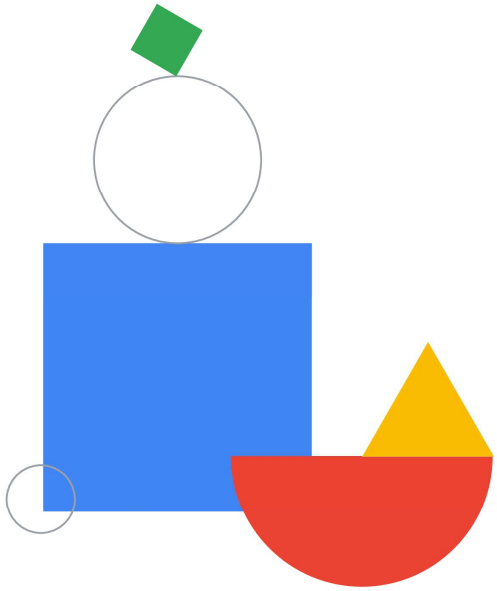
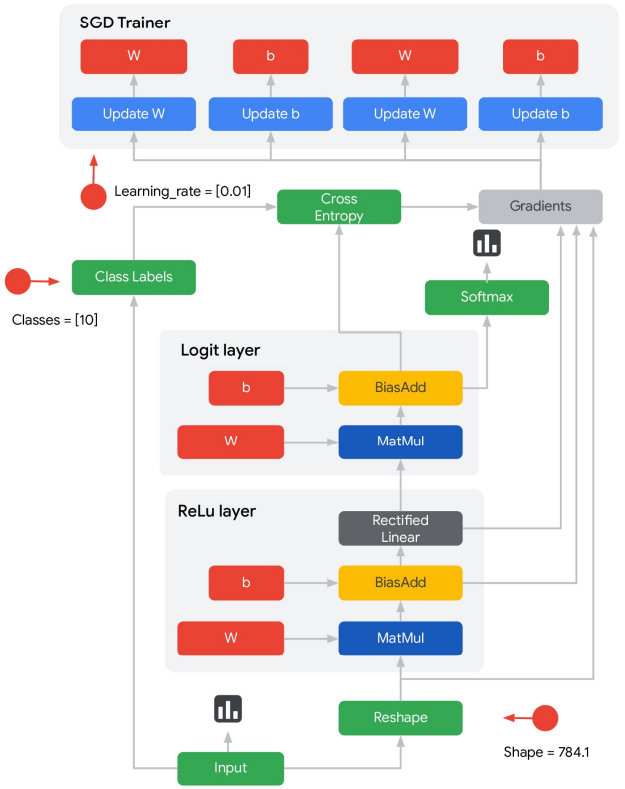
Google Cloud



TensorFlow on Google Cloud

TensorFlow is an open-source, high-performance library for numerical computation that uses directed graphs



