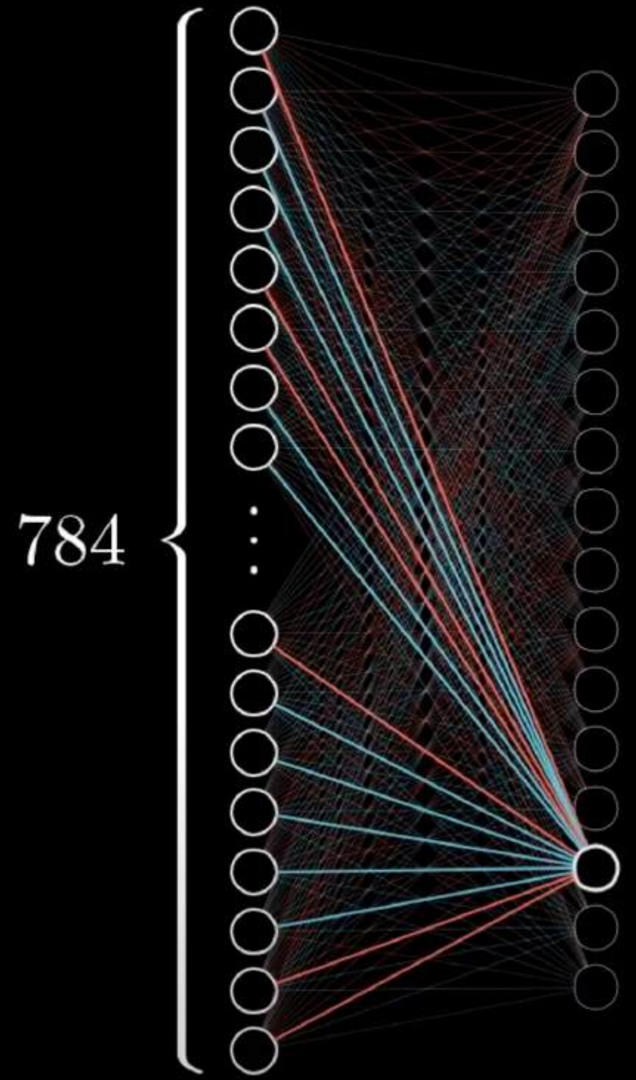


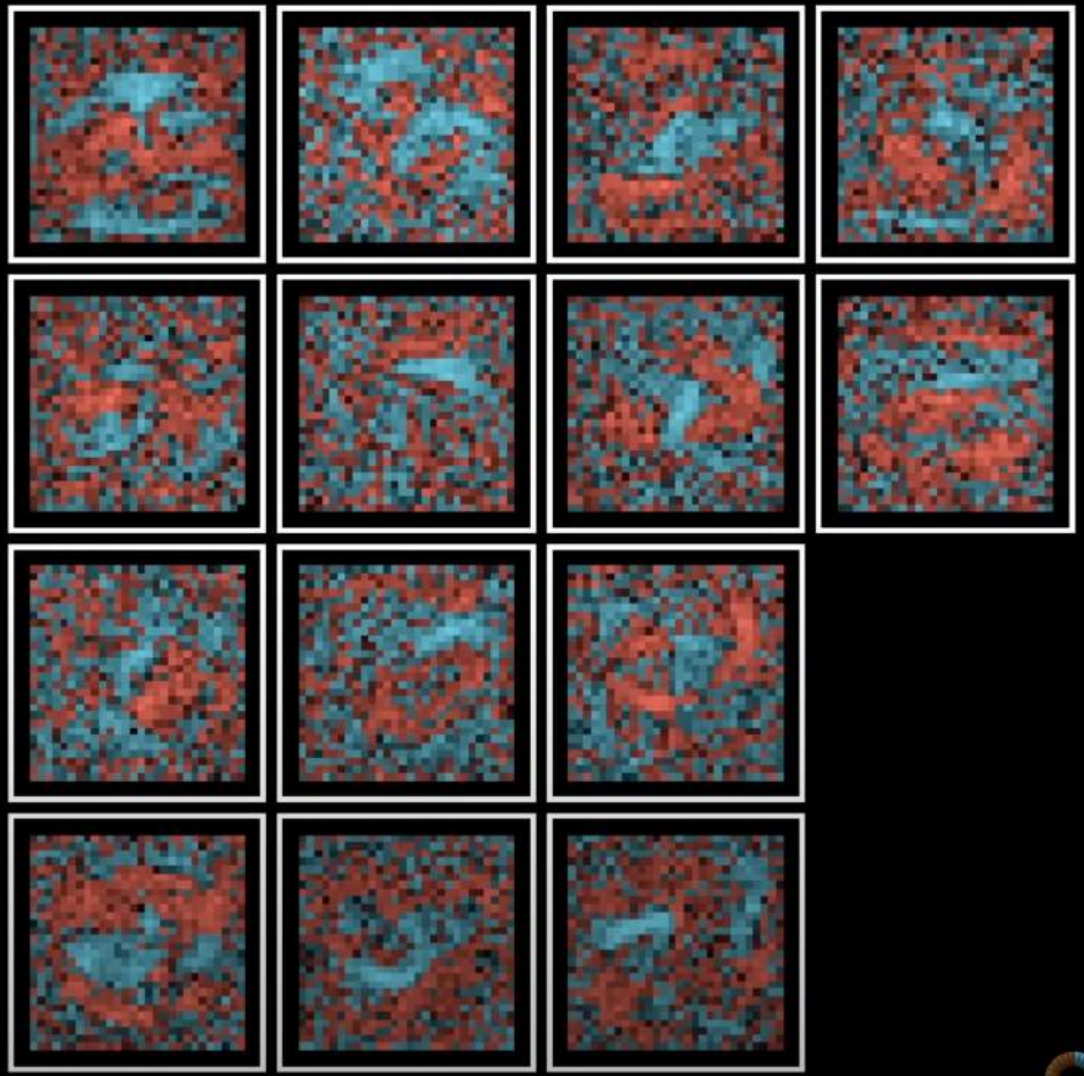
Extracts from videos https://www.3blue1brown.com/topics/neural-networks



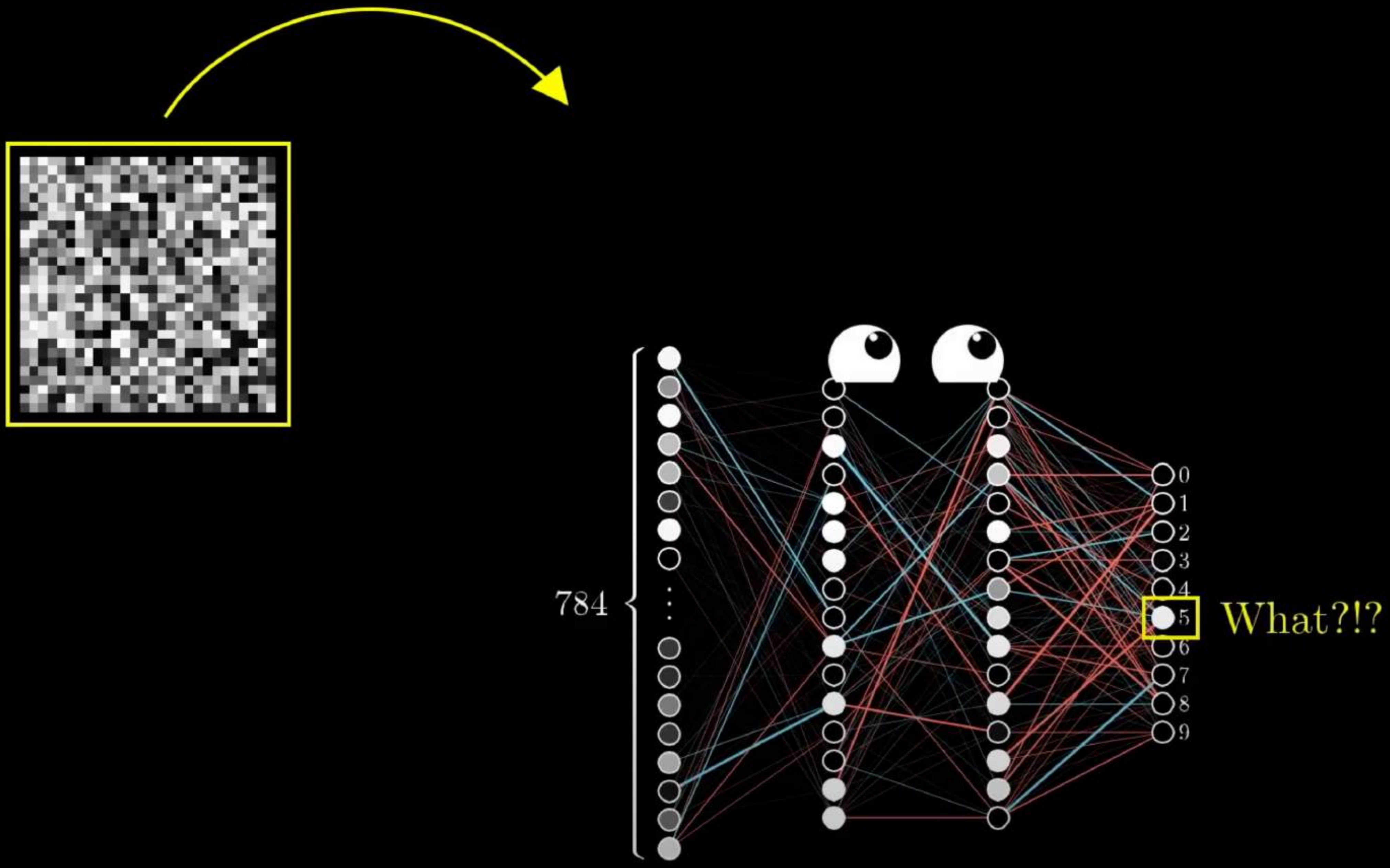
Extracts from videos https://www.3blue1brown.com/topics/neural-networks

