TensorFlow2教程-mlp及深度學習常見技巧

我們將以mlp對為，基礎模型，然後介紹一些深度學習常見技巧， 如： 權重初始化， 啟動函數， 優化器， 批規範化， dropout，模型集成

In [1]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
print(tf.__version__)
```

```
2.3.1
```

1.導入數據

In [2]:
```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
x_train = x_train.reshape([x_train.shape[0], -1])
x_test = x_test.reshape([x_test.shape[0], -1])
print(x_train.shape, ' ', y_train.shape)
print(x_test.shape, ' ', y_test.shape)
```

```
(60000, 784)   (60000,)
(10000, 784)   (10000,)
```

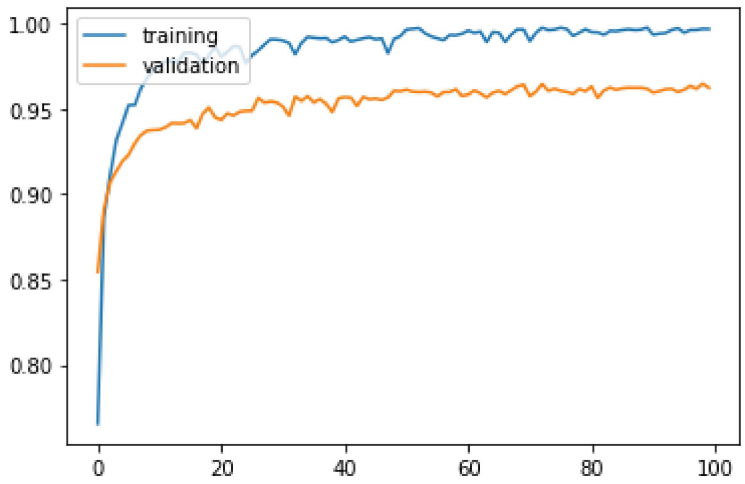
2.基礎模型

In [3]:
```
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(784,)),
    layers.Dense(64, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer=keras.optimizers.Adam(),
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                50240
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 59,210
Trainable params: 59,210
Non-trainable params: 0
_________________________________________________________________
```

In [4]:
```
history = model.fit(x_train, y_train, batch_size=256, epochs=100, validation_spli
```

In [5]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc='upper left')
plt.show()
```



In [6]:
```
result = model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 0s 978us/step - loss: 0.3450 - accur
acy: 0.9695
```

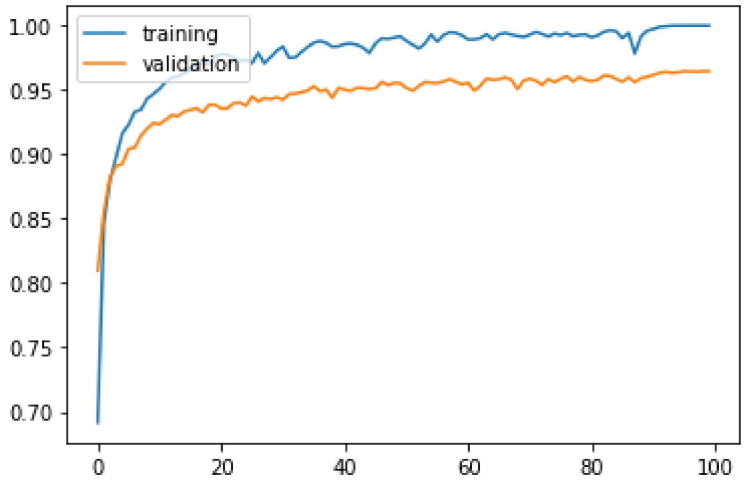
3.權重初始化

In [7]:
```
model = keras.Sequential([
    layers.Dense(64, activation='relu', kernel_initializer='he_normal', input_sha
    layers.Dense(64, activation='relu', kernel_initializer='he_normal'),
    layers.Dense(64, activation='relu', kernel_initializer='he_normal'),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer=keras.optimizers.Adam(),
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 64)                50240
_________________________________________________________________
dense_5 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_6 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_7 (Dense)              (None, 10)                650
=================================================================
Total params: 59,210
Trainable params: 59,210
Non-trainable params: 0
_________________________________________________________________
```

