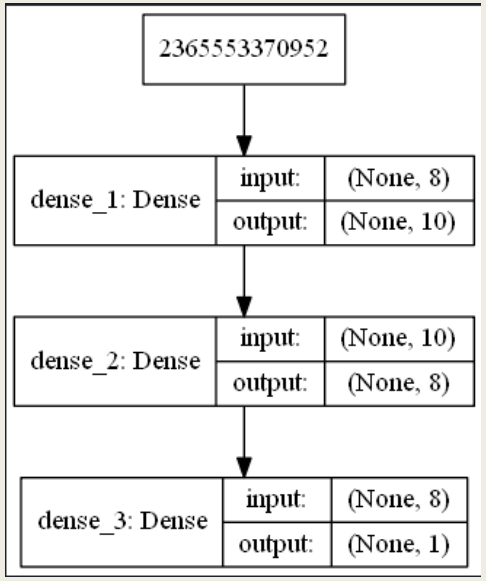
第十六章

深度學習模型存成圖片

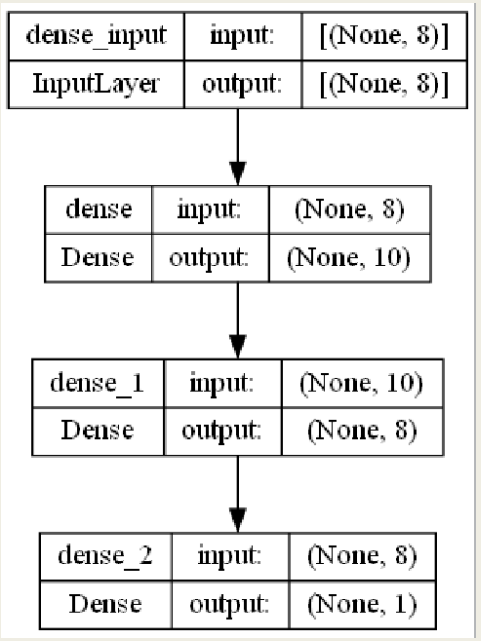
```python
from keras.models import Sequential
from keras.layers import Dense

# 定義模型
model = Sequential()
model.add(Dense(10, input_shape=(8,), activation="relu"))
model.add(Dense(8, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
model.summary()  # 顯示模型摘要資訊
# 繪出模型圖片
from keras.utils import plot_model

plot_model(model, to_file="Ch16_1.png", show_shapes=True)

```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                90

 dense_1 (Dense)             (None, 8)                 88

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 187
Trainable params: 187
Non-trainable params: 0
_________________________________________________________________
```

舊版(課本範例)



新版



顯示個神經層名稱及輸出

```python
# 定義模型
model = Sequential()
model.add(Conv2D(16, kernel_size=(5, 5), padding="same",
                 input_shape=(28, 28, 1), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, kernel_size=(5, 5), padding="same",
                 activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation="softmax"))
model.summary()    # 顯示模型摘要資訊
```

```
神經層數:  9
0 conv2d
1 max_pooling2d
2 conv2d_1
3 max_pooling2d_1
4 dropout
5 flatten
6 dense_3
7 dropout_1
8 dense_4
```

```python
# 顯示各神經層
print("神經層數: ", len(model.layers))
for i in range(len(model.layers)):
    print(i, model.layers[i].name)
print("每一層的輸入張量: ")
for i in range(len(model.layers)):
    print(i, model.layers[i].input)
print("每一層的輸出張量: ")
for i in range(len(model.layers)):
    print(i, model.layers[i].output)
```

```
每一層的輸入張量:
0 KerasTensor(type_spec=TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name='conv2d_input'), name='conv2d_input', description="created by layer 'conv2d_input'")
1 KerasTensor(type_spec=TensorSpec(shape=(None, 28, 28, 16), dtype=tf.float32, name=None), name='conv2d/Relu:0', description="created by layer 'conv2d'")
2 KerasTensor(type_spec=TensorSpec(shape=(None, 14, 14, 16), dtype=tf.float32, name=None), name='max_pooling2d/MaxPool:0', description="created by layer 'max_pooling2d'")
3 KerasTensor(type_spec=TensorSpec(shape=(None, 14, 14, 32), dtype=tf.float32, name=None), name='conv2d_1/Relu:0', description="created by layer 'conv2d_1'")
4 KerasTensor(type_spec=TensorSpec(shape=(None, 7, 7, 32), dtype=tf.float32, name=None), name='max_pooling2d_1/MaxPool:0', description="created by layer 'max_pooling2d_1'")
5 KerasTensor(type_spec=TensorSpec(shape=(None, 7, 7, 32), dtype=tf.float32, name=None), name='dropout/Identity:0', description="created by layer 'dropout'")
6 KerasTensor(type_spec=TensorSpec(shape=(None, 1568), dtype=tf.float32, name=None), name='flatten/Reshape:0', description="created by layer 'flatten'")
7 KerasTensor(type_spec=TensorSpec(shape=(None, 128), dtype=tf.float32, name=None), name='dense_3/Relu:0', description="created by layer 'dense_3'")
8 KerasTensor(type_spec=TensorSpec(shape=(None, 128), dtype=tf.float32, name=None), name='dropout_1/Identity:0', description="created by layer 'dropout_1'")
```

```
每一層的輸出張量:
0 KerasTensor(type_spec=TensorSpec(shape=(None, 28, 28, 16), dtype=tf.float32, name=None), name='conv2d/Relu:0', description="created by layer 'conv2d'")
1 KerasTensor(type_spec=TensorSpec(shape=(None, 14, 14, 16), dtype=tf.float32, name=None), name='max_pooling2d/MaxPool:0', description="created by layer 'max_pooling2d'")
2 KerasTensor(type_spec=TensorSpec(shape=(None, 14, 14, 32), dtype=tf.float32, name=None), name='conv2d_1/Relu:0', description="created by layer 'conv2d_1'")
3 KerasTensor(type_spec=TensorSpec(shape=(None, 7, 7, 32), dtype=tf.float32, name=None), name='max_pooling2d_1/MaxPool:0', description="created by layer 'max_pooling2d_1'")
4 KerasTensor(type_spec=TensorSpec(shape=(None, 7, 7, 32), dtype=tf.float32, name=None), name='dropout/Identity:0', description="created by layer 'dropout'")
5 KerasTensor(type_spec=TensorSpec(shape=(None, 1568), dtype=tf.float32, name=None), name='flatten/Reshape:0', description="created by layer 'flatten'")
6 KerasTensor(type_spec=TensorSpec(shape=(None, 128), dtype=tf.float32, name=None), name='dense_3/Relu:0', description="created by layer 'dense_3'")
7 KerasTensor(type_spec=TensorSpec(shape=(None, 128), dtype=tf.float32, name=None), name='dropout_1/Identity:0', description="created by layer 'dropout_1'")
8 KerasTensor(type_spec=TensorSpec(shape=(None, 10), dtype=tf.float32, name=None), name='dense_4/Softmax:0', description="created by layer 'dense_4'")
```

取得MLP個神經層的權重

```python
from keras.models import Sequential
from keras.models import load_model

# 建立Keras的Sequential模型
model = Sequential()
model = load_model("titanic.h5")
model.summary()   # 顯示模型摘要資訊
# 編譯模型
model.compile(loss="binary_crossentropy", optimizer="adam",
              metrics=["accuracy"])
# 顯示各層的權重形狀
for i in range(len(model.layers)):
    print(i, model.layers[i].name, ":")
    weights = model.layers[i].get_weights()
    for j in range(len(weights)):
        print("==>", j, weights[j].shape)
```

```
0 dense_97 :
==> 0 (9, 11)
==> 1 (11,)
1 dense_98 :
==> 0 (11, 11)
==> 1 (11,)
2 dense_99 :
==> 0 (11, 1)
==> 1 (1,)
```

取得CNN各神經層的權重

```python
from keras.models import Sequential
from keras.models import load_model

# 建立Keras的Sequential模型
model = Sequential()
model = load_model("mnist.h5")
model.summary()   # 顯示模型摘要資訊
# 編譯模型
model.compile(loss="categorical_crossentropy", optimizer="adam",
              metrics=["accuracy"])
# 顯示各層的權重形狀
for i in range(len(model.layers)):
    print(i, model.layers[i].name, ":")
    weights = model.layers[i].get_weights()
    for j in range(len(weights)):
        print("==>", j, weights[j].shape)
```

```
0 conv2d_1 :
==> 0 (5, 5, 1, 16)
==> 1 (16,)
1 max_pooling2d_1 :
2 conv2d_2 :
==> 0 (5, 5, 16, 32)
==> 1 (32,)
3 max_pooling2d_2 :
4 dropout_2 :
5 flatten_1 :
6 dense_51 :
==> 0 (1568, 128)
==> 1 (128,)
7 dropout_3 :
8 dense_52 :
==> 0 (128, 10)
==> 1 (10,)
```

取得RNN、LSTM、GRU神經層的權重

RNN

```python
from keras.models import Sequential
from keras.models import load_model

# 建立Keras的Sequential模型
model = Sequential()
model = load_model("imdb_rnn.h5")
model.summary()   # 顯示模型摘要資訊
# 編譯模型
model.compile(loss="binary_crossentropy", optimizer="rmsprop",
              metrics=["accuracy"])
# 顯示SimpleRNN層的權重形狀
print(2, model.layers[2].name, ":")
weights = model.layers[2].get_weights()
for i in range(len(weights)):
    print("==>", i, weights[i].shape)
```

```
2 simple_rnn_1 :
==> 0 (32, 32)
==> 1 (32, 32)
==> 2 (32,)
```

LSTM

```python
model = load_model("imdb_lstm.h5")
```

```
2 lstm_1 :
==> 0 (32, 128)
==> 1 (32, 128)
==> 2 (128,)
```

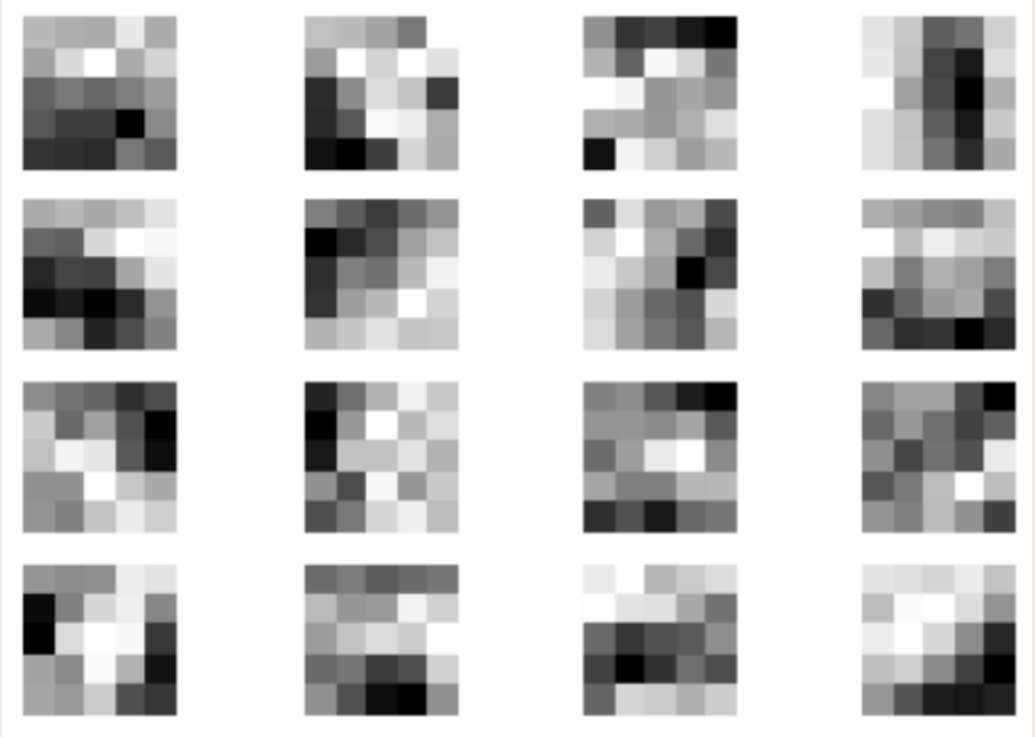
GRU

```python
model = load_model("imdb_gru.h5")
```

```
2 gru_1 :
==> 0 (32, 96)
==> 1 (32, 96)
==> 2 (96,)
```

第1層 Conv2D 層過濾器視覺化

```python
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import load_model
import matplotlib.pyplot as plt
# 指定亂數種子
seed = 7
np.random.seed(seed)
# 載入資料集
(X_train, Y_train), (_, _) = mnist.load_data()
# 將圖片轉換成 4D 張量
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype("float32")
# 因為是固定範圍，所以執行正規化，從 0-255 至 0-1
X_train = X_train / 255
# 建立Keras的Sequential模型
model = Sequential()
print("載入模型...")
model = load_model("mnist.h5")
# 編譯模型
model.compile(loss="categorical_crossentropy", optimizer="adam",
              metrics=["accuracy"])
# 顯示各神經層
print("神經層數: ", len(model.layers))
for i in range(len(model.layers)):
    print(i, model.layers[i].name)
# 第1個 Conv2D 的 filters 形狀
print(model.layers[0].get_weights()[0].shape)
# 繪出第1個 Conv2D 的 16 個 filters
weights = model.layers[0].get_weights()[0]
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.imshow(weights[:,:,0,i], cmap="gray",
               interpolation="none")
    plt.axis("off")
```



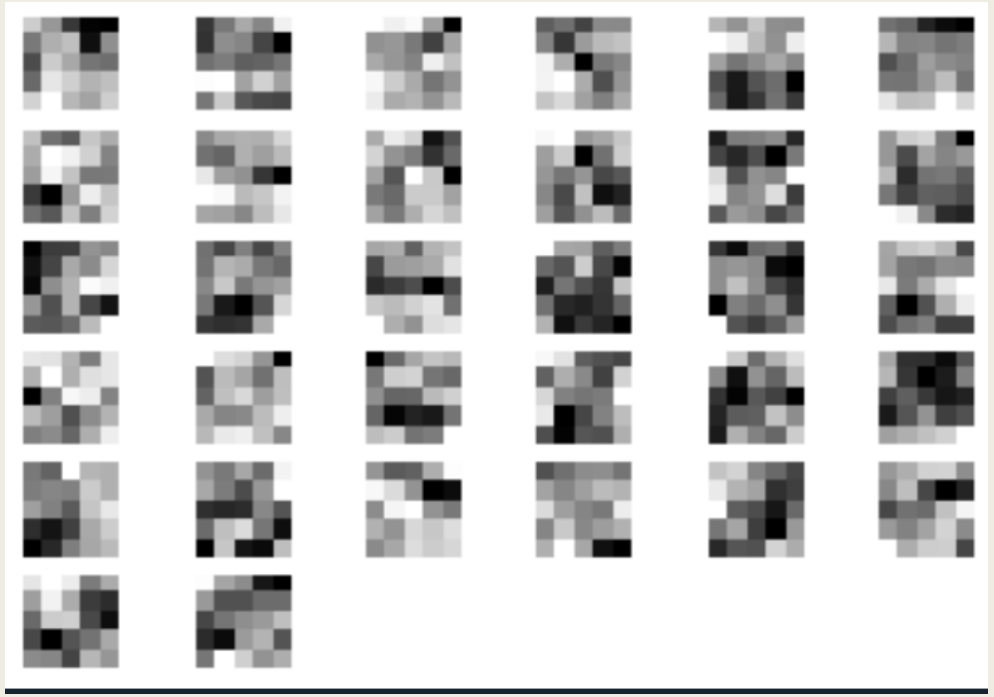
Get.weights取出索引值0的權重

```
(5, 5, 1, 16)
```

16個5X5的過濾器

第2層 Conv2D 層過濾器視覺化

```python
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import load_model
import matplotlib.pyplot as plt
# 指定亂數種子
seed = 7
np.random.seed(seed)
# 載入資料集
(X_train, Y_train), (_, _) = mnist.load_data()
# 將圖片轉換成 4D 張量
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype("float32")
# 因為是固定範圍，所以執行正規化，從 0-255 至 0-1
X_train = X_train / 255
# 建立Keras的Sequential模型
model = Sequential()
print("載入模型...")
model = load_model("mnist.h5")
# 編譯模型
model.compile(loss="categorical_crossentropy", optimizer="adam",
              metrics=["accuracy"])
# 顯示各神經層
print("神經層數: ", len(model.layers))
for i in range(len(model.layers)):
    print(i, model.layers[i].name)
# 第2個 Conv2D 的 filters 形狀
print(model.layers[2].get_weights()[0].shape)
# 繪出第2個 Conv2D 的 32 個 filters
weights = model.layers[2].get_weights()[0]
for i in range(32):
    plt.subplot(6,6,i+1)
    plt.imshow(weights[:,:,0,i], cmap="gray",
               interpolation="none")
    plt.axis("off")
```



`(5, 5, 16, 32)` 32個5X5的過濾器

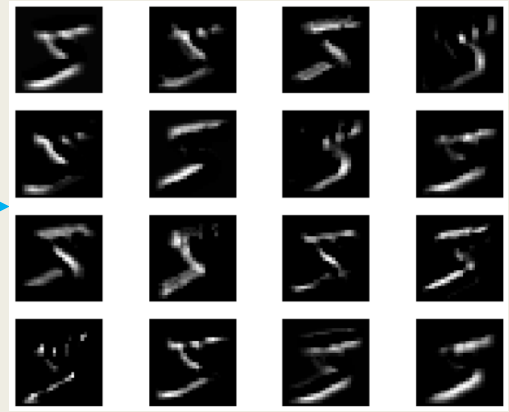
繪出第1層卷積層輸出的特徵圖

```python
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Conv2D
from keras.models import load_model
import matplotlib.pyplot as plt
# 指定亂數種子
seed = 7
np.random.seed(seed)
# 載入資料集
(X_train, Y_train), (_, _) = mnist.load_data()
# 將圖片轉換成 4D 張量
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype("float32")
# 因為是固定範圍，所以執行正規化，從 0-255 至 0-1
X_train = X_train / 255
# 建立Keras的Sequential模型
model = Sequential()
model = load_model("mnist.h5")
model.summary()   # 顯示模型摘要資訊
# 編譯模型
model.compile(loss="categorical_crossentropy", optimizer="adam",
              metrics=["accuracy"])
# 使用 Sequential 建立 Conv2D 層
model_test = Sequential()
model_test.add(Conv2D(16, kernel_size=(5, 5), padding="same",
                      input_shape=(28, 28, 1), activation="relu"))
for i in range(len(model_test.layers)):
    model_test.layers[i].set_weights(model.layers[i].get_weights())

output = model_test.predict(X_train[0].reshape(1,28,28,1))
# 繪出第1個 Conv2D 層的輸出
plt.figure(figsize=(10,8))
for i in range(0,16):
    plt.subplot(4,4,i+1)
    plt.imshow(output[0,:,:,i], cmap="gray")
    plt.axis("off")
```

建立一層conv2D模型

使用set.weights指定權重，CNN第一層Conv2D層的權重，因為model_test.layers的值為1

複製minst.h5的權重後，就可使用此模型預測訓練資料集的第一張圖



另一個方式是用Functional API建立model_test模型，Functional API重組CNN現有Conv2D神經層，就不用複製權重

```python
from keras.models import Model
layer_name = "conv2d_1"
model_test = Model(inputs=model.input,
                   outputs=model.get_layer(layer_name).output)
output = model_test.predict(X_train[0].reshape(1,28,28,1))
```

繪出第1層池化層輸出的特徵圖

第一層池化層經啟動函數輸出的特徵圖

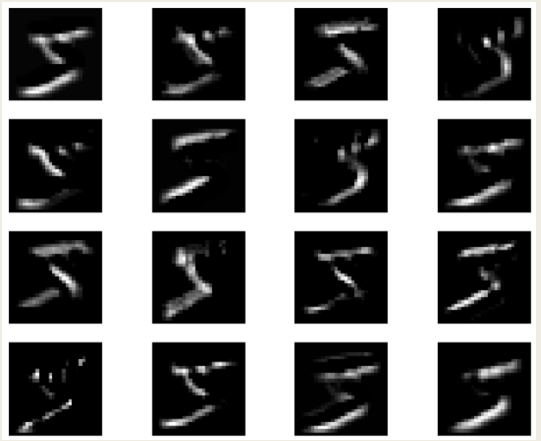
```python
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.models import load_model
import matplotlib.pyplot as plt
# 指定亂數種子
seed = 7
np.random.seed(seed)
# 載入資料集
(X_train, Y_train), (_, _) = mnist.load_data()
# 將圖片轉換成 4D 張量
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype("float32")
# 因為是固定範圍，所以執行正規化，從 0-255 至 0-1
X_train = X_train / 255
# 建立Keras的Sequential模型
model = Sequential()
model = load_model("mnist.h5")
model.summary()   # 顯示模型摘要資訊
# 編譯模型
model.compile(loss="categorical_crossentropy", optimizer="adam",
              metrics=["accuracy"])
# 使用 Sequential 建立 Conv2D 和 MaxPooling 層
model_test = Sequential()
model_test.add(Conv2D(16, kernel_size=(5, 5), padding="same",
                      input_shape=(28, 28, 1), activation="relu"))
model_test.add(MaxPooling2D(pool_size=(2, 2)))
for i in range(len(model_test.layers)):
    model_test.layers[i].set_weights(model.layers[i].get_weights())
output = model_test.predict(X_train[0].reshape(1,28,28,1))
# 繪出第1層 MaxPooling 層的輸出
plt.figure(figsize=(10,8))
for i in range(0,16):
    plt.subplot(4,4,i+1)
    plt.imshow(output[0,:,:,i], cmap="gray")
    plt.axis("off")
```

使用 Sequential 建立 Conv2D 和 MaxPooling 的 model_test模型

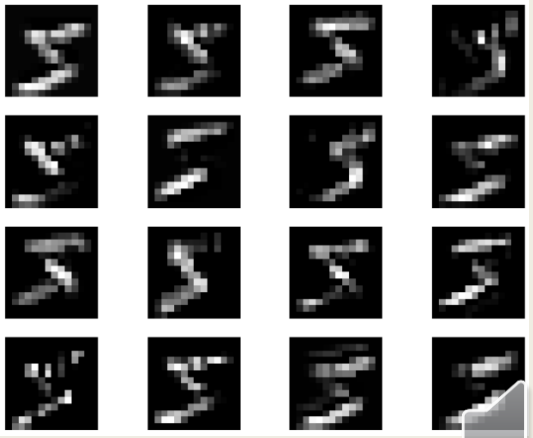
用for迴圈複製這兩層的權重後就可以產生第一個maxpooling2D輸出層的特徵圖

最後使用matplotlib繪出池化層輸出的特徵圖

卷積層特徵圖



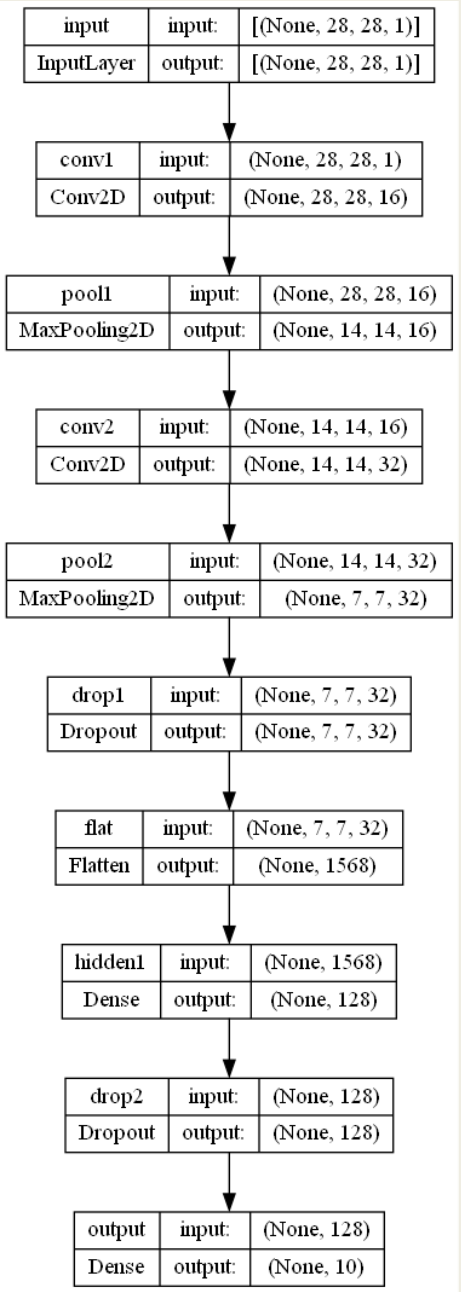
池化層特徵圖



Functional API建立CNN模型

```
23  mnist_input = Input(shape=(28, 28, 1),
24                      name="input")
25  conv1 = Conv2D(16, kernel_size=(5, 5), padding="same",
26                 activation="relu", name="conv1")(mnist_input)
27  pool1 = MaxPooling2D(pool_size=(2, 2),
28                       name="pool1")(conv1)
29  conv2 = Conv2D(32, kernel_size=(5, 5), padding="same",
30                 activation="relu", name="conv2")(pool1)
31  pool2 = MaxPooling2D(pool_size=(2, 2),
32                       name="pool2")(conv2)
33  drop1 = Dropout(0.5, name="drop1")(pool2)
34  flat = Flatten(name="flat")(drop1)
35  hidden1 = Dense(128, activation="relu", name="hidden1")(flat)
36  drop2 = Dropout(0.5, name="drop2")(hidden1)
37  output = Dense(10, activation="softmax",
38                 name="output")(drop2)
39  model = Model(inputs=mnist_input, outputs=output)
40  model.summary()    # 顯示模型摘要資訊
```

經過上述程式碼的各神經層使用name參數指定神經層名稱，建立右方的CNN模型圖

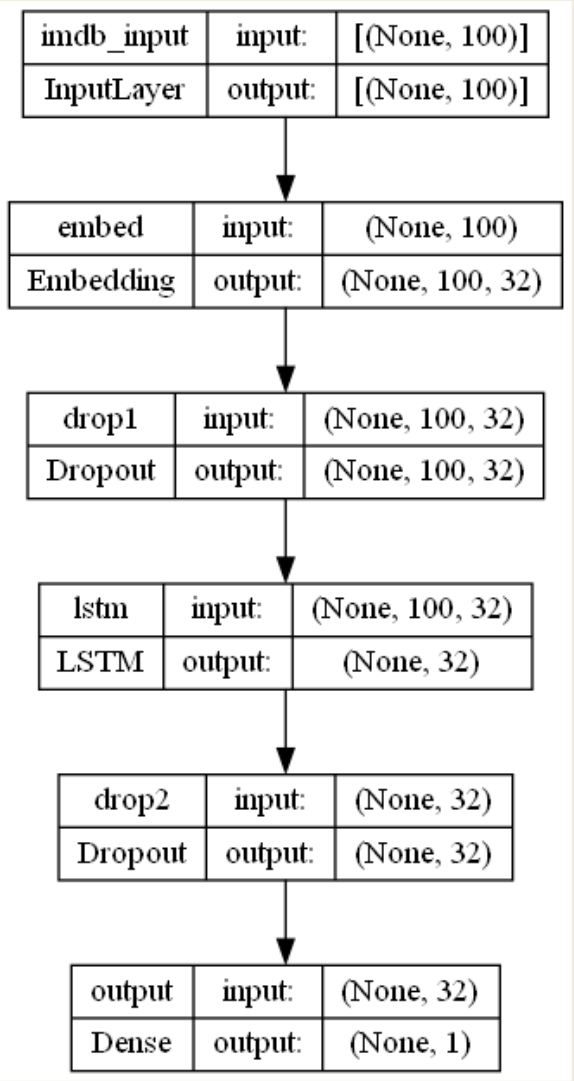


Functional API建立LSTM模型

使用範本會出錯，依照網路上的更改方式能成功

```
(base) C:\Users\marslin>pip install keras_preprocessing
```

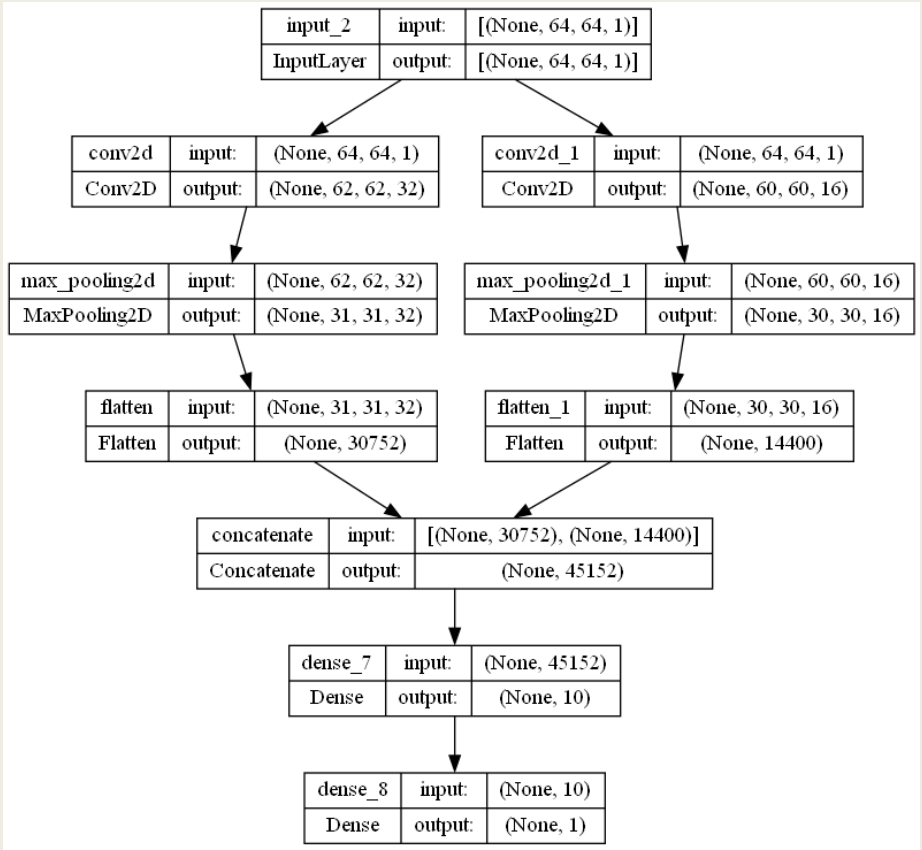
```python
import numpy as np
from keras.datasets import imdb
from keras_preprocessing.sequence import pad_sequences
from keras.models import Model
from keras.layers import Dense, Input, Dropout, Embedding, LSTM

seed = 10
np.random.seed(seed)  # 指定亂數種子
# 載入 IMDb 資料集
top_words = 1000
(X_train, Y_train), (X_test, Y_test) = imdb.load_data(
                                    num_words=top_words)
# 資料預處理
max_words = 100
X_train = pad_sequences(X_train, maxlen=max_words)
X_test = pad_sequences(X_test, maxlen=max_words)
# 定義模型
imdb_input = Input(shape=(100,), dtype="int32",
                   name="imdb_input")
embed = Embedding(top_words, 32, input_length=max_words,
                  name="embed")(imdb_input)
drop1 = Dropout(0.25, name="drop1")(embed)
lstm = LSTM(32, name="lstm")(drop1)
drop2 = Dropout(0.25, name="drop2")(lstm)
output = Dense(1, activation="sigmoid",
               name="output")(drop2)
model = Model(inputs=imdb_input, outputs=output)
model.summary()    # 顯示模型摘要資訊
```



共享輸入層

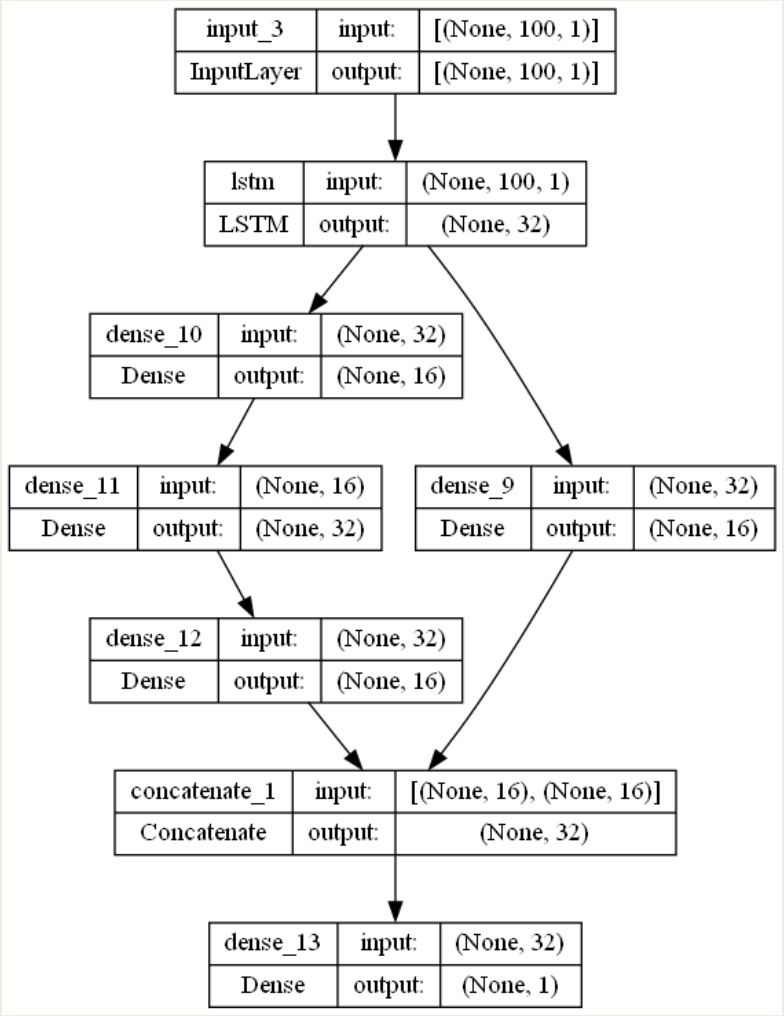
使用範本會出錯，依照網路上的更改方式能成功

```python
from keras.models import Model
from keras.layers import Input, Dense, Flatten, Conv2D, MaxPooling2D
from keras.layers import concatenate

# 定義模型(建立輸入層)
shared_input = Input(shape=(64, 64, 1))
# 建立第1個共享輸入層的卷積和池化層
conv1 = Conv2D(32, kernel_size=3, activation="relu")(shared_input)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
flat1 = Flatten()(pool1)
# 建立第2個共享輸入層的卷積和池化層
conv2 = Conv2D(16, kernel_size=5, activation="relu")(shared_input)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flat2 = Flatten()(pool2)
# 合併前面建立的兩個共享輸入層的卷積和池化層
merge = concatenate([flat1, flat2])
# 合併後建立分類器的Dense層
hidden1 = Dense(10, activation="relu")(merge)
output = Dense(1, activation="sigmoid")(hidden1)
model = Model(inputs=shared_input, outputs=output)
model.summary()    # 顯示模型摘要資訊
from keras.utils import plot_model

plot_model(model, to_file="Ch16_4_1.png", show_shapes=True)
```



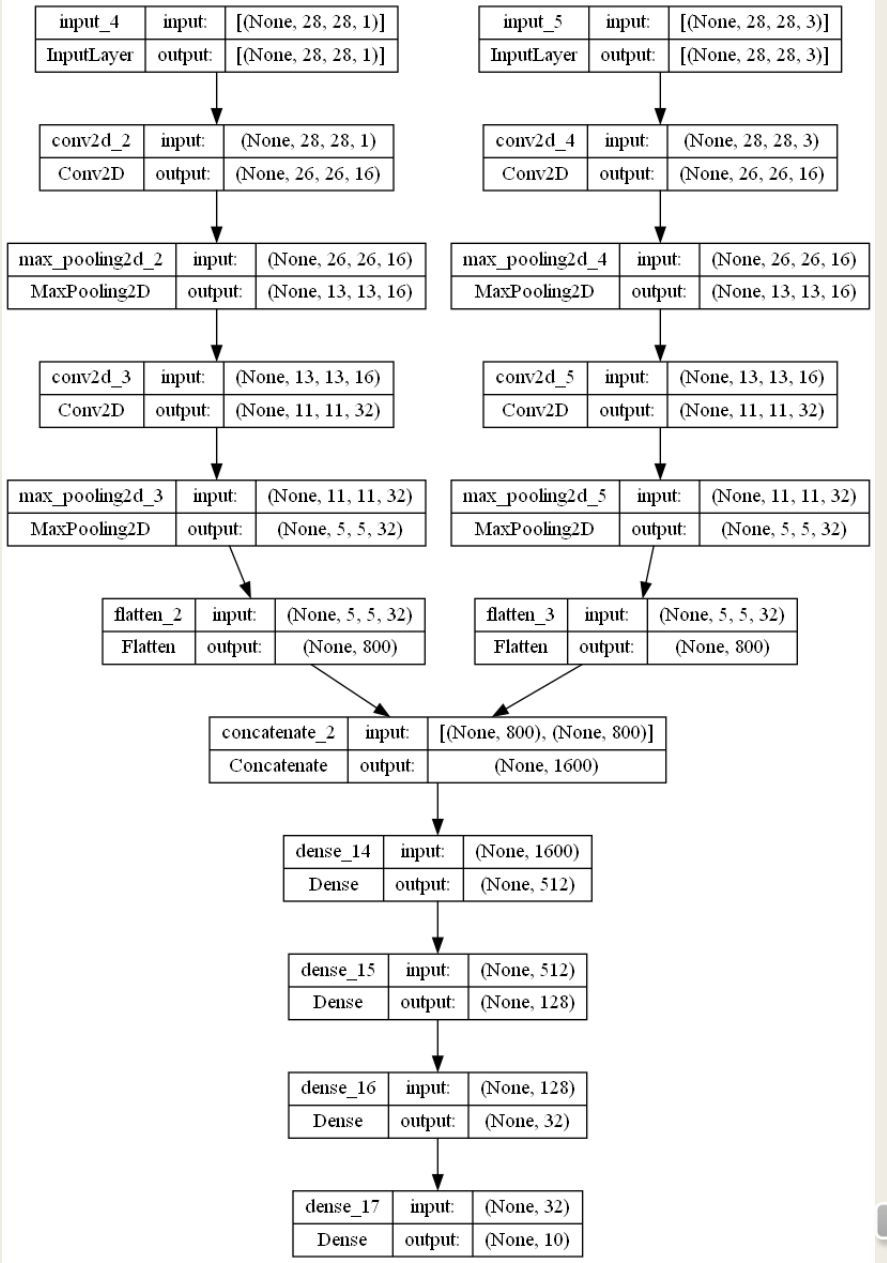
共享特徵萃取層

```python
from keras.models import Model
from keras.layers import Input, Dense, LSTM
from keras.layers import concatenate

# 定義模型
model_input = Input(shape=(100, 1))
lstm = LSTM(32)(model_input)
# 第 1 個共享特徵提取層的1個解釋層
extract1 = Dense(16, activation="relu")(lstm)
# 第 2 個共享特徵提取層的3個解釋層
dense1 = Dense(16, activation="relu")(lstm)
dense2 = Dense(32, activation="relu")(dense1)
extract2 = Dense(16, activation='relu')(dense2)
# 合併 2 個共享特徵提取層的解釋層
merge = concatenate([extract1, extract2])
output = Dense(1, activation="sigmoid")(merge)
model = Model(inputs=model_input, outputs=output)
model.summary()   # 顯示模型摘要資訊
from keras.utils import plot_model

plot_model(model, to_file="Ch16_4_2.png", show_shapes=True)
```



多輸入模型

```python
from keras.models import Model
from keras.layers import Input, Dense, Flatten, Conv2D, MaxPooling2D
from keras.layers import concatenate

# 定義模型
# 第 1 個灰階圖片輸入
input1 = Input(shape=(28, 28, 1))
conv11 = Conv2D(16, (3,3), activation="relu")(input1)
pool11 = MaxPooling2D(pool_size=(2,2))(conv11)
conv12 = Conv2D(32, (3,3), activation="relu")(pool11)
pool12 = MaxPooling2D(pool_size=(2,2))(conv12)
flat1 = Flatten()(pool12)
# 第 2 個彩色圖片輸入
input2 = Input(shape=(28, 28, 3))
conv21 = Conv2D(16, (3,3), activation="relu")(input2)
pool21 = MaxPooling2D(pool_size=(2,2))(conv21)
conv22 = Conv2D(32, (3,3), activation="relu")(pool21)
pool22 = MaxPooling2D(pool_size=(2,2))(conv22)
flat2 = Flatten()(pool22)
# 合併 2 個輸入
merge = concatenate([flat1, flat2])
# 送入最後的分類器(4個Dense層)
dense1 = Dense(512, activation="relu")(merge)
dense2 = Dense(128, activation="relu")(dense1)
dense3 = Dense(32, activation="relu")(dense2)
output = Dense(10, activation="softmax")(dense3)
# 定義多輸入模型
model = Model(inputs=[input1, input2], outputs=output)
model.summary()   # 顯示模型摘要資訊
from keras.utils import plot_model

plot_model(model, to_file="Ch16_5_1.png", show_shapes=True)
```



多輸出模型

```python
from keras.models import Model
from keras.layers import Dense, Input

# 定義模型，建立MLP神經網路
model_input = Input(shape = (784,))
dense1 = Dense(512, activation="relu")(model_input)
dense2 = Dense(128, activation="relu")(dense1)
dense3 = Dense(32, activation ="relu")(dense2)
# 第 1 個分類輸出(1個Dense層)
output = Dense(10, activation="softmax")(dense3)
# 建立第 2 個輸出層，這是自編碼器輸出
up_dense1 = Dense(128, activation="relu")(dense3)
up_dense2 = Dense(512, activation="relu")(up_dense1)
decoded_outputs = Dense(784)(up_dense2)
# 定義多輸出模型
model = Model(model_input, [output, decoded_outputs])
model.summary()   # 顯示模型摘要資訊
from keras.utils import plot_model

plot_model(model, to_file="Ch16_5_2.png", show_shapes=True)
```