Need help with Deep Learning? Take the FREE Mini-Course

How to Use the Keras Functional API for Deep Learning

by **Jason Brownlee** on October 27, 2017 in **Deep Learning**

The Keras Python library makes creating deep learning models fast and easy.

The sequential API allows you to create models layer-by-layer for most problems. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs.

The functional API in Keras is an alternate way of creating models that offers a lot more flexibility, including creating more complex models.

In this tutorial, you will discover how to use the more fl ning models.

After completing this tutorial, you will know:

- The difference between the Sequential and Funct
- How to define simple Multilayer Perceptron, Conv Network models using the functional API.
- How to define more complex models with shared

Let's get started.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

- **Update Nov/2017**: Update note about hanging dimension for input layers only affecting 1D input, thanks Joe.

Tutorial Overview

This tutorial is divided into 6 parts; they are:

1. Keras Sequential Models
2. Keras Functional Models
3. Standard Network Models
4. Shared Layers Model
5. Multiple Input and Output Models
6. Best Practices

1. Keras Sequential Models

As a review, Keras provides a Sequential model API.

This is a way of creating deep learning models where an instance of the Sequential class is created and model layers are created and added to it.

For example, the layers can be defined and passed to the Sequential as an array:

```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential([Dense(2, input_dim=1), Dense(1)])
```

Layers can also be added piecewise:

```
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(2, input_dim=1))
model.add(Dense(1))
```

The Sequential model API is great for developing deep learning models in most situations, but it also has some limitations.

For example, it is not straightforward to define models
produce multiple output destinations or models that re

2. Keras Functional Models

The Keras functional API provides a more flexible way

It specifically allows you to define multiple input or out ore
than that, it allows you to define ad hoc acyclic networ

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Models are defined by creating instances of layers and connecting them directly to each other in pairs, then defining a Model that specifies the layers to act as the input and output to the model.

Let's look at the three unique aspects of Keras functional API in turn:

1. Defining Input

Unlike the Sequential model, you must create and define a standalone Input layer that specifies the shape of input data.

The input layer takes a shape argument that is a tuple that indicates the dimensionality of the input data.

When input data is one-dimensional, such as for a multilayer Perceptron, the shape must explicitly leave room for the shape of the mini-batch size used when splitting the data when training the network. Therefore, the shape tuple is always defined with a hanging last dimension when the input is one-dimensional (2,), for example:

```
from keras.layers import Input
visible = Input(shape=(2,))
```

2. Connecting Layers

The layers in the model are connected pairwise.

This is done by specifying where the input comes from when defining each new layer. A bracket notation is used, such that after the layer is created, the layer from which the input to the current layer comes from is specified.

Let's make this clear with a short example. We can create the input layer as above, then create a hidden layer as a Dense that receives input only from the input layer.

```
from keras.layers import Input
from keras.layers import Dense
visible = Input(shape=(2,))
hidden = Dense(2)(visible)
```

Note the (visible) after the creation of the Dense layer
the dense hidden layer.

It is this way of connecting layers piece by piece that
you can see how easy it would be to start defining ad

3. Creating the Model

After creating all of your model layers and connecting

As with the Sequential API, the model is the thing you
predictions.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Keras provides a Model class that you can use to create a model from your created layers. It requires that you only specify the input and output layers. For example:

```
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
visible = Input(shape=(2,))
hidden = Dense(2)(visible)
model = Model(inputs=visible, outputs=hidden)
```

Now that we know all of the key pieces of the Keras functional API, let's work through defining a suite of different models and build up some practice with it.

Each example is executable and prints the structure and creates a diagram of the graph. I recommend doing this for your own models to make it clear what exactly you have defined.

My hope is that these examples provide templates for you when you want to define your own models using the functional API in the future.

3. Standard Network Models

When getting started with the functional API, it is a good idea to see how some standard neural network models are defined.

In this section, we will look at defining a simple multilayer Perceptron, convolutional neural network, and recurrent neural network.

These examples will provide a foundation for understanding the more elaborate examples later.

Multilayer Perceptron

In this section, we define a multilayer Perceptron model for binary classification.

The model has 10 inputs, 3 hidden layers with 10, 20, and 10 neurons, and an output layer with 1 output. Rectified linear activation functions are used in each hidden layer and a sigmoid activation function is used in the output layer, for binary classification.

```
# Multilayer Perceptron
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
visible = Input(shape=(10,))
hidden1 = Dense(10, activation='relu')(visibl
hidden2 = Dense(20, activation='relu')(hidden
hidden3 = Dense(10, activation='relu')(hidden
output = Dense(1, activation='sigmoid')(hidde
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='multilayer_percept
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

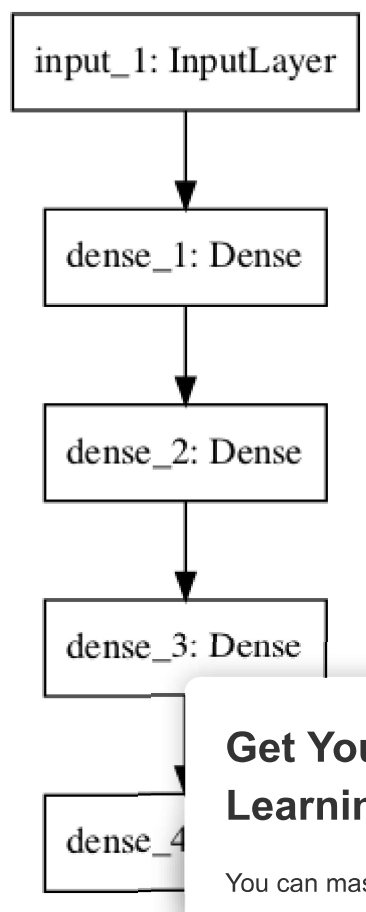
Email Address

START MY EMAIL COURSE

Running the example prints the structure of the network.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 10)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 20)                220
_________________________________________________________________
dense_3 (Dense)              (None, 10)                210
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 551
Trainable params: 551
Non-trainable params: 0
_________________________________________________________________
```

A plot of the model graph is also created and saved to file.