In [8]:
```
history = model.fit(x_train, y_train, batch_size=256, epochs=100, validation_spli
```

In [10]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc='upper left')
plt.show()
```

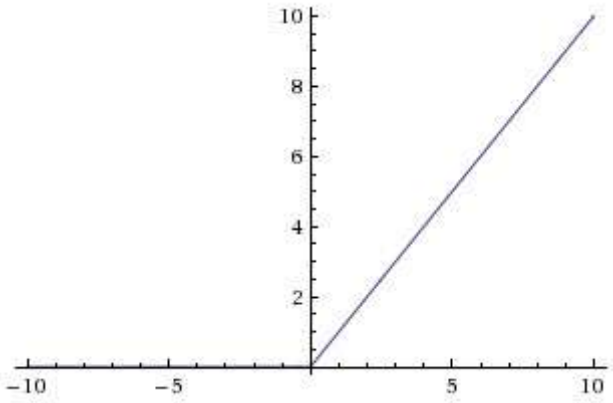


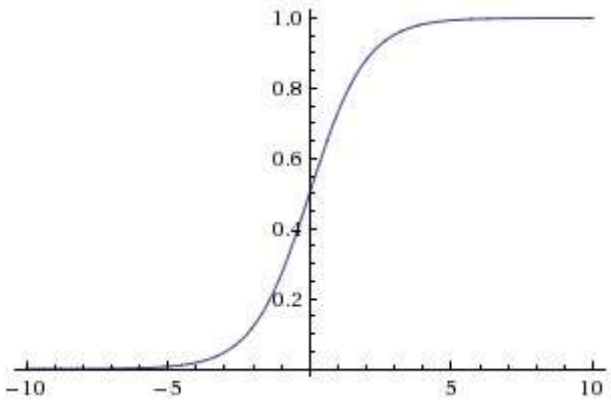
In [11]:
```
result = model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.3618 - accurac
y: 0.9662
```

4.啟動函數

relu和sigmoid對比



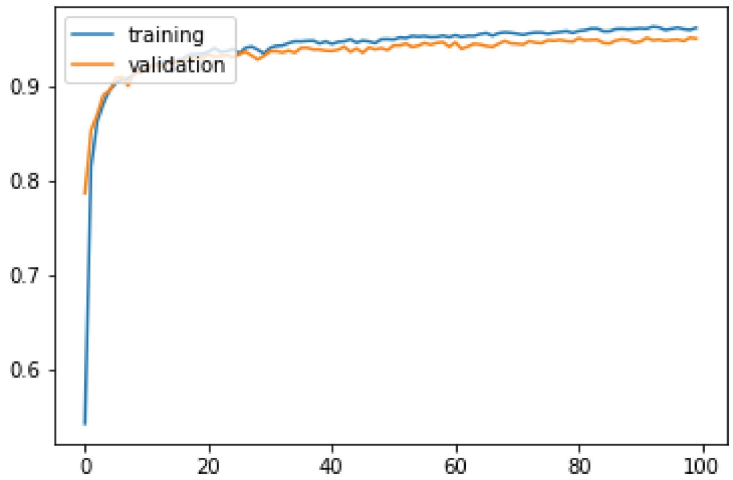


In [12]:
```
model = keras.Sequential([
    layers.Dense(64, activation='sigmoid', input_shape=(784,)),
    layers.Dense(64, activation='sigmoid'),
    layers.Dense(64, activation='sigmoid'),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer=keras.optimizers.Adam(),
