PYTHON

TensorFlow

FULL COURSE

Keras • Neural Networks • AI



LONG VIDEO • 12 MIN

PYTHON • TENSORFLOW FULL COURSE

Python TensorFlow Full Course

Tensors, Keras, neural networks, image classification and deployment

FREE FULL COURSE • MyWebUniversity.com



Wahidullah Lutfy
Founder, MyWebUniversity • CS Teacher

TENSORFLOW

What You Will Learn

Machine learning
What it is, and how it differs from normal programming.

Tensors
Shapes, types, operations and GPUs.

Gradients
How models learn: loss and GradientTape.

Keras models
Layers, compile, fit, evaluate and predict.

Image classification
Recognize handwritten digits with MNIST.

Real-world skills
Overfitting, callbacks, saving and deployment.

Python TensorFlow • Full Course

PART 1 · FOUNDATIONS

TENSORFLOW

What Is Machine Learning?

Normal programming

Rules + data → answers. You write every rule yourself.

Machine learning

Data + answers → rules. The computer learns the rules from examples.

Neural networks

Layers of simple units with adjustable weights.

TensorFlow and Keras

TensorFlow does the math fast; Keras makes building models simple.

TENSORFLOW

Install TensorFlow and Meet Tensors

Python 3.12

```
$ pip install tensorflow
>>> import tensorflow as tf
>>> tf.__version__
'2.18.0'

>>> t = tf.constant([[1., 2.], [3., 4.]])
>>> t.shape, t.dtype
(TensorShape([2, 2]), tf.float32)
>>> t * 10, tf.reduce_sum(t), t @ t

>>> v = tf.Variable(1.0)
>>> v.assign_add(0.5)
>>> t.numpy()
array([[1., 2.], [3., 4.]], dtype=float32)
```

Check versions

TensorFlow supports specific Python versions. Check tensorflow.org if pip finds no match.

Tensors are like arrays

Shapes, dtypes and math like NumPy, but they can run on a GPU.

Constant vs Variable

Variables can change: they hold the weights a model learns.

TENSORFLOW

How Models Learn: Loss and Gradients

learn.py — Python 3.12

```
w = tf.Variable(0.0)                    # start with a bad guess
x, y_true = 3.0, 6.0                    # we want w * 3 = 6

for step in range(50):
   with tf.GradientTape() as tape:
       loss = (w * x - y_true) ** 2     # how wrong are we?
   grad = tape.gradient(loss, w)        # which way to move?
   w.assign_sub(0.01 * grad)            # take a small step

print(w.numpy())
2.0
```

Loss

A number that measures how wrong the model is. Lower is better.

Gradient

Tells us which direction to change each weight to reduce the loss.

Gradient descent

Repeat small steps downhill. This is how every neural network learns.

Python TensorFlow • Full Course

PART 2 · BUILDING MODELS WITH KERAS

TENSORFLOW

Your First Keras Model

first_model.py — Python 3.12

```
import numpy as np, tensorflow as tf
xs = np.array([-1, 0, 1, 2, 3, 4], dtype=float)
ys = 2 * xs - 1                              # the hidden rule

model = tf.keras.Sequential([tf.keras.Input(shape=(1,)),
tf.keras.layers.Dense(1)])
model.compile(optimizer="sgd", loss="mean_squared_error")
model.fit(xs, ys, epochs=500, verbose=0)

print(model.predict(np.array([[10.0]]), verbose=0))
[[18.98]]
model.summary()
```

Sequential
A stack of layers, one after another.

compile
Choose the optimizer (how to learn) and the loss (what to minimize).

fit and predict
fit trains on examples; predict uses what was learned.

TENSORFLOW

Project: Recognize Handwritten Digits (MNIST)

mnist.py — Python 3.12

```
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0     # scale pixels to
0..1

model = tf.keras.Sequential([
   tf.keras.Input(shape=(28, 28)),
   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(128, activation="relu"),
   tf.keras.layers.Dropout(0.2),
   tf.keras.layers.Dense(10)])

model.compile(optimizer="adam", metrics=["accuracy"],
   loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True))
model.fit(x_train, y_train, epochs=5, validation_split=0.1)
model.evaluate(x_test, y_test)    # about 98% accuracy
```

The data

70,000 images of digits, 28 × 28 pixels, with the correct label for each.

The layers

Flatten → Dense 128 with ReLU → Dropout → 10 outputs, one per digit.

Evaluate on test data

Accuracy on images the model has never seen is the true score.

TENSORFLOW

Making Predictions

mnist.py — Python 3.12

```
probs = tf.nn.softmax(model.predict(x_test[:3], verbose=0)).numpy()
probs.argmax(axis=1), y_test[:3]
(array([7, 2, 1]), array([7, 2, 1], dtype=uint8))


probs[0].round(3)
array([0.   , 0.   , 0.   , 0.001, 0.   , 0.   , 0.   , 0.998, 0.   ,
0.001], dtype=float32)
```

softmax
Turns the 10 raw scores into probabilities that add up to 1.

argmax
The digit with the highest probability is the prediction.

Confidence
99.8% sure it is a 7. Always look at confidence, not just the answer.

Python TensorFlow • Full Course

PART 3 · REAL-WORLD SKILLS

TENSORFLOW

Overfitting, Callbacks and Better Models

mnist.py — Python 3.12

```
stop = tf.keras.callbacks.EarlyStopping(monitor="val_loss", patience=2,
restore_best_weights=True)
history = model.fit(x_train, y_train, epochs=30, validation_split=0.1,
callbacks=[stop])


cnn = tf.keras.Sequential([tf.keras.Input(shape=(28, 28, 1)),
   tf.keras.layers.Conv2D(32, 3, activation="relu"),
tf.keras.layers.MaxPooling2D(),
   tf.keras.layers.Flatten(), tf.keras.layers.Dense(10)])
```

Overfitting

Great on training data, poor on new data: the model memorized instead of learning.

EarlyStopping

Stops training when validation loss stops improving, and keeps the best weights.

CNNs for images

Conv2D layers learn local patterns like edges and curves. Often over 99% on MNIST.

TENSORFLOW

Save, Load and Deploy

deploy.py — Python 3.12

```
model.save("digits.keras")
loaded = tf.keras.models.load_model("digits.keras")


converter = tf.lite.TFLiteConverter.from_keras_model(loaded)
open("digits.tflite", "wb").write(converter.convert())


print(tf.config.list_physical_devices("GPU"))
[]   # empty means CPU only
```

.keras files

Save the architecture, weights and settings in one file.

Mobile and edge

TensorFlow Lite runs models on phones and small devices.

GPU check

GPUs speed up training a lot. On Windows, use WSL2 for GPU support.

TENSORFLOW

Recap and Practice

Remember

Data + answers → rules. Build, compile, fit, evaluate, predict.

Try it yourself

Train on Fashion-MNIST (clothes instead of digits) and compare accuracy.

Watch out

Judge models on test data. Check for bias, and use AI responsibly.