What second layer neurons look for





Extracts from videos https://www.3blue1brown.com/topics/neural-networks



Extracts from videos https://www.3blue1brown.com/topics/neural-networks

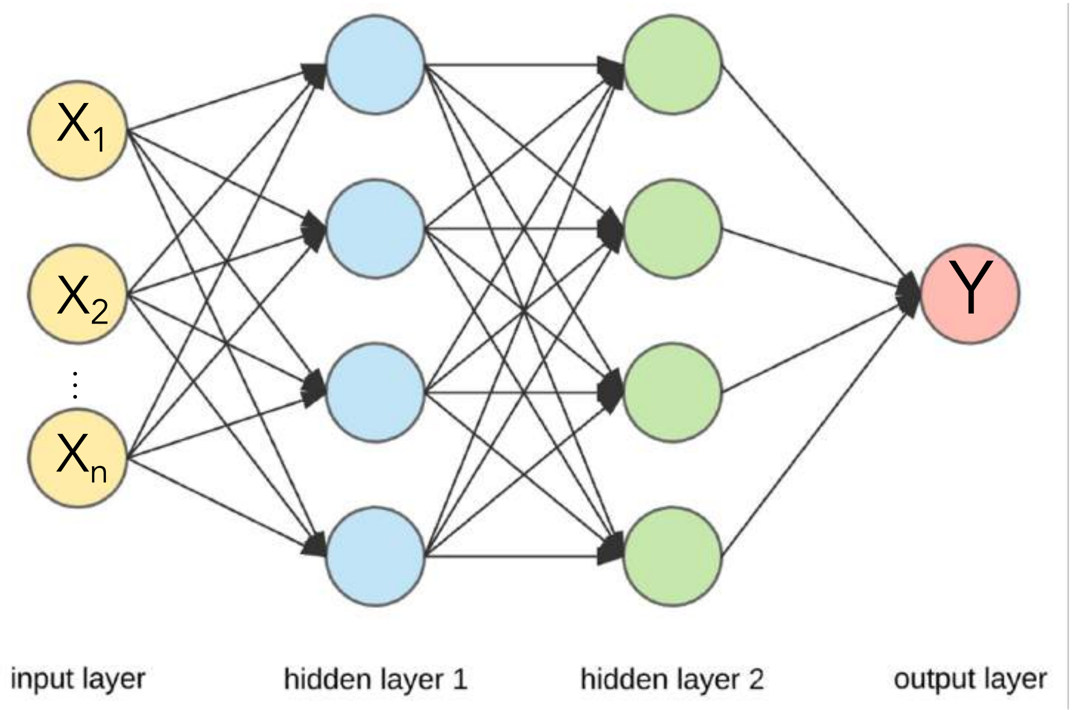
A very nice introduction to NN

- 3Blue1Brown playlist on Neural Networks
  - But what is a neural network?
    - Chapter 1 – Deep learning
    - https://youtu.be/aircAruvnKk
  - Gradient descent, how neural networks learn
    - Chapter 2 – Deep learning
    - https://youtu.be/IHZwWFHWa-w
  - What is backpropagation really doing?
    - Chapter 3 – Deep learning
    - https://youtu.be/Ilg3gGewQ5U
  - (Optional) Backpropagation calculus
    - Chapter 4 – Deep learning
    - https://youtu.be/tIeHLnjs5U8

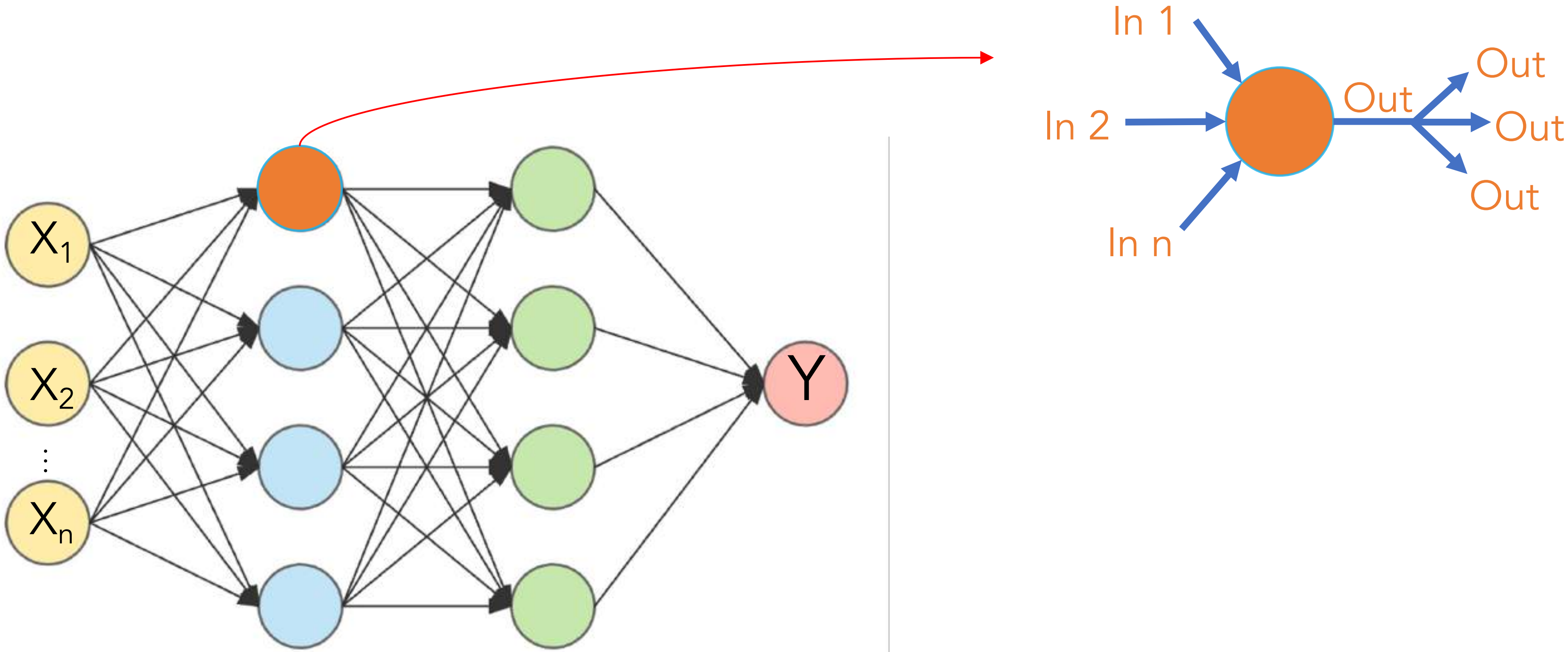
https://www.3blue1brown.com/topics/neural-networks

