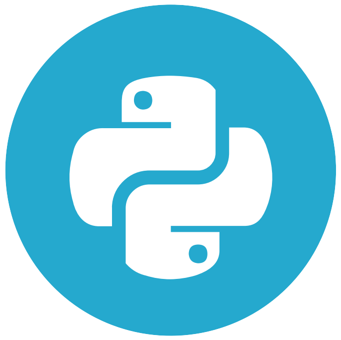


# Keras Cheat Sheet: Neural Networks in Python

April 25th, 2017 in Python

Karlijn Willems

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

We recently launched one of the first online interactive deep learning course using Keras 2.0, called "Deep Learning in Python.

Now, DataCamp has created a Keras cheat sheet for those who have already taken the course and that still want a handy one-page reference or for those who need an extra push to get started.

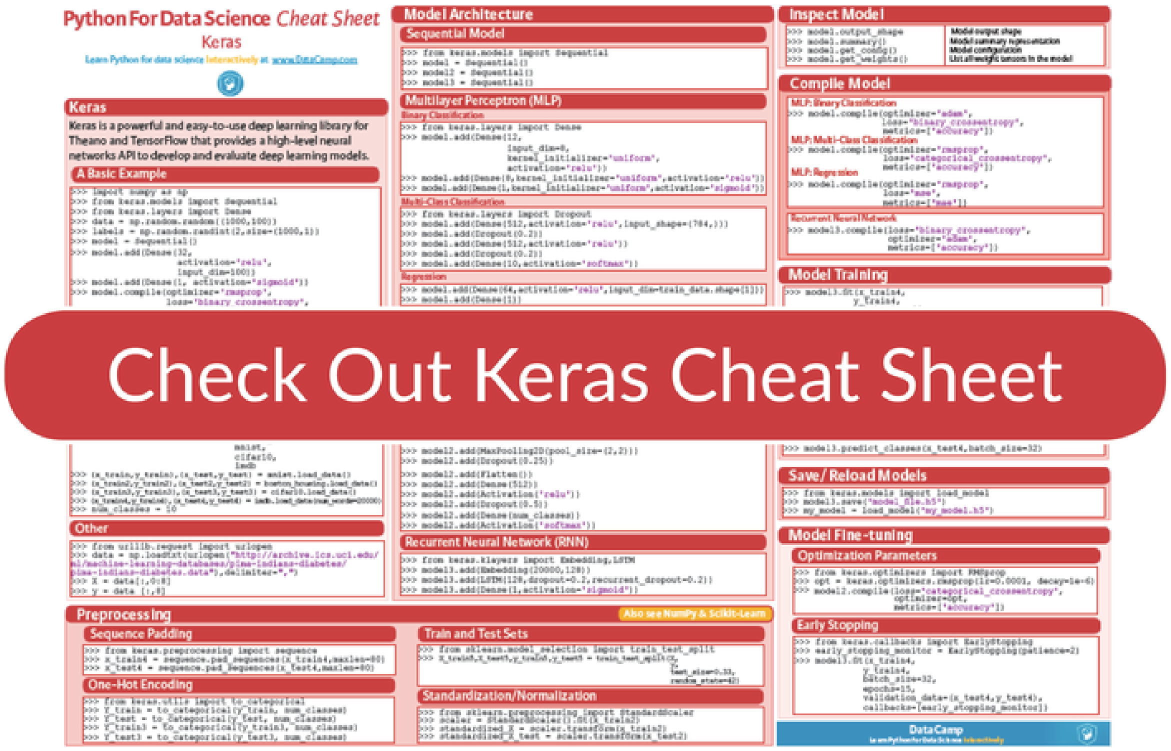
In no time, this Keras cheat sheet will make you familiar with how you can load datasets from the library itself, preprocess the data, build up a model architecture, and compile, train, and evaluate it. As there is a considerable amount of freedom in how you build up

Furthermore, you'll also see some examples of how to inspect your model, and how you can save a
find examples of how you can predict values for test data and how you can fine tune your models b
parameters and early stopping.

learners and start one of our interactive tutorials today!

**Learn R** **Learn Python**

In short, you'll see that this cheat sheet not only presents you with the six steps that you can go through to make neural networks in Python with the Keras library.



**Python For Data Science** *Cheat Sheet*

Keras

Learn Python for data science Interactively at www.DataCamp.com

**Keras**

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

**A Basic Example**

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2,size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32,
                    activation='relu',
                    input_dim=100))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
                  loss='binary_crossentropy',
```

**Model Architecture**

**Sequential Model**

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

**Multilayer Perceptron (MLP)**

Binary Classification

```
>>> from keras.layers import Dense
>>> model.add(Dense(12,
                    input_dim=8,
                    kernel_initializer='uniform',
                    activation='relu'))
>>> model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
>>> model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
```

Multi-Class Classification

```
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

Regression

```
>>> model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

**Inspect Model**

| | |
|---|---|
| `>>> model.output_shape` | Model output shape |
| `>>> model.summary()` | Model summary representation |
| `>>> model.get_config()` | Model configuration |
| `>>> model.get_weights()` | List all weight tensors in the model |

**Compile Model**

MLP: Binary Classification

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

MLP: Multi-Class Classification

