An Introduction to Deep Learning With Python

[7.1] Introduction to the functional API

Prof. Yuzo Iano

pgs: 236 - 248

In [1]:
```
from keras import Input, layers

input_tensor = Input(shape=(32,))
dense = layers.Dense(32, activation='relu')
output_tensor = dense(input_tensor)
```

```
Using TensorFlow backend.
```

```
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\python\fra
mework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and will
be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [2]:
```
from keras.models import Sequential, Model
from keras.layers import Dense
from keras import Input

seq_model = Sequential()
seq_model.add(Dense(32, activation='relu', input_shape=(64,)))
seq_model.add(Dense(32, activation='relu'))
seq_model.add(Dense(10, activation='softmax'))

input_tensor = Input(shape=(64,))
x = Dense(32, activation='relu')(input_tensor)
x = Dense(32, activation='relu')(x)
output_tensor = Dense(10, activation='softmax')(x)

model = Model(input_tensor, output_tensor)
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_6 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_7 (Dense)              (None, 10)                330
=================================================================
Total params: 3,466
Trainable params: 3,466
Non-trainable params: 0
_________________________________________________________________
```

In [3]:
```
unrelated_input = Input(shape=(32,))
```

In [4]:
```
bad_model = model = Model(unrelated_input, output_tensor)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-e887ec670abe> in <module>
----> 1 bad_model = model = Model(unrelated_input, output_tensor)

~\Python\envs\DAVID\lib\site-packages\keras\legacy\interfaces.py in wrapper(*args, **kwargs)
     89                 warnings.warn('Update your `' + object_name + '` call to the ' +
     90                               'Keras 2 API: ' + signature, stacklevel=2)
---> 91             return func(*args, **kwargs)
     92         wrapper._original_function = func
     93         return wrapper

~\Python\envs\DAVID\lib\site-packages\keras\engine\network.py in __init__(self, *args, **kwargs)
     91                 'inputs' in kwargs and 'outputs' in kwargs):
     92             # Graph network
---> 93             self._init_graph_network(*args, **kwargs)
     94         else:
     95             # Subclassed network

~\Python\envs\DAVID\lib\site-packages\keras\engine\network.py in _init_graph_network(self, inputs, output
s, name)
    229         # Keep track of the network's nodes and layers.
    230         nodes, nodes_by_depth, layers, layers_by_depth = _map_graph_network(
--> 231             self.inputs, self.outputs)
    232         self._network_nodes = nodes
    233         self._nodes_by_depth = nodes_by_depth

~\Python\envs\DAVID\lib\site-packages\keras\engine\network.py in _map_graph_network(inputs, outputs)
   1441                                         'The following previous layers '
   1442                                         'were accessed without issue: ' +
-> 1443                                         str(layers_with_complete_input))
   1444                 for x in node.output_tensors:
   1445                     computable_tensors.append(x)

ValueError: Graph disconnected: cannot obtain value for tensor Tensor("input_2:0", shape=(?, 64), dtype=fl
oat32) at layer "input_2". The following previous layers were accessed without issue: []
```

In [5]:
```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')

import numpy as np
x_train = np.random.random((1000, 64))
y_train = np.random.random((1000, 10))

model.fit(x_train, y_train, epochs=10, batch_size=128)

score = model.evaluate(x_train, y_train)
print('Score: ', score)
```

```
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\python\ops
\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a f
uture version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/10
1000/1000 [==============================] - 0s 471us/step - loss: 11.7223
Epoch 2/10
1000/1000 [==============================] - 0s 12us/step - loss: 11.5871
Epoch 3/10
1000/1000 [==============================] - 0s 19us/step - loss: 11.5710
Epoch 4/10
1000/1000 [==============================] - 0s 12us/step - loss: 11.5614
Epoch 5/10
1000/1000 [==============================] - 0s 16us/step - loss: 11.5546
Epoch 6/10
1000/1000 [==============================] - 0s 14us/step - loss: 11.5497
Epoch 7/10
1000/1000 [==============================] - 0s 13us/step - loss: 11.5463
Epoch 8/10
1000/1000 [==============================] - 0s 17us/step - loss: 11.5437
Epoch 9/10
1000/1000 [==============================] - 0s 22us/step - loss: 11.5412
Epoch 10/10
1000/1000 [==============================] - 0s 16us/step - loss: 11.5389
1000/1000 [==============================] - 0s 97us/step
Score:  11.53592537689209
```

Multi-input models

In [6]:
```
from keras.models import Model
from keras import layers
from keras import Input

text_vocabulary_size = 10000
question_vocabulary_size = 10000
answer_vocabulary_size = 500

text_input = Input(shape=(None,), dtype='int32', name='text')
embedded_text = layers.Embedding(text_vocabulary_size, 64)(text_input) #change 64 to 2nd position
encoded_text = layers.LSTM(32)(embedded_text)
question_input = Input(shape=(None,),
                       dtype='int32',
                       name='question')

embedded_question = layers.Embedding(question_vocabulary_size, 32)(question_input) #change 64 to 2nd posit
ion
encoded_question = layers.LSTM(16)(embedded_question)

concatenated = layers.concatenate([encoded_text, encoded_question], axis=-1)
answer = layers.Dense(answer_vocabulary_size,activation='softmax')(concatenated)

model = Model([text_input, question_input], answer)

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

In [7]:
```
import numpy as np

num_samples = 1000
max_length = 100

text = np.random.randint(1, text_vocabulary_size, size=(num_samples, max_length))
question = np.random.randint(1, question_vocabulary_size, size=(num_samples, max_length))
answers = np.random.randint(0, 1, size=(num_samples, answer_vocabulary_size))

model.fit([text, question], answers, epochs=10, batch_size=128)

model.fit({'text': text, 'question': question}, answers, epochs=10, batch_size=128)
```

```
Epoch 1/10
1000/1000 [==============================] - 4s 4ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 2/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 3/10
1000/1000 [==============================] - 1s 899us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 4/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 5/10
1000/1000 [==============================] - 1s 973us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 6/10
1000/1000 [==============================] - 1s 959us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 7/10
1000/1000 [==============================] - 1s 958us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 8/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 9/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 10/10
1000/1000 [==============================] - 1s 900us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 1/10
1000/1000 [==============================] - 1s 946us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 2/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 3/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 4/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 5/10
1000/1000 [==============================] - 1s 992us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 6/10
1000/1000 [==============================] - 1s 963us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 7/10
1000/1000 [==============================] - 1s 990us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 8/10
1000/1000 [==============================] - 1s 987us/step - loss: 0.0000e+00 - acc: 1.0000e-03
Epoch 9/10
1000/1000 [==============================] - 1s 1ms/step - loss: 0.0000e+00 - acc: 1.0000e-03s - loss: 0.0
000e+00 - acc: 0.001
Epoch 10/10
1000/1000 [==============================] - 1s 977us/step - loss: 0.0000e+00 - acc: 1.0000e-03
```

Out[7]:
```
<keras.callbacks.History at 0x1ce35c05208>
```

Multi-output models

Functional API Implementation of a three-output model

In [8]:
```
from keras import layers
from keras import Input
from keras.models import Model

vocabulary_size = 50000
num_income_groups = 10

posts_input = Input(shape=(None,), dtype='int32', name='posts')
embedded_posts = layers.Embedding(256, vocabulary_size)(posts_input)
x = layers.Conv1D(128, 5, activation='relu')(embedded_posts)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.MaxPooling1D(5)(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.Conv1D(256, 5, activation='relu')(x)
x = layers.GlobalMaxPooling1D()(x)
x = layers.Dense(128, activation='relu')(x)
age_prediction = layers.Dense(1, name='age')(x)
income_prediction = layers.Dense(num_income_groups, activation='softmax', name='income')(x)
gender_prediction = layers.Dense(1, activation='sigmoid', name='gender')(x)

model = Model(posts_input, [age_prediction, income_prediction, gender_prediction])
```

Compilation options of a multi-output model: multiple losses

In [9]:
```
model.compile(optimizer='rmsprop',
              loss=['mse', 'categorical_crossentropy', 'binary_crossentropy'])

model.compile(optimizer='rmsprop',
              loss={'age': 'mse','income': 'categorical_crossentropy','gender': 'binary_crossentropy'})
```

Compilation options of a multi-output model: loss weighting

In [10]:
```
model.compile(optimizer='rmsprop',
              loss=['mse', 'categorical_crossentropy', 'binary_crossentropy'],
              loss_weights=[0.25, 1., 10.])

model.compile(optimizer='rmsprop',
              loss={'age': 'mse','income': 'categorical_crossentropy','gender': 'binary_crossentropy'},
              loss_weights={'age': 0.25,'income': 1.,'gender': 10.})
```

Feeding data to a multi-output model

In [ ]:
```
model.fit(posts_input,
          [age_targets, income_targets, gender_targets],
          epochs=10,
          batch_size=64)

model.fit(posts_input,
          {'age': age_targets, 'income': income_targets, 'gender': gender_targets},
          epochs=10,
          batch_size=64)
```

Directed acyclic graphs of layers

In [ ]:
```
from keras import layers
branch_a = layers.Conv2D(128, 1,
                         activation='relu', strides=2)(x)
branch_b = layers.Conv2D(128, 1, activation='relu')(x)
branch_b = layers.Conv2D(128, 3, activation='relu', strides=2)(branch_b)

branch_c = layers.AveragePooling2D(3, strides=2)(x)
branch_c = layers.Conv2D(128, 3, activation='relu')(branch_c)

branch_d = layers.Conv2D(128, 1, activation='relu')(x)
branch_d = layers.Conv2D(128, 3, activation='relu')(branch_d)
branch_d = layers.Conv2D(128, 3, activation='relu', strides=2)(branch_d)
output = layers.concatenate([branch_a, branch_b, branch_c, branch_d], axis=-1)
```

Here's how to implement a residual connection in Keras when the feature-map sizes are the same, using identity residual connections. This example assumes the existence of a 4D input tensor x:

In [ ]:
```
from keras import layers

x = ...
y = layers.Conv2D(128, 3, activation='relu', padding='same')(x)
y = layers.Conv2D(128, 3, activation='relu', padding='same')(y)
y = layers.Conv2D(128, 3, activation='relu', padding='same')(y)

y = layers.add([y, x])
```

And the following implements a residual connection when the feature-map sizes differ, using a linear residual connection (again, assuming the existence of a 4D input tensor x):

In [ ]:
```
from keras import layers

x = ...
y = layers.Conv2D(128, 3, activation='relu', padding='same')(x)
y = layers.Conv2D(128, 3, activation='relu', padding='same')(y)
y = layers.MaxPooling2D(2, strides=2)(y)

residual = layers.Conv2D(128, 1, strides=2, padding='same')(x)

y = layers.add([y, residual])
```

Layer weight sharing

Here's how to implement such a model using layer sharing (layer reuse) in the Keras functional API:

In [ ]:
```
from keras import layers
from keras import Input
from keras.models import Model

lstm = layers.LSTM(32)
left_input = Input(shape=(None, 128))
left_output = lstm(left_input)
right_input = Input(shape=(None, 128))
right_output = lstm(right_input)
merged = layers.concatenate([left_output, right_output], axis=-1)
predictions = layers.Dense(1, activation='sigmoid')(merged)
model = Model([left_input, right_input], predictions)

model.fit([left_data, right_data], targets)
```

Models as layers

In [ ]:
```
y = model(x)
y1, y2 = model([x1, x2])
```

In [12]:
```
from keras import layers
from keras import applications
from keras import Input

xception_base = applications.Xception(weights=None, include_top=False)
left_input = Input(shape=(250, 250, 3))
right_input = Input(shape=(250, 250, 3))
left_features = xception_base(left_input)
right_input = xception_base(right_input)
merged_features = layers.concatenate([left_features, right_input], axis=-1)
```

Pablo Minango

- pablodavid218@gmail.com