… in summary

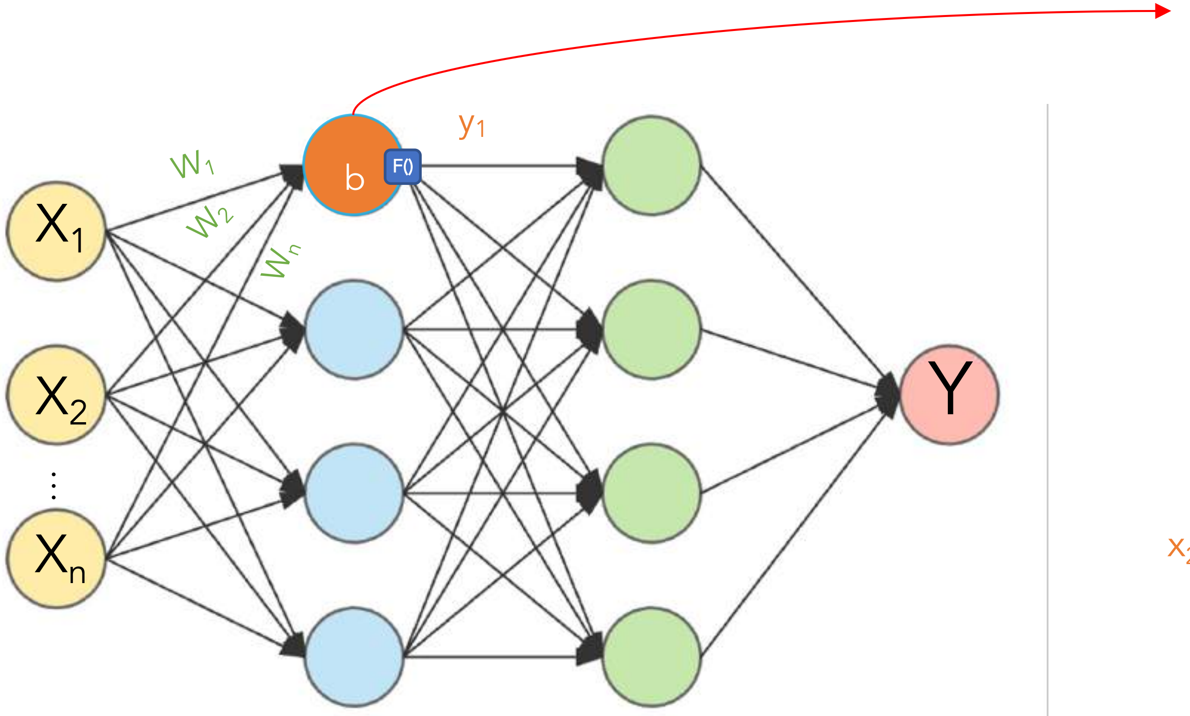
Dense Neural Networks – DNNs

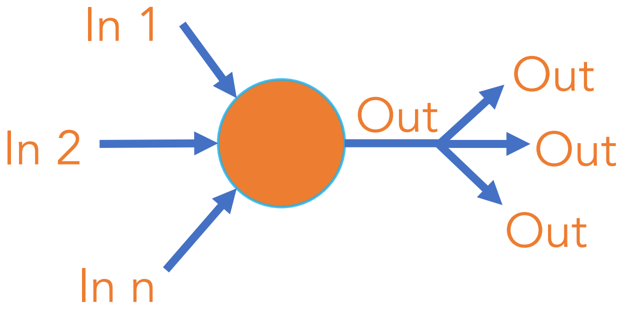


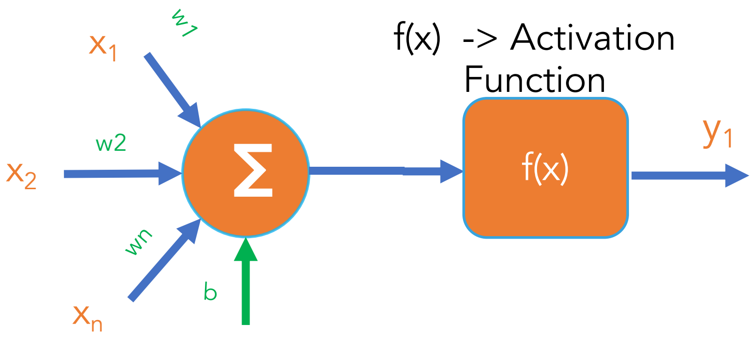
DNNs



DNNs



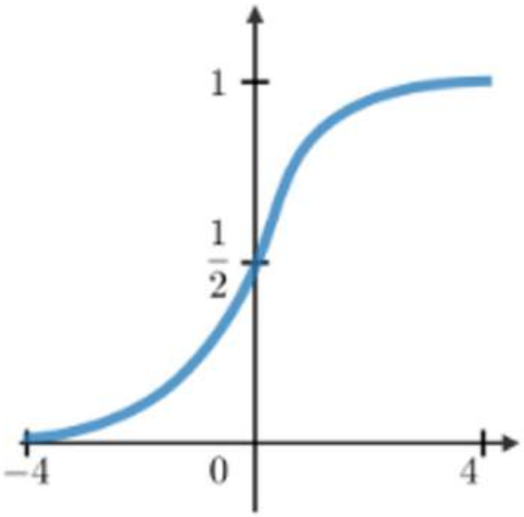


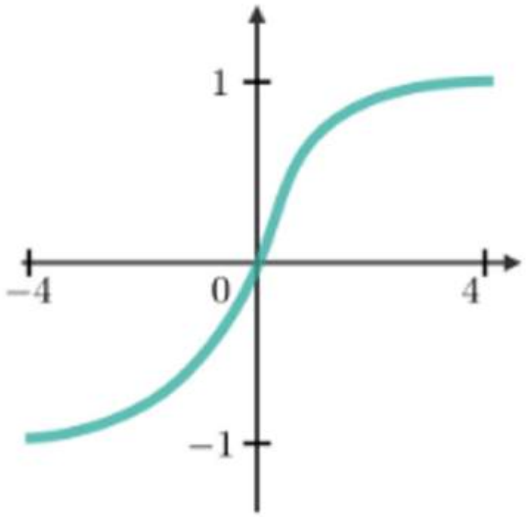


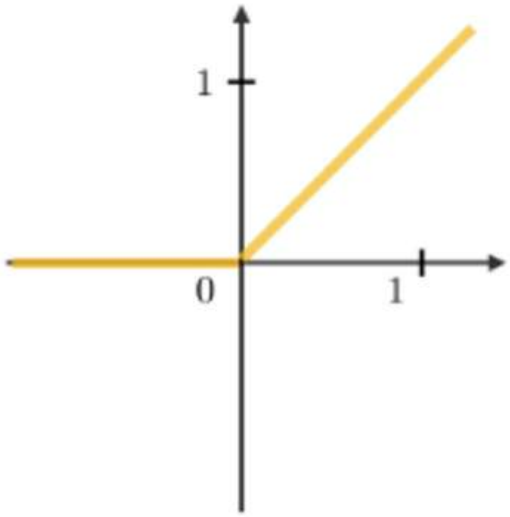
$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

DNNs – Activation Functions

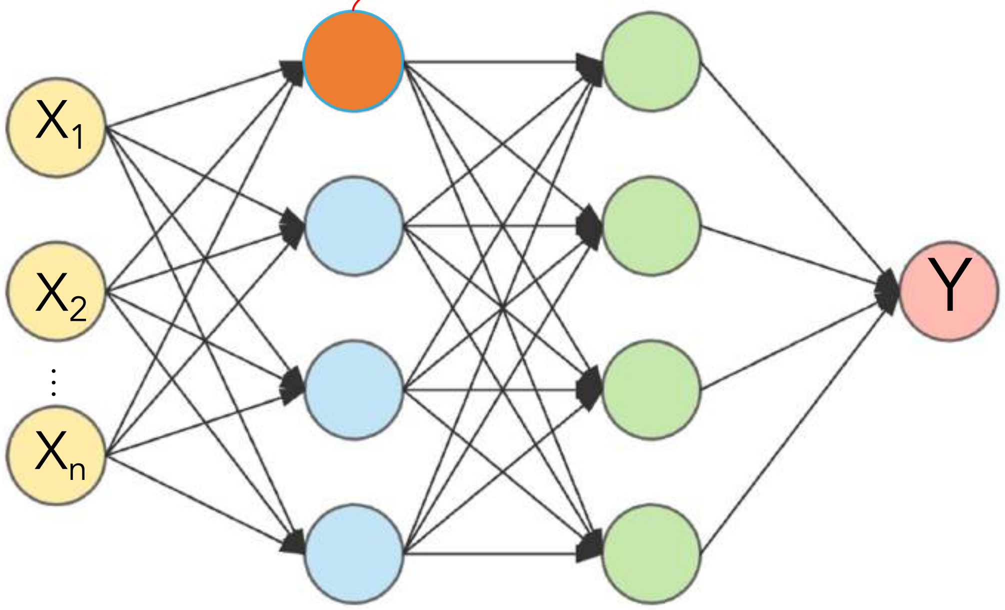
Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	$g(z) = \max(0, z)$