```
>>> model.compile(optimizer='rmsprop',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

MLP: Regression

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

Recurrent Neural Network

```
>>> model3.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

**Model Training**

```
>>> model3.fit(x_train4,
               y_train4,
```

Check Out Keras Cheat Sheet

```
                        mnist,
                        cifar10,
                        imdb
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4),(x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
```

**Other**

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/
ml/machine-learning-databases/pima-indians-diabetes/
pima-indians-diabetes.data"),delimiter=",")
>>> X = data[:,0:8]
>>> y = data [:,8]
```

```
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

**Recurrent Neural Network (RNN)**

```
>>> from keras.klayers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

```
>>> model3.predict_classes(x_test4,batch_size=32)
```

**Save/ Reload Models**

```
>>> from keras.models import load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

**Model Fine-tuning**

**Optimization Parameters**

```
>>> from keras.optimizers import RMSprop
>>> opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
                   optimizer=opt,
                   metrics=['accuracy'])
```

**Early Stopping**

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               validation_data=(x_test4,y_test4),
               callbacks=[early_stopping_monitor])
```

**Preprocessing** Also see NumPy & Scikit-Learn

**Sequence Padding**

```
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

**One-Hot Encoding**

```
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
```

**Train and Test Sets**

```
>>> from sklearn.model_selection import train_test_split
>>> X_train5,X_test5,y_train5,y_test5 = train_test_split(X,
                                                         y,
                                                         test_size=0.33,
                                                         random_state=42)
```

**Standardization/Normalization**

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(x_train2)
>>> standardized_X = scaler.transform(x_train2)
>>> standardized_X_test = scaler.transform(x_test2)
```

DataCamp

Learn Python for Data Science Interactively

**(Click above to download a printable version or read the online version below.)**

## Python For Data Science Cheat Sheet: Keras

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

## A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2,size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32, activation='relu', input_dim=100))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
>>>  model.fit(data,labels,epochs=10,batch_size=32)
>>>  predictions = model.predict(data)
```

## Data

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation` .

### Keras Data Sets

```
>>> from keras.datasets import boston_housing, mnist, cifar10, imdb
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
```

### Other

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-i
>>> X = data[:,0:8]
>>> y = data [:,8]
```

## Preprocessing

### Sequence Padding

```
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

### One-Hot Encoding

```
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
```

### Train And Test Sets

```
>>> from sklearn.model_selection import train_test_split
>>> X_train5, X_test5, y_train5, y_test5 = train_test_split(X, y, test_size=0.33, random_state=42)
```

### Standardization/Normalization

```
>>> standardized_X_test = scaler.transform(x_test2)
```

## Model Architecture

### Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

### Multi-Layer Perceptron (MLP)

### Binary Classification

```
>>> from keras.layers import Dense
>>> model.add(Dense(12, input_dim=8, kernel_initializer='uniform', activation='relu'))
>>> model.add(Dense(8, kernel_initializer='uniform', activation='relu'))
>>> model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))
```

### Multi-Class Classification

```
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

### Regression

### Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation, Conv2D, MaxPooling2D, Flatten
>>> model2.add(Conv2D(32, (3,3), padding='same', input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32, (3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Conv2D(64, (3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64, (3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

### Recurrent Neural Network (RNN)

```
>>> from keras.klayers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

## Inspect Model

### Model output shape

```
>>> model.output_shape
>>> model.summary()
```

### Model configuration

```
>>> model.get_config()
```

### List all weight tensors in the model

```
>>> model.get_weights()
```

## Compile Model

### Multi-Layer Perceptron (MLP)

### MLP: Binary Classification

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

### MLP: Multi-Class Classification

```
>>> model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

### MLP: Regression

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

## Model Training

```
>>> model3.fit(x_train4, y_train4, batch_size=32, epochs=15, verbose=1, validation_data=(x_test4, y_test4))
```

## Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test, y_test, batch_size=32)
```

## Prediction

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

## Save/Reload Models

```
>>> from keras.models import load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

## Model Fine-Tuning

### Optimization Parameters

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4, y_train4, batch_size=32, epochs=15, validation_data=(x_test4, y_test4), callbacks=[early_st
```

## Going Further

Begin with our Keras tutorial for beginners, in which you'll learn in an easy, step-by-step way how to explore and preprocess the wine quality data set, build up a multi-layer perceptron for classification and regression tasks, compile, fit and evaluate the model and fine-tune the model that you have built.

Also, don't miss out on our Scikit-Learn cheat sheet, NumPy cheat sheet and Pandas cheat sheet!

What do you think?

Python

## Up Next

learners and start one of our interactive tutorials today!

**Learn R** **Learn Python**

## New Course! Supervised Learning in R: Classification

R Programming

73 views

This beginner-level introduction to machine learning covers four of the most common classification algorithms. You will come away with a b…

September 27th, 2017 in Blog

## New Course: Case Studies in Statistical Thinking!

Python

1,135 views

Hone your applied data science skills in Python by doing a variety of case studies across multiple disciplines. Use data science to solve …

September 20th, 2017 in Blog