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 64)                50240
_________________________________________________________________
dense_9 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_10 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_11 (Dense)             (None, 10)                650
=================================================================
Total params: 59,210
Trainable params: 59,210
Non-trainable params: 0
_________________________________________________________________
```

In [13]:
```
history = model.fit(x_train, y_train, batch_size=256, epochs=100, validation_spli
```

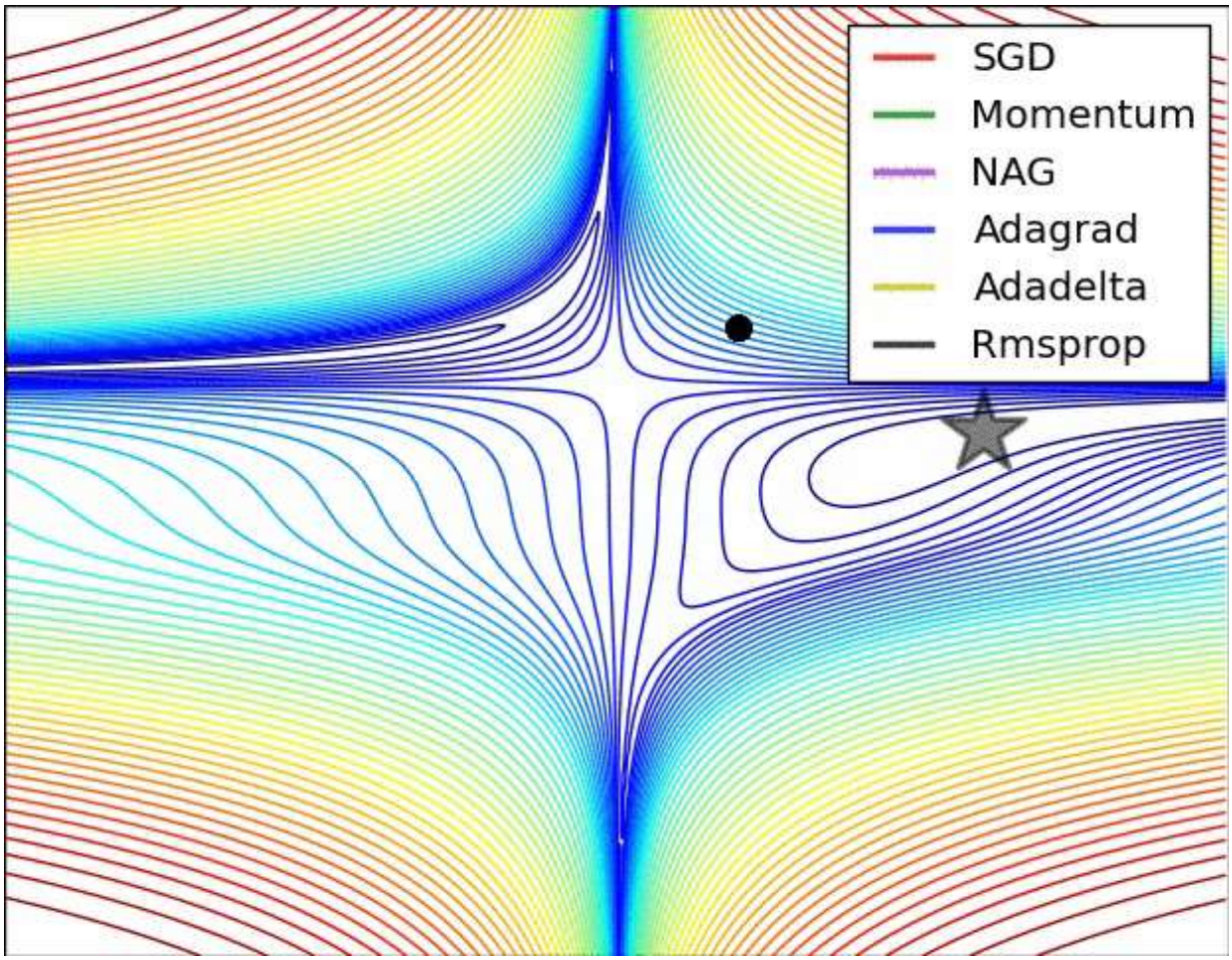
In [34]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc='upper left')
plt.show()
```