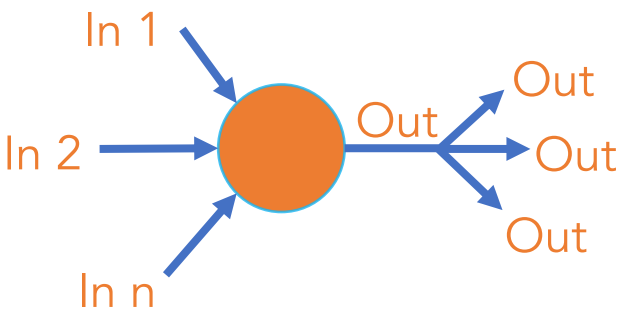


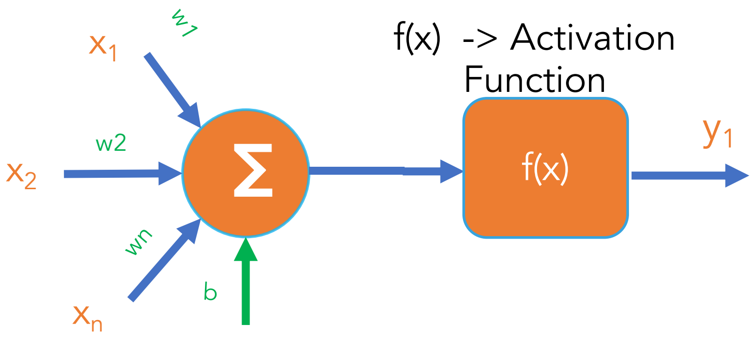




DNNs



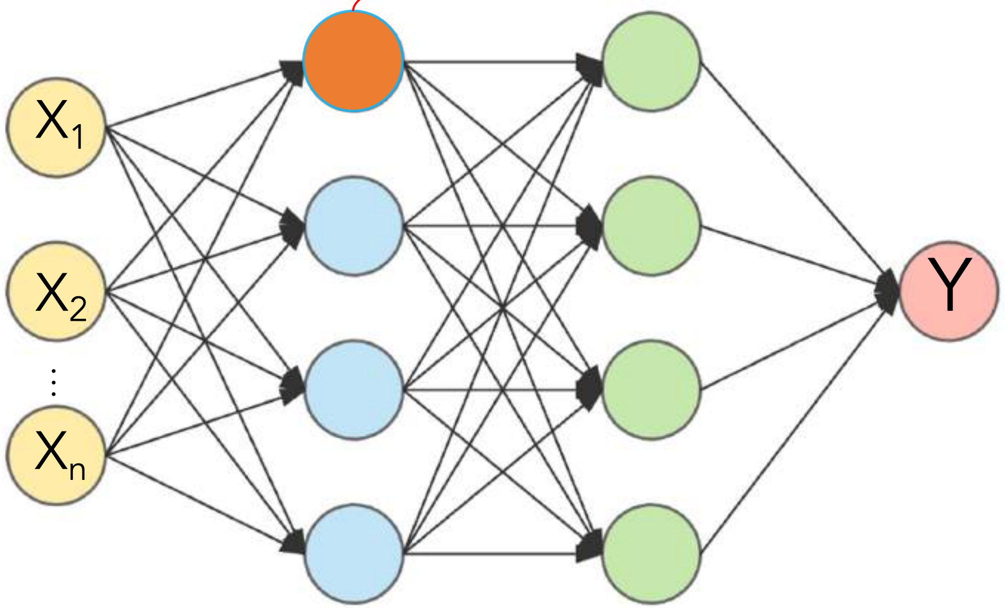


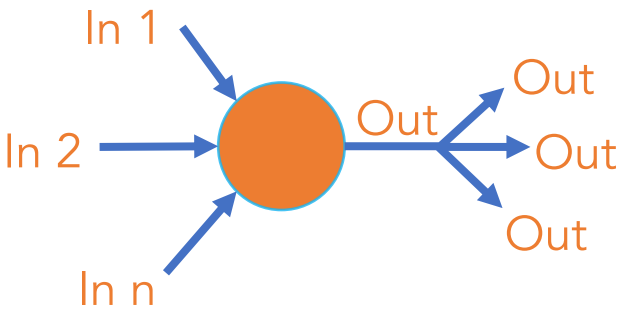


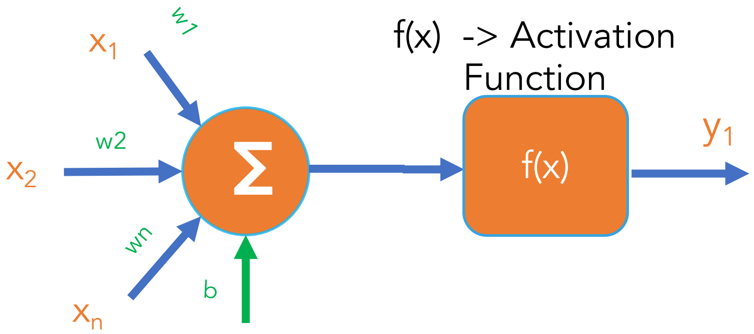
Parameters to be found during training, to reach minimum error

$$y = f(\sum_{i=1}^{n} x_i w_i + b)$$

DNNs





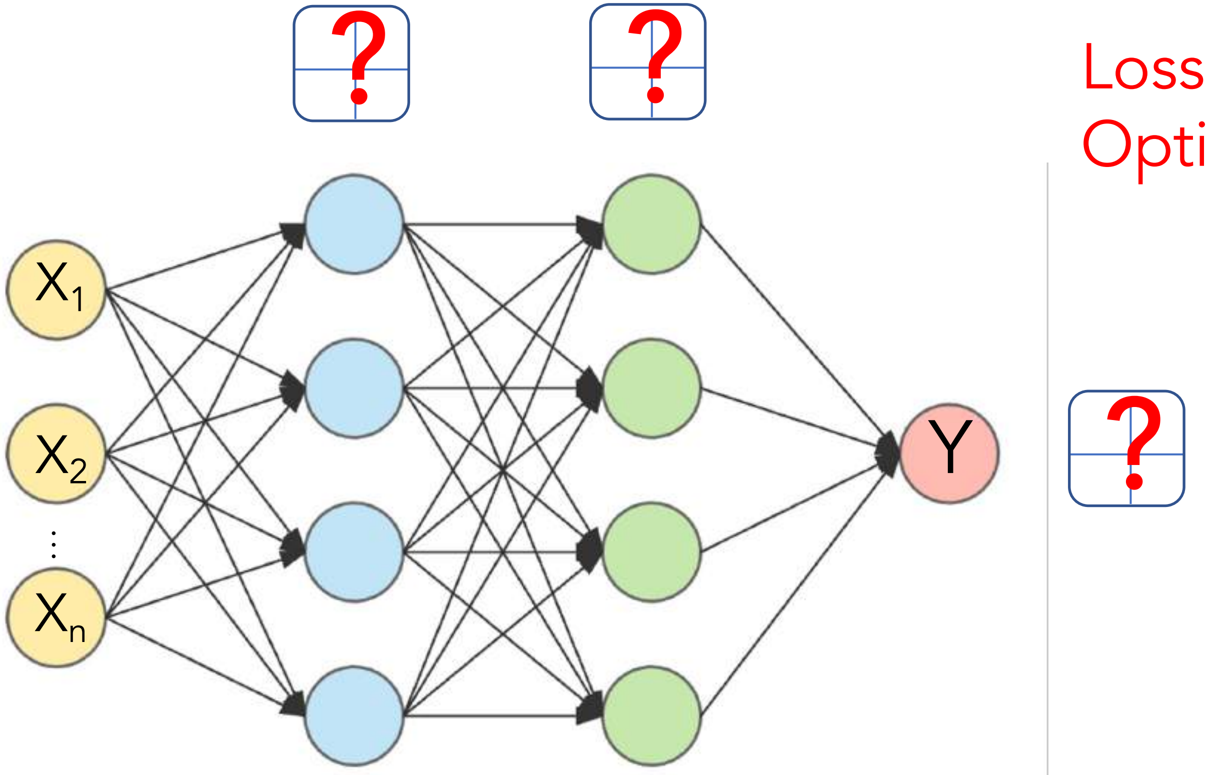


- Error Measurement (Loss)
- Optimization

Parameters to be found during training, to reach minimum error

$$y = f(\sum_{i=1}^{n} x_i w_i + b)$$

DNNs



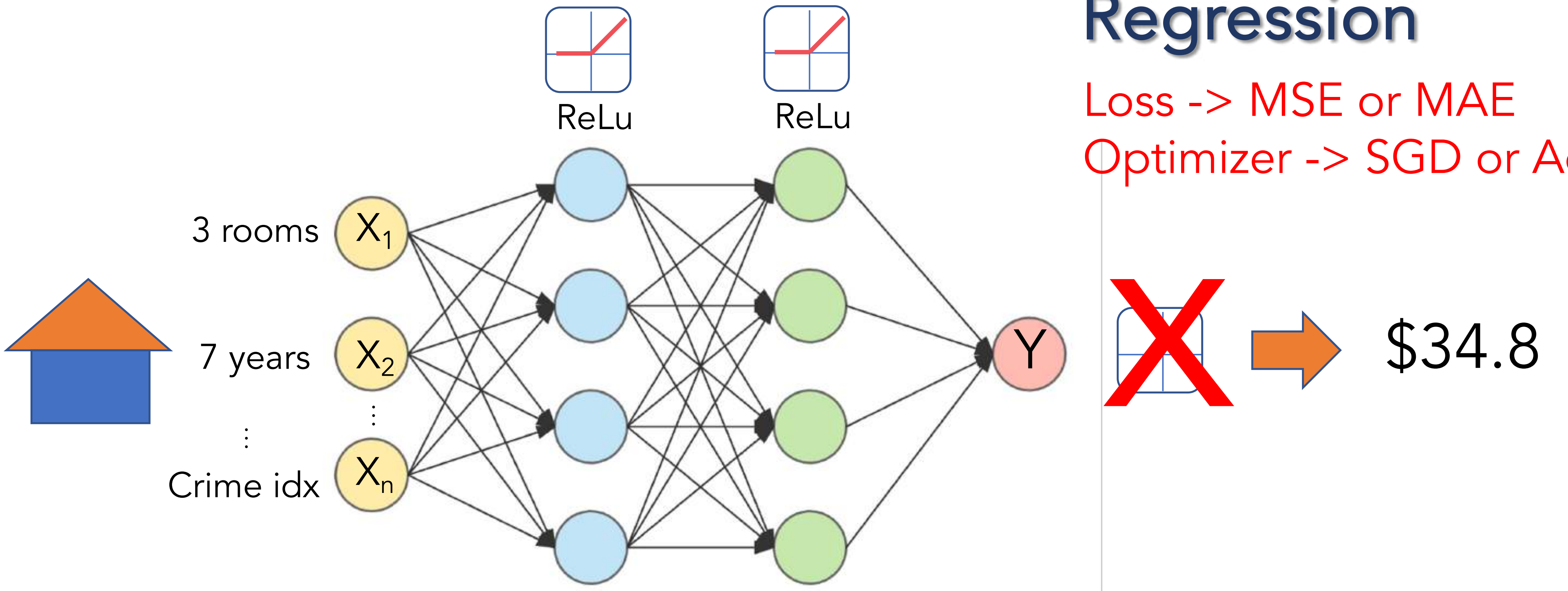
Loss -> ?

Optimizer -> ?

