How to Use the Keras Functional API for Deep Learning

by **Jason Brownlee** on May 28, 2020 in **Deep Learning** 313

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 flexible functional API in Keras to define deep learning models.

After completing this tutorial, you will know:

- 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.

Kick-start your project with my new book Deep Learning With Python, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.

- **Update Nov/2017**: Added note about hanging dimension for input layers.
- **Update Nov/2018**: Added missing flatten layer for CNN, thanks Konstantin.

- **Update Nov/2018**: Added description of the functional API Python syntax.

Tutorial Overview

This tutorial is divided into 7 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
7. **NEW**: Note on the Functional API Python Syntax

1. Keras Sequential Models

As a review, Keras provides a Sequential model API.

If you are new to Keras or deep learning, see this step-by-step Keras tutorial.

The Sequential model API 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 that may have multiple different input sources, produce multiple output destinations or models that re-use layers.

2. Keras Functional Models

The Keras functional API provides a more flexible way for defining models.

It specifically allows you to define multiple input or output models as well as models that share layers. More than that, it allows you to define ad hoc acyclic network graphs.

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:

```
1 from keras.layers import Input
2 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.

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

Note the (visible) after the creation of the Dense layer that connects the input layer output as the input to the dense hidden layer.

It is this way of connecting layers piece by piece that gives the functional API its flexibility. For example, you can see how easy it would be to start defining ad hoc graphs of layers.

3. Creating the Model

After creating all of your model layers and connecting them together, you must define the model.

As with the Sequential API, the model is the thing you can summarize, fit, evaluate, and use to make predictions.

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.

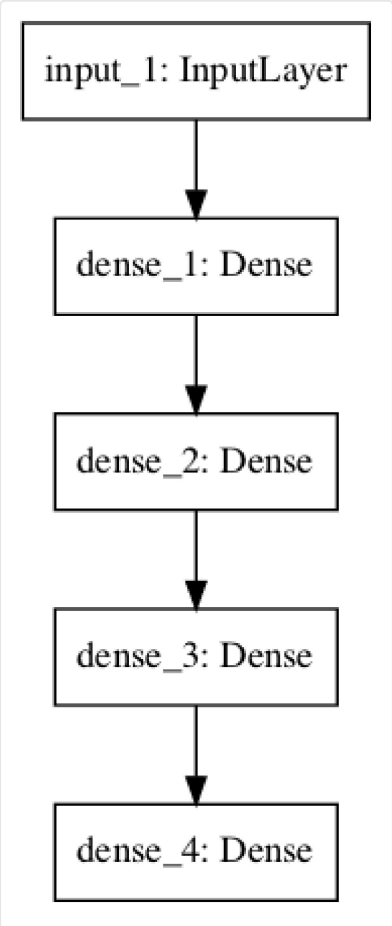
```
# Multilayer Perceptron
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
visible = Input(shape=(10,))
```

```
hidden1 = Dense(10, activation='relu')(visible)
hidden2 = Dense(20, activation='relu')(hidden1)
hidden3 = Dense(10, activation='relu')(hidden2)
output = Dense(1, activation='sigmoid')(hidden3)
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='multilayer_perceptron_graph.png')
```

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 Perceptron Network Graph

Convolutional Neural Network

In this section, we will define a convolutional neural network for image classification.

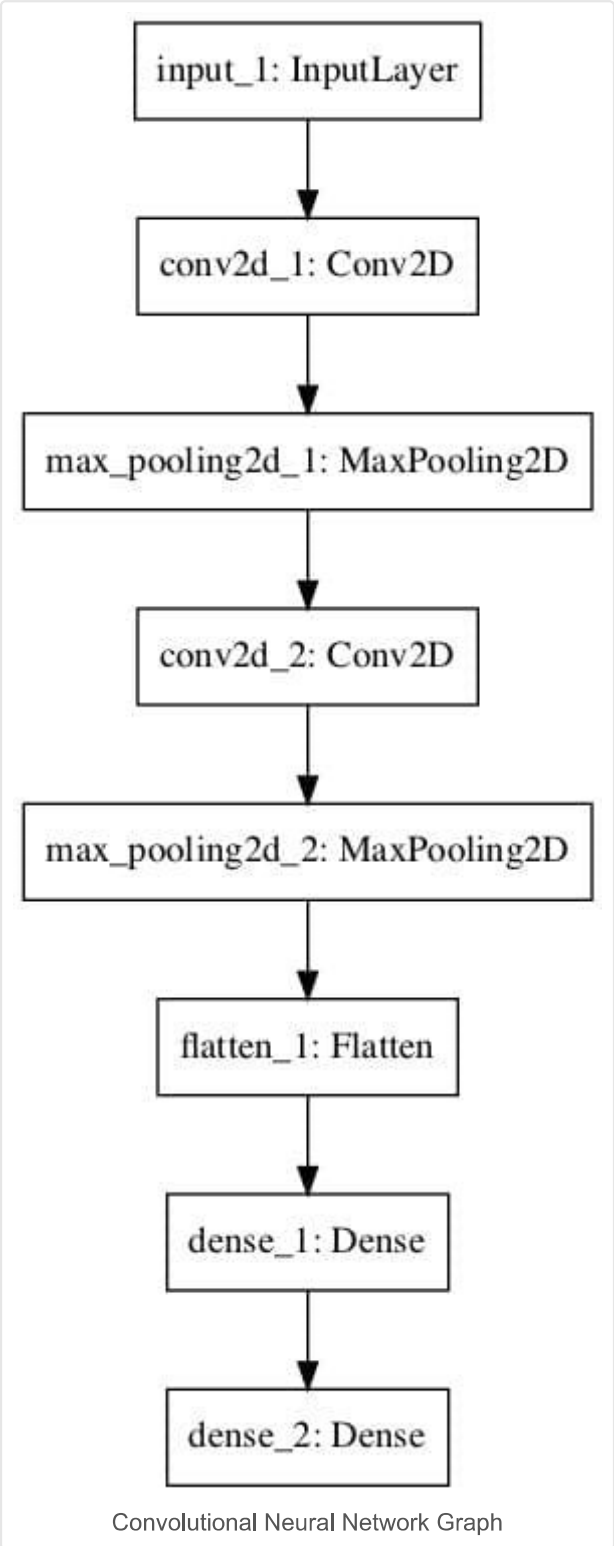
The model receives black and white 64×64 images as input, then has a sequence of two convolutional and pooling layers as feature extractors, followed by a fully connected layer to interpret the features and an output layer with a sigmoid activation for two-class predictions.

```
# 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 import Flatten
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)
flat = Flatten()(pool2)
hidden1 = Dense(10, activation='relu')(flat)
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
_________________________________________________________________
flatten_1 (Flatten)          (None, 2704)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                27050
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 35,813
Trainable params: 35,813
Non-trainable params: 0
_________________________________________________________________
```

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



Convolutional Neural Network Graph

Recurrent Neural Network

In this section, we will define a long short-term memory recurrent neural network for sequence classification.

The model expects 100 time steps of one feature as input. The model has a single LSTM hidden layer to extract features from the sequence, followed by a fully connected layer to interpret the LSTM output,

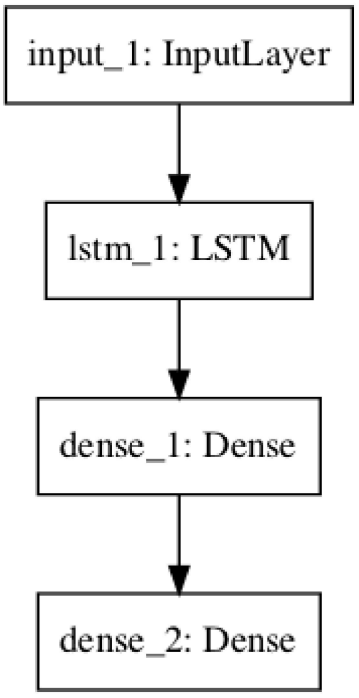
followed by an output layer for making binary predictions.

```
# 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
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.



Recurrent Neural Network Graph

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.

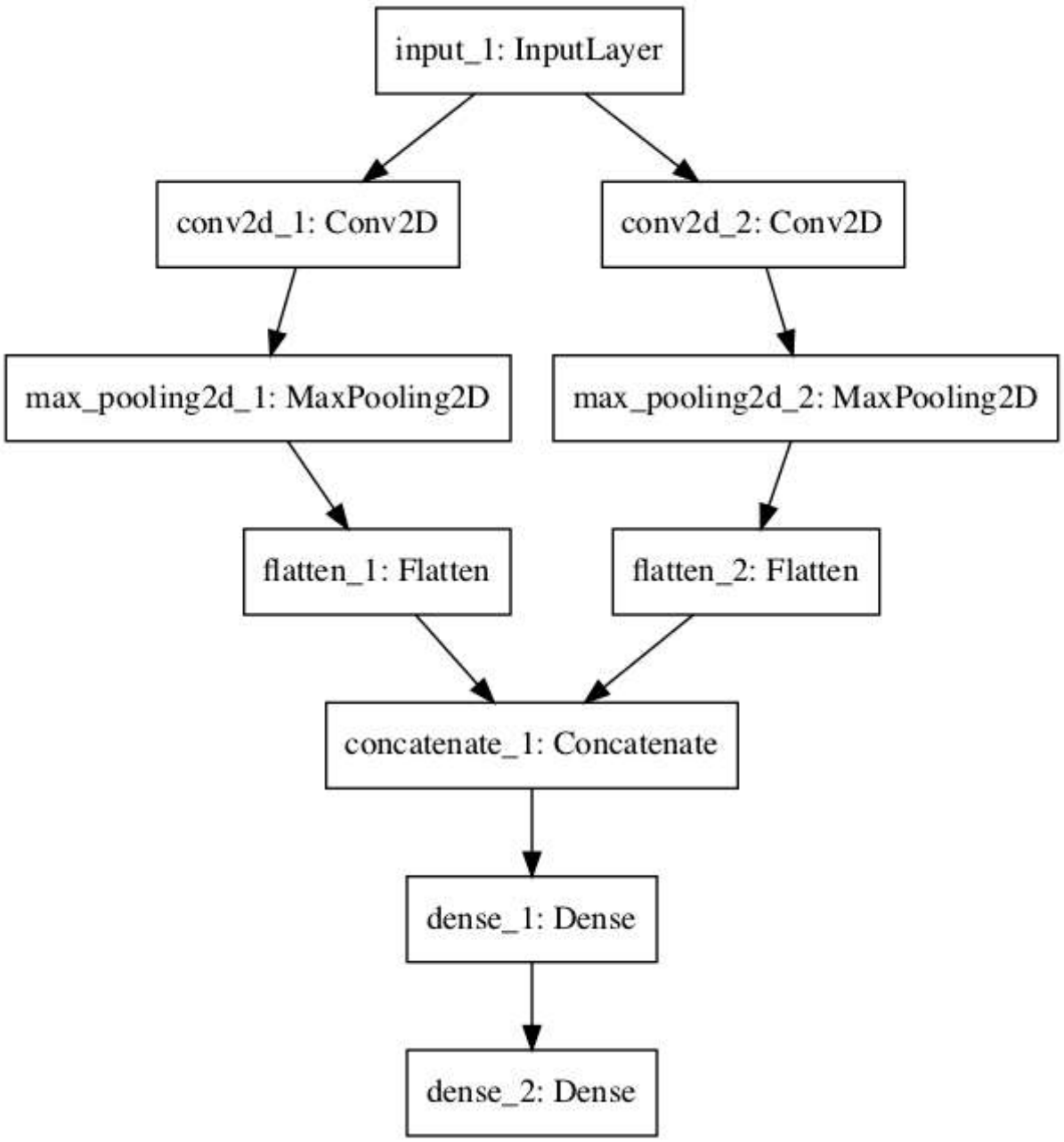
```
# Shared Input Layer
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
```

```
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')(hidden1)
model = Model(inputs=visible, outputs=output)
# summarize layers
print(model.summary())
# plot graph
plot_model(model, to_file='shared_input_layer.png')
```

Running the example summarizes the model layers.

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 64, 64, 1)     0
____________________________________________________________________________________________________
conv2d_1 (Conv2D)                (None, 61, 61, 32)    544         input_1[0][0]
____________________________________________________________________________________________________
conv2d_2 (Conv2D)                (None, 57, 57, 16)    1040        input_1[0][0]
____________________________________________________________________________________________________
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.



Neural Network Graph With Shared Inputs

Shared Feature Extraction Layer

In this section, we will use two parallel submodels to interpret the output of an LSTM feature extractor for sequence classification.

The input to the model is 100 time steps of 1 feature. An LSTM layer with 10 memory cells interprets this sequence. The first interpretation model is a shallow single fully connected layer, the second is a deep 3 layer model. The output of both interpretation models are concatenated into one long vector that is passed to the output layer used to make a binary prediction.