A tensor is an N-dimensional array of data

Rank 0 Tensor

scalar

Rank 1 Tensor

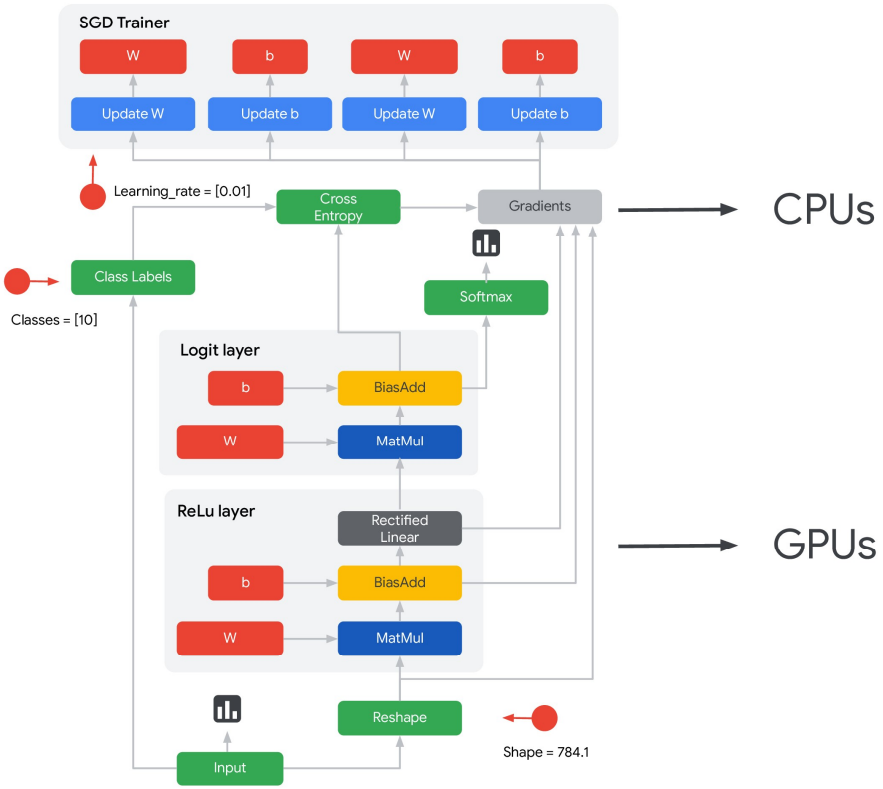
vector

Rank 2 Tensor

matrix

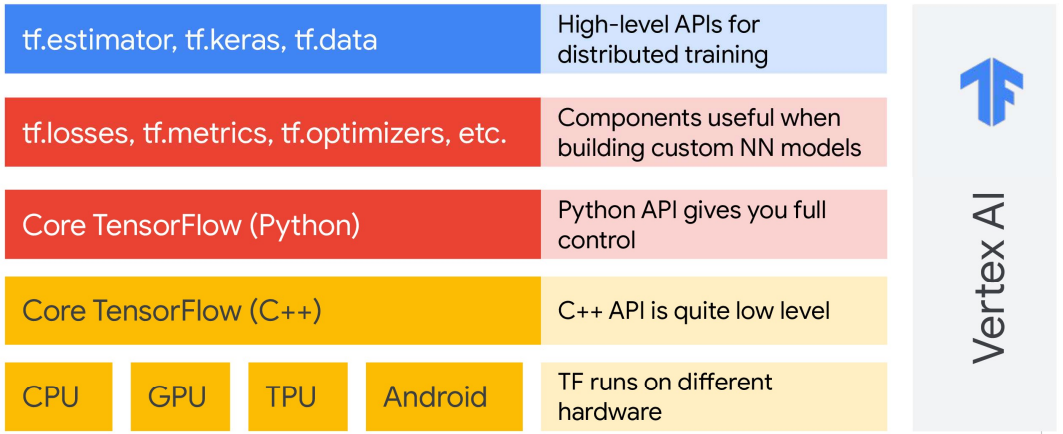
Rank 3 Tensor

Rank 4 Tensor



TensorFlow graphs are portable between different devices

TensorFlow contains multiple abstraction layers.



Note that AI Platform is now Vertex AI.

A tensor is an N-dimensional array of data

	Common name	Rank (Dimension)	Example	Shape of example
	Scalar	0	`x = tf.constant(3)`	()
	Vector	1	`x = tf.constant([3, 5, 7])`	(3,)
	Matrix	2	`x = tf.constant([[3, 5, 7],` `[4, 6, 8]])`	(2, 3)

	3D Tensor	3	`tf.constant([[[3, 5, 7],[4, 6, 8]],`
			`[[1, 2, 3],[4, 5, 6]]`
			`])`
				(2, 2, 3)

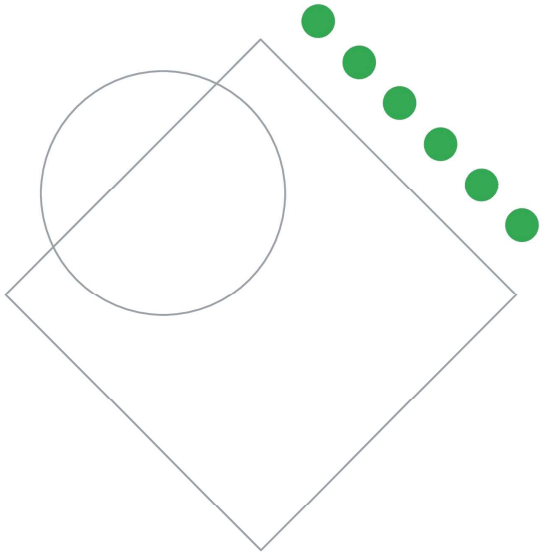
	nD Tensor	n	`x1 = tf.constant([2, 3, 4])`
			`x2 = tf.stack([x1, x1])`
			`x3 = tf.stack([x2, x2, x2, x2])`
			`x4 = tf.stack([x3, x3])`
			`...`
				(3,)
				(2, 3)
				(4, 2, 3)
				(2, 4, 2, 3)