DNNs

Regression

Loss -> MSE or MAE
Optimizer -> SGD or Adam

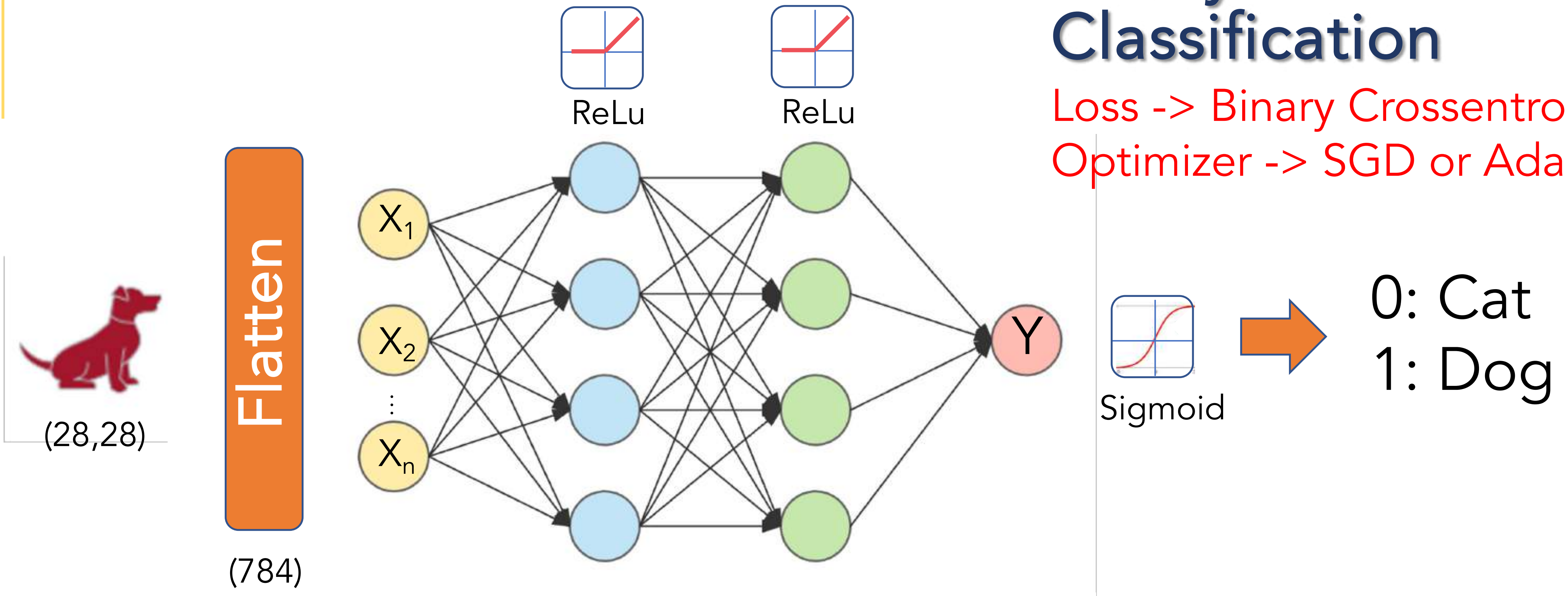


DNNs

Binary Classification

Loss -> Binary Crossentropy

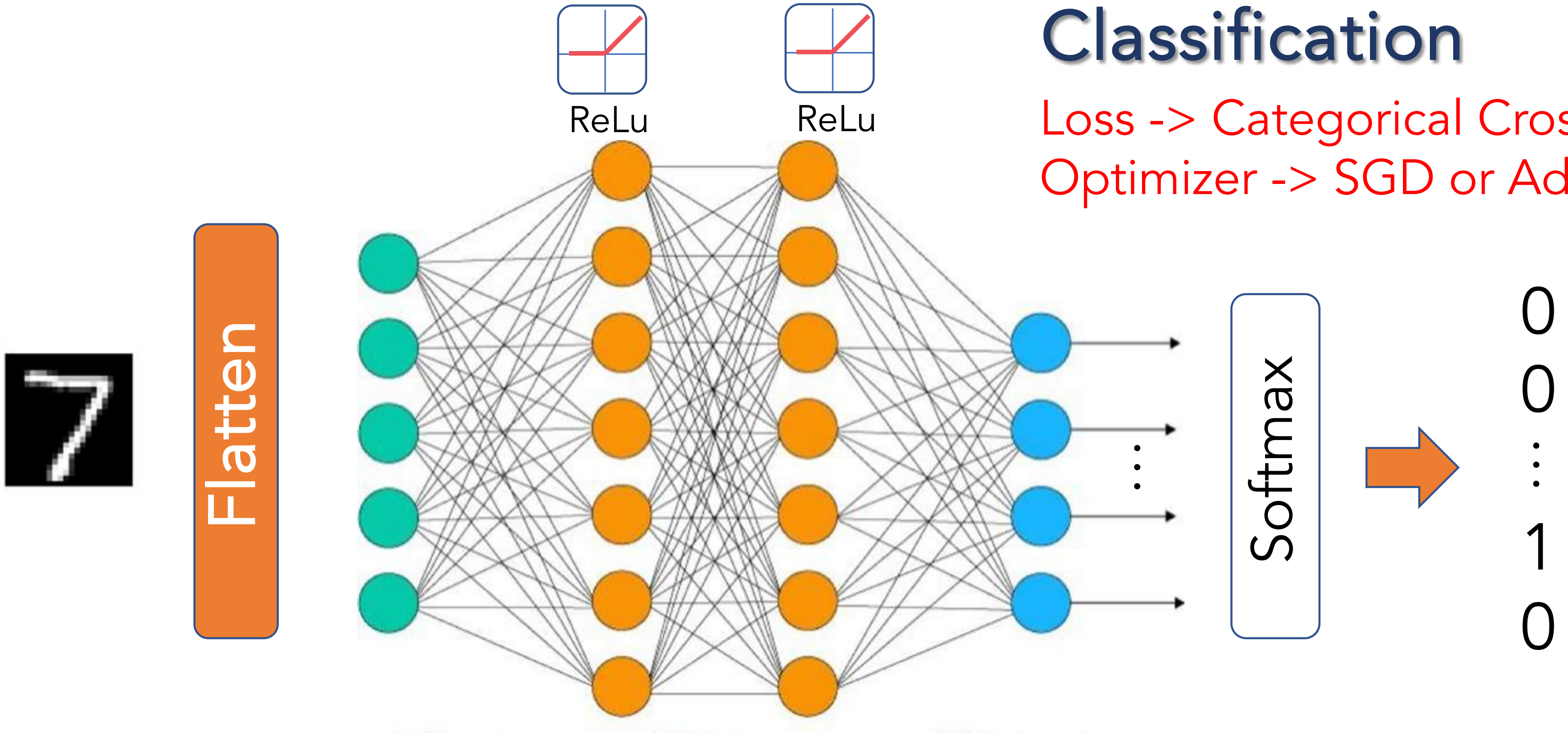
Optimizer -> SGD or Adam



DNNs

Multi-class Classification

Loss -> Categorical Crossentropy *
Optimizer -> SGD or Adam



* or "Sparse Categorical Crossentropy" if label is 1, 2, 3, …

and some issues?




Steps to take

- Get as many examples of shoes as possible
- Train using these examples
- Profit!

FINLAND
na
višjem

Steps to take

- Get as many examples of shoes as possible

- Train using these examples

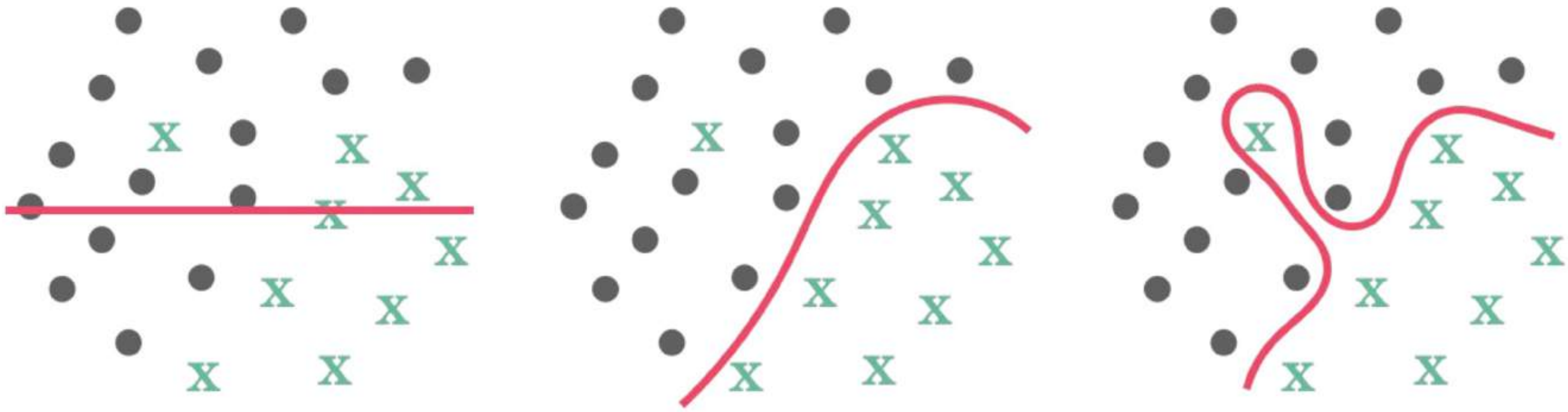
- $\boxed{\text{Profit!}}$

Training accuracy: .920
Training accuracy: .935
Training accuracy: .947
Training accuracy: .961
Training accuracy: .977
Training accuracy: .995

$\boxed{\text{Training accuracy: 1.00}}$

Data

The network **'sees' everything**.
Has no context for measuring how well it does with data it has never previously been exposed to.