Multilayer Perc
Gra

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Convolutional Neural Network

In this section, we will define a convolutional neural ne

The model receives black and white 64×64 images as input, then has a sequence of two convolutional and pooling layers as feature extractors, followed by a fully connected layer to interpret the features and an output layer with a sigmoid activation for two-class predictions.

```
# Convolutional Neural Network
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
visible = Input(shape=(64,64,1))
conv1 = Conv2D(32, kernel_size=4, activation='relu')(visible)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(16, kernel_size=4, activation='relu')(pool1)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
hidden1 = Dense(10, activation='relu')(pool2)
output = Dense(1, activation='sigmoid')(hidden1)
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='convolutional_neural_network.png')
```

Running the example summarizes the model layers.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64, 64, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 61, 32)        544
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 27, 27, 16)        8208
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 16)        0
_________________________________________________________________
dense_1 (Dense)              (None, 13, 13, 10)        170
_________________________________________________________________
dense_2 (Dense)              (None, 13, 13, 1)         11
=================================================================
Total params: 8,933
Trainable params: 8,933
Non-trainable params: 0
_________________________________________________________________
```

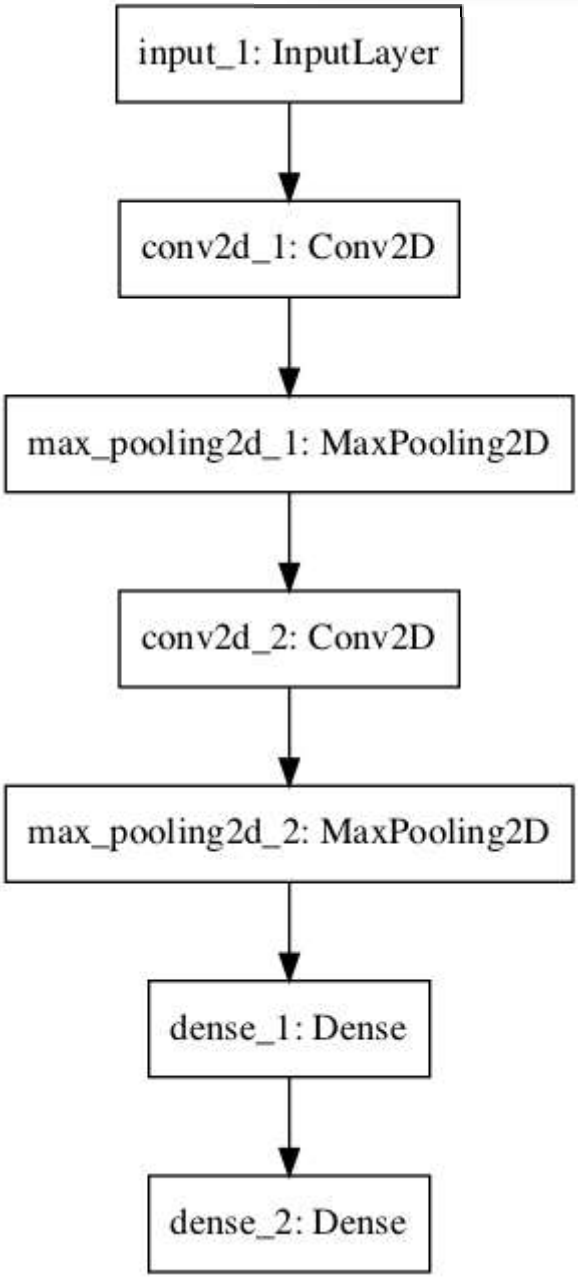
A plot of the model graph is also created and saved to

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Convolutional Neural Network Graph

Recurrent Neural Network

In this section, we will define a long short-term memor
classification.

The model expects 100 time steps of one feature as i
extract features from the sequence, followed by a fully
followed by an output layer for making binary predictio

```
# Recurrent Neural Network
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers.recurrent import LSTM
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

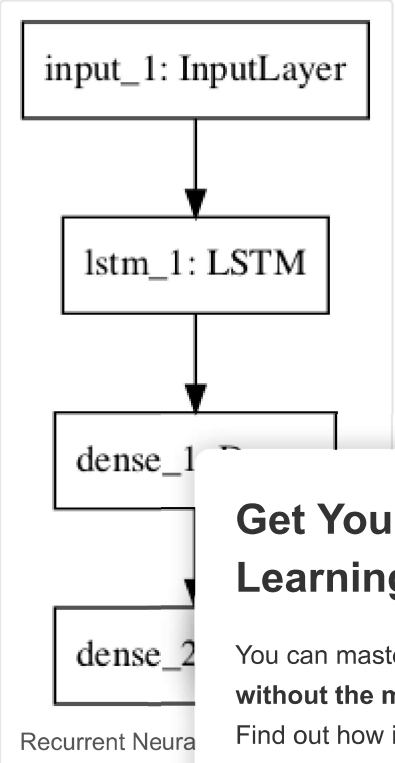
START MY EMAIL COURSE

```
visible = Input(shape=(100,1))
hidden1 = LSTM(10)(visible)
hidden2 = Dense(10, activation='relu')(hidden1)
output = Dense(1, activation='sigmoid')(hidden2)
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='recurrent_neural_network.png')
```

Running the example summarizes the model layers.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 100, 1)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 10)                480
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 601
Trainable params: 601
Non-trainable params: 0
_________________________________________________________________
```

A plot of the model graph is also created and saved to file.



Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

4. Shared Layers Model

Multiple layers can share the output from one layer.

For example, there may be multiple different feature extraction layers from an input, or multiple layers used to interpret the output from a feature extraction layer.

Let's look at both of these examples.

Shared Input Layer

In this section, we define multiple convolutional layers with differently sized kernels to interpret an image input.

The model takes black and white images with the size 64×64 pixels. There are two CNN feature extraction submodels that share this input; the first has a kernel size of 4 and the second a kernel size of 8. The outputs from these feature extraction submodels are flattened into vectors and concatenated into one long vector and passed on to a fully connected layer for interpretation before a final output layer makes a binary classification.