A tensor is an N-dimensional array of data

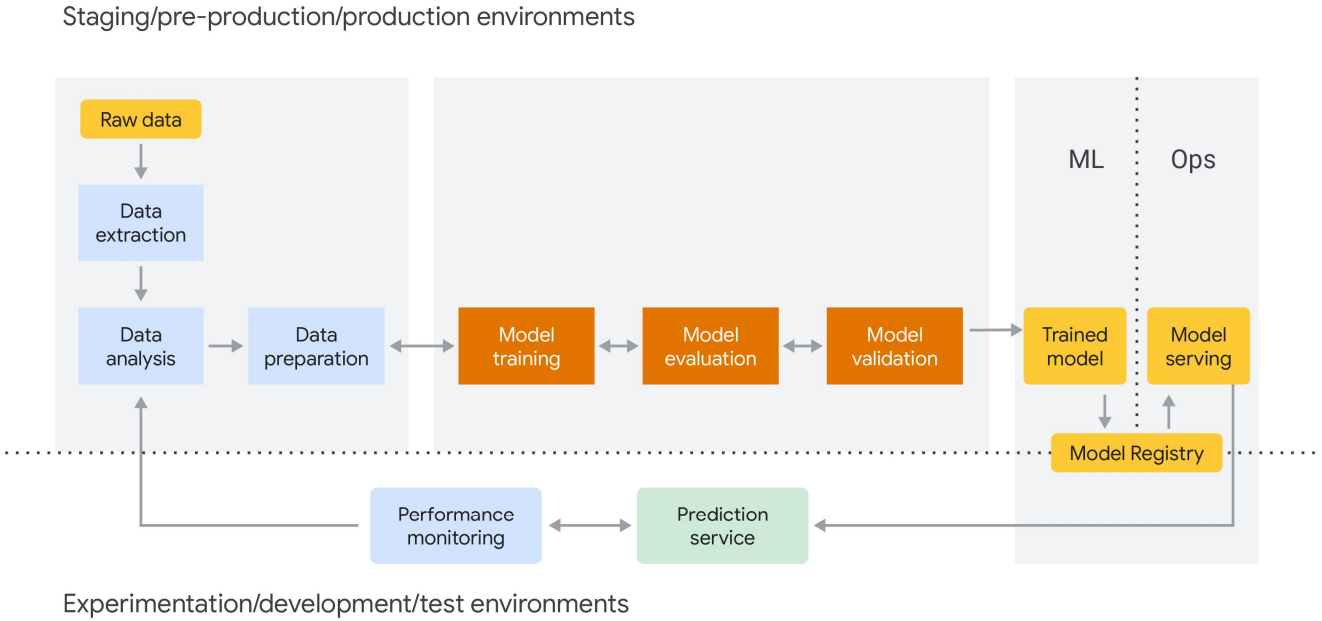
They behave like numpy n-dimensional arrays except that:

- tf.constant produces constant tensors
- tf.Variable produces tensors that can be modified