Underfitting

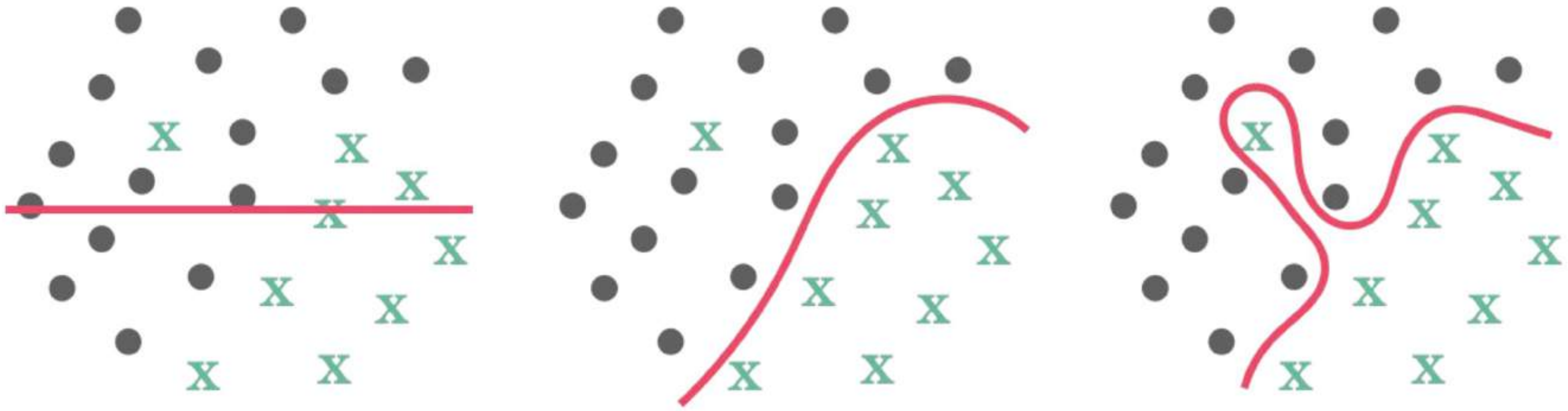
Desired

Overfitting

Correct vs. Overfit Model

Model fitting refers to the accuracy of the model's underlying function as it attempts to analyze data with which it is not familiar.

Underfitting and overfitting are common problems that degrade the quality of the model, as the model fits either not well enough or too well.



Underfitting Desired Overfitting

https://microsoft.github.io/ML-For-Beginners/#/1-Introduction/4-techniques-of-ML/README?id=techniques-of-machine-learning

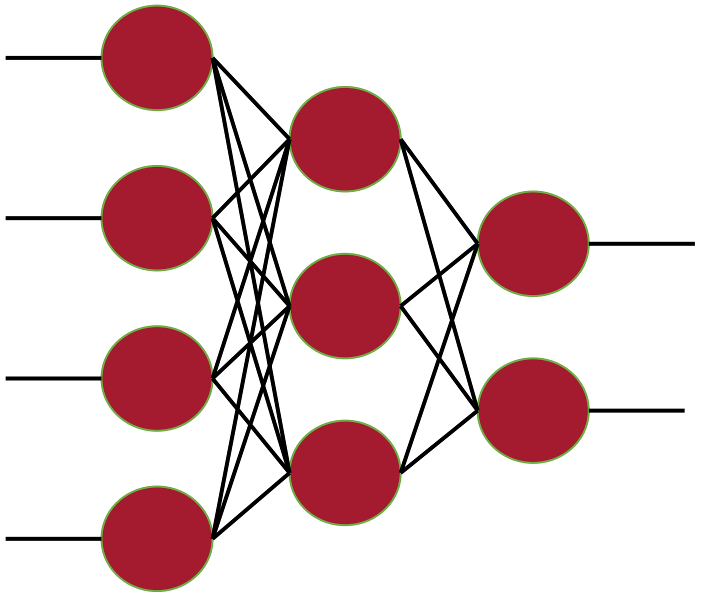
Data	Validation Data

The network 'sees' a subset of your data. You can use the rest to measure its performance against previously unseen data.

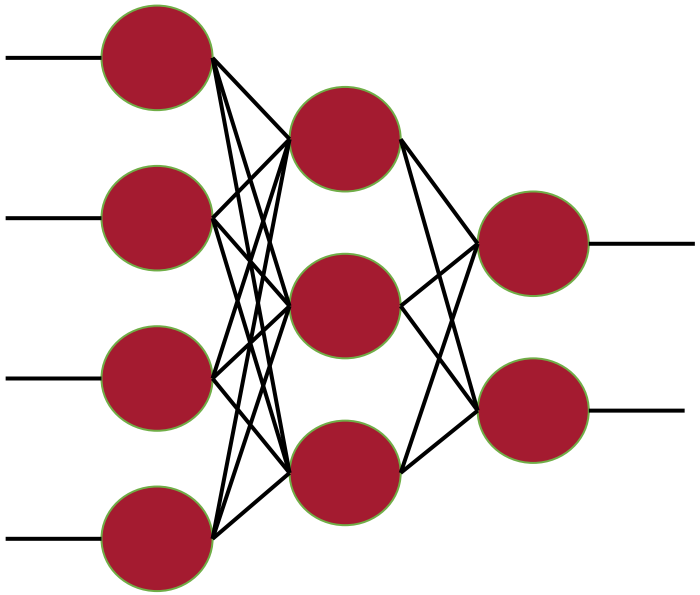
Data	Validation Data	Test Data

The network 'sees' a subset of your data. You can use an unseen subset to measure its accuracy while training (validation), and then another subset to measure its accuracy after it's finished training (test).

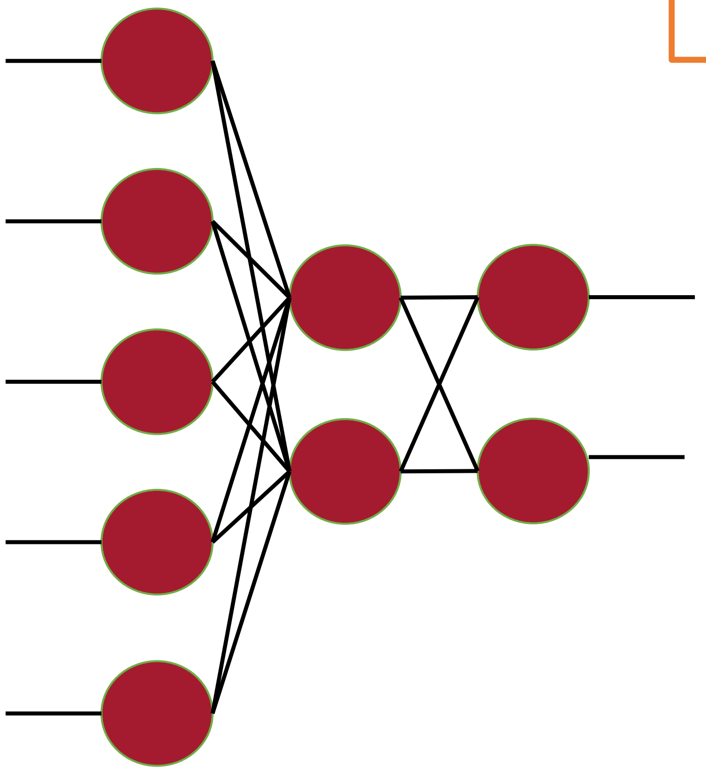
Data	Validation Data	Test Data
Accuracy: 0.999	Accuracy: 0.920	Accuracy: 0.800



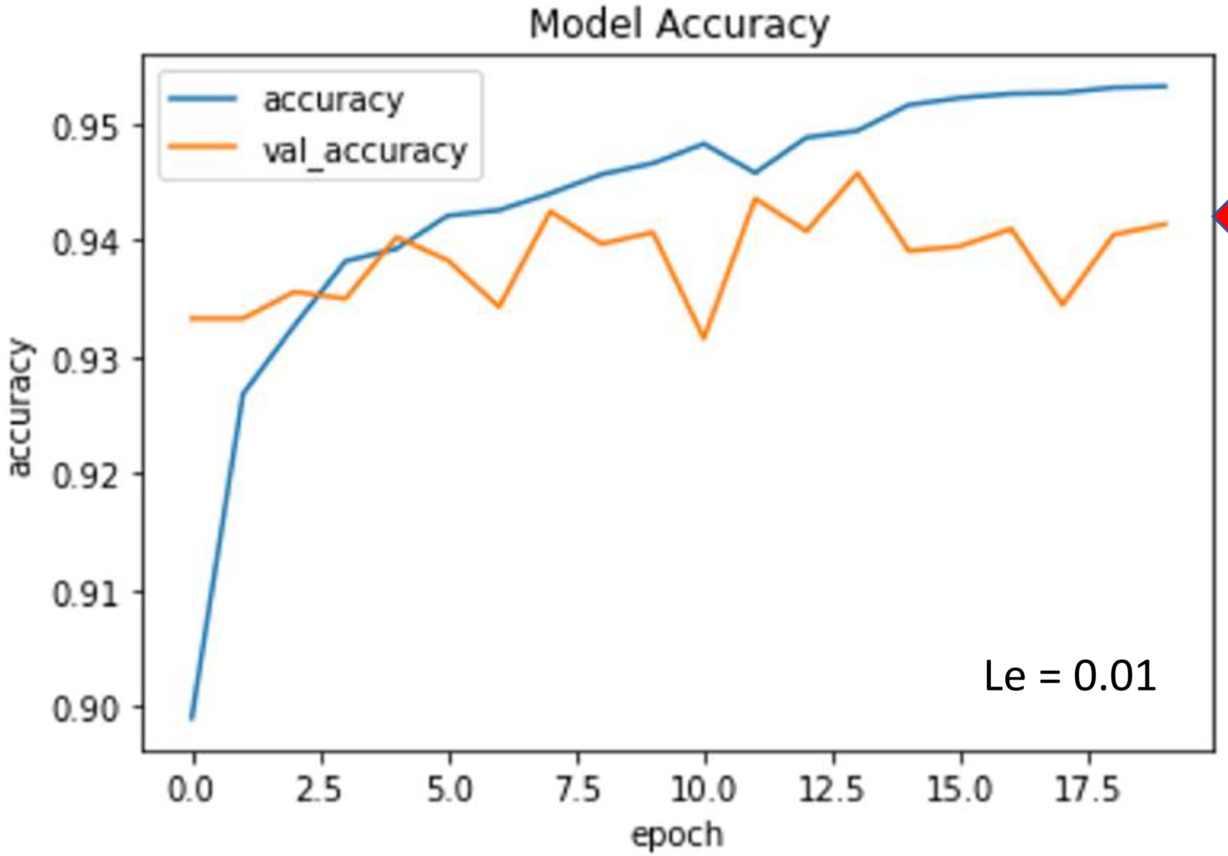
Data	Validation Data	Test Data
Accuracy: 0.999	Accuracy: 0.920	Accuracy: 0.800