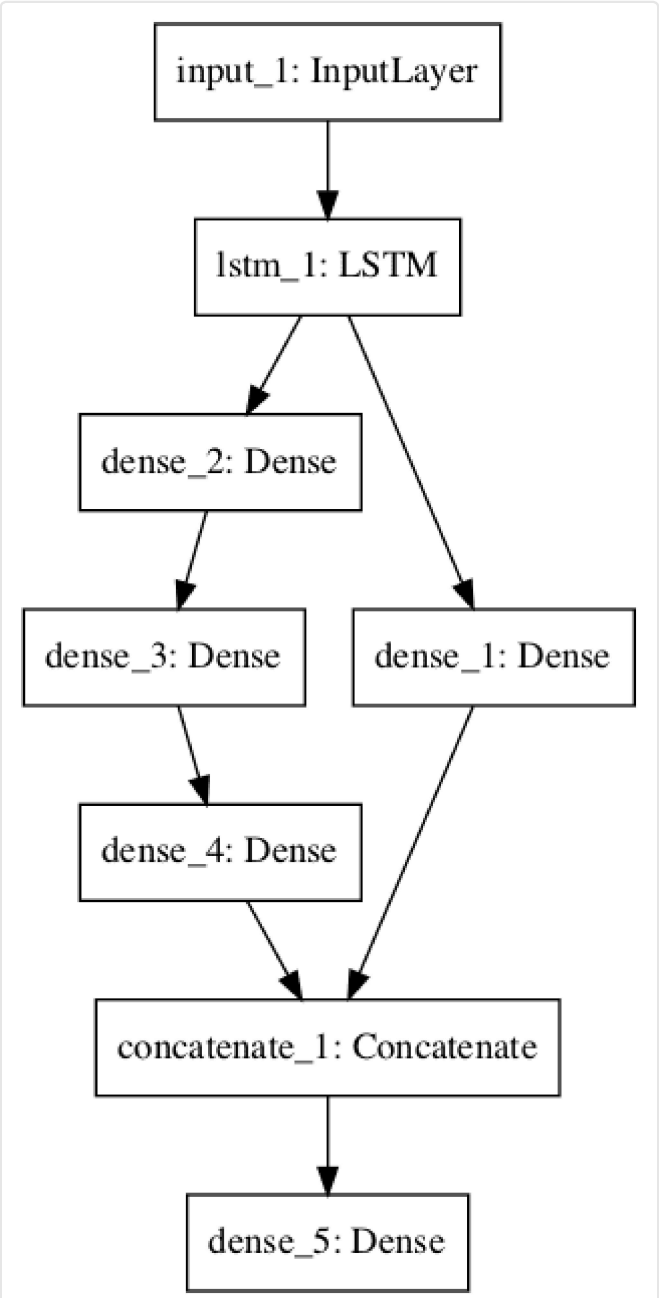
```
1  # Shared Feature Extraction Layer
2  from keras.utils import plot_model
3  from keras.models import Model
```

```
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 file.



Neural Network Graph With Shared Feature Extraction Layer

5. Multiple Input and Output Models

The functional API can also be used to develop more complex models with multiple inputs, possibly with different modalities. It can also be used to develop models that produce multiple outputs.

We will look at examples of each in this section.

Multiple Input Model

We will develop an image classification model that takes two versions of the image as input, each of a different size. Specifically a black and white 64×64 version and a color 32×32 version. Separate feature extraction CNN models operate on each, then the results from both models are concatenated for interpretation and ultimate prediction.

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.

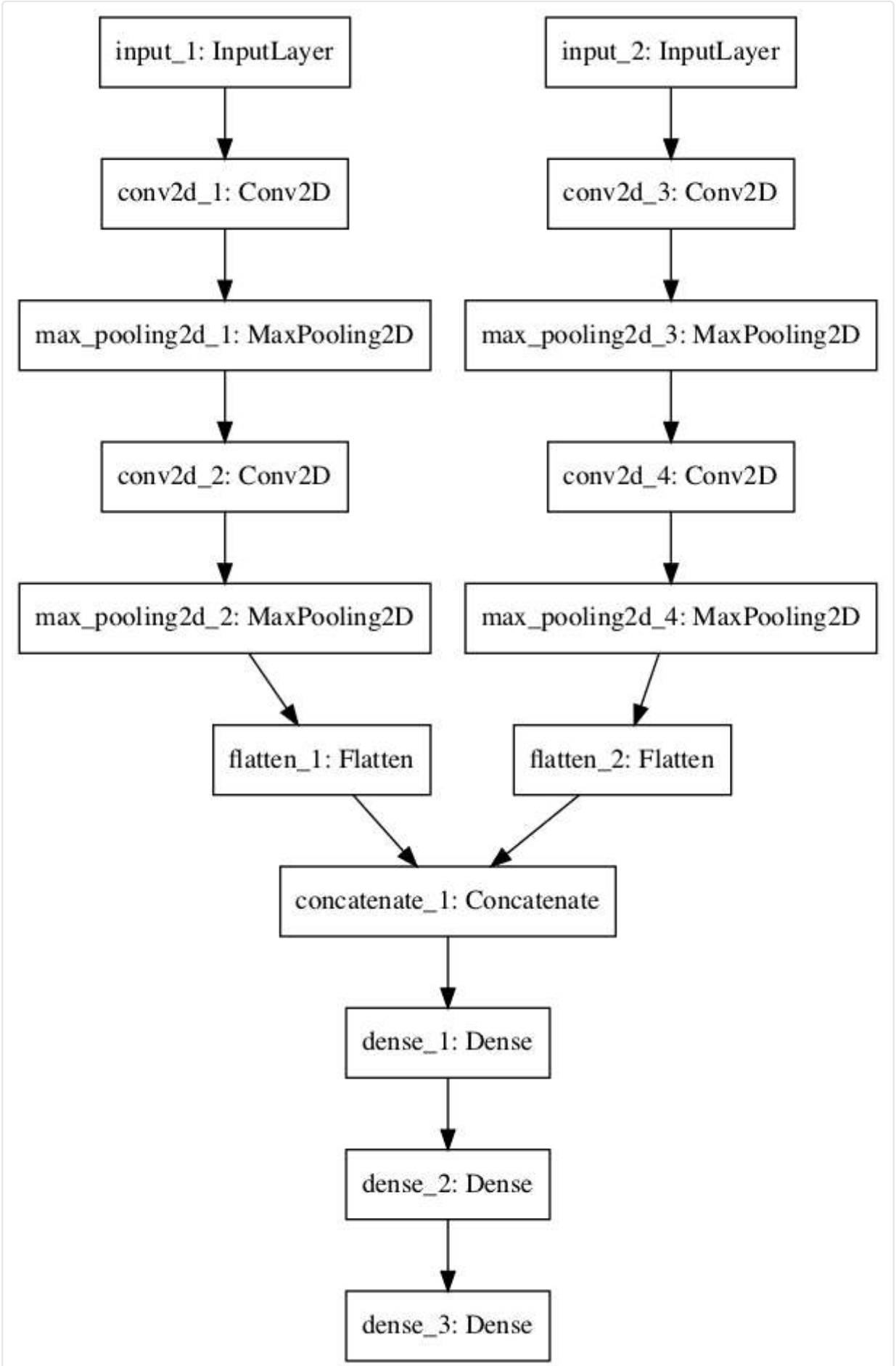
```
# Multiple Inputs
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import 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          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 64, 64, 1)     0
____________________________________________________________________________________________________
input_2 (InputLayer)             (None, 32, 32, 3)     0
____________________________________________________________________________________________________
conv2d_1 (Conv2D)                (None, 61, 61, 32)    544         input_1[0][0]
____________________________________________________________________________________________________
conv2d_3 (Conv2D)                (None, 29, 29, 32)    1568        input_2[0][0]
____________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)   (None, 30, 30, 32)    0           conv2d_1[0][0]
____________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)   (None, 14, 14, 32)    0           conv2d_3[0][0]
____________________________________________________________________________________________________
```

```
conv2d_2 (Conv2D)               (None, 27, 27, 16)    8208        max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 11, 11, 16)    8208        max_pooling2d_3[0][0]
__________________________________________________________________________________________________
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.



Neural Network Graph With Multiple Inputs

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.

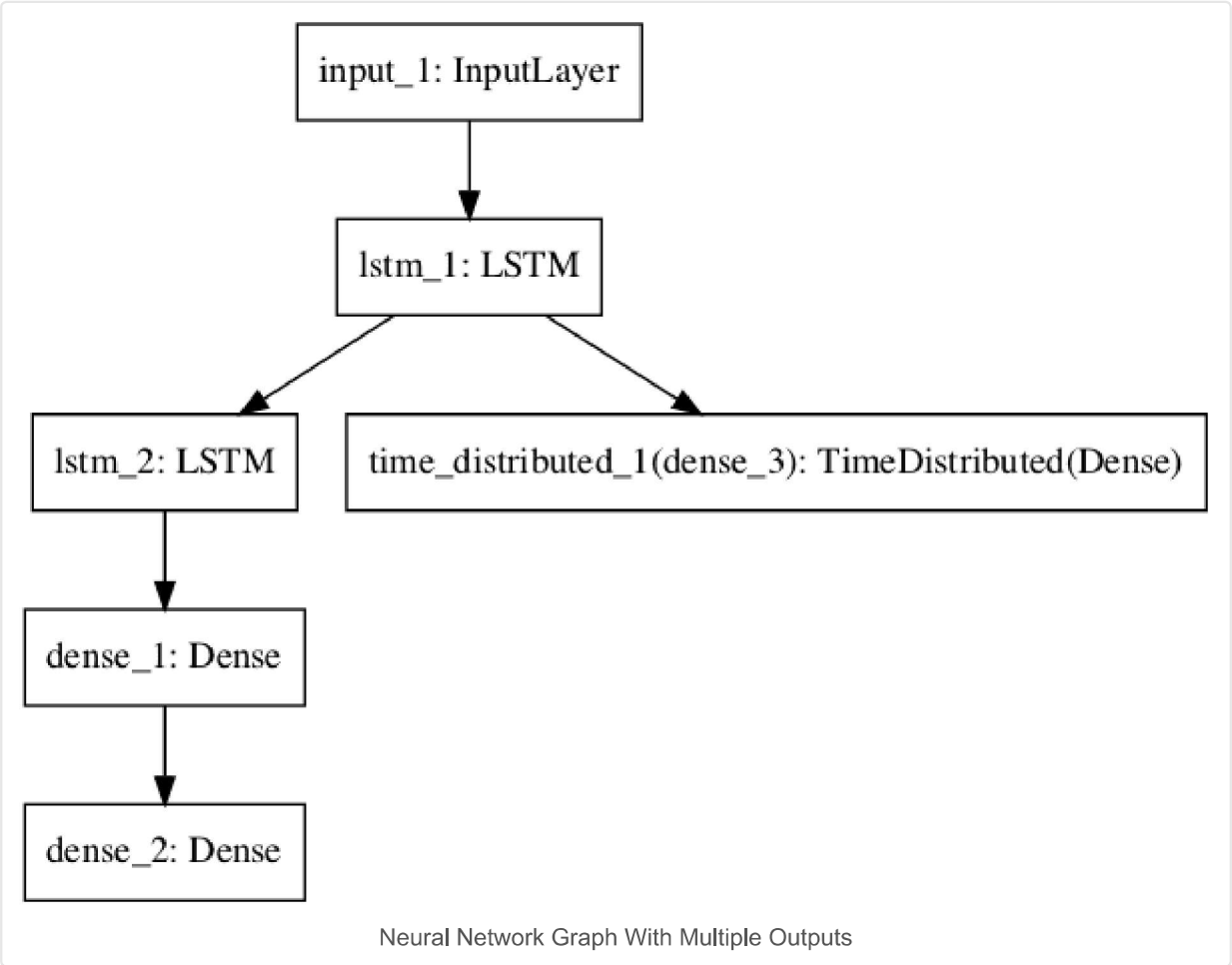
An LSTM layer interprets the input sequence and returns the hidden state for each time step. The first output model creates a stacked LSTM, interprets the features, and makes a binary prediction. The second output model uses the same output layer to make a real-valued prediction for each input time step.

```
# 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 #
=================================================================
input_1 (InputLayer)         (None, 100, 1)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 10)           480
_________________________________________________________________
lstm_2 (LSTM)                (None, 10)                840
_________________________________________________________________
dense_1 (Dense)              (None, 10)                110
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
_________________________________________________________________
time_distributed_1 (TimeDistribu (None, 100, 1)            11
=================================================================
Total params: 1,452
Trainable params: 1,452
Non-trainable params: 0
_________________________________________________________________
```

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



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 expected.
- **Review Graph Plots.** Always create a plot of the model graph and review it to ensure that everything was put together as you intended.
- **Name the layers.** You can assign names to layers that are used when reviewing summaries and plots of the model graph. For example: Dense(1, name='hidden1').
- **Separate Submodels.** Consider separating out the development of submodels and combine the submodels together at the end.

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

7. Note on the Functional API Python Syntax

If you are new or new-ish to Python the syntax used in the functional API may be confusing.

