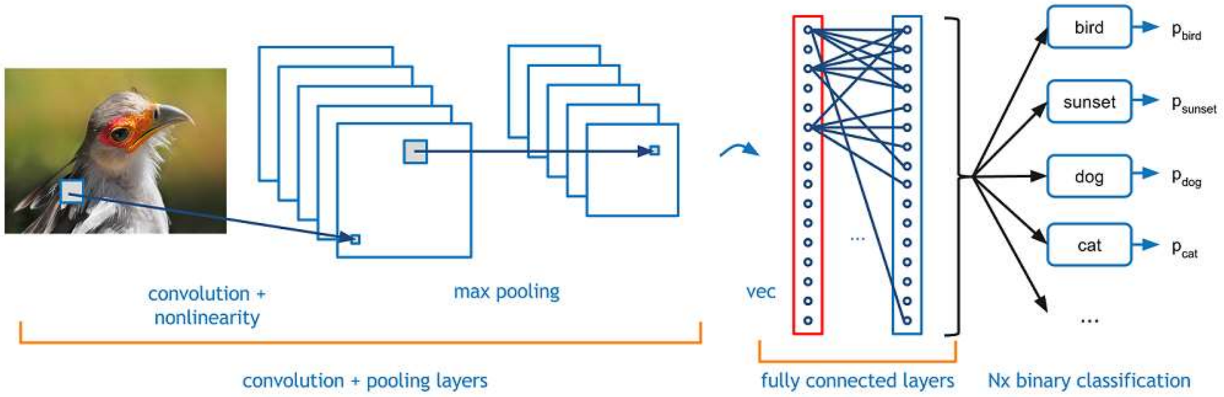
tensorflow2-基礎CNN網路



In [13]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
print(tf.__version__)
```

```
2.3.1
```

1.構造數據

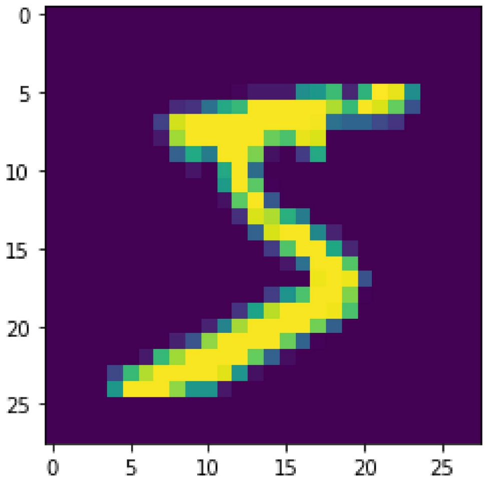
In [14]:

```

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
print(x_train.shape, ' ', y_train.shape)
print(x_test.shape, ' ', y_test.shape)

```

```
(60000, 28, 28)   (60000,)
(10000, 28, 28)   (10000,)
```

In [15]:

```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
plt.show()
```



In [16]:

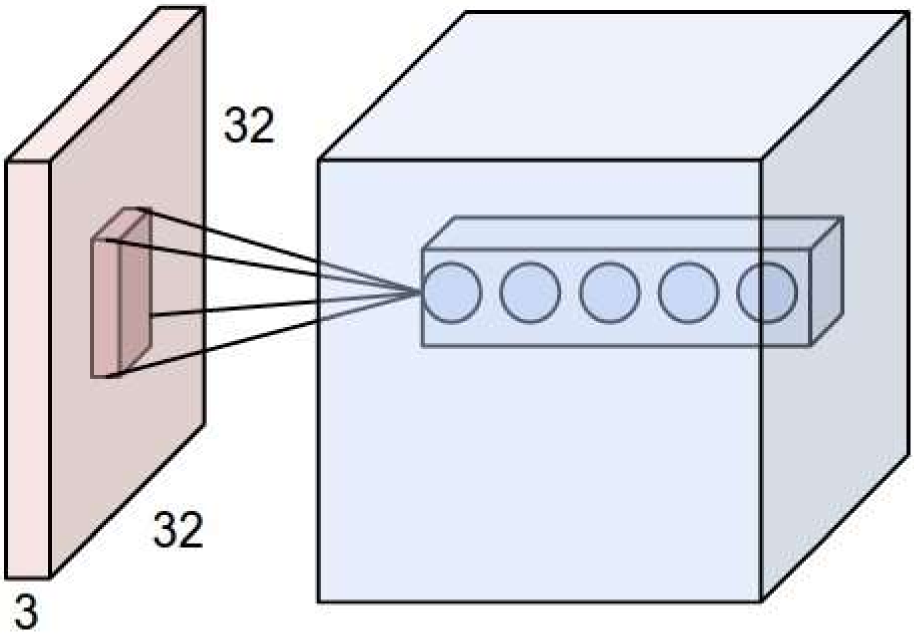
```
x_train = x_train.reshape((-1,28,28,1))
x_test = x_test.reshape((-1,28,28,1))
```

2.構造網路

In [17]:

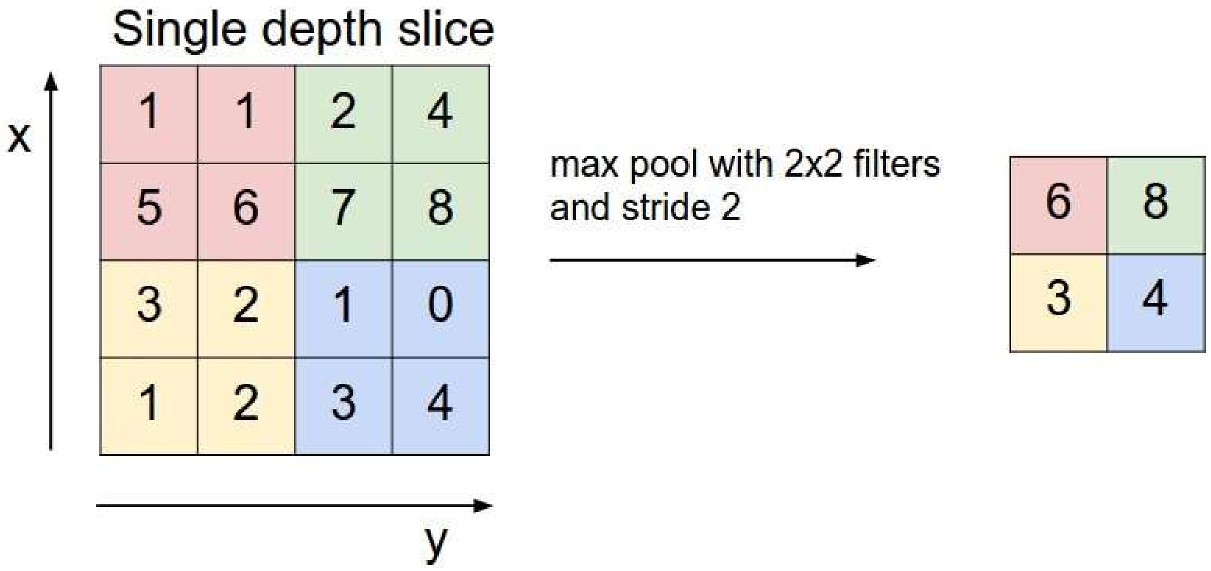
```
model = keras.Sequential()
```

卷積層



In [18]:
```
model.add(layers.Conv2D(input_shape=(x_train.shape[1], x_train.shape[2], x_train.
                        filters=32, kernel_size=(3,3), strides=(1,1), padding='va
                       activation='relu'))
```

池化層



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

In [19]:
```
model.add(layers.MaxPool2D(pool_size=(2,2)))
```

全連接層

In [20]:
```
model.add(layers.Flatten())
model.add(layers.Dense(32, activation='relu'))
# 分類層
model.add(layers.Dense(10, activation='softmax'))
```

3.模型配置

In [21]:
```
model.compile(optimizer=keras.optimizers.Adam(),
             # loss=keras.losses.CategoricalCrossentropy(),  # 需要使用to_categor
             loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 5408)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 32)                173088    
_________________________________________________________________
dense_3 (Dense)              (None, 10)                330       
=================================================================
Total params: 173,738
Trainable params: 173,738
Non-trainable params: 0
_________________________________________________________________
```

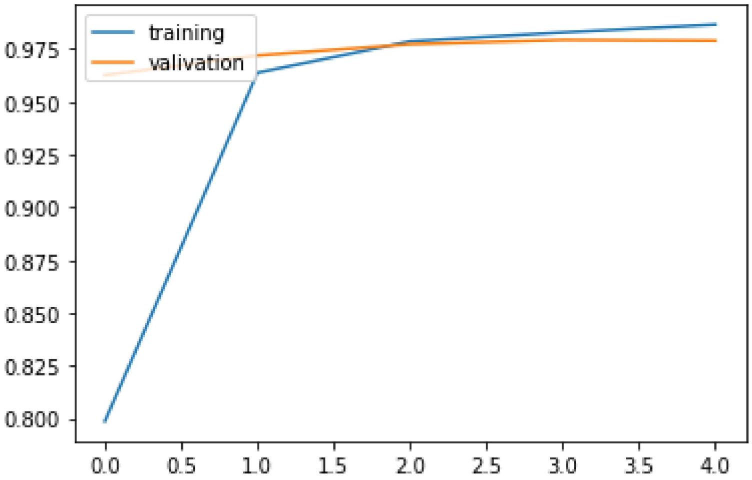
4.模型訓練

In [22]:
```
history = model.fit(x_train, y_train, batch_size=64, epochs=5, validation_split=
```

```
Epoch 1/5
844/844 [==============================] - 11s 13ms/step - loss: 0.7368 - accur
acy: 0.7987 - val_loss: 0.1450 - val_accuracy: 0.9627
Epoch 2/5
844/844 [==============================] - 11s 13ms/step - loss: 0.1334 - accur
acy: 0.9638 - val_loss: 0.1044 - val_accuracy: 0.9720
Epoch 3/5
844/844 [==============================] - 11s 13ms/step - loss: 0.0754 - accur
acy: 0.9787 - val_loss: 0.0842 - val_accuracy: 0.9773
Epoch 4/5
844/844 [==============================] - 12s 14ms/step - loss: 0.0582 - accur
acy: 0.9829 - val_loss: 0.0919 - val_accuracy: 0.9793
Epoch 5/5
844/844 [==============================] - 12s 14ms/step - loss: 0.0425 - accur
acy: 0.9866 - val_loss: 0.0929 - val_accuracy: 0.9790
```

In [23]:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'valivation'], loc='upper left')
plt.show()
```



In [24]:

```
res = model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0988 - accurac
y: 0.9742
```