Data	Validation Data	Test Data
Accuracy: 0.942	Accuracy: 0.930	Accuracy: 0.925

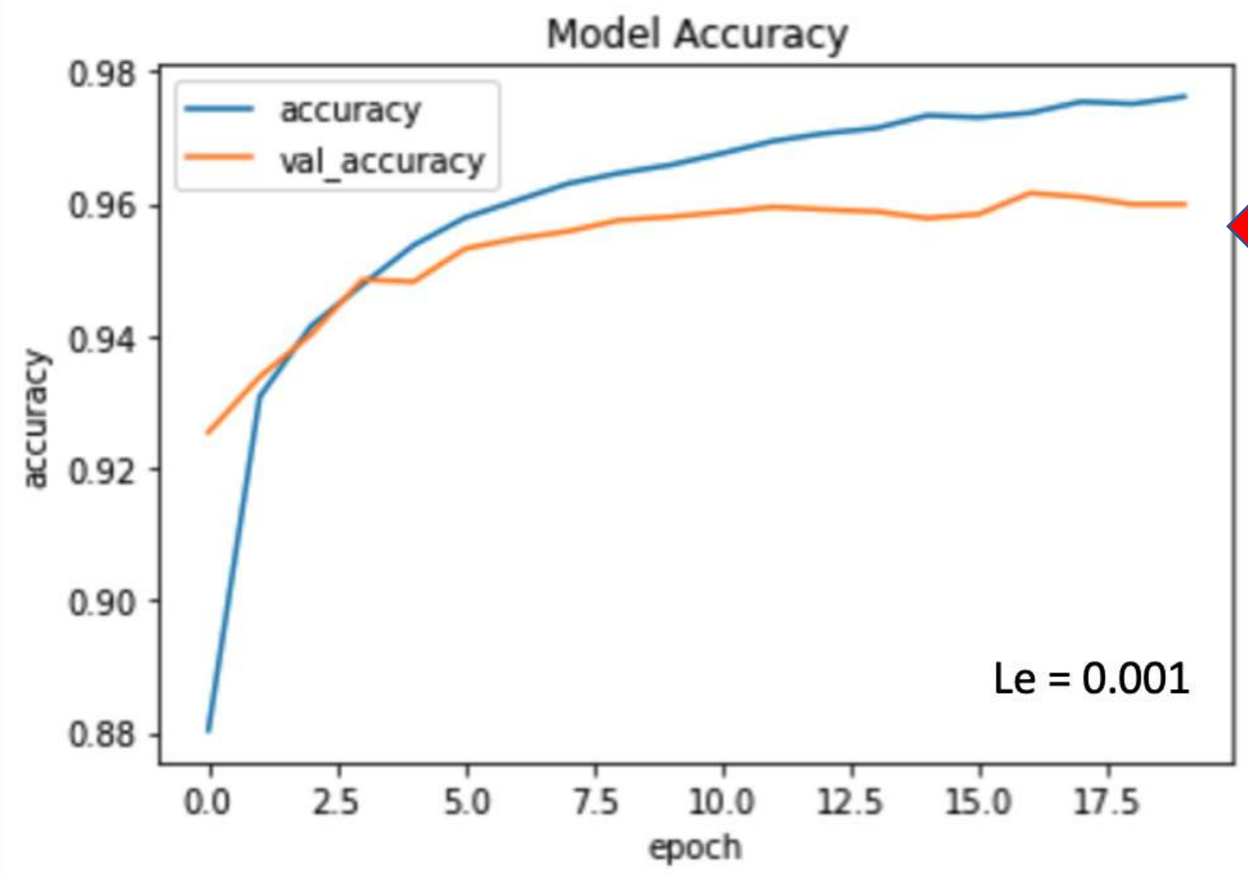


Prevent Overfitting and Imbalanced Data



If <u>validation accuracy seems unstable</u>, could be that Learning Rate is high (try to reduce it).

Prevent Overfitting and Imbalanced Data



If <u>**validation accuracy goes down**</u> (or became stable) , even if train accuracy goes up, means that probably the model is overfitting. In this case the training (epochs) should terminate.

In summary

Model	Train Accuracy	Test Accuracy
A	99,9%	95%
B	87%	87%
C	99,9%	45%

Test accuracy should be lower than train accuracy, but how much less accurate?

Model A is better than model B because it has a higher test accuracy, regardless its difference with the train accuracy.

Model C is a clear case of overfitting as the train accuracy is very high but the test accuracy isn't anywhere near as high.

This distinction is subjective, but comes from knowledge of your problem and data, and what magnitudes of error are acceptable.

https://docs.microsoft.com/en-us/azure/machine-learning/concept-manage-ml-pitfalls

In summary

Training Data -> Used to train model parameters

Validation Data -> Used to determine what model hyperparameters to adjust (and re-training)

Test Data -> Used to get model final performance metric

Classification Model Performance Metrics

Class = [1]

Class = [0]

actual = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

prediction = [0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1]

Confusion Matrix

		predicted condition	
	12 pictures, 8 of cats and 4 of dogs	Cat [1]	Dog [0]
true condition	Cat [1]	6	2
	Dog [0]	1	3

Confusion Matrix

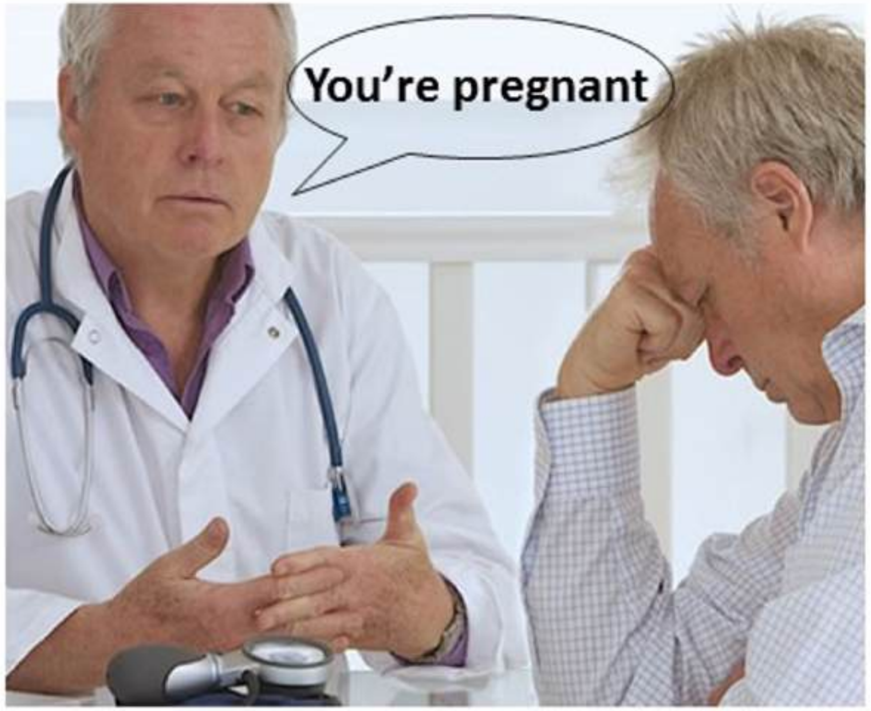
		predicted condition	
	12 pictures, 8 of cats and 4 of dogs	Cat [1]	Dog [0]
true condition	Cat [1]	**True Positive (TP)** 6	**False Negative (FN)** (type II error) 2
	Dog [0]	**False Positive (FP)** (Type I error) 1	**True Negative (TN)** 3

Confusion Matrix

		predicted condition	
	total population (P + N)	prediction positive (PP)	prediction negative (PN)
true condition	condition positive (P)	**True Positive (TP)**	**False Negative (FN)** (type II error)
	condition negative (N)	**False Positive (FP)** (Type I error)	**True Negative (TN)**

Confusion Matrix

Type I error (false positive)



Type II error (false negative)

Precision vs. Accuracy

In a set of measurements:

- **Accuracy** is closeness of the measurements to a specific value
- **Precision** is the closeness of the measurements to each other.