```
# Shared Input Layer
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
# input layer
visible = Input(shape=(64,64,1))
# first feature extractor
conv1 = Conv2D(32, kernel_size=4, activation='relu')(visible)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
flat1 = Flatten()(pool1)
# second feature extractor
conv2 = Conv2D(16, kernel_size=8, activation='relu')(visible)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flat2 = Flatten()(pool2)
# merge feature extractors
merge = concatenate([flat1, flat2])
# interpretation layer
hidden1 = Dense(10, activation='relu')(merge)
# prediction output
output = Dense(1, activation='sigmoid')(hidde
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='shared_input_layer
```

Running the example summarizes the model layers.

```
__________________________________________________
Layer (type)                    Output Shape
==================================================
input_1 (InputLayer)            (None, 64, 6
__________________________________________________
conv2d_1 (Conv2D)               (None, 61, 6
__________________________________________________
conv2d_2 (Conv2D)               (None, 57, 5
__________________________________________________
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

```
max_pooling2d_1 (MaxPooling2D)   (None, 30, 30, 32)    0           conv2d_1[0][0]
____________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)   (None, 28, 28, 16)    0           conv2d_2[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 28800)         0           max_pooling2d_1[0][0]
____________________________________________________________________________________________________
flatten_2 (Flatten)              (None, 12544)         0           max_pooling2d_2[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 41344)         0           flatten_1[0][0]
                                                                   flatten_2[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 10)            413450      concatenate_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 1)             11          dense_1[0][0]
====================================================================================================
Total params: 415,045
Trainable params: 415,045
Non-trainable params: 0
____________________________________________________________________________________________________
```

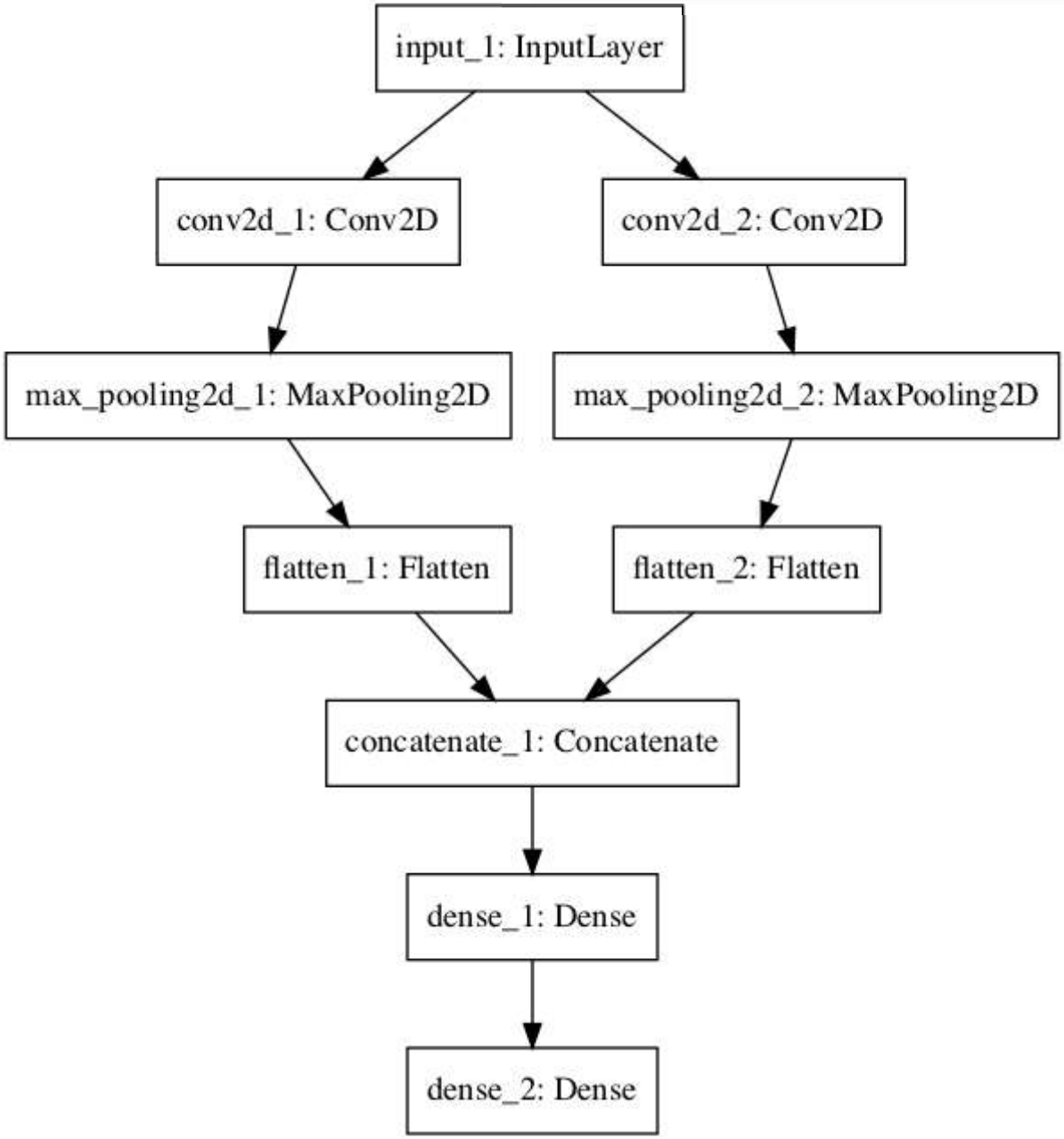
A plot of the model graph is also created and saved to file.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Neural Network Graph With Shared Inputs

Shared Feature Extraction Layer

In this section, we will two parallel submodels to interp
sequence classification.