For example, given:

```
...
dense1 = Dense(32)(input)
...
```

What does the double bracket syntax do?

What does it mean?

It looks confusing, but it is not a special python thing, just one line doing two things.

The first bracket "*(32)*" creates the layer via the class constructor, the second bracket "*(input)*" is a function with no name implemented via the *__call__()* function, that when called will connect the layers.

The __call__() function is a default function on all Python objects that can be overridden and is used to "*call*" an instantiated object. Just like the __init__() function is a default function on all objects called just after instantiating an object to initialize it.

We can do the same thing in two lines:

```
# create layer
dense1 = Dense(32)
# connect layer to previous layer
dense1(input)
```

I guess we could also call the *__call__()* function on the object explicitly, although I have never tried:

```
# create layer
dense1 = Dense(32)
# connect layer to previous layer
dense1.__call_(input)
```

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.

More On This Topic

How to Load, Convert, and Save Images With the Keras API

Functional Programming in Python

A Gentle Introduction to the tensorflow.data API

Use Keras Deep Learning Models with Scikit-Learn in Python

How to Use Metrics for Deep Learning with Keras in Python

How to Use Word Embedding Layers for Deep Learning…

About Jason Brownlee

Jason Brownlee, PhD 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 Develop a Seq2Seq Model for Neural Machine Translation in Keras

Deep Convolutional Neural Network for Sentiment Analysis (Text Classification) ›

313 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!

Fathi January 7, 2019 at 3:22 am #

REPLY

I'm using a keras API and I'm using the shared layers with 2 inputs and one output and have a problem with the fit model

```
model.fit([train_images1, train_images2],
batch_size=batch_size,
epochs=epochs,
validation_data=([test_images1, test_images2]))
```

I have this error :

ValueError: Error when checking model input: the list of Numpy arrays that you are passing to your model is not the size the model expected. Expected to see 2 array(s), but instead got the following list of 1 arrays: [array([[[[0.2 , 0.5019608 , 0.8862745 , …, 0.5686275 , 0.5137255 , 0.59607846],

[0.24705882, 0.3529412 , 0.31764707, …, 0.5803922 ,
0.57254905, 0.6],

Jason Brownlee January 7, 2019 at 6:37 am #

REPLY

Perhaps double check the data has the shape expected by each model head.

Camilo February 4, 2021 at 2:49 pm #

REPLY

could you fix it?

Arbaz August 22, 2021 at 2:38 am #

REPLY

You haven't provided the output in the fit method, that's why its asking for 2 numpy arrays.

Ahmad August 22, 2021 at 7:23 pm #

REPLY

Fathi, could you fix ??

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 blog, where did you hide all of this 😉
Can you please write a book where you implement more of the functional API?
I'm sure the book will be a real success

Best regards
Thabet

Tom October 27, 2017 at 12:18 pm #

REPLY

I agree with Thabet Ali, need a book on advance functional API part.

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/human+activity+recognition+using+smartphones

Where there are three time series inputs and 5 different classes as output

Franco April 19, 2018 at 5:34 am #

REPLY

Hi Jason, are you asking for a wish list? 🙂
– Autoencoders / Anomaly detection with Keras's API

Thanks!

Jason Brownlee April 19, 2018 at 6:38 am #

REPLY

Thanks for the suggestion!

Franco April 19, 2018 at 5:32 am #

REPLY

Yes, totally agree.

Alexander October 27, 2017 at 6:50 pm #

REPLY

Jason, thank you for very interest blog. Beautiful article which can open new doors.

Jason Brownlee October 28, 2017 at 5:08 am #

REPLY

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 #

REPLY

Thank you for the awesome tutorial.

For anyone wants to directly visualize in Jupyter Notebooks use the following lines.

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

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

Jason Brownlee November 7, 2017 at 9:49 am #

REPLY

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 something here?

Jason Brownlee November 25, 2017 at 10:23 am #

REPLY

Yes, I was incorrect. This is only the case for 1D input. I have updated the post, thanks.

Gerhard Lemmer October 14, 2019 at 6:57 pm #

REPLY

You are still incorrect. There is no “hanging last dimension”, even in the case of 1D input. It’s a “trailing comma” used by Python to disambiguate 0- or 1-tuples from parenthesis.

Jason Brownlee October 15, 2019 at 6:09 am #

REPLY

Thanks for your note.

Franek December 1, 2017 at 9:14 pm #

REPLY

Jason, I’ve been reading about keras and AI for some time and I always find your articles more clear and straightforward than all the other stuff on the net 🙄
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, ahead)

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

hidden = 50

#LSTM
model = Sequential()
model.add(LSTM(input_dim=features, output_dim=hidden,  return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dropout(0.2))
model.add(Dense(input_dim=hidden, output_dim=ahead))
model.add(Activation('linear'))
model.compile(loss='mae', optimizer='adam')
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)

```

```
7 m = Model(inputs=[inputs], outputs=[outputA, outputB])
8 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?

Owolabi June 15, 2020 at 3:58 pm #

REPLY

Check the write-up on lstms, this will greatly help. The output series must have dimensions as n,2,21 ; n data with 2 timestep and 21 variables.
Y.reshape((440/2),2,21)

Owolabi June 15, 2020 at 3:47 pm #

REPLY

The x dimension is already in place, change the y dimension to have 3 dimensions. Num. Samples, time step, variables

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 am #

REPLY

Hii Jason, this was a great post, specially for beginners to learn the Functional API. Would you mind to write a post to explain the code for a image segmentation problem step by step for beginners?

Jason Brownlee January 5, 2018 at 5:27 am #

REPLY

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 #

REPLY

Good tutorial
Thanks alot

Jason Brownlee January 28, 2018 at 8:22 am #

REPLY

Thanks.

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

REPLY

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?

Joseph February 15, 2018 at 3:42 am #

REPLY

I was wondering if was possible to have two separate layers as inputs to the output layer, without concatenating them in a new layer and then having the concatenated layer project to the output layer. If you can't do this with Keras, could you suggest another library that allows you to do this? I am new to neural networks, so would prefer a library that is suitable for newbies.

Jason Brownlee February 15, 2018 at 8:50 am #

REPLY

There are a host of merge layers to choose from besides concat.

Does that help?

vinayakumar r February 16, 2018 at 12:16 am #

REPLY

I have two images, first image and its label is good, second images and its label is bad. I want to pass both images at a time to deep learning model for training. While testing I will have two images (unlabelled) and I want to detect which one is good and which one is bad. Could you please tell how to do it?

Jason Brownlee February 16, 2018 at 8:34 am #

REPLY

You will need a model with two inputs, one for each image.

vinayakumar r February 24, 2018 at 5:57 pm #

REPLY

Any examples you have to give two inputs to a model

Jason Brownlee February 25, 2018 at 7:41 am #

REPLY

Yes, see this caption example for inputting a photo and text:
https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

John February 25, 2018 at 7:28 am #

REPLY

Thanks so much for the tutorial! It is much appreciated.

Where I am confused is for a model with multiple inputs and multiple outputs. To make it simple, lets say we have two input layers, some shared layers, and two output layers. I was able to build this in Keras and get a model printout that looks as expected, but when I go to fit the model Keras complains: ValueError: All input arrays (x) should have the same number of samples

Is it not possible to feed in inputs of different sizes?

Jason Brownlee February 25, 2018 at 7:47 am #

REPLY

Correct. Inputs must be padded to the same length.

saira March 8, 2018 at 6:54 am #

REPLY

Hi,

In the start of the post, you talked about hanging dimension to entertain the mini batch size. Could you kindly explain this a little.
My feature Matrix is a numpy N-d Array, in one -hot -encoded form: (6000,200) , and my batch size = 150.

Does this mean, I should give shape=(200,) ?

saira March 8, 2018 at 6:55 am #

REPLY

* batch size = 50.

Thanks!

Jason Brownlee March 8, 2018 at 2:53 pm #

REPLY

Sorry, it means a numpy array where there is really only 1D of data with the second dimension not specified.

For example:

```
from numpy import array
a = array([1,2,3])
print(a.shape)
```

Results in:

```
(3,)
```

It's 1D, but looks confusing to beginners.

saira March 8, 2018 at 5:09 pm #

REPLY

This means I should use shape(200,).

Thanks a lot for the prompt reply !!!

H Hua March 9, 2018 at 1:32 am #

REPLY

how to do a case with multi-input and multi-output cases

Jason Brownlee March 9, 2018 at 6:24 am #

REPLY

Simply combine some of the examples from this post.

Kieu My March 9, 2018 at 5:18 am #

REPLY

Hi Jason,
Thank you very much for your blog, it's easy to understand via some examples, I recognize that learning from some example is one of the fast way to learn new things.
In your post, I have a little confuse that in case multi input. If you have 1 image but you want get RGB 32x32x3 version and 64x64x1 gray-scale version for each Conv branch. How can the network know that.

Because when we define the network we only said the input_shape, we don't say which kind of image we want to led into the Conv in branch 1 or branch 2? In fit method we also have to said input and output, not give the detail. And if I want in the gray-scale version is: 32x32x3 (3 because I want to channel-wise, triple gray-scale version). And how can the network recognize the first branch is for gray-scale. Sorry for my question if is there any easy thing I don't know. Thanks again for your post. I always follow your post.

Jason Brownlee March 9, 2018 at 6:28 am #

REPLY

You could run all images through one input and pad the smaller image to the dimensions of the larger image. Or you can use a multiple input model and define two separate input shapes.

saira March 9, 2018 at 5:32 pm #

REPLY

Hi,

When I run this functional API in model for k fold cross validation, the numbers in the naming the dense layer is increasing in the return fitted model of each fold.
Like in first fold it's "dense_2_acc", then in 2nd fold its "dense_5_acc".

By my model summary shows my model is correct. Could you kindly tell why is it changing the names in the fitted model "history" object of each fold?

regards,

Jason Brownlee March 10, 2018 at 6:23 am #

REPLY

Sorry, i have not seen this behavior. Perhaps it's a fault? You could try posting to the keras list: https://machinelearningmastery.com/get-help-with-keras/

Jane March 16, 2018 at 4:33 pm #

REPLY

This was a fantastic and concise beginner tutorial for building neural networks with Keras. Great job !

Jason Brownlee March 17, 2018 at 8:31 am #

REPLY

Thanks, I'm glad to hear that.

HARISANKAR HARIDAS March 19, 2018 at 10:44 pm #

REPLY

Thanks for the tutorial.

“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,)

…
visible = Input(shape=(2,))

”

I was a bit confused at first after reading these 2 sentences.

regarding trailing comma: A trailing comma is always required when we have a tuple with a single element. Otherwise (2) returns only the value 2, not a tuple with the value 2.
https://docs.python.org/3/reference/expressions.html#expression-lists

As the shape parameter for Input should be a tuple (https://keras.io/layers/core/#input), we do not have any option other than to add a comma when we have a single element to be passed.

So, I’m not able to get the meaning implied in “the shape must explicitly leave room for the shape of the mini-batch size … Therefore, the shape tuple is always defined with a hanging last dimension”

Jeremy March 27, 2018 at 2:50 am #

REPLY

Hi Jason,

Thanks so much for such a great post.

So, in your case in shared input layers section, you have the same CNN models for feature extraction, and the output can be concated since both features produced binary classification result.

But what if we have separate categorical classification model (for sequence classification) and regression model (for time series) which relies on the same input data. So is it possible to concate categorical classification model (which produces more than two classes) with a regression model, and the final result after model concatenation is binary classification?

Your opinion, in this case, is much appreciated.

Thank you.

Jason Brownlee March 27, 2018 at 6:39 am #

REPLY

Not sure I follow. Perhaps try it and as many variations as you can think of, and see.

REPLY

Dhruv May 2, 2018 at 1:59 pm #

Hi Jason, thanks for the neat post.

REPLY

Jason Brownlee May 3, 2018 at 6:30 am #

You're welcome, I'm glad it helped.

REPLY

Wayne Satz May 5, 2018 at 8:11 am #

Thanks Jason, great and helpful post
Can you go over combining wide and deep models using th functional api?

REPLY

Jason Brownlee May 6, 2018 at 6:19 am #

Thanks for the suggestion.

Do you have a specific question or concern with the approach?

REPLY

Ankush Chandna May 15, 2018 at 11:31 pm #

Thanks Jason,

Your articles are the best and the consistency across articles is something to be admired.

Can you also explain residual nets using functional api.

Thanks

REPLY

Jason Brownlee May 16, 2018 at 6:04 am #

Thanks, and thanks for the suggestion.

REPLY

mina May 25, 2018 at 7:17 pm #

Thank you so much for your great post.
though I have one question, I use the Multiple Input and Output Models with same network for my inputs. I wanna share the weights between them, can you please point out how should I address that?

REPLY

Jason Brownlee May 26, 2018 at 5:53 am #

Copy them between layers or use a wrapper that lets you reuse a layer, e.g. like timedistributed.