High Precision, High Accuracy

Low Precision, High Accuracy

High Precision, Low Accuracy

Low Precision, Low Accuracy

Accuracy , Precision and Recall

$$\text{Accuracy} = \frac{TP + TN}{(P + N)} = \frac{TP + TN}{(TP + TN + FP + FN)} = \frac{6 + 3}{(6 + 3 + 1 + 2)} = \frac{9}{12} = 0.75$$

$$\text{Precision} = \frac{TP}{(TP + FP)} = \frac{6}{(6 + 1)} = \frac{6}{7} = 0.86 \qquad \frac{\text{Total Positive}}{\text{Total Predict Positive}}$$

$$\text{Recall} = \frac{TP}{(TP + FN)} = \frac{6}{(6 + 2)} = \frac{6}{8} = 0.75 \qquad \frac{\text{Total Positive}}{\text{Total Actual Positive}}$$

(or Sensitivity)

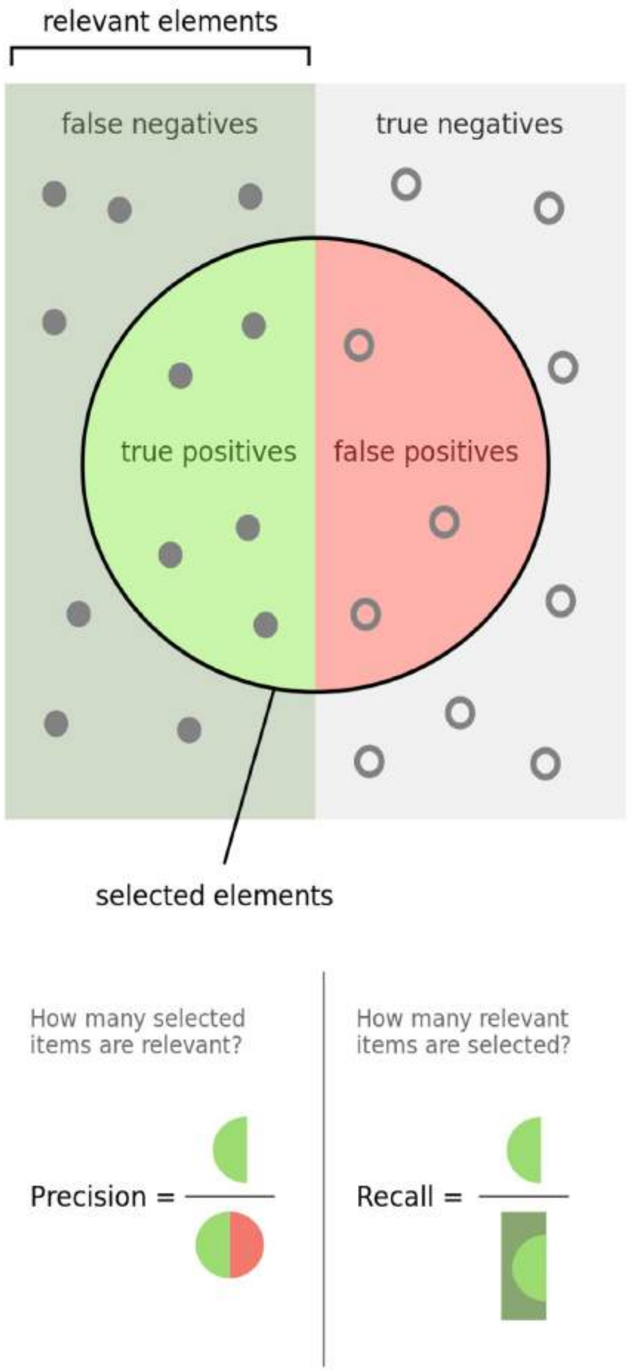
F1-Score

$$F1 = 2 \times \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

$$F1 = 2 \times \frac{(0.86 * 0.75)}{(0.86 + 0.75)} = 2 \times \frac{0.65}{1.61} = 0.80$$

The F1-score is a way of combining the precision and recall of the model

https://en.wikipedia.org/wiki/F-score#Formulation



En resumen

- Alta precisión y alto recall: el modelo de Machine Learning escogido maneja perfectamente esa clase.
- Alta precisión y bajo recall: el modelo de Machine Learning escogido no detecta la clase muy bien, pero cuando lo hace es altamente confiable.
- Baja precisión y alto recall: El modelo de Machine Learning escogido detecta bien la clase, pero también incluye muestras de la otra clase.
- Baja precisión y bajo recall: El modelo de Machine Learning escogido no logra clasificar la clase correctamente.

https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Consejos generales

- La precisión es un gran estadístico, pero es útil únicamente cuando se tienen datasets simétricos (la cantidad de casos de la clase 1 y de las clase 2 tienen magnitudes similares)
- El indicador F1 de la matriz de confusión es útil si se tiene una distribución de clases desigual.
- Elija mayor precisión para conocer qué tan seguro está de los verdaderos positivos, mientras que la sensibilidad o Recall le servirá para saber si no está perdiendo positivos.
- Las Falsas Alarmas: si cree que es mejor en su caso tener falsos positivos que falsos negativos, utilice una sensibilidad alta (Recall), cuando la aparición de falsos negativos le resulta inaceptable pero no le importa tener falsos positivos adicionales (falsas alarmas).
  - Prefiere que algunas personas sanas sean etiquetadas como diabéticas, en lugar de dejar a una persona diabética etiquetada como sana.
- Elija precisión si quiere estar más seguro de sus verdaderos positivos, por ejemplo, correos electrónicos no deseados.
  - Prefiere tener algunos correos electrónicos “no deseados” en su bandeja de entrada, en lugar de tener correos electrónicos “reales” en su bandeja de SPAM.

https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

```python
from sklearn.metrics import classification_report
```

```python
actual = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
prediction = [0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1]
```

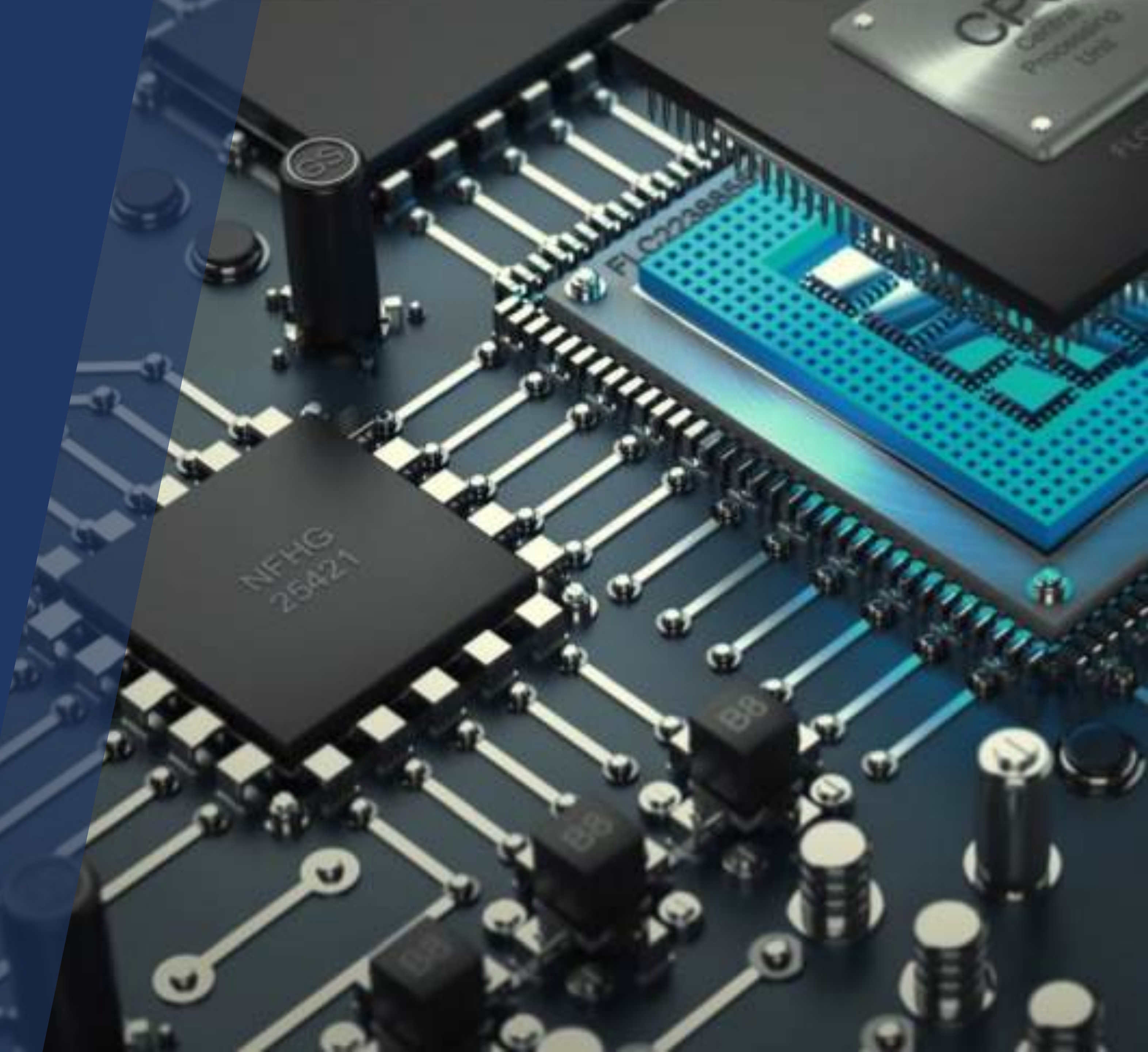
```python
target_names = ['Dogs', 'Cats']
```

```python
print(classification_report(actual, prediction, target_names=target_names))
```

```
              precision    recall  f1-score   support

        Dogs       0.60      0.75      0.67         4
        Cats       0.86      0.75      0.80         8

    accuracy                           0.75        12
   macro avg       0.73      0.75      0.73        12
weighted avg       0.77      0.75      0.76        12
```

Open in Colab

Gracias!



Prof. Diego Méndez Chaves, Ph.D

Associate Professor - Electronics Engineering Department
Director of the Master Program in Internet of Things
Director of the Master Program in Electronics Engineering

https://perfilesycapacidades.javeriana.edu.co/en/persons/diego-mendez-chaves

diego-mendez@javeriana.edu.co