The input to the model is 100 time steps of 1 feature. A s
sequence. The first interpretation model is a shallow s 3
layer model. The output of both interpretation models ed
to the output layer used to make a binary prediction.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE


```
# Shared Feature Extraction Layer
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
```

```
from keras.layers.recurrent import LSTM
from keras.layers.merge import concatenate
# define input
visible = Input(shape=(100,1))
# feature extraction
extract1 = LSTM(10)(visible)
# first interpretation model
interp1 = Dense(10, activation='relu')(extract1)
# second interpretation model
interp11 = Dense(10, activation='relu')(extract1)
interp12 = Dense(20, activation='relu')(interp11)
interp13 = Dense(10, activation='relu')(interp12)
# merge interpretation
merge = concatenate([interp1, interp13])
# output
output = Dense(1, activation='sigmoid')(merge)
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='shared_feature_extractor.png')
```

Running the example summarizes the model layers.

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 100, 1)        0
____________________________________________________________________________________________________
lstm_1 (LSTM)                    (None, 10)            480         input_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 10)            110         lstm_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 20)            220         dense_2[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 10)            110         lstm_1[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 10)            210         dense_3[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 20)            0           dense_1[0][0]
                                                                   dense_4[0][0]
____________________________________________________________________________________________________
dense_5 (Dense)                  (None, 1)             21          concatenate_1[0][0]
====================================================================================================
Total params: 1,151
Trainable params: 1,151
Non-trainable params: 0
____________________________________________________________________________________________________
```

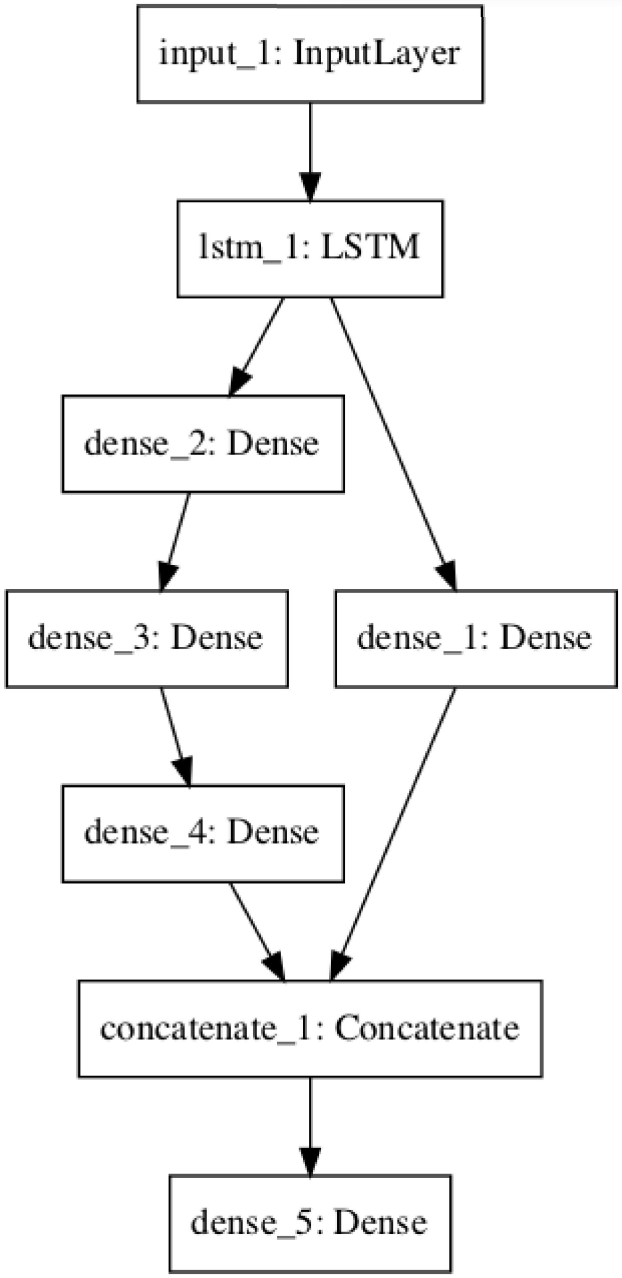
A plot of the model graph is also created and saved to

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Neural Network Graph With Shared Feature Extraction Layer

5. Multiple Input and Output Mo

The functional API can also be used to develop more
different modalities. It can also be used to develop mo

We will look at examples of each in this section.

Multiple Input Model

We will develop an image classification model that tak
different size. Specifically a black and white 64×64 ve
extraction CNN models operate on each, then the resu
interpretation and ultimate prediction.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Note that in the creation of the Model() instance, that we define the two input layers as an array. Specifically:

```
model = Model(inputs=[visible1, visible2], outputs=output)
```

The complete example is listed below.

```
# Multiple Inputs
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
# first input model
visible1 = Input(shape=(64,64,1))
conv11 = Conv2D(32, kernel_size=4, activation='relu')(visible1)
pool11 = MaxPooling2D(pool_size=(2, 2))(conv11)
conv12 = Conv2D(16, kernel_size=4, activation='relu')(pool11)
pool12 = MaxPooling2D(pool_size=(2, 2))(conv12)
flat1 = Flatten()(pool12)
# second input model
visible2 = Input(shape=(32,32,3))
conv21 = Conv2D(32, kernel_size=4, activation='relu')(visible2)
pool21 = MaxPooling2D(pool_size=(2, 2))(conv21)
conv22 = Conv2D(16, kernel_size=4, activation='relu')(pool21)
pool22 = MaxPooling2D(pool_size=(2, 2))(conv22)
flat2 = Flatten()(pool22)
# merge input models
merge = concatenate([flat1, flat2])
# interpretation model
hidden1 = Dense(10, activation='relu')(merge)
hidden2 = Dense(10, activation='relu')(hidden1)
output = Dense(1, activation='sigmoid')(hidden2)
model = Model(inputs=[visible1, visible2], outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='multiple_inputs.png')
```

Running the example summarizes the model layers.