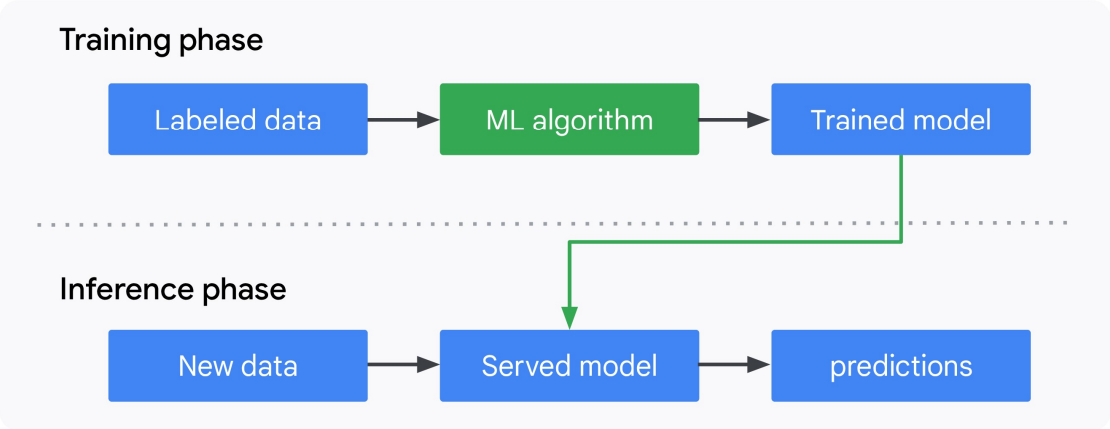
Design and Build an Input Data Pipeline



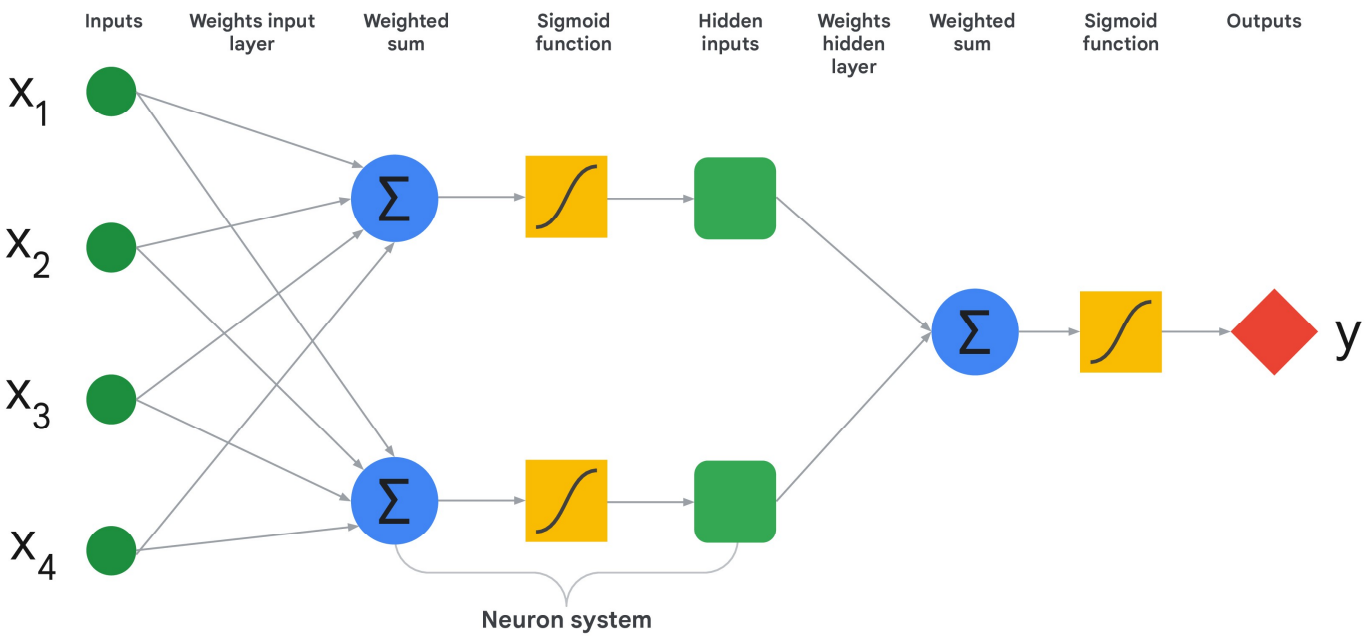
An ML recap


Staging/pre-production/production environments
Raw data
Data extraction
Data analysis
Data preparation
Model training
Model evaluation
Model validation
ML
Ops
Trained model
Model serving
Model Registry
Performance monitoring
Prediction service
Experimentation/development/test environments

An ML recap


Training phase
Labeled data
ML algorithm
Trained model
Inference phase
New data
Served model
predictions