In [35]:
```
result = model.evaluate(x_test, y_test)
```

```
10000/10000 [==============================] - 0s 29us/sample - loss: 0.1526 -
accuracy: 0.9529
```

5.優化器

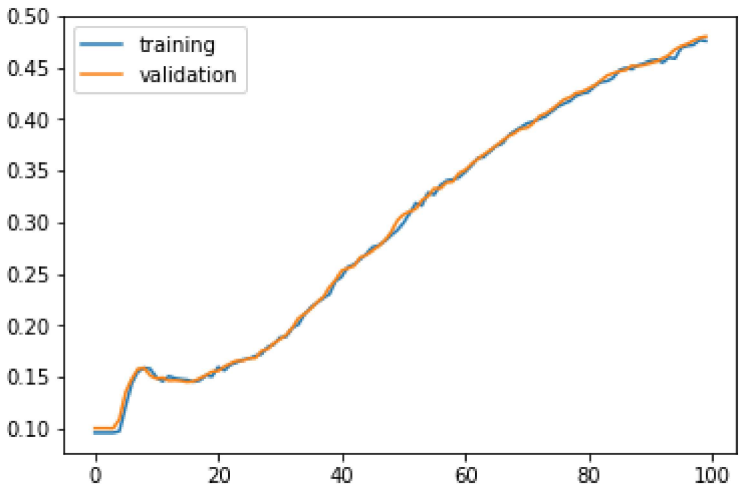


In [36]:
```
model = keras.Sequential([
    layers.Dense(64, activation='sigmoid', input_shape=(784,)),
    layers.Dense(64, activation='sigmoid'),
    layers.Dense(64, activation='sigmoid'),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer=keras.optimizers.SGD(),
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 64)                50240
_________________________________________________________________
dense_28 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_29 (Dense)             (None, 64)                4160
_________________________________________________________________
dense_30 (Dense)             (None, 10)                650
=================================================================
Total params: 59,210
Trainable params: 59,210
Non-trainable params: 0
_________________________________________________________________
```

In [37]:
```
history = model.fit(x_train, y_train, batch_size=256, epochs=100, validation_spli
```

In [38]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc='upper left')
plt.show()
```



In [39]:
```
result = model.evaluate(x_test, y_test)
```

```
10000/10000 [==============================] - 0s 44us/sample - loss: 2.1199 -
accuracy: 0.4749
```

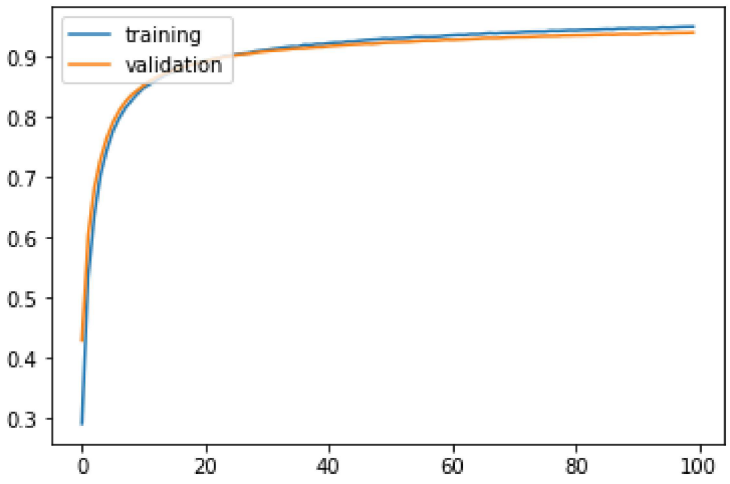
6.批正则化

In [41]:
```
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(784,)),
    layers.BatchNormalization(),
    layers.Dense(64, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(64, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer=keras.optimizers.SGD(),
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_31 (Dense)             (None, 64)                50240
_________________________________________________________________
batch_normalization_v2 (Batc (None, 64)                256
_________________________________________________________________
dense_32 (Dense)             (None, 64)                4160
_________________________________________________________________
batch_normalization_v2_1 (Ba (None, 64)                256
_________________________________________________________________
dense_33 (Dense)             (None, 64)                4160
_________________________________________________________________
batch_normalization_v2_2 (Ba (None, 64)                256
_________________________________________________________________
dense_34 (Dense)             (None, 10)                650
=================================================================
Total params: 59,978
Trainable params: 59,594
Non-trainable params: 384
_________________________________________________________________
```

In [42]:
```
history = model.fit(x_train, y_train, batch_size=256, epochs=100, validation_spli
```

In [43]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc='upper left')
plt.show()
```

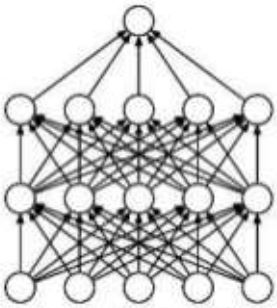


In [44]:
```
result = model.evaluate(x_test, y_test)
```

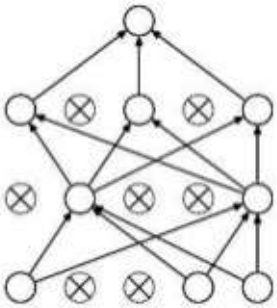
```
10000/10000 [==============================] - 0s 25us/sample - loss: 0.1863 -
accuracy: 0.9447
```

7.dropout

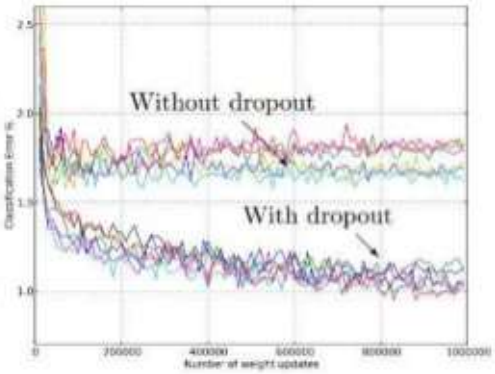
Dropout



(a) Standard Neural Net



(b) After applying dropout.



Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

Table 6: Results on the ILSVRC-2012 validation/test set.

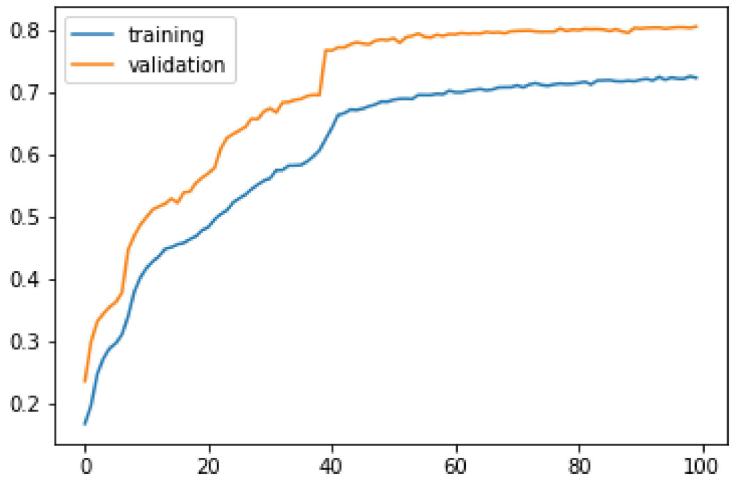
Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

In [45]:
```
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(784,)),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(10, activation='softmax')
])
model.compile(optimizer=keras.optimizers.SGD(),
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_35 (Dense)             (None, 64)                50240
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_36 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_37 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_38 (Dense)             (None, 10)                650
=================================================================
Total params: 59,210
Trainable params: 59,210
Non-trainable params: 0
_________________________________________________________________
```