```
__________________________________________________
Layer (type)                    Output Shape
==================================================
input_1 (InputLayer)            (None, 64, 6
__________________________________________________
input_2 (InputLayer)            (None, 32, 3
__________________________________________________
conv2d_1 (Conv2D)               (None, 61, 6
__________________________________________________
conv2d_3 (Conv2D)               (None, 29, 2
__________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 30, 3
__________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 14, 1
__________________________________________________
conv2d_2 (Conv2D)               (None, 27, 2
__________________________________________________
conv2d_4 (Conv2D)               (None, 11, 1
```


Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

```
____________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)   (None, 13, 13, 16)    0           conv2d_2[0][0]
____________________________________________________________________________________________________
max_pooling2d_4 (MaxPooling2D)   (None, 5, 5, 16)      0           conv2d_4[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 2704)          0           max_pooling2d_2[0][0]
____________________________________________________________________________________________________
flatten_2 (Flatten)              (None, 400)           0           max_pooling2d_4[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 3104)          0           flatten_1[0][0]
                                                                   flatten_2[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 10)            31050       concatenate_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 10)            110         dense_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 1)             11          dense_2[0][0]
====================================================================================================
Total params: 49,699
Trainable params: 49,699
Non-trainable params: 0
____________________________________________________________________________________________________
```

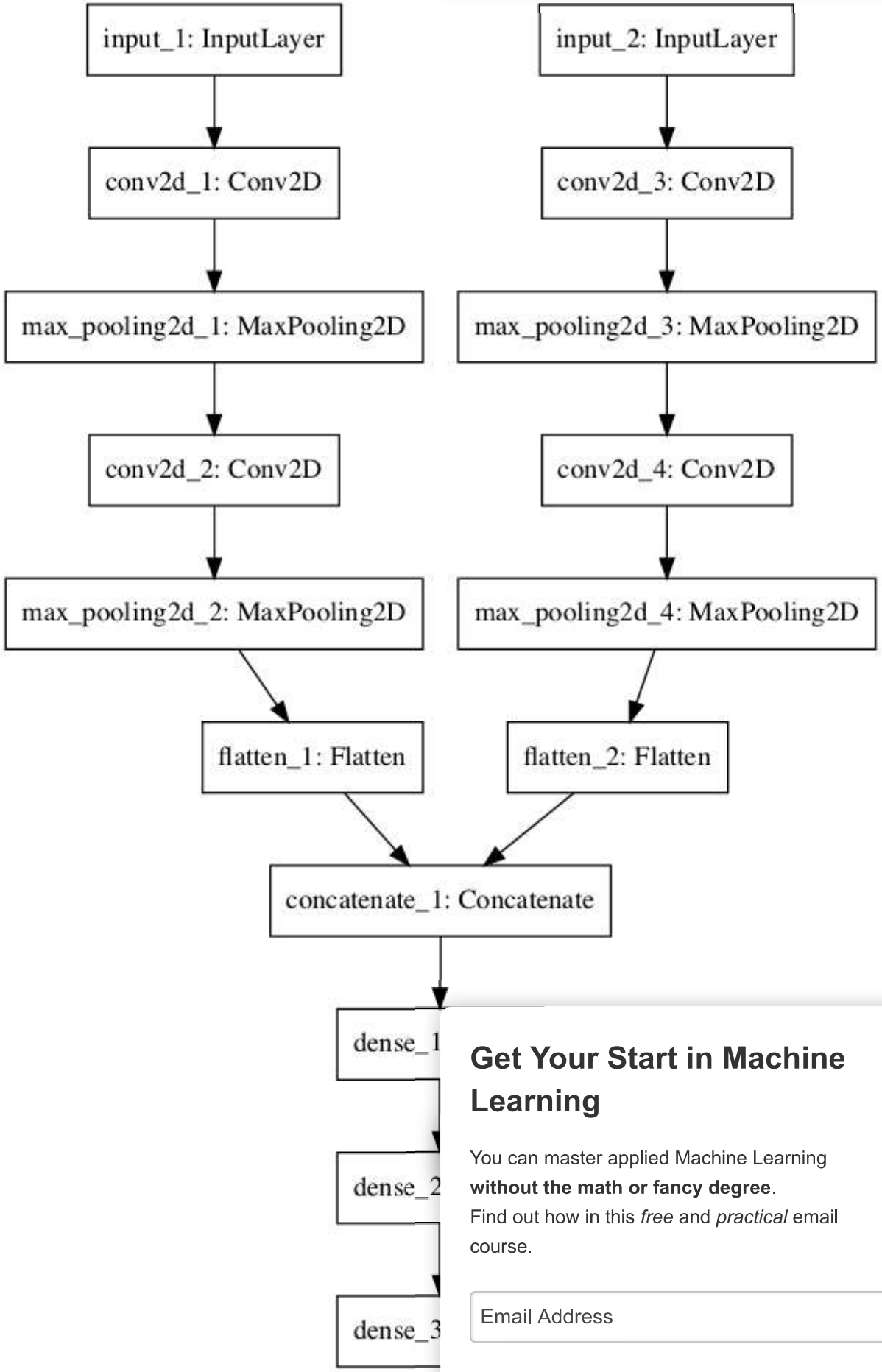
A plot of the model graph is also created and saved to file.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Neural Network Grapl

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Multiple Output Model

In this section, we will develop a model that makes two different types of predictions. Given an input sequence of 100 time steps of one feature, the model will both classify the sequence and output a new sequence with the same length.

An LSTM layer interprets the input sequence and returns the hidden state for each time step. The first output model creates a stacked LSTM, interprets the features, and makes a binary prediction. The second output model uses the same output layer to make a real-valued prediction for each input time step.

```
# Multiple Outputs
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers.recurrent import LSTM
from keras.layers.wrappers import TimeDistributed
# input layer
visible = Input(shape=(100,1))
# feature extraction
extract = LSTM(10, return_sequences=True)(visible)
# classification output
class11 = LSTM(10)(extract)
class12 = Dense(10, activation='relu')(class11)
output1 = Dense(1, activation='sigmoid')(class12)
# sequence output
output2 = TimeDistributed(Dense(1, activation='linear'))(extract)
# output
model = Model(inputs=visible, outputs=[output1, output2])
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='multiple_outputs.png')
```

Running the example summarizes the model layers.

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 100, 1)        0
____________________________________________________________________________________________________
lstm_1 (LSTM)                    (None, 100, 10)       480         input_1[0][0]
____________________________________________________________________________________________________
lstm_2 (LSTM)                    (None, 10)
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 10)
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 1)
____________________________________________________________________________________________________
time_distributed_1 (TimeDistribu (None, 100,
====================================================================================================
Total params: 1,452
Trainable params: 1,452
Non-trainable params: 0
____________________________________________________________________________________________________
```

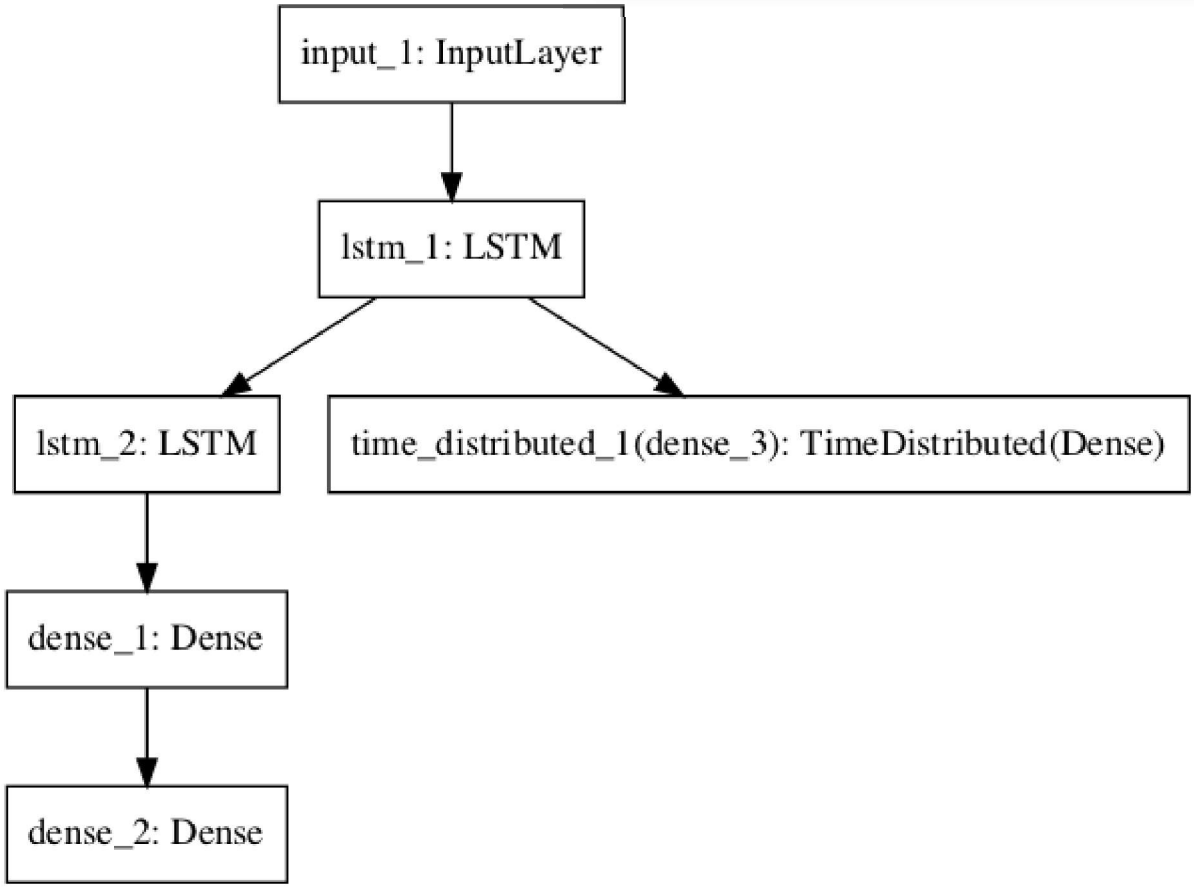
A plot of the model graph is also created and saved to

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Neural Network Graph With Multiple Outputs

6. Best Practices

In this section, I want to give you some tips to get the most out of the functional API when you are defining your own models.

- **Consistent Variable Names.** Use the same variable name for the input (visible) and output layers (output) and perhaps even the hidden layers (hidden1, hidden2). It will help to connect things together correctly.
- **Review Layer Summary.** Always print the model summary and review the layer outputs to ensure that the model was connected together as you expect
- **Review Graph Plots.** Always create a plot of the ng was put together as you intended.
- **Name the layers.** You can assign names to layer ots of the model graph. For example: Dense(1, name
- **Separate Submodels.** Consider separating out t submodels together at the end.

Do you have your own best practice tips when using t Let me know in the comments.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Further Reading

This section provides more resources on the topic if you are looking go deeper.

- The Sequential model API
- Getting started with the Keras Sequential model
- Getting started with the Keras functional API
- Model class Functional API

Summary

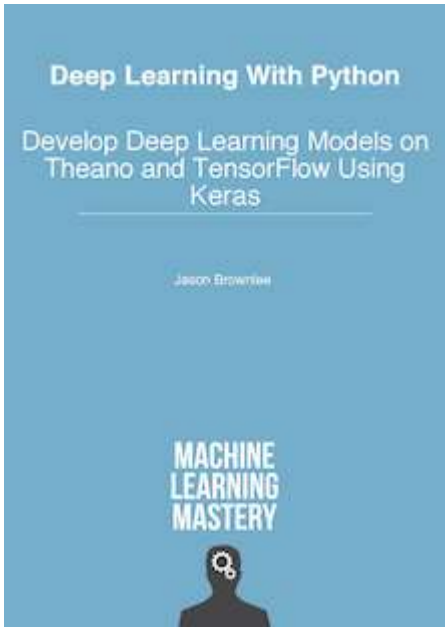
In this tutorial, you discovered how to use the functional API in Keras for defining simple and complex deep learning models.