REPLY

Ahmed Sahlol June 11, 2018 at 1:55 am #

All ur posts r awesome. God bless u 🙂

REPLY

Jason Brownlee June 11, 2018 at 6:09 am #

hanks, I'm glad they help.

REPLY

Ahmed Sahlol June 11, 2018 at 9:32 am #

Really, amazing tutorial.
Why don't u complete it with the testing step "predict"?
Thanks again 🙂

REPLY

Jason Brownlee June 11, 2018 at 1:50 pm #

Thanks for the suggestion.

I explain how to make predictions here:
https://machinelearningmastery.com/how-to-make-classification-and-regression-predictions-for-deep-learning-models-in-keras/

REPLY

Ahmed Sahlol June 12, 2018 at 1:24 am #

The "Shared Input Layer" is very interesting. I wonder if the 2 convolutional structures can be replaced by 2 pre-trained models (let's say VGG16 and Inception). What do u think?

Jason Brownlee June 12, 2018 at 6:45 am #

REPLY

Sure, try it.

Shardul Singh June 12, 2018 at 7:22 pm #

REPLY

This question is with reference to your older post on “Multilayer Perceptron Using the Window Method” : https://machinelearningmastery.com/time-series-prediction-with-deep-learning-in-python-with-keras/

In your code there, you have successfully created a model using a multidimensional array input, without having to flatten it.
Is it possible to do this with the keras functional API as well? Every solution i find seems like it requires flattening of data, however i’m trying to do a time series analysis and flattening would lead to loss of information.

Jason Brownlee June 13, 2018 at 6:16 am #

REPLY

The shape of the input is unrelated to use the use of the functional API.

You can use either API regardless of the shape of the data.

Time series data must be transformed into a supervised learning problem:
https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

Joshna June 14, 2018 at 3:09 pm #

REPLY

Hey,
Thanks for the blog. I want to know how can I extract features form intermediate layer in Alexnet model . I am using functional api .

Jason Brownlee June 14, 2018 at 4:09 pm #

REPLY

You can create a new model that ends at the layer of interest, then use a forward propagation (e.g. call predict()) to get the features.

I give an example for VGG in the image captioning tutorial:
https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

joost zeeuw June 15, 2018 at 6:13 pm #

REPLY

Hi Jason!

Great blog once again, thank you. I have a question regarding the current research on multi-input models.

I'm building a model that combines text-sequences and patient-characteristics. For this I'm using an LSTM 'branch' that i concat with a normal 'branch' in a neural network. I was wondering whether you came across some nice papers/articles that go a little deeper into such architectures, possibly giving me some insights in how to optimize this model and understand it thoroughly.

With kind regards,
Joost Zeeuw

Jason Brownlee June 16, 2018 at 7:25 am #

REPLY

Not off hand. I recommend experimenting a lot with the architecture and see what works best for your dataset.

I'd love to hear how to you go.

Isaac July 3, 2018 at 6:27 am #

REPLY

Hi Jason! Really great blog!

My question is: how to feed this kind of models with a generator? Well two generators actually, one for test, and one for train. I'm trying to do phoneme classification BTW

I have tried something like:

```
#model
input_data = Input(name='the_input', shape=(None, self.n_feats))
```

x = Bidirectional(LSTM(20, return_sequences=False, dropout=0.3), merge_mode='sum')(input_data)
y_pred = Dense(39, activation="softmax"), name="out")(x)

labels = Input(name='the_labels', shape=[39], dtype='int32') # not sure of this but how to compare labels otherwise??
self.model = Model(inputs=[input_data, labels], outputs=y_pred)
...
I'm gonna omit the optimization and compile steps for simplicty

my generator yields something like this:

return ({'the_input':data_x, 'the_labels':labels},{'out':np.zeros([batch_size, np.max(seq_lens), num_classes])})

Also, just to be sure for sequence classification (many-to-one) I should use return_sequences=False in recurrent layers and Dense instead of TimeDistributed rigth?

Thanks!
Isaac

REPLY

Jason Brownlee July 3, 2018 at 6:30 am #

I have an example of using a generator here (under progressive loading):
https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

REPLY

priya July 11, 2018 at 3:22 am #

Hi Jason,

Thanks for the blog. It is very interesting. After reading your blog, I got one doubt if you can help me out in solving that – what if one wants to extract feature from an intermediate layer from a fine-tuned Siamese network which is pre-trained with a feed-forward multi-layer perceptron.

Is there any lead that you can provide. It would be very helpful to me.

REPLY

Jason Brownlee July 11, 2018 at 6:01 am #

You can get the weights for a network via layer.get_weights()

REPLY

Hee July 16, 2018 at 5:06 pm #

Hi, Thanks for your article.
I have one question.
What is the more efficient way to combine discrete and continuous features layers?

REPLY

Jason Brownlee July 17, 2018 at 6:12 am #

Often an integer encoding, one hot encoding or an embedding layer are effective for categorical variables.

REPLY

Vivek July 17, 2018 at 7:03 am #

Hi, Jason your blog is very good. I want to add custom layer in keras. Can you please explain how can I do?

REPLY

Jason Brownlee July 17, 2018 at 2:30 pm #

Thanks.

I hope to cover that topic in the future.

REPLY

Matt July 20, 2018 at 3:48 pm #

Hi Jason,

Thanks for the excellent post. I attempted to implement a 1 hidden layer with 2 neurons followed by an output layer, both dense with sigmoid activation to train on XOR input – classical problem, that of course has a solution. However, without specifying a particular initialisation, I was unable to train this minimal neuron network toward a solution (with high enough number of neurons, I think it is working independent of initialisation). Could you include such a simple example as a test case of Keras machinery and perhaps comment on the pitfalls where presumably the loss function has multiple critical points?

Cheers,
Matt

REPLY

Jason Brownlee July 21, 2018 at 6:30 am #

Thanks for the suggestion.

XOR is really only an academic exercise anyway, perhaps focus on some real datasets?

REPLY

Sola July 22, 2018 at 10:40 am #

Thanks for your excellent tutorials. I am trying to use Keras Functional API for my problem. I have two different sets of input which I am trying to use a two input – one output model.
My model looks like your “Multiple Input Model” example and as you mentioned I am doing the same thing as :
model = Model(inputs=[visible1, visible2], outputs=output)
and I am fitting the model with this code:
model.fit([XTrain1, XTrain2], [YTrain1, YTrain2], validation_split=0.33, epochs=100, batch_size=150, verbose=2), but I’m receiving error regarding the size mismatching.

The output TensorShape has a dimension of 3 and YTrain1 and YTrain2 has also the shape of (–, 3). Do you have any suggestion on how to resolve this error? I would be really thankful.

Jason Brownlee July 23, 2018 at 6:04 am #

REPLY

If the model has one output, you only need to specify one yTrain.

Sola August 10, 2018 at 5:11 am #

REPLY

Hi

Thank you for your reply.
I have another question which I will be grateful if you could help me with that.
In your Multilayer Perceptron example, which the input data is 1-D, if I add a reshape module at the end of the Dense4 to reshape the output into a 2D object, then is it possible to see this 2D feature space as an image?
Is there any syntax to plot this 2D tensor object?
Thanks

Jason Brownlee August 10, 2018 at 6:21 am #

REPLY

If you fit an MLP on an image, the image pixels must be flattened to 1D before being provided as input.

tuan anh July 25, 2018 at 5:45 pm #

REPLY

Thanks, Jason
Can you give me an example of how to combine Conv1D => BiLSTM => Dense
I try to do but can’t figure out how to combine them

Jason Brownlee July 26, 2018 at 7:37 am #

REPLY

This will help as a start:
https://machinelearningmastery.com/cnn-long-short-term-memory-networks/

tuan anh July 26, 2018 at 12:24 pm #

REPLY

Thank you so much for quick reply Jason, I read this article, very useful!

But when I apply, I face that it has a very strange thing, I don't know why:

Let see my program, it runs normally, but the val_acc, I don't know why it always .] – ETA: 0s – loss: 0.2195 – acc: 0.8978
Epoch 00046: loss improved from 0.22164 to 0.21951,

40420/40420 [==============================] – 386s – loss: 0.2195 – acc: 0.8978 – val_loss: 5.2004 – val_acc: 0.2399
Epoch 48/100
40416/40420 [==============================>.] – ETA: 0s – loss: 0.2161 – acc: 0.9010
Epoch 00047: loss improved from 0.21951 to 0.21610,

40420/40420 [==============================] – 390s – loss: 0.2161 – acc: 0.9010 – val_loss: 5.0661 – val_acc: 0.2369
Epoch 49/100
40416/40420 [==============================>.] – ETA: 0s – loss: 0.2274 – acc: 0.8965
Epoch 00048: loss did not improve
40420/40420 [==============================] – 393s – loss: 0.2276 – acc: 0.8964 – val_loss: 5.1333 – val_acc: 0.2412
Epoch 50/100
40416/40420 [==============================>.] – ETA: 0s – loss: 0.2145 – acc: 0.9028
Epoch 00049: loss improved from 0.21610 to 0.21455,

40420/40420 [==============================] – 395s – loss: 0.2146 – acc: 0.9027 – val_loss: 5.3898 – val_acc: 0.2344
Epoch 51/100
40416/40420 [==============================>.] – ETA: 0s – loss: 0.2100 – acc: 0.9051
Epoch 00050: loss improved from 0.21455 to 0.20999,

Jason Brownlee July 26, 2018 at 2:25 pm #

REPLY

You may need to tune the network to your problem.

tuan anh July 26, 2018 at 3:28 pm #

I tried many times, but even it overfits all database, val_acc still low.
I know it overfits all because I use predict program to predict all database, acc high as training acc.
Thank you

Jason Brownlee July 27, 2018 at 5:45 am #

Perhaps try adding some regularization like dropout?

Perhaps getting more data?

Perhaps try reducing the number of training epochs?

Perhaps try reducing the size of the model?

tuan anh July 30, 2018 at 12:29 pm #

REPLY

thank you, Jason,
– I am trying to test by adding some dropout layers,
– the number of epochs when training doesn't need to reduce because I observe it frequently myself,
– about the size of the model, I am training 4 programs in parallel to check it.
– the last one, getting more data, I will do if all of above have better results

Jason Brownlee July 30, 2018 at 2:17 pm #

REPLY

Sounds great.

sahand September 1, 2018 at 10:27 pm #

REPLY

hi Jason tnx for this awesome post
really helpful

when i run this code:

```
l_input = Input(shape=(336, 25))
adense = GRU(256)(l_input)
bdense = Dense(64, activation='relu')(adense)
.
.
.
```

i'll get this error:

ValueError: Invalid reduction dimension 2 for input with 2 dimensions. for 'model_1/gru_1/Sum' (op: 'Sum') with input shapes: [?,336], [2] and with computed input tensors: input[1] = .

i'm really exhausted and i didn't find the answer anywhere.
what should i do?
i appreciate your help

Jason Brownlee September 2, 2018 at 5:31 am #

REPLY

Sounds like the data and expectations of the model do not match. Perhaps change the data or the model?

Bradley Elfman September 15, 2018 at 5:33 am #

REPLY

This is a particularly helpful tutorial, but I cannot begin to use without data source.

Jason Brownlee September 15, 2018 at 6:19 am #

REPLY

Thanks.

Bradley Elfman September 15, 2018 at 6:00 am #

REPLY

I left a previous reply about needing data sources, I see other readers not having this problem, but seems I am still at the stage where I don't see what data to input or how to preprocess for these examples. I am also confused, as looks like a png is common source.

I am particularly interested in example that takes text and question and returns an answer – where would I find such input and how to fit into your code?

Bradley Elfman September 15, 2018 at 10:46 pm #

REPLY

Jason, What dataset from your github datasets would be good for this LSTM tutorial? Or is there an online dataset you could recommend. I am interested in both LSTM for text processing (not IMDB) and Keras functional API

Jason Brownlee September 16, 2018 at 6:02 am #

REPLY

Not sure I follow what you are trying to achieve?

Mark October 2, 2018 at 5:44 pm #

REPLY

Any chance of a tutorial on this using some real/toy data as a vehicle

REPLY

Jason Brownlee October 3, 2018 at 6:14 am #

I have many deep learning tutorials on real dataset, you can get started here:
https://machinelearningmastery.com/start-here/#deeplearning

And here:
https://machinelearningmastery.com/start-here/#nlp

And here:
https://machinelearningmastery.com/start-here/#deep_learning_time_series

REPLY

rim October 4, 2018 at 9:25 pm #

Thank you Mr.Jason,

Can you help me to predict solar radiation using kalman filter?

Have you a matlab code about kalman filter for solar radiation prediction.

Best regards

REPLY

Jason Brownlee October 5, 2018 at 5:36 am #

Sorry, I don't have examples in matlab nor an example of a kalman filter.