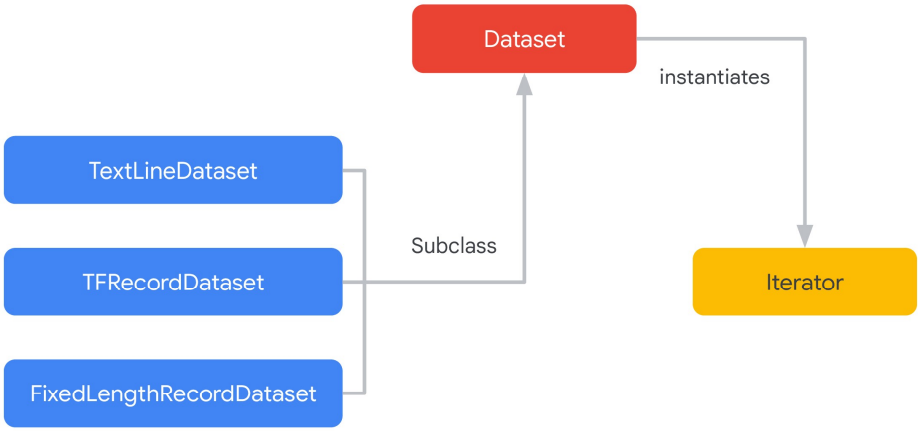
An ML recap


Inputs
Weights input layer
Weighted sum
Sigmoid function
Hidden inputs
Weights hidden layer
Weighted sum
Sigmoid function
Outputs
x_1
x_2
x_3
x_4
Σ
Σ
Σ
y
Neuron system

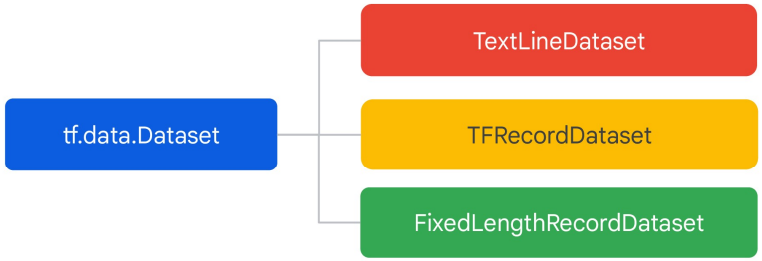
tf.data API

1. Build complex input pipelines from simple, reusable pieces.
2. Build pipelines for multiple data types
3. Handle large amounts of data; perform complex transformations

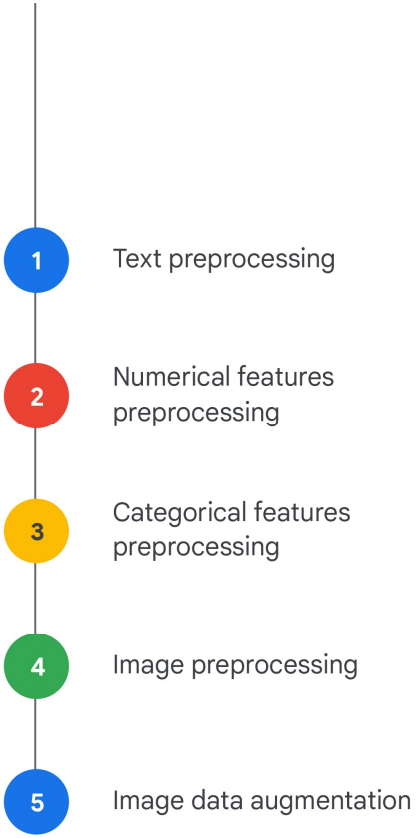
Multiple ways to feed TensorFlow models with data



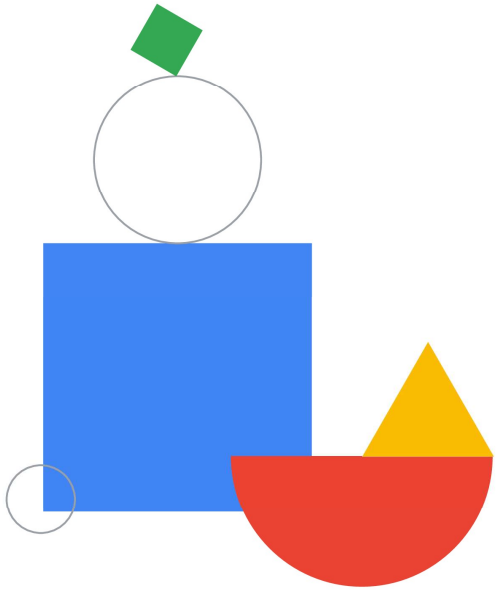
Datasets can be created from different file formats



Keras preprocessing layers



Google Cloud



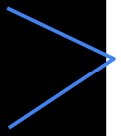
Building Neural Networks with the TensorFlow and Keras API

Keras is built-in to TF 2.x

- User-friendly
- Modular and composable
- Easy to extend

```
%tensorflow_version 2.x
import tensorflow as tf
from tensorflow import keras
(x_train, y_train), (x_test, y_test) =
keras.datasets.mnist.load_data()

# Define your model
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```



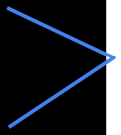
A deeper neural network

Compiling a Keras model

```
def rmse(y_true, y_pred):
    return tf.sqrt(tf.reduce_mean(tf.square(y_pred - y_true)))


model.compile(optimizer="adam", loss="mse", metrics=[rmse, "mse"])
```

```
%tensorflow_version 2.x
import tensorflow as tf
from tensorflow import keras
(x_train, y_train), (x_test, y_test) =
keras.datasets.mnist.load_data()
# Define your model
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
# Configure and train
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```



A deeper neural network

Once trained, the model can be used for prediction

```
predictions = model.predict(input_samples,        )
```

returns a Numpy array of predictions

With the predict() method you can pass

- a Dataset instance
- Numpy array
- a TensorFlow tensor, or list of tensors
- a generator of input samples

To serve our model for others to use, we export the model file and deploy the model as a service.

SavedModel is the universal serialization format for TensorFlow models

```
OUTPUT_DIR = "./export/savedmodel"
shutil.rmtree(OUTPUT_DIR, ignore_errors=True)

EXPORT_PATH = os.path.join(OUTPUT_DIR,
datetime.datetime.now().strftime("%Y%m%d%H%M%S"))

tf.saved_model.save(model, EXPORT_PATH)
```

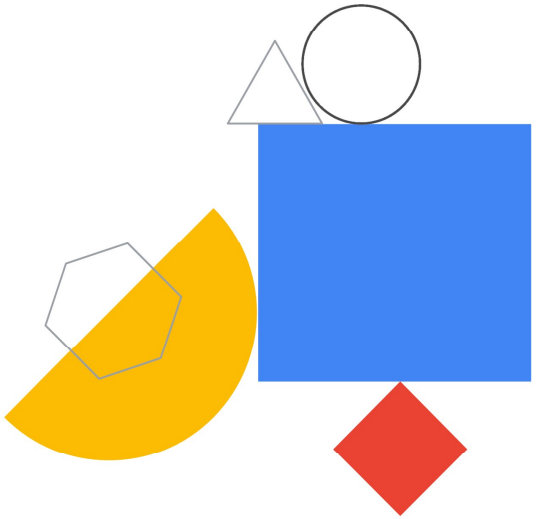
the directory in which to write the SavedModel

exports a model object to a SavedModel format

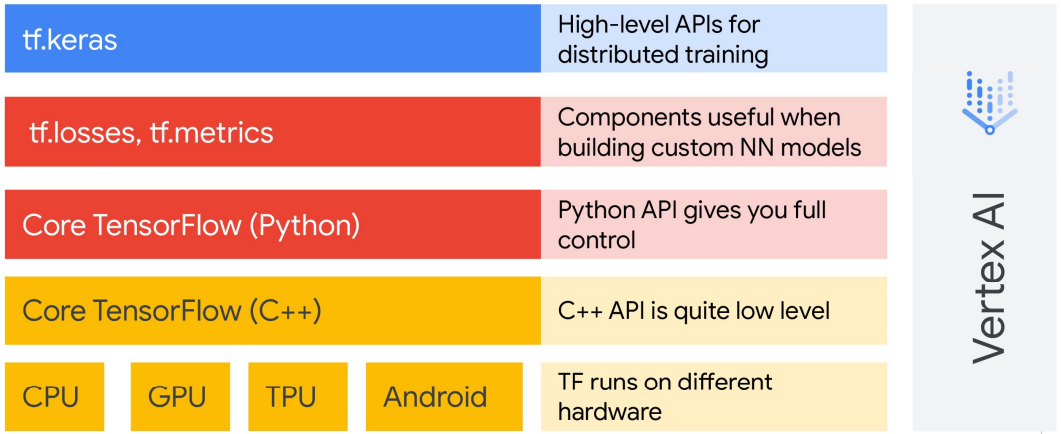
a trackable object such as a trained keras model

Google Cloud

Training at Scale with Vertex AI



We will use distributed TensorFlow on Vertex AI



Thanks