HW4 Pattern Recognition

2019150445 신백록

1. Download MNIST dataset

In [1]:
```
import numpy as np
from tensorflow.keras.datasets import mnist

(X,y),(X_test,y_test)=mnist.load_data()
print(X.shape)
print(X_test.shape)
print(y.shape)
print(y_test.shape)
```

```
Init Plugin
Init Graph Optimizer
Init Kernel
(60000, 28, 28)
(10000, 28, 28)
(60000,)
(10000,)
```

In [2]:
```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1, stra
```

In [3]:
```
print(X_train.shape)
print(X_val.shape)
```

```
(54000, 28, 28)
(6000, 28, 28)
```

In [4]:
```
X_train=np.reshape(X_train,(-1,28,28,1))
X_test=np.reshape(X_test,(-1,28,28,1))
X_val=np.reshape(X_val,(-1,28,28,1))
```

In [5]:
```
from tensorflow.keras.utils import to_categorical

X_train=X_train/255.
X_val=X_val/255.
X_test=X_test/255.

y_train=to_categorical(y_train)
y_val=to_categorical(y_val)
y_test=to_categorical(y_test)
print(X_train.shape)
print(y_train.shape)
```

```
(54000, 28, 28, 1)
(54000, 10)
```

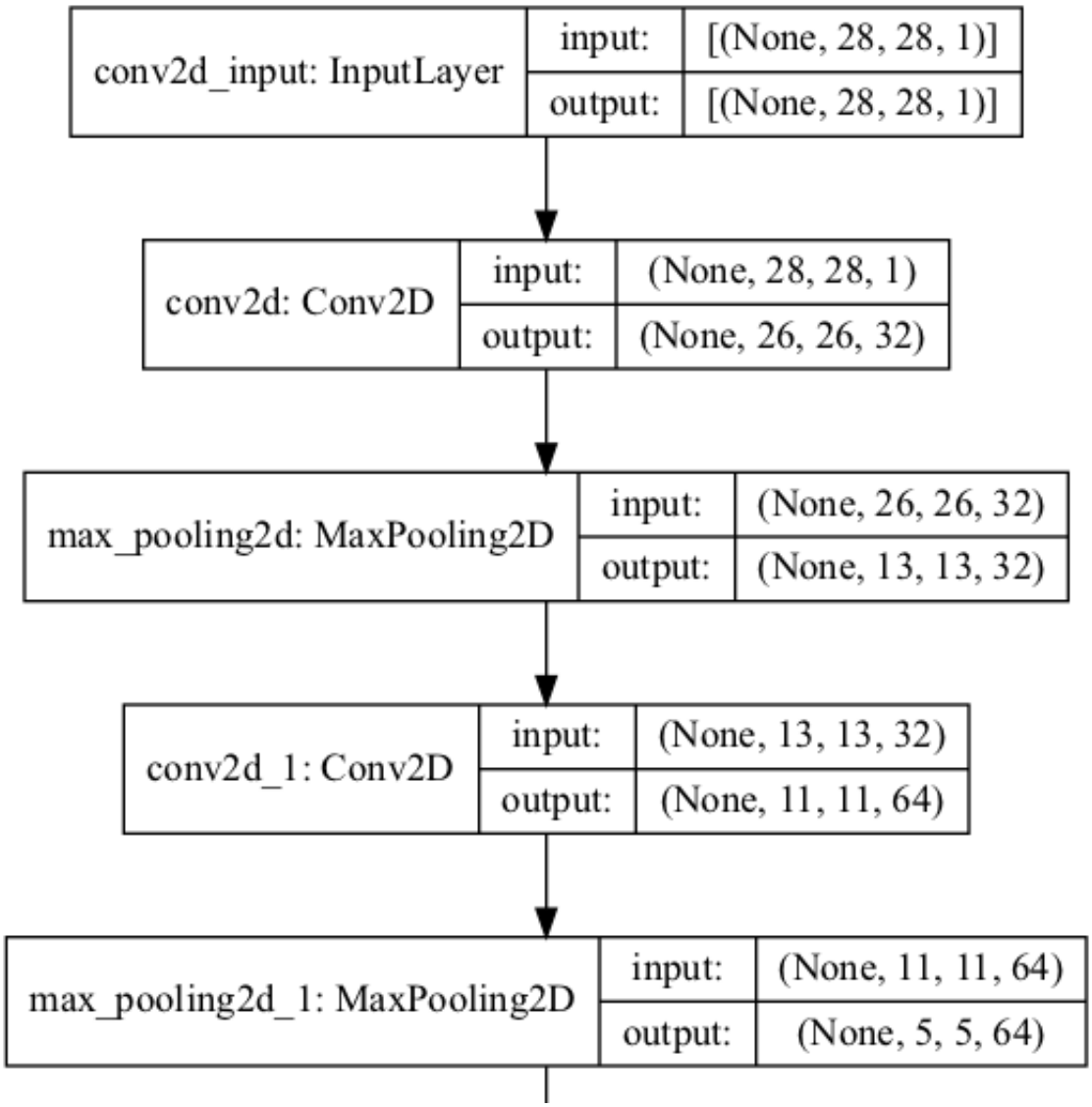
2. Build the CNN Model

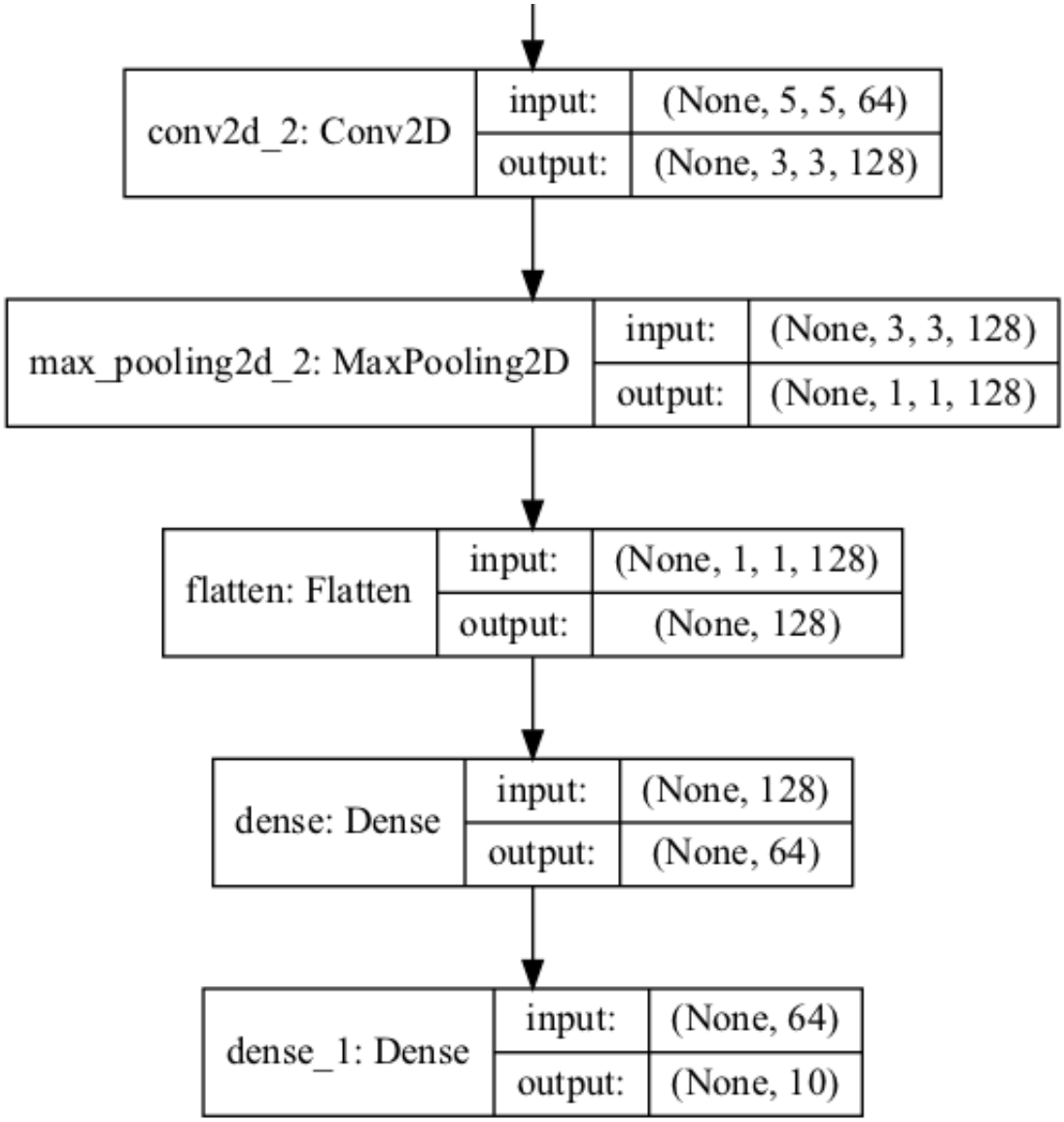
In [6]:
```
from tensorflow.keras.layers import Conv2D, MaxPooling2D,Flatten,Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import plot_model

A_net=Sequential()
A_net.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
A_net.add(MaxPooling2D((2,2), strides=2))
A_net.add(Conv2D(64,(3,3),activation='relu'))
A_net.add(MaxPooling2D((2,2), strides=2))
A_net.add(Conv2D(128,(3,3),activation='relu'))
A_net.add(MaxPooling2D((2,2), strides=2))
A_net.add(Flatten())
A_net.add(Dense(64,activation='relu'))
A_net.add(Dense(10,activation='softmax'))

A_net.summary()
plot_model(A_net,show_shapes=True)
```

```
Metal device set to: Apple M1
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
```

```
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 101,578
Trainable params: 101,578
Non-trainable params: 0
_________________________________________________________________
2021-12-02 00:25:27.405725: I tensorflow/core/common_runtime/pluggable_devi
ce/pluggable_device_factory.cc:305] Could not identify NUMA node of platfor
m GPU ID 0, defaulting to 0. Your kernel may not have been built with NUMA
support.
2021-12-02 00:25:27.405825: I tensorflow/core/common_runtime/pluggable_devi
ce/pluggable_device_factory.cc:271] Created TensorFlow device (/job:localho
st/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical PluggableDev
ice (device: 0, name: METAL, pci bus id: <undefined>)
```