Specifically, you learned:

- The difference between the Sequential and Functional APIs.
- How to define simple Multilayer Perceptron, Convolutional Neural Network, and Recurrent Neural Network models using the functional API.
- How to define more complex models with shared layers and multiple inputs and outputs.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.

Frustrated With Your Progress In Deep Learning?



What If You Could Develop A Network in Minutes

…with just a few lines of Python

Discover how in my new Ebook: Deep Learning With Python

It covers **self-study tutorials** and **end-to-end projects** on topics like: *Multilayer Perceptrons*, *Convolutional Nets* and *Recurrent Neural Nets*, and more…

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

About Jason Brownlee

Jason Brownlee, Ph.D. is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.

View all posts by Jason Brownlee →

‹ How to Define an Encoder-Decoder Sequence-to-Sequence Model for Neural Machine Translation in Keras

How to Develop a Word Embedding Model for Predicting Movie Review Sentiment ›

39 Responses to *How to Use the Keras Functional API for Deep Learning*

Salameh October 27, 2017 at 5:36 am #

REPLY

Thank you
I have been waiting fir this tutorial

Jason Brownlee October 27, 2017 at 2:52 pm #

REPLY

Thanks, I hope it helps!

Thabet Ali October 27, 2017 at 7:21 am #

REPLY

You're awesome Jason!

I just can't wait to see more from you on this wonderful
Can you please write a book where you implement mo
I'm sure the book will be a real success

Best regards
Thabet

Tom October 27, 2017 at 12:18 pm #

I agree with Thabet Ali, need a book on a

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Jason Brownlee October 27, 2017 at 2:57 pm # REPLY

What aspects of the functional API do you need more help with Tom?

Jason Brownlee October 27, 2017 at 2:54 pm # REPLY

Thanks!

What problems are you having with the functional API?

Thabet October 27, 2017 at 7:00 pm # REPLY

I would like to know more on how to implement autoencoders on multi input time series signals with a single output categorical classification, using the functional API

Jason Brownlee October 28, 2017 at 5:10 am # REPLY

Thanks.

LSTMs, for example, can take multiple time series directly and don't require parallel input models.

Why do you want to use autoencoders for time series?

Thabet October 28, 2017 at 8:27 am #

Yes I think it's because the *output sequence is shorter* than the input sequence Like seen in this dataset:

https://archive.ics.uci.edu/ml/datasets/

Where there are three time series inpu

Alexander October 27, 2017 at 6:50 pm #

Jason, thank you for very interest blog. Beaut

Jason Brownlee October 28, 2017 at 5:08 am

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Thanks Alexander.

Leon October 28, 2017 at 5:28 pm #

REPLY

Dr. Brownlee,
How do you study mathematics behind so many algorithms that you implement?
regards
Leon

Jason Brownlee October 29, 2017 at 5:51 am #

REPLY

I read a lot 🙂

Alex November 4, 2017 at 5:32 am #

REPLY

Can I consider the initial_state of an LSTM layer as an input branch of the architecture? Say that for each data I have a sequence s1, s2, s3 and a context feature X. I define an LSTM with 128 neurons, for each batch I want to map X to a 128 dimensional feature through a Dense(128) layer and set the initial_state for that batch training, meanwhile the sequence s1, s2,s3 is fed to the LSTM as the input sequence.

Jason Brownlee November 4, 2017 at 5:34 am #

REPLY

Sorry Alex, I'm not sure I follow. If you have some ideas, perhaps design an experiment to test them?

Jithu R Jacob November 6, 2017 at 8:38 pm #

Thank you for the awesome tutorial.

For anyone wants to directly visualize in Jupyter Noteb

from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

SVG(model_to_dot(model).create(prog='dot', format='

Jason Brownlee November 7, 2017 at 9:49 a

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Awesome, thanks for the tip!

Luis November 7, 2017 at 2:09 am #

REPLY

Thanks for this blog. It is really helpful and well explained.

Jason Brownlee November 7, 2017 at 9:51 am #

REPLY

Thanks Luis, I'm glad to hear that. I appreciate your support!

Cal Almodovar November 11, 2017 at 8:52 am #

REPLY

Hi Jason – I am wondering if the code should be:

visible = Input(image_file, shape=(height, width, color channels))

if not, I wonder how the code references the image in question….

Surprised no one else asked this…

YOU ROCK JASON!

joe November 25, 2017 at 5:32 am #

REPLY

In section 1 you write that "the shape tuple is always defined with a hanging last dimension", but when you define the convolutional network, you define it as such:

visible = Input(shape=(64,64,1))

without any hanging last dimension. Am I missing som

Jason Brownlee November 25, 2017 at 10:23

Yes, I was incorrect. This is only the case

Franek December 1, 2017 at 9:14 pm #

Jason, I've been reading about keras and AI f
clear and straightforward than all the other stuff on the

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Thanks!

Jason Brownlee December 2, 2017 at 8:55 am #

REPLY

Thanks Franek.

Franek December 1, 2017 at 9:17 pm #

REPLY

Jason, for some reasons I need to know the output tensors of the layers in my model. I've tried to experiment with the layer.get_output, get_output_et etc following the keras documentation, but I always fail to get anything sensible.

I tried to look for this subject on your blog, but I couldn't find anything. Are you planning to write a post on this? 🙂 That would really help!

Jason Brownlee December 2, 2017 at 8:57 am #

REPLY

Sorry, I don't have good advice for getting the output tensor. Perhaps post to the keras google group or slack channel?

See here:
https://machinelearningmastery.com/get-help-with-keras/

Nicola December 2, 2017 at 12:32 am #

REPLY

Hi Jason, yet another great post.
Using one of your past posts I created an LSTM that, using multiple time series, predicts several step ahead of a specific time series. Currently I have this structure

```
#reshape the time series
x_tr, y_tr = get_data(training, back, ahea

n_samples = x_tr.shape[0]
features = x_tr.shape[2]

hidden = 50

#LSTM
model = Sequential()
model.add(LSTM(input_dim=features, output_
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dropout(0.2))
model.add(Dense(input_dim=hidden, output_di
model.add(Activation('linear'))
model.compile(loss='mae', optimizer='adam'
```


Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

```
model.fit(x_tr, y_tr)
```

where:
– x_tr has (440, 7, 6) dimensions (440 samples, 7 past time steps, 6 variables/time series)
– y_tr has (440, 21) dimensions, that is 440 samples and 21 ahead predicted values.

Now, I'd like to extend my network so that it predicts the (multi-step ahead) values of two time series. I tried this code:

```
inputs = Input(shape=(7,6)) # 7 steps ahead and 6 variables
m = Dense(64,activation='relu')(inputs)
m = Dense(64,activation='relu')(m)
outputA = Dense(1,activation='tanh')(m)
outputB = Dense(1,activation='tanh')(m)

m = Model(inputs=[inputs], outputs=[outputA, outputB])
m.fit(x,[y1,y2])
```

where y1 and y2 both have (440, 21) dimensions, but I have this error:

"Error when checking target: expected dense_4 to have 3 dimensions, but got array with shape (440, 21)".

How should I reshape y1 and y2 so that they fit with the network?

Jason Brownlee December 2, 2017 at 9:03 am # REPLY

Sorry, I cannot debug your code for you, perhaps post to stack overflow?

Davor December 22, 2017 at 3:12 am # REPLY

Great tutorial, it really helped me understand Model API better! THANKS!

Jason Brownlee December 22, 2017 at 5:36 am # REPLY

I'm glad to hear that.

Debarati Bhattacharjee January 5, 2018 at 2:27

Hii Jason, this was a great post, specially for I
mind to write a post to explain the code for a image se

Jason Brownlee January 5, 2018 at 5:27 am

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Thanks for the suggestion.

Harminder January 6, 2018 at 10:10 pm #

REPLY

Hi Jason, thank you for such a great post. It helped me a lot to understand functional API's in keras. Could you please explain how we define the model.compile statement in multiple output case where each sub-model has a different objective function. For example, one output might be regression and the other classification as you mentioned in this post.

Jason Brownlee January 7, 2018 at 5:06 am #

REPLY

I believe you must have one objective function for the whole model.

Harry Garrison January 23, 2018 at 7:57 pm #

REPLY

Splendid tutorial as always!

Do you think you could make a tutorial about siamese neural nets in the future? It would be particularly interesting to see how a triplet loss model can be created in keras, one that recognizes faces, for example. The functional API must be the way to go, but I can't imagine exactly how the layers should be connected.

Jason Brownlee January 24, 2018 at 9:53 am #

REPLY

Thanks for the suggestion Harry.

Akhtar Ali January 27, 2018 at 8:20 pm #

Good tutorial
Thanks alot

Jason Brownlee January 28, 2018 at 8:22 am

Thanks.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Paul the Student February 2, 2018 at 2:49 am #

Thank you very much for your tutorial! He helped me a lot!

I have a question about cost functions. I have one request and several documents: 1 relevant and 4 irrelevant. I would like a cost function that both maximizes SCORE(Q, D+) and minimizes SCORE(Q, D-).
So, I could have Delta = SUM{ Score(Q,D+) – Score(Q,Di-) } for i in (1..4)
Using the Hinge Loss cost function, I have L = max(0, 4 – Delta)

I wanted to know if taking the 4 documents, calculating their score with the NN and sending everything in the cost function is a good practice?

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery

Hi, I'm Jason Brownlee, Ph.D.
My goal is to make practitioners like YOU

Read More

Finally Get Started W

Get Your Start in Machine Learning

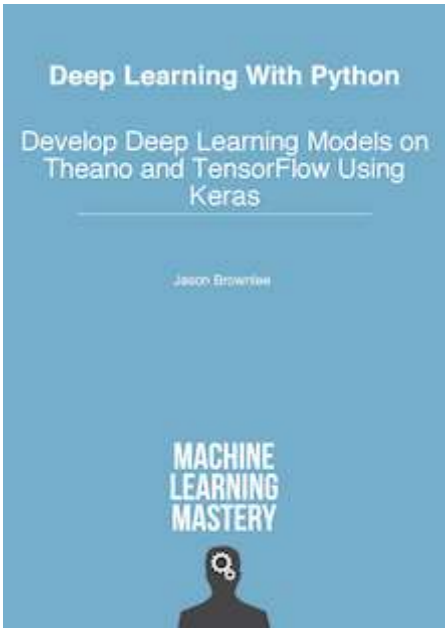
You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Sick of the fancy math and need for super computers?
Looking for step-by-step tutorials?
Want end-to-end projects?

Get Started With Deep Learning in Python Today!



POPULAR

Your First Machine Learning Project in Python Step-By-Step
JUNE 10, 2016

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
JULY 21, 2016

Multivariate Time Series Forecasting with LSTMs in Keras
AUGUST 14, 2017

Develop Your First Neural Network in Python With Keras Step-By-Step
MAY 24, 2016

How to Setup a Python Environment for Machine L
MARCH 13, 2017

Sequence Classification with LSTM Recurrent Neur
JULY 26, 2016

Time Series Forecasting with the Long Short-Term
APRIL 7, 2017

Multi-Class Classification Tutorial with the Keras Deep Learn

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

JUNE 2, 2016

Regression Tutorial with the Keras Deep Learning Library in Python

JUNE 9, 2016

How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras

AUGUST 9, 2016

© 2018 Machine Learning Mastery. All Rights Reserved.

Privacy | Contact | About

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE