In [69]:
```
# # define input data
# import numpy as np
# from numpy import asarray
# from keras.models import Sequential
# from keras.layers import Conv2D
# data = [[0, 0, 0, 1, 1, 0, 0, 0],
#  [0, 0, 0, 1, 1, 0, 0, 0],
#  [0, 0, 0, 1, 1, 0, 0, 0],
#  [0, 0, 0, 1, 1, 0, 0, 0],
#  [0, 0, 0, 1, 1, 0, 0, 0],
#  [0, 0, 0, 1, 1, 0, 0, 0],
#  [0, 0, 0, 1, 1, 0, 0, 0]]
# data = asarray(data)
# data = data.reshape(1, 8, 8, 1)
```

In [82]:
```
import numpy as np
from keras.models import Sequential
from keras.layers import Conv2D, Activation, AveragePooling2D, MaxPooling2D, GlobalAvera

# Define input data
data = np.array([[0, 0, 0, 1, 1, 0, 0, 0],
                 [0, 0, 0, 1, 1, 0, 0, 0],
                 [0, 0, 0, 1, 1, 0, 0, 0],
                 [0, 0, 0, 1, 1, 0, 0, 0],
                 [0, 0, 0, 1, 1, 0, 0, 0],
                 [0, 0, 0, 1, 1, 0, 0, 0],
                 [0, 0, 0, 1, 1, 0, 0, 0],
                 [0, 0, 0, 1, 1, 0, 0, 0]])
data = data.reshape(1, 8, 8, 1)

# Create model
model = Sequential()

# Add first convolutional layer
model.add(Conv2D(32, (3, 3), padding='same', input_shape=(8, 8, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Add second convolutional layer
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten the feature maps
model.add(Flatten())


model.add(Dense(128))
model.add(Activation('relu'))


model.add(Dense(1))
model.add(Activation('relu'))


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


model.summary()
```

```
loss, accuracy = model.evaluate(data, np.array([1]), verbose=0)
print("Accuracy:", accuracy)
```

```
Model: "sequential_55"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_85 (Conv2D)          (None, 8, 8, 32)          320

 activation_32 (Activation)  (None, 8, 8, 32)          0

 max_pooling2d_49 (MaxPoolin  (None, 4, 4, 32)         0
 g2D)

 conv2d_86 (Conv2D)          (None, 4, 4, 64)          18496

 activation_33 (Activation)  (None, 4, 4, 64)          0

 max_pooling2d_50 (MaxPoolin  (None, 2, 2, 64)         0
 g2D)

 flatten_9 (Flatten)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 activation_34 (Activation)  (None, 128)               0

 dense_3 (Dense)             (None, 1)                 129

 activation_35 (Activation)  (None, 1)                 0

=================================================================
Total params: 51,841
Trainable params: 51,841
Non-trainable params: 0
_________________________________________________________________
Accuracy: 0.0
```

In [ ]:
```
# define a vertical line detector
detector = [[[[0]],[[1]],[[0]]],
           [[[0]],[[1]],[[0]]],
           [[[0]],[[1]],[[0]]]]
weights = [asarray(detector), asarray([0.0])]
# store the weights in the model
model.set_weights(weights)
# confirm they were stored
print(model.get_weights())
```

In [ ]:
```
# apply filter to input data
yhat = model.predict(data)
```

In [ ]:
```
for r in range(yhat.shape[1]):
 # print each column in the row
 print([yhat[0,r,c,0] for c in range(yhat.shape[2])])
```

In [ ]:
```
# example of calculation 2d convolutions
from numpy import asarray
from keras.models import Sequential
from keras.layers import Conv2D
# define input data
data = [[0, 0, 0, 1, 1, 0, 0, 0],
                    [0, 0, 0, 1, 1, 0, 0, 0],
                    [0, 0, 0, 1, 1, 0, 0, 0],
                    [0, 0, 0, 1, 1, 0, 0, 0],
```

```
                    [0, 0, 0, 1, 1, 0, 0, 0],
                    [0, 0, 0, 1, 1, 0, 0, 0],
                    [0, 0, 0, 1, 1, 0, 0, 0]]
data = asarray(data)
data = data.reshape(1, 8, 8, 1)
# create model
model = Sequential()
model.add(Conv2D(1, (3,3), input_shape=(8, 8, 1)))
# define a vertical line detector
detector = [[[[0]],[[1]],[[0]]],
           [[[0]],[[1]],[[0]]],
           [[[0]],[[1]],[[0]]]]
weights = [asarray(detector), asarray([0.0])]
# store the weights in the model
model.set_weights(weights)
# confirm they were stored
print(model.get_weights())
# apply filter to input data
yhat = model.predict(data)
for r in range(yhat.shape[1]):
        # print each column in the row
        print([yhat[0,r,c,0] for c in range(yhat.shape[2])])
```

In [ ]:

```
from numpy import asarray
print(asarray([1, 1, 1]).dot(asarray([1, 1, 1])))
```

In [ ]:

```
from numpy import asarray
from numpy import tensordot
m1 = asarray([[0, 1, 0],
   [0, 1, 0],
   [0, 1, 0]])
m2 = asarray([[0, 1, 1],
   [0, 1,1],
   [0, 1, 1]])
print(tensordot(m1, m2))
```