Out[6]:





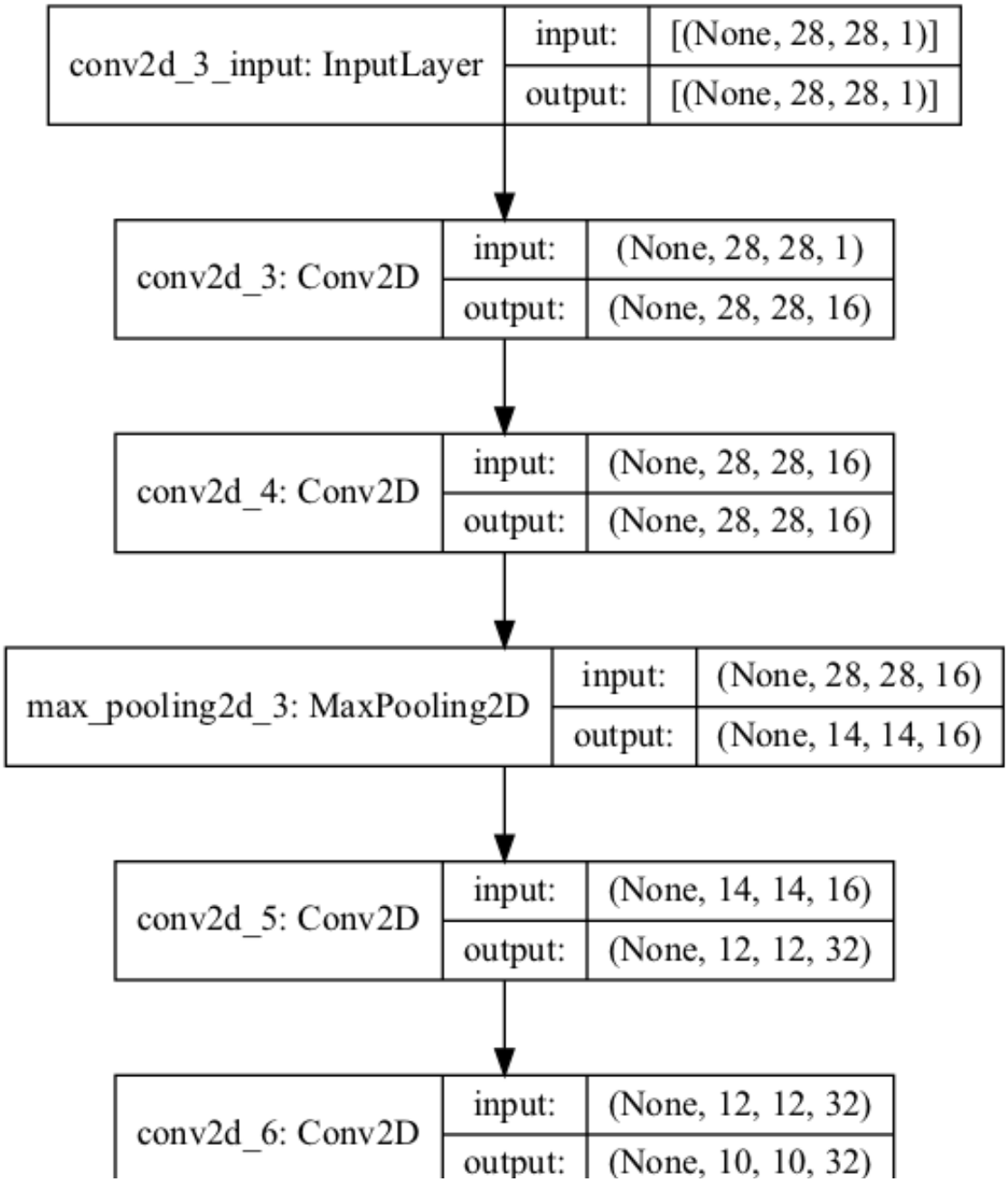
In [7]:

