Keras is a powerful and easy-to-use deep learning library for TensorFlow that provides a high-level neural network API to develop and evaluate deep learning models.

A Basic Example
>>> import numpy as np
>>> from tensorflow.keras.models import Sequential
>>> model = Sequential()
>>> model.add(Dense(12, activation='relu'))
>>> model.add(Dense(8, activation='relu'))
>>> model.add(Dense(4, activation='relu'))
>>> model.add(Dense(2, activation='relu'))
>>> model.add(Dense(10))

You can also resort to the

Data
Keras Data Sets
>>> from tensorflow.keras.datasets import boston_housing, imdb, mnist, cifar10, coco
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()

Other
>>> from urllib.request import urlopen

Preprocessing
Sequence Padding
>>> from tensorflow.keras.preprocessing import sequence
>>> x_train = sequence.pad_sequences(x_train, maxlen=80)

One-Hot Encoding
>>> from tensorflow.keras.utils import to_categorical
>>> x_train = to_categorical(x_train, num_classes)

Train and Test Sets
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)

Optimization Parameters
from tensorflow.keras.optimizers import Adam
opt = Adam(learning_rate=0.001)

Evaluation Your Model’s Performance
score = model.evaluate(x_test, y_test)

Save/Reload Models
model2.save('my_model.h5')
model2 = load_model('my_model.h5')

Model Fine-tuning
Model Architecture
Sequential Model
>>> from tensorflow.keras.models import Sequential
>>> model = Sequential()
>>> model.add(Dense(10))

Multilayer Perceptron (MLP)
Binary Classification
>>> from tensorflow.keras.models import Sequential
>>> model = Sequential()
>>> model.add(Dense(10, activation='sigmoid'))

Convolutional Neural Network (CNN)
>>> from tensorflow.keras.layers import Conv2D, Flatten
>>> model.add(Conv2D(32, (3,3), padding='same'))

Recurrent Neural Network (RNN)
>>> from tensorflow.keras.layers import LSTM
>>> model.add(LSTM(128, return_sequences=True))

Inspect Model
MLP: Binary Classification
>>> model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

MLP: Multi-Class Classification
>>> model.compile(optimizer='adam', loss='softmax', metrics=['accuracy'])

MLP: Regression
>>> model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

Compile Model
Model Training
>>> model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

Evaluate Your Model’s Performance
>>> score = model.evaluate(x_test, y_test)

Save/Reload Models
>>> model2.save('my_model.h5')

Model2 = load_model('my_model.h5')

Model Fine-tuning
Optimization Parameters
from tensorflow.keras.optimizers import Adam
opt = Adam(learning_rate=0.001)

Early Stopping
from tensorflow.keras.callbacks import EarlyStopping
early_stopping_monitor = EarlyStopping(monitor='val_loss', patience=3)

Train and Test Sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)

Data
Keras Data Sets
>>> from tensorflow.keras.datasets import boston_housing, imdb, mnist, cifar10, coco
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()

Other
>>> from urllib.request import urlopen

Preprocessing
Sequence Padding
>>> from tensorflow.keras.preprocessing import sequence
>>> x_train = sequence.pad_sequences(x_train, maxlen=80)

One-Hot Encoding
>>> from tensorflow.keras.utils import to_categorical
>>> x_train = to_categorical(x_train, num_classes)

Train and Test Sets
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)

Standardization/Normalization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler(feature_range=(0,1))
standaized_X = scaler.transform(x_train)
standaized_X_test = scaler.transform(x_test)