REPLY

Gledson October 7, 2018 at 3:08 am #

Hello how are you? Sorry for the inconvenience. I'm following up on his explanations of Keras using neural networks and convolutional neural networks. I'm trying to perform a convolution using a set of images that three channels each image and another set of images that has one channel each image. When I run a CNN with Keras for each type of image, I get a result. So I have two inputs and one output. The entries are X_train1 with size of (24484,227,227,1) and X_train2 with size of (24484,227,227,3). So I perform a convolution separately for each input and then I use the "merge" command from KERAS, then I apply the "merge" on a CNN. However, I get the following error:
ValueError: could not broadcast input array from shape (24484,227,227,1) into shape (24484,227,227).

I already tried to take the number 1 and so stick with the shape (24484,227,227). So it looks like it's right. But the error happens again in X_train2 with the following warning:
ValueError: could not broadcast input array from shape (24484,227,227,3) into shape (24484,227,227).
However, I can not delete the number "3".

Could you help me to eliminate this error?

My code is:

X_train1: shape of (24484,227,227,1)
X_train2: shape of (24484,227,227,3)
X_val1: shape of (2000,227,227,1)
X_val2: shape of (2000,227,227,3)

```
batch_size=64
num_epochs=30
DROPOUT = 0.5

model_input1 = Input(shape = (img_width, img_height, 1))
DM = Convolution2D(filters = 64, kernel_size = (1,1), strides = (1,1), activation = “relu”)(model_input1)
DM = Convolution2D(filters = 64, kernel_size = (1,1), strides = (1,1), activation = “relu”)(DM)

model_input2 = Input(shape = (img_width, img_height, 3))
RGB = Convolution2D(filters = 64, kernel_size = (1,1), strides = (1,1), activation = “relu”)(model_input2)
RGB = Convolution2D(filters = 64, kernel_size = (1,1), strides = (1,1), activation = “relu”)(RGB)

merge = concatenate([DM, RGB])

# First convolutional Layer
z = Convolution2D(filters = 96, kernel_size = (11,11), strides = (4,4), activation = “relu”)(merge)
z = BatchNormalization()(z)
z = MaxPooling2D(pool_size = (3,3), strides=(2,2))(z)

# Second convolutional Layer
z = ZeroPadding2D(padding = (2,2))(z)
z = Convolution2D(filters = 256, kernel_size = (5,5), strides = (1,1), activation = “relu”)(z)
z = BatchNormalization()(z)
z = MaxPooling2D(pool_size = (3,3), strides=(2,2))(z)

# Rest 3 convolutional layers
z = ZeroPadding2D(padding = (1,1))(z)
z = Convolution2D(filters = 384, kernel_size = (3,3), strides = (1,1), activation = “relu”)(z)

z = ZeroPadding2D(padding = (1,1))(z)
z = Convolution2D(filters = 384, kernel_size = (3,3), strides = (1,1), activation = “relu”)(z)

z = ZeroPadding2D(padding = (1,1))(z)
z = Convolution2D(filters = 256, kernel_size = (3,3), strides = (1,1), activation = “relu”)(z)

z = MaxPooling2D(pool_size = (3,3), strides=(2,2))(z)
z = Flatten()(z)

z = Dense(4096, activation=”relu”)(z)
z = Dropout(DROPOUT)(z)

z = Dense(4096, activation=”relu”)(z)
z = Dropout(DROPOUT)(z)
```

```
model_output = Dense(num_classes, activation='softmax')(z)
model = Model([model_input1,model_input2], model_output)
model.summary()

sgd = SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
optimizer=sgd,
metrics=['accuracy'])

print('RGB_D')

datagen_train = ImageDataGenerator(rescale=1./255)
datagen_val = ImageDataGenerator(rescale=1./255)

print("fit_generator")
# Train the model using the training set…
Results_Train = model.fit_generator(datagen_train.flow([X_train1,X_train2], [Y_train1,Y_train2], batch_size =
batch_size),
steps_per_epoch = nb_train_samples//batch_size,
epochs = num_epochs,
validation_data = datagen_val.flow([X_val1,X_val1], [Y_val1,Y_val2],batch_size = batch_size),
shuffle=True,
verbose=1)

print(Results_Train.history)
```

Jason Brownlee October 7, 2018 at 7:26 am #

REPLY

Looks like a mismatch between your data and your model. You can reshape your data or change the expectations of your model.

Fathi January 7, 2019 at 4:46 am #

REPLY

Thank you for all the informations.

Do you have any example with Keras API using shared layers with 2 inputs and one output.

I want to knew how to use every input to get it's : Xtrain, Ytrain and Xtest, Ytest,

I think, It will be more simple with an example.

Thank you.

Jason Brownlee January 7, 2019 at 6:40 am #

REPLY

Perhaps this will help:
https://machinelearningmastery.com/lstm-autoencoders/

Leeor October 9, 2018 at 12:43 am #

REPLY

Hi Jason.
Do you know of a way to combine models each with a different loss function?

Leeor.

Jason Brownlee October 9, 2018 at 8:44 am #

REPLY

Yes, as an ensemble after they are trained.

Konstantin November 1, 2018 at 11:34 pm #

REPLY

Hi Jason,
thank you for your wonderful tutorials!

I just wonder about the “Convolutional Neural Network” example. Isn’t there a Flatten layer missing between max_pooling2d_2 and dense_1?

Something like:
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flatt = Flatten()(pool2)
hidden1 = Dense(10, activation=’relu’)(flatt)

Beste regards

Jason Brownlee November 2, 2018 at 5:51 am #

REPLY

I think you’re right! Fixed.

Nada November 6, 2018 at 3:31 pm #

REPLY

Hi Jason

In Multiple Input Model, How did you naming the layers?

```
Layer (type) Output Shape Param # Connected to
==================================================================================================
input_1 (InputLayer) (None, 64, 64, 1) 0
__________________________________________________________________________________________________
conv2d_1 (Conv2D) (None, 61, 61, 32) 544 input_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D) (None, 57, 57, 16) 1040 input_1[0][0]
__________________________________________________________________________________________________
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
__________________________________________________________________________________________________
```

Jason Brownlee November 7, 2018 at 5:57 am #

REPLY

If the model is built at one time, the default names are fine.

If the models are built at different times, I give arbitrary names to each head, like: name = ‘head_1_’ + name

REPLY

Or Bennatan November 6, 2018 at 10:29 pm #

Very good insight into Keras.
I also read your Deep_Learning_Time_Series_Forcasting and it was very helpful

REPLY

Jason Brownlee November 7, 2018 at 6:02 am #

Thanks.

It was your email that prompted me to update this post with the Python syntax explanation!

REPLY

Jonathan Roy November 8, 2018 at 2:01 am #

Wow thank a lot for all your post, you save me a lot of time in my learning and prototyping experience!

I use an LSTM layer and want to use the ouput to feed a Dense layer to get an first predictive value ans insert this new value to the first LSTM output and feed an new LSTM layer. Im stuck with the dimension problem…

```
main_inputs = Input(shape=(train_X.shape[1], train_X.shape[2]), name='main_inputs')

ly1 = LSTM(100, return_sequences=False)(main_inputs)
auxiliary_output = Dense(1, activation='softmax', name='aux_output')(ly1)

merged_input = concatenate([main_inputs, auxiliary_output])

ly2 = LSTM(100, return_sequences=True)(merged_input)
main_output = Dense(1, activation= 'softmax', name='main_output')(ly2)
```

Any suggestion is welcome

REPLY

Jason Brownlee November 8, 2018 at 6:12 am #

What’s the problem exactly?

I don’t have the capacity to debug your code, perhaps post to stackoverflow?

REPLY

Alaya November 11, 2018 at 5:25 am #

Thanks Jason for the post 🙂

In the multi-input CNN model example, does the two images enter to the model at the same time has the same index?

does the two images enter to the model at the same time has the same class?

In training, does each black image enters to the model many times (with all colored images) or each black image enters to the model one time (with only one colored image)?

Thanks..

Jason Brownlee November 11, 2018 at 6:12 am #

REPLY

Both images are provided to the model at the same time.

Alaya November 11, 2018 at 8:18 am #

REPLY

does the two images enter to the model at the same time has the same class?

Jason Brownlee November 12, 2018 at 5:34 am #

REPLY

It really depends on the problem that you are solving.

bit_scientist February 11, 2020 at 5:13 pm #

REPLY

Hi Jason, I was looking for the comments hoping someone would ask a similar question.

I have images and their corresponding numeric values. If I am to construct a fine-tuned VGG model for images and MLP (or any other) for numeric values and concatenate them just how you did in this post, how do I need to keep the correspondence between them?

Is it practically possible to input images (and numeric values) into the model by some criteria, say, names of images? Because my images' names and one column in my numeric dataset keeps the names for samples.

Thanks a lot.

Jason Brownlee February 12, 2020 at 5:42 am #

REPLY

You can use a multi-input model, one input for the image, one for the number.

The training dataset would be 2 arrays, one array of image one of numbers. The rows would correspond.

Does that help?

bit_scientist February 12, 2020 at 11:39 am #

Thank you very much. To clarify for myself, please let me know your feedback for these:

1. Do you mean that I need to convert the images into an array and,based on their names, append to the numeric data file (csv) as a new column ?

2. I haven't seen such a numeric data where it contains RGB values of images as an array in one column. Could you post some related links/ sources?

I appreciate your feedback. Thanks again!

Jason Brownlee February 12, 2020 at 1:37 pm #

No. There would be one array of images and one array of numbers and the rows between them would correspond. e.g. row 0 in the first array would be an image that would relate to the number in row 0 of the second array.

This will help you work with arrays if it is new:
https://machinelearningmastery.com/gentle-introduction-n-dimensional-arrays-python-numpy/

This will help you load images as arrays:
https://machinelearningmastery.com/how-to-load-and-manipulate-images-for-deep-learning-in-python-with-pil-pillow/

Meiling November 14, 2018 at 2:46 am #

REPLY

Hi, now I want to use a 1-D data like wave.shape=(360,) as input, and 3-D data like velocity.shape=(560,7986,3) as output. I want to ask if this problem can be solved by multilayers perceptron to tain these data? I have tried, but the shape problem is not solved, it shows "ValueError: Error when checking target: expected dense_3 to have 2 dimensions, but got array with shape (560, 7986, 3)"

Jason Brownlee November 14, 2018 at 7:35 am #

REPLY

Perhaps, it really comes down to what the data represents.

CHYOO December 2, 2018 at 12:33 am #

REPLY

Hello, thank you for sharing the contents 🙂
Is it same that the ‘Multi-output’ case with ‘multi-task deep learning’ ?

I am trying to build up the multi-task deep learning model and found here.
Thank you again.

Jason Brownlee December 2, 2018 at 6:21 am #

REPLY

Yes, it can be.

Antonio December 7, 2018 at 11:06 pm #

REPLY

Jason, a very modest contribution for now. Just a typo. In,

…Shared Feature Extraction Layer
In this section, we will two parallel submodels to interpret the output…

it looks like we are missing a verb or something in the sentence, it sounds strange.
If it sounds OK to you, just disregard, it must be me being tired.

I hope to support more substantially in the future this extraordinary site.

(You don’t need to post this comment)

Regards
Antonio

Jason Brownlee December 8, 2018 at 7:10 am #

REPLY

Thanks, fixed!

(I like to keep these comments in there, to show that everything is a work in progress and getting incrementally better)

Liu December 14, 2018 at 2:16 am #

REPLY

Jason, thank you very much for your tutorial, it is very helpful! I have a question, how would the ModelCheckpoint callback work with multiple outputs? If I set save_best_only = True what will be saved? Is it the model that yields the best overall result for both outputs, or will there be two models saved?

Jason Brownlee December 14, 2018 at 5:32 am #

REPLY

Really good question, I go into this in great detail in this post: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Rojin December 15, 2018 at 10:41 am #

REPLY

Can you put a post on how to make partially connected layers in Keras? With a predefined neurons connections (prev layer to the next layer)

Jason Brownlee December 16, 2018 at 5:19 am #

REPLY

Thanks for the suggestion.

alija January 2, 2019 at 2:49 am #

REPLY

Hi Jason,

If we want to write predictions to a separate txt file, what we have to add at the end of the code?

Thanks.

Jason Brownlee January 2, 2019 at 6:41 am #

REPLY

You can save the numpy array to a csv file directly with the savetext() function.