In [46]:
```
history = model.fit(x_train, y_train, batch_size=256, epochs=100, validation_spli
```

In [47]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc='upper left')
plt.show()
```



In [48]:
```
result = model.evaluate(x_test, y_test)
```

```
10000/10000 [==============================] - 0s 27us/sample - loss: 0.6157 -
accuracy: 0.8132
```

8.模型集成

下面是使用投票的方法進行模型集成

In [54]:
```
import numpy as np
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score

def mlp_model():
    model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(784,)),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(10, activation='softmax')
    ])
    model.compile(optimizer=keras.optimizers.SGD(),
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
    return model
model1 = KerasClassifier(build_fn=mlp_model, epochs=100, verbose=0)
model2 = KerasClassifier(build_fn=mlp_model, epochs=100, verbose=0)
model3 = KerasClassifier(build_fn=mlp_model, epochs=100, verbose=0)
```

In [55]:
```
ensemble_clf = VotingClassifier(estimators=[
    ('model1', model1), ('model2', model2), ('model3', model3)
], voting='soft')
```

In [56]:
```
ensemble_clf.fit(x_train, y_train)
```

Out[56]:
```
VotingClassifier(estimators=[('model1', <tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at 0x7f4ed7d4c518>), ('model2', <tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at 0x7f4ed7d4c470>), ('model3', <tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at 0x7f4ed7d4c588>)],
         flatten_transform=None, n_jobs=None, voting='soft', weights=None)
```

In [57]:
```
y_pred = ensemble_clf.predict(x_test)
print('acc: ', accuracy_score(y_pred, y_test))
```

```
acc:  0.9504
```

9.全部使用

In [5]:
```
from tensorflow.keras import layers

import numpy as np
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score

def mlp_model():
    model = keras.Sequential([
    layers.Dense(64, activation='relu', kernel_initializer='he_normal', input_sha
    layers.BatchNormalization(),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu', kernel_initializer='he_normal'),
    layers.BatchNormalization(),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu', kernel_initializer='he_normal'),
    layers.BatchNormalization(),
    layers.Dropout(0.2),
    layers.Dense(10, activation='softmax')
    ])
    model.compile(optimizer=keras.optimizers.SGD(),
             loss=keras.losses.SparseCategoricalCrossentropy(),
             metrics=['accuracy'])
    return model
model1 = KerasClassifier(build_fn=mlp_model, epochs=100, verbose=0)
model2 = KerasClassifier(build_fn=mlp_model, epochs=100, verbose=0)
model3 = KerasClassifier(build_fn=mlp_model, epochs=100, verbose=0)
model4 = KerasClassifier(build_fn=mlp_model, epochs=100, verbose=0)
ensemble_clf = VotingClassifier(estimators=[
    ('model1', model1), ('model2', model2), ('model3', model3),('model4', model4)
```

In [9]:
```
ensemble_clf.fit(x_train, y_train)
```

Out[9]:
```
VotingClassifier(estimators=[('model1', <tensorflow.python.keras.wrappers.sciki
t_learn.KerasClassifier object at 0x7f7183d246a0>), ('model2', <tensorflow.pyth
on.keras.wrappers.scikit_learn.KerasClassifier object at 0x7f7183d245c0>), ('mo
del3', <tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at
0x7f7183d5a198>), ('model4', <tensorflow.python.keras.wrappers.scikit_learn.Ker
asClassifier object at 0x7f71839c94e0>)],
         flatten_transform=None, n_jobs=None, voting='hard', weights=None)
```

In [10]:
```
y_predict = ensemble_clf.predict(x_test)
```

In [11]:
```
print('acc: ', accuracy_scoreecuracy_scorecuracy_score(y_pred, y_test))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-c1dea66b1d3d> in <module>
----> 1 print('acc: ', accuracy_score(y_pred, y_test))

NameError: name 'y_pred' is not defined
```

In [ ]:
```
```