Ritesh January 11, 2019 at 8:17 pm #

REPLY

Hi Jason, I really loved this article. I have an one application in that data is in csv file with text data which has four columns and i'm considering first three columns as input data to predict fourth column data as my output. I need to take first three column as input because data is dependent. can you guide me using glove how can i train model?
Thanks

Jason Brownlee January 12, 2019 at 5:39 am #

REPLY

Perhaps this tutorial will help:
https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

Kaiche January 31, 2019 at 7:41 pm #

REPLY

Hi Jason

Thanks again for good tutorial, i want to know the different when concatenate two layer as feature extraction. You use merge = concatenate([interp1, interp13])

Other people use merge = concatenate([interp1, interp13], axis = -1). I want to know is there different between the two and how different is it

Jason Brownlee February 1, 2019 at 5:35 am #

REPLY

They are the same thing. The axis=-1 is the default and does not need to be specified. Learn more here:
https://keras.io/layers/merge/

majed February 3, 2019 at 7:55 pm #

REPLY

Hi Jason

Thanks for good tutorial, i want use Multiple Input Model with fit generator

```
model.fit_generator(generator=fit_generator,
steps_per_epoch=steps_per_epoch_fit,
epochs=epochs,
verbose=1,
validation_data=val_generator,
validation_steps=steps_per_epoch_val)
```

but i get ValueError: Error when checking model input: the list of Numpy arrays that you are passing to your model is not the size the model expected. Expected to see 2 array(s), but instead got the following list of 1 arrays

keras code

```
input1 = Input(shape=(145,53,63,40))
input2 = Input(shape=(145,133560))
#feature 1
conv1 = Conv3D(16, (3,3,3),data_format ='channels_last')(input1)
pool1 = MaxPooling3D(pool_size=(2, 2,2))(conv1)
batch1 = BatchNormalization()(pool1)
```

ac1=Activation('relu')(batch1)

.....

ac3=Activation('relu')(batch3)

flat1 = Flatten()(ac3)
#feature 2

gr1=GRU(8,return_sequences=True)(input2)
batch4 = BatchNormalization()(gr1)
ac4=Activation ('relu')(batch4)

flat2= Flatten()(ac4)

merge feature extractors

merge = concatenate([flat1, flat2])

output = Dense(2)(merge)
out=BatchNormalization()(output)
out1=Activation('softmax')(out)

prediction output

model = Model(inputs=[input1,input2], outputs=out1)

opt2 = optimizers.Adam(lr=learn_rate, decay=decay)

model.compile(loss='categorical_crossentropy', optimizer=opt2,
metrics=['accuracy'])

Jason Brownlee February 4, 2019 at 5:46 am #

REPLY

I'm eager to help, but I don't have the capacity to debug your code, I have some suggestions here:
https://machinelearningmastery.com/faq/single-faq/can-you-read-review-or-debug-my-code

---

sanker February 22, 2019 at 4:28 pm #

REPLY

i tried the given section, (5. Multiple Input and Output Models)
please help me to fit the data . i got the error
AttributeError: 'NoneType' object has no attribute 'shape'
my input :

```
visible1 = Input(shape=(200,2, 48))
        conv11 = Conv2D(25, kernel_size=4, activation='relu')(visible1)
        pool11 = MaxPooling2D(pool_size=(2, 2))(conv11)

        flat1 = Flatten()(pool11)
        # second input model
        visible2 = Input(shape=(200,2,48))
```

```
conv21 = Conv2D(25, kernel_size=4, activation='relu')(visible2)
pool21 = MaxPooling2D(pool_size=(2, 2))(conv21)

flat2 = Flatten()(pool21)
# merge input models
merge = concatenate([flat1, flat2])
#print(merge)
# 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())
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
model.fit([ X_train,X_train] ,y_train, epochs=20)
```

Jason Brownlee February 23, 2019 at 6:25 am #

REPLY

Nice work.

Perhaps try some of the suggestions here:
https://machinelearningmastery.com/faq/single-faq/can-you-read-review-or-debug-my-code

Teja March 1, 2019 at 4:29 am #

REPLY

I'am knew to the Machine Learning. After fitting the model using functional API i got training accuracy and loss. but i don't know how to find test accuracy using keras functional API.

Jason Brownlee March 1, 2019 at 6:26 am #

REPLY

You can evaluate the model on the a test dataset by calling model.evaluate()

David March 6, 2019 at 7:40 pm #

REPLY

I want make CNN + convLSTM model,
but, at last line, occured dim error .. how can i fit shape
: Input 0 is incompatible with layer conv_lst_m2d_27: expected ndim=5, found ndim=4
input = Input(shape=(30,30,3), dtype='float32")

conv1 = Convolution2D(kernel_size=1, filters=32, strides=(1,1), activation="selu")(input)
conv2 = Convolution2D(kernel_size=2, filters=64, strides=(2,2), activation="selu")(conv1)
conv3 = Convolution2D(kernel_size=2, filters=128, strides=(2,2), activation="selu")(conv2)
conv4 = Convolution2D(kernel_size=2, filters=256, strides=(2,2), activation='selu')(conv3)

ConvLSTM_1 = ConvLSTM2D(32, 2, strides=(1,1))(conv1)

Jason Brownlee March 7, 2019 at 6:46 am #

REPLY

Perhaps change the input shape to match the expectations of the model or change the model to meet the expectations of the data shape?

Xu Zhang March 12, 2019 at 5:54 am #

REPLY

Thank you so much for your great article.

What should I set up my model with API if my problem is like this:
First, classify if the chemicals are toxic or not, then if they are toxic, what toxic scores they are. I can create two models separately. But combine them together is a good regularization that some papers said that. I think it is a multi-input and multi-output problem. I have a dataset which has the same input shape, combineoutput will be y1 = 0 or 1, and y2= numerical scores if y1=1.

I don't know where and how to put if statment in the combined model.

Any advice is highly appreciated.

Jason Brownlee March 12, 2019 at 7:00 am #

REPLY

Perhaps start here:
https://machinelearningmastery.com/start-here/#deeplearning

Xu Zhang March 12, 2019 at 8:56 am #

REPLY

Maybe I didn't explain my problem clearly.

I can build models for classification and regression separately using Keras without any problems. My problem is how to combine these two into one multi-input, multi-output model with "an IF STATEMENT". From the link you provided, I couldn't find the solutions. Could you please make it clear? Many thanks.

Jason Brownlee March 12, 2019 at 2:30 pm #

REPLY

What would be the inputs and outputs exactly?

Xu Zhang March 13, 2019 at 9:39 am #

It is like this:

ex x1 x2 x3 x4 y1 y2
1 0.1 0.5 0.7 0.4 1 0.3
2 0.5 0.2 0.4 0.1 0
3 0.7 0.6 0.3 0.2 0
4 0.12 0.33 0.05 0.77 1 0.55
.. ..

Only when y1 = 1, y2 has a value

Above is not a real dataset.

Jason Brownlee March 13, 2019 at 2:08 pm #

There may be many ways to approach this problem.

Perhaps you could setup the model to always predict something, and only pay attention to y2 when y1 has a specific value?

Xu Zhang March 14, 2019 at 3:56 am #

First of all, thank you so much for your fast response.
I still didn't get it. You said there are many ways to approach it, but I don't know any of them. There are two datasets (X1, y1), (X2, y2). So I think it is a multi-input and multi-output problem. Should the number of samples between two datasets be equal?

Jason Brownlee March 14, 2019 at 9:28 am #

Yes.

Helina April 21, 2019 at 8:33 pm #

REPLY

Thank you, Jason, for yet another awesome tutorial! You are a very talented teacher!

Jason Brownlee April 22, 2019 at 6:22 am #

REPLY

Thanks, I'm glad it helped.

Zabit May 22, 2019 at 7:55 pm #

REPLY

Hi Jason, I always follow your blogs and book (Linear algebra for ML) and they are extremely helpful. Do you have post related to LSTM layer followed by 2D CNN/Maxpool?. If you have already have a post then please provide a link to it.

Actually I have some problem with dimensions after using the direction given in this post. https://stackoverflow.com/questions/53904688/using-keras-to-build-a-lstmconv2d-model/56253458#56253458

But it did not solve my problem. It will be great if you provide help in this regard.

Thank you

Jason Brownlee May 23, 2019 at 5:58 am #

REPLY

I don't I have an LSTM feeding a CNN, I don't think that makes sense.

Why would you want to do that?

Mostafa Shahriari June 5, 2019 at 1:00 am #

REPLY

Hello,
Firstly thank you so much for the great webpage you have. It has been a great help from the first day I started to work on deep learning. I have a question, though. What is the corresponding loss function for the model with multiple inputs and one output that you have in subsection Multiple Input Model? I want to know if you have X1 and X2 as inputs and Y as outputs, what would be the mathematical expression for the loss function. I want to do the same. However, I am not sure what I will be minimizing.

Another question is about loss weights. If I have, for example, two outputs and their two corresponding loss functions if I set the first loss weights equal to zero, would it deactivate the training process for the part related to the first output?

Thank you so much in advance for your help.

Regards,

Jason Brownlee June 5, 2019 at 8:47 am #

REPLY

The model will have one loss function per output.

Y is only one output though, a single output model with 2 inputs X1 and X2.

If you have a vector output, it is only one loss function, average across each item in the vector output.

On non-trivial problems, you will never hit zero loss unless you overfit the training data.

Mostafa Shahriari June 7, 2019 at 12:48 am #

REPLY

Thank you so much for your reply. But, you misunderstood my question regarding multiple outputs. However, I figured that one out. But my question regarding the network containing two inputs (X1 and X2) and one output is still unanswered. I know it has only one loss, but I am not sure what is the loss. It can be any of the following:

1. [F1(x1)+F2(x2) – Y]

2. [F(x1,x2) – Y]

I am not sure which one will be the loss here. I appreciate if you can help me. Thank you so much.

Jason Brownlee June 7, 2019 at 8:04 am #

REPLY

The input in a multi-input are aggregated, therefore loss is a function of both inputs.

Balaji June 5, 2019 at 3:19 pm #

REPLY

Great Blog!
Can you help us with one example code architecture without using Dense layer?
Finding it difficult to understand the last part to get rid of Dense layers.

Thanks

Jason Brownlee June 6, 2019 at 6:19 am #

REPLY

You can use any output layer you wish.

Why do you want to get rid of the dense layers?

Karthik S June 21, 2019 at 9:06 am #

REPLY

Amazing. Thanks a lot for sharing. Really helpful! You are a wizard…

Jason Brownlee June 21, 2019 at 2:01 pm #

REPLY

I'm happy it helped.

Wesly Vazkez June 28, 2019 at 12:05 am #

REPLY

Amazing post! It's both easy to understand and complex enough to generalize several types of networks. It helps me so much! Thanks!!

Jason Brownlee June 28, 2019 at 6:04 am #

REPLY

Thanks, I'm happy that it helped!

Jake July 23, 2019 at 2:38 am #

REPLY

Hi Jason,

You have help me out a lot and i am back to hitting wall.

I want molding a multiple output multiple input linear regression model and i cant find anything on the internet. i tried using different key words like multi-target linear regression using keras, multi depended Valarie, multivariate. i cant find anything.

Do you have any materials on this?

Jason Brownlee July 23, 2019 at 8:11 am #

REPLY

You can specify the number of targets as the number of nodes in the output layer.

Yes, I have many examples for time series, you can get started here:
https://machinelearningmastery.com/start-here/#deep_learning_time_series

Also this:
https://machinelearningmastery.com/faq/single-faq/how-do-you-use-lstms-for-multi-step-time-series-forecasting

Does that help?

Jake July 23, 2019 at 11:00 pm #

REPLY

Hi Jason,
Thanks a lot for the help for the two links, i didn't know you have your stuff so organized.

I am new to machine learning and I think am a bit confuse.
lets say i have a bunch of data which are all time average from a time series data.
do i really need to use time series? I have 10 columns of input and 5 columns of output. however i am going to be dealing with the time series version of this data set.

also the definition of time series = "n descriptive modeling, or time series analysis, a time series is modeled to determine its components in terms of seasonal patterns, trends, relation to external factors, and the like. … In contrast, time series forecasting uses the information in a time series (perhaps with additional information) to forecast future values of that series" from your blog

What if my data dont have trends, or patterns? like my data are industrial data like, data from a maybe a engine? i want to predict how it will run at certain time or things like that

also i remember reading your post saying that LSTM are not really good for time series forecasting. what different methods can i use ? i dont have image so i cant use CNN.

is RNN good?

sorry for the long read.

Jake July 23, 2019 at 11:09 pm #

REPLY

Update, i just read your blog on CNN for time series molding , I believe you do not require images , i am so sorry for saying "i dont have image so i cant use CNN."

https://machinelearningmastery.com/how-to-develop-convolutional-neural-network-models-for-time-series-forecasting/

For the link above you seems to only be using sequences, do my data have to be sequences?

Jason Brownlee July 24, 2019 at 7:58 am #

Yes. To use a 1D CNN, the model assumes a sequence as input.

Jason Brownlee July 24, 2019 at 7:57 am #

REPLY

The time series may or may not have useful temporal structure that a model can use. You can choose to try to capture that or not – you're right.

But, you must respect the temporal ordering of observations, e.g. train on past, test on future, never mix the two. Otherwise, your model evaluation will be invalid (optimistic).

See this framework for methods to try and the order to try them:
https://machinelearningmastery.com/how-to-develop-a-skilful-time-series-forecasting-model/

luka August 5, 2019 at 5:23 am #

REPLY

Amazing post like always !! thanks Jason.

I have two questions about flattening :

can we replace flatten with an Bi-LSTM?

If i want to use a Bi-LSTM after Embedding, should i flatten the output of Embeddig before go to the Bi-Lstm?

Thanks.

Jason Brownlee August 5, 2019 at 6:54 am #

REPLY

No need to flatten.

priyanshu Khullar August 26, 2019 at 1:12 am #

REPLY

Just Awesome

Jason Brownlee August 26, 2019 at 6:20 am #

REPLY

Thanks, I'm glad it helped.

Hamed September 7, 2019 at 3:58 am #

REPLY

Great as always! But would you mind if you let me know how to address the loss function in case of MIMO? It should be calculated separately for each input/output I guess. Thank you!

Jason Brownlee September 7, 2019 at 5:37 am #

REPLY

No different, if multiple outputs are numbers, mse is a great start.

Rekha September 18, 2019 at 3:31 pm #

REPLY

Can functional models applied to predict stock market

Jason Brownlee September 19, 2019 at 5:50 am #

REPLY

This is a common question that I answer here:
https://machinelearningmastery.com/faq/single-faq/can-you-help-me-with-machine-learning-for-finance-or-the-stock-market

Abdul Damodu September 23, 2019 at 9:08 pm #

REPLY

Thanks a lot for the insightful write-up. I am looking for a way to combine RNN (LSTM) and HMM to predict stock prices based on the combined strength of the two paradigms to achieve better result than ordinary RNN (LSTM). Thank you

Jason Brownlee September 24, 2019 at 7:42 am #

REPLY

Sounds like a fun project, let me know how you go.

Abdul Damodu September 24, 2019 at 9:10 am #

REPLY

Thanks for the response. But I need an insight from you in this regard. Thank you.

Jason Brownlee September 24, 2019 at 1:05 pm #

REPLY

Sorry, I don’t have any examples of working with HMMs or integrating them with LSTMs.

I would recommend developing some prototypes in order to discover what works.

haider jafri February 5, 2020 at 6:36 am #

REPLY

I have created a two output LSTM model to predict the angular and linear velocity, the loss is low in angular velocity and but loss is high in linear velocity. Please tell me, how to reduce the loss.

Epoch 9/10
– 18s – loss: 0.4790 – A_output_loss: 0.0826 – L_output_loss: 0.3964 – A_output_acc: 0.6077 – L_output_acc: 0.4952 – val_loss: 0.6638 – val_A_output_loss: 0.0958 – val_L_output_loss: 0.5680 – val_A_output_acc: 0.6059 – val_L_output_acc: 0.3166
Epoch 10/10
– 18s – loss: 0.4752 – A_output_loss: 0.0821 – L_output_loss: 0.3931 – A_output_acc: 0.6084 – L_output_acc: 0.4996 – val_loss: 0.6503 – val_A_output_loss: 0.0970 – val_L_output_loss: 0.5533 – val_A_output_acc: 0.6052 – val_L_output_acc: 0.3176

Jason Brownlee February 5, 2020 at 8:23 am #

REPLY

Here are some suggestions:
https://machinelearningmastery.com/start-here/#better

Sam February 16, 2020 at 8:06 pm #

REPLY

In the shared layer CNN example, why does the shape changed from 64 to 61, I understand kernel size is 4, but 64/4 has no remainder. Also, do you know if mxnet has similar methods or tutorial on this?

Jason Brownlee February 17, 2020 at 7:45 am #

REPLY

Yes, it comes down to same vs valid padding I believe:
https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/

Sorry, I don't know about mxnet.

Hilbert van Pelt February 19, 2020 at 2:29 pm #

REPLY

Hi Jason,

Thanks for your great website and your great books (we have most of them).

I do have a question I hope you can help we with.

In the article above you describe a large number of different network structures that you can implement. Are there any rules of thumb that describe which network structure works best with which problem? I do bit of work in time series forecasting and anything that I have read tells me to just try different structures, but given the amount of different structures this is quite unpractical.

For example, if you have multiple input sources of data do you concatenate them into a single input for an MLP or do you use a multiple input model?

I would love to hear your though.

Jason Brownlee February 20, 2020 at 6:06 am #

REPLY

You're welcome.

Not really, you must use controlled experiments to discover what works best for your dataset.

This might help as a general guide:
https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

saipavankumar March 10, 2020 at 8:40 pm #

REPLY

Hi Jason, I have been following your blog since I started my college project, I got stuck on this page, My problem is I have a dataset of a bioreactor(fermentation process) which has 100 batches of data and each batch has 1000 timesteps and 200 parameters(variables).
so 100 * 1000 = 1,00,000 timesteps of 200 variables, I wanted to develop a ' Y ' like architecture(like MIMO in your post), therefore from one side of 'Y' inputs are 'observed variables' and from the other side ' controlled variables ', I want to pass observed variables through a LSTM layer where I am confused with input dimensions and the other is how can I use this model
1. Will I be predicting the next batch given the current batch?
2. Will I be predicting the next time step (t+1) given t?
3. Is it necessary to pass the whole batch size when we wanted to make a prediction bcuz when we were building the model we used dim(1000 * timesteps * 200)

My Goal: given the current state of the process(3 or 4 or 'n' time steps) I want my model to be predicting n+1 or n+10 time steps also give the controlled variables from other side of 'Y'.

Jason Brownlee March 11, 2020 at 5:23 am #

REPLY

This will help you to better understand the input shape for LSTMs:
https://machinelearningmastery.com/faq/single-faq/what-is-the-difference-between-samples-timesteps-and-features-for-lstm-input

saipavankumar March 11, 2020 at 9:45 pm #

REPLY

Thanks a lot, I got 1 more question if you have time to answer and I am sorry to bother you with too many questions.

If Xa and Xb are inputs to 2 different networks (Xa-> LSTM) and (Xb-> Dense) and they want to share a common Dense layer in the future to give an output ' y '.

Xa and Xb share the same time index so if I reshape Xa to be 3D(samples, 10, 5) then how should I reshape Xb?

REPLY

Jason Brownlee March 12, 2020 at 8:47 am #

You would concat the output of each sub-model. You might need to flatten LSTM output prior to the concat.

REPLY

Iamiaa ali April 8, 2020 at 11:42 am #

please dr jason
i can't understand what does None mean and how it is processed from flatten layer to the Dense layer
as i read it is (no.of batches proceesed in parallel, no. of features)
also i want to know how to code Python to visualize this flatten output (batches included and the features)

thank you very much

REPLY

Jason Brownlee April 8, 2020 at 1:20 pm #

None means not specified, so it will process whatever number is provided.

REPLY

Zaidur Rahman May 5, 2020 at 8:41 am #

Thank you very much! The whole reading was very helpful!! Especially, the last note on Functional API Python Syntax. No one would care to add this to their tutorial of Deep Learning!

REPLY

Jason Brownlee May 5, 2020 at 1:36 pm #

Thanks, I'm happy to hear it was helpful!

REPLY

Ricardo May 5, 2020 at 2:33 pm #

Sir your tutorials are great. Thank you so much.

REPLY

Jason Brownlee May 6, 2020 at 6:20 am #

You're welcome!

Mostafa May 12, 2020 at 6:09 pm #

REPLY

Thanks for your informative tutorial.

I have 2 directories containing RGB images. I am going to use 2 data-generators to read them and then feed these 2 generators into a CNN.

Question1:
How should I combine these 2 data-generators and more importantly how to use function "fit_generator" for multiple generators so that network can train on whole samples (both 2 directories)?

Question2:
If I merge these 2 datasets manually (copy all files from one directory to another) to form one single dataset and then use 1 single data-generator to read them and then feed it into CNN.
In comparison to method 1 (mentioned in queston 1), does it have effect on output? It means does it increase or decrease accuracy, loss or other metrics?

Thanks.

Jason Brownlee May 13, 2020 at 6:31 am #

REPLY

A separate generator is used for each directory.

Probably. Try it and see.

nkm May 29, 2020 at 4:13 am #

REPLY

Hello Mr Jason,

I also have same queries. Readers will be grateful if you can kindly share any reference code.

Thanks and regards

Jason Brownlee May 29, 2020 at 6:35 am #

REPLY

Thanks for the suggestion.

James Kajdasz June 1, 2020 at 3:12 pm #

REPLY

Best explanation of the functional API I've found so far.

REPLY

Jason Brownlee June 2, 2020 at 6:09 am #

Thanks James!

REPLY

Negar June 4, 2020 at 8:34 pm #

Hi Jason,
Thanks for great blog, the links you have provided for further reading don't work. Could you please update them? Thanks again!

REPLY

Jason Brownlee June 5, 2020 at 8:10 am #

Thanks, which links?

REPLY

S.Gowri pooja June 9, 2020 at 11:40 am #

Hi , Jason. I have a doubt . why keras is called as keras api?.It is a library and how it will become an api?

REPLY

Jason Brownlee June 9, 2020 at 1:21 pm #

A code library is a collection of reusable code:
https://en.wikipedia.org/wiki/Library_(computing)

The API is the standard interface for using a library (or any software):
https://en.wikipedia.org/wiki/Application_programming_interface

REPLY

S.Gowri pooja June 9, 2020 at 7:57 pm #

Thank you jason , Then all the libraries in python are called api?

REPLY

Jason Brownlee June 10, 2020 at 6:12 am #

They are libraries, each offers an API.

S.Gowri pooja June 10, 2020 at 2:32 pm #

Thank you jason

Jason Brownlee June 11, 2020 at 5:49 am #

You're welcome.

hoan June 16, 2020 at 9:44 pm #

REPLY

This is amazing. Thank you so much for this article.

Jason Brownlee June 17, 2020 at 6:24 am #

REPLY

You're welcome!

Sina July 4, 2020 at 7:44 am #

REPLY

Thank you, Jason,

Such a useful post. For the case of Multiple Inputs and Multiple Outputs, it seems that number of examples/samples in each input should be the same for training data. I have a single mode, with two different sets of inputs, which gives two different sets of outputs. A different loss is applied to each (one loss for each, 2 in total).

However, I get this error: "All input arrays (x) should have the same number of samples. "

It seems that it is a common issue and no one has a solution for that. Do you have any thoughts?

==========

#number of examples are different to input/output 1 and 2
model = Model(inputs= [input1, input2], outputs=[outputs1, outputs2])

model.compile(loss=[loss1,loss2], optimizer=opt, metrics=['accuracy'])

model.fit({‘inputs1’:input_data_array1, ‘inputs2′:input_data_array2},
{outputs1′:y_arrray1,outputs2’:y_arrray2,
, epochs=100)

Jason Brownlee July 5, 2020 at 6:47 am #

REPLY

Yes, the input of input samples must match the number of output samples, e.g. for both targets.

David July 21, 2020 at 7:46 am #

REPLY

I will nominate Jason brownlee for australian of the year!

Jason Brownlee July 21, 2020 at 1:43 pm #

REPLY

Thanks, you are very kind!

Febin August 4, 2020 at 10:55 pm #

REPLY

Hello Jaison,

Thanks for the blog. I am new to machine learning and I have the following question.

I created multiple inputs(CNN, LSTM) and a single output model. Since the model is taking image input and text input together, how can I add image augmentation?

model.fit([image_train,text_train], label_train, batch_size=BATCH_SIZE,
epochs=40, verbose=1, callbacks=callbacks, validation_data=([image_test,text_test], label_test))

Jason Brownlee August 5, 2020 at 6:14 am #

REPLY

You could create augmented images with copies of the text – unaugmented. You might need a custom data generator for this purpose.

Jim August 13, 2020 at 7:41 pm #

REPLY

Hi Jason,

Thanks for the blog.

Since I'm a Python beginner, this is probably a question more related to Python syntax rather than Keras.

For Keras Functional API example:

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

The last syntax simillar to type casting in Java, what's it called and doing in Python ?

Jason Brownlee August 14, 2020 at 6:02 am #

REPLY

No, not casting.

It is calling a function on the object returned from the constructor which just so happens to be a function to connect the object's input to the output of another layer passed as an argument.

Terrible to read – I know, but it's easy to write.

Jim August 14, 2020 at 3:03 pm #

REPLY

So according to your explanation, can I write this way?

```
hidden = visible(Dense(2))
```

Jason Brownlee August 15, 2020 at 6:15 am #

REPLY

No. It would have a different effect, e.g. visible would take the output of Dense(2) as input and the ref to Dense(2) is now not available.

Xu Zhang September 3, 2020 at 5:22 pm #

REPLY

Thank you so much for your great post.

If I have two inputs but with different number of samples, is it possible to use multi-input, multi-outputs API to build and train a model?

My problem is if I give you an image, first, I want to know whether a person is in the image. If yes, I want to know how old he/she is. So I have to prepare two datesets. For example. one dataset includes 10,000 images, in which 5,000 have a person in it and 5,000 don't have a person in it. Another dataset has 1,000 images with a person in it and label them with age. These two datasets have different sample numbers, some images maybe appear in both datasets.

For this problem, it is a multi-input, multi-output problem, but two inputs have different sample numbers, Can I use the Keras' API to build a model? If not, any other methods would you like to suggest? Many thanks

Jason Brownlee September 4, 2020 at 6:21 am #

REPLY

You're welcome.

No, I believe the number of samples must be the same for each input.

Xu Zhang September 4, 2020 at 8:06 am #

REPLY

Thank you for your replay. If the number of samples are the same for each input, when I prepare these two inputs, do I need to pair them? I mean the features of dataset1 and dataset2 have to represent the same sample. Like I gave the example, Number 1 image in the dataset1 have to Number 1 image in the dataset2, and so on.

Jason Brownlee September 4, 2020 at 1:34 pm #

REPLY

Yes, inputs need to be paired.

Xu Zhang September 5, 2020 at 7:56 am #

Thank you very much!

Jason Brownlee September 5, 2020 at 8:07 am #

You're welcome.

Sai September 28, 2020 at 9:41 pm #

REPLY

Hi, nice post! Gives a quick and clear introduction to the functional API.

Jason Brownlee September 29, 2020 at 5:37 am #

REPLY

Thanks!

saeed October 21, 2020 at 10:30 pm #

REPLY

Hi Jason,

Thanks for the blog.
I,m splitt the image into blocks. What you'r suggestions to extract features to discriminate between stego and cover image.

Jason Brownlee October 22, 2020 at 6:44 am #

REPLY

You're welcome.

What is "stego and cover image"?

Helene October 23, 2020 at 1:14 am #

REPLY

Hi Jason,

Thanks a lot for this great tutorial!

I am trying to implement a model with multiple Input layers using your example "Multiple Input Model". I experience some issues when calling the fit function regarding the shape of the input data. Could you please provide an example of how to call the fit function on your example?

Jason Brownlee October 23, 2020 at 6:13 am #

REPLY

Call fit() as per normal and provide a list of inputs with one element in the list for each input in the model.

I give many examples of multi-input models on the blog. Perhaps try searching.

Helene October 23, 2020 at 4:37 pm #

REPLY

Thanks!

Jason Brownlee October 24, 2020 at 6:55 am #

REPLY

You're welcome.

Jyothsna November 5, 2020 at 2:48 pm #

REPLY

Thank you so much for your clear explanation. I like to know how to add BatchNormalization and Relu in that. Actually, I am trying to write code for dncnn using functional API.

code for sequential API:
model=Sequential()
model.add(Conv2D(64,(3,3),padding="same",input_shape=(None,None,1)))
model.add(Activation('relu'))
for layers in range(2,16+1):
model.add(Conv2D(64,(3,3),padding="same"))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Conv2D(1,(3,3),padding="same"))

I like to know how to implement using Functional API

Jason Brownlee November 6, 2020 at 5:52 am #

REPLY

You can add a batch norm layer via the functional API just like any other layer, such as a dense. No special syntax required.

Sarasa Jyothsna Kamireddi March 16, 2021 at 4:18 pm #

REPLY

It is very nice explanation sir. I would like to know how to achieve this:
1. Create a model and train it. (I understood this)
2. Use this model with this trained weights as a layer in other model.
It will be very helpful if you can provide information regarding this.

Jason Brownlee March 17, 2021 at 6:00 am #

REPLY

Thanks.

Yes, this is called transfer learning, see this:
https://machinelearningmastery.com/?s=transfer+learning&post_type=post&submit=Search

Grace March 30, 2021 at 9:51 pm #

REPLY

Hello, Jason.
Could you explain the theory to me a little? How the input data is converted to the tensor of the first Lstm layer. For example, if the input is (None, 1,10), and the first layer of the lstm has 64 neurons, that is (None, 64). Is there some kind of matrix multiplication going on?

Jason Brownlee March 31, 2021 at 6:03 am #

REPLY

No theory is involved. This will help you prepare data for an LSTM:
https://machinelearningmastery.com/faq/single-faq/what-is-the-difference-between-samples-timesteps-and-features-for-lstm-input

Shashank Gurnalkar May 27, 2021 at 4:13 pm #

REPLY

Thanks for this great tutorial. It is really helpful.

Jason Brownlee May 28, 2021 at 6:45 am #

REPLY

You're welcome.

Maqsood June 3, 2021 at 12:26 am #

REPLY

Hello, Jason.

Thanks a lot for everything. you are just genius.

I have question about multi input layers (CNN). I got everything but i don't know how to train this model if i have two images as input? can you please guide.

Usually we call fit model like this with single x and y but i am confused with two inputs.
eg:
model.fit(trainImages, trainPrices, validation_data=(testImages, testPrices),epochs=300, batch_size=batchSize)

I really appreciate your help, if you do for me. thanks

Jason Brownlee June 3, 2021 at 5:37 am #

REPLY

You provide one input dataset for each input to the model, provided as a list when calling fit, e.g. model.fit([trainX1, trainX2], …)

I have many examples on the blog.

REPLY

Maqsood June 4, 2021 at 7:37 pm #

Thank you Jason Brownlee,

I got it and worked. god bless you and have a more success in your life.

REPLY

Jason Brownlee June 5, 2021 at 5:28 am #

Well done!

REPLY

Atheer June 21, 2021 at 6:11 am #

hello Jason
Thank you so much for your hard work
I've been reading your tutorial these past months and they were really helpful
can I ask if you have or read about an example for a DL model that have 2 input like text and binary features .. and thank you

REPLY

Jason Brownlee June 22, 2021 at 6:26 am #

Yes, you can use a multi-input model with one input for each data type/stream. I have a few examples, e.g. the photo captioning tutorial might give you ideas.

REPLY

Lidya June 25, 2021 at 3:00 pm #

Hello, may I ask about how to do autoencoder feature selection and put it into deep neural network model?
Thankyou

REPLY

Jason Brownlee June 26, 2021 at 4:53 am #

Perhaps this will help:
https://machinelearningmastery.com/autoencoder-for-classification/

Pankaj July 15, 2021 at 12:51 am #

REPLY

Great Post @Jason Brownlee

Any example on how to use Deep Learning for clustering?

Jason Brownlee July 15, 2021 at 5:32 am #

REPLY

Not yet, perhaps in the future.

Reza July 16, 2021 at 12:26 am #

REPLY

Thank you, it was the best explanations that I have read 🙂

Jason Brownlee July 16, 2021 at 5:26 am #

REPLY

You're welcome.

Akilu Muhammad August 4, 2021 at 2:09 am #

REPLY

Interesting complicated stuff put out concisely. I wished I had come across this post much earlier. Thank you so much Jason.

Jason Brownlee August 4, 2021 at 5:15 am #

REPLY

You're welcome!

nkm August 16, 2021 at 3:25 am #

REPLY

Thanks, Dr. Jason for this great blog.

Further, I would like to ask that how to CONCATENATE CNN (using images) and LSTM (using time-series) models, where CNN is trained using Keras flow_from_directory and generators.

Any example or guidance will guide in practical implementation.

Thanks and Regards

Adrian Tam August 17, 2021 at 7:33 am #

REPLY

There is a Keras CNN post that might give you some sample code to start with: https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

But I am not sure what you want to concatenate with CNN and LSTM? Are you assuming the CNN output as LSTM input? In that case you just drop your last layer at CNN and take that as input to LSTM model.

nkm August 17, 2021 at 4:52 pm #

REPLY

Thanks for valuable suggections.

I am willing to extract features from time-series data using LSTM and features from images using CNN (using Keras generators). Further, I need to concatenate both features before feeding it to the Fully connected (Dense) Layers followed by softmax layer. I am NOT feeding CNN output as LSTM input.

Regards

Adrian Tam August 18, 2021 at 2:27 am #

REPLY

That works. Keras has a concatenate function that allows you to combine them. An example is in the code on this post: https://machinelearningmastery.com/how-to-implement-pix2pix-gan-models-from-scratch-with-keras/

liedji August 19, 2021 at 2:19 am #

REPLY

Thank you. Very easy to follow post. Liked it!

Adrian Tam August 19, 2021 at 4:10 am #

REPLY

Thank you. Hope you enjoyed.

Alexandrea Marks October 13, 2021 at 1:47 am #

REPLY

Interesting complicated stuff put out concisely. I wished I had come across this post much earlier. Thank you so much Jason.

REPLY

Adrian Tam October 13, 2021 at 7:34 am #

Thanks. Glad you like it.

REPLY

Vaishnavi March 26, 2022 at 1:23 am #

Thank you for this blog.
I would like to ask if I want to use functions on the last hidden layer neurons which includes trigonometric functions. So, how do I separate the inputs in tensor and apply functions to them and give outputs to the output layer?

REPLY

Vaishnavi March 26, 2022 at 7:53 pm #

I defined a custom layer to which the input type is tensor. Then I have used a tf.split() function which I thought that will separate the inputs from previous hidden layer into 5 inputs. Then, I used 6 functions on the 5 inputs and saved them in a list. Then, I converted the list into tensor using tf.convert_to_tensor() function. In the end, the custom layer will return this tensor. I am getting an error. Is it really possible to split a tensor into 5 variables? Also, will the batch size affect the split?

REPLY

Jaydeep June 12, 2022 at 2:22 pm #

I am having dataset with columns

year, w_1_1, w_1_2,…., w_1_52, w_2_1, w_2_2,…,w_2_52, w_3_1,…, w_3_52, w_4_1,.., w_4_52, w_5_1,…, w_5_52, w_6_1,…, w_6_52, yield

where yield is the single value that I want my model to predict.

I want to create 6 1D CNN model to train on different parts of data, first on w_1_1 to w_1_52, second on w_2_1 to w_2_52 and third on w_3_1 to w_3_52 so on till sixth model train on w_6_1 to w_6_52 and finally to concatenate their output.

But I dont understand how to I resize different parts of data for 1D convolutions

REPLY

James Carmichael June 13, 2022 at 11:32 am #

Hi Jaydeep…The same concept is applicable to CNNs as presented in the following resource:

https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/

Zack August 4, 2022 at 6:03 pm #

REPLY

Hello James, thanks for the tutorial, supper helpful.

Do you have any tutorial or complete implementation of using the Keras functional API? For instance using the keras functional API to compile and fit a multiple input multiple output model using ImageDataGenarator function.

Thanks.

James Carmichael August 5, 2022 at 9:32 am #

REPLY

Hi Jack…You are very welcome! For your question, I would recommend the following resources as a starting point:

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/

efuye October 9, 2022 at 6:06 am #

REPLY

I'm trying to concatenate multiclass two features (hog and cnn) CNN using python, could you help me?

James Carmichael October 10, 2022 at 9:53 am #

REPLY

Hi efuye…Please elaborate on what you have already done and perhaps what is not working so that we may better assist you.

Henriko August 23, 2023 at 8:29 am #

REPLY

Hey Jason, thanks for your great blog post!

How would you specify the layer_input shape in this case: I have a three-dimensional balanced monthly panel data set with 2500 individual cross-section observations per month, 134 features for each cross sections and a time period from 2006-01 to 2019-12 (168 months), so 2500 x 134 x 168.
The data set thus has 2500*168 = 420000 rows and 134 columns.
The batch size (time dimension) is left out as you say, so I specified 'shape = c(2500, 134)' in the input layer.

However I receive the error ‘ValueError: Input 0 of layer “model_6” is incompatible with the layer: expected shape=(None, 2537, 132), found shape=(None, 132)’

Can you tell me how to specify the shape of the input layer correctly in such a three dimensional panel data case?

James Carmichael August 23, 2023 at 9:46 am #

REPLY

Hi Henriko…You are very welcome! The following resources may be of interest to you:

https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/

https://stackoverflow.com/questions/52562133/keras-how-to-shape-inputs-for-cnn-and-lstm-layers

Rahul Gulia January 26, 2024 at 4:22 am #

REPLY

I am trying to use the “Multiple Input, Multiple Output Models” for my dataset.

```
# merge input models
merge = concatenate([flat1, flat2])
```

I wanted to know if the 2 input datasets should be concatenated for the whole dataset, or should it be for training and testing data separately.

Looking forward to any kind of suggestion on this. Thank you.

Rahul Gulia January 26, 2024 at 4:27 am #

REPLY

I am trying to use the ” 5. Multiple Input and Output Models ” for my dataset.

In this line of code,

```
# merge input models
merge = concatenate([flat1, flat2])
```

Do I have to concatenate the train and test dataset separately, or should I do it on the whole dataset at once?

James Carmichael January 26, 2024 at 11:08 am #

REPLY

Hi Rahul…The following resource provides best practices for train, test split.

https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Rahul Gulia January 26, 2024 at 4:28 am #

REPLY

Looking forward to any kind of suggestion on this.

Leave a Reply

Name (required)

Email (will not be published) (required)

SUBMIT COMMENT

Welcome!
I'm *Jason Brownlee* PhD
and I **help developers** get results with **machine learning**.
Read more

Never miss a tutorial:

Picked for you:


Your First Deep Learning Project in Python with Keras Step-by-Step


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


Regression Tutorial with the Keras Deep Learning Library in Python


Multi-Class Classification Tutorial with the Keras Deep Learning Library


How to Save and Load Your Keras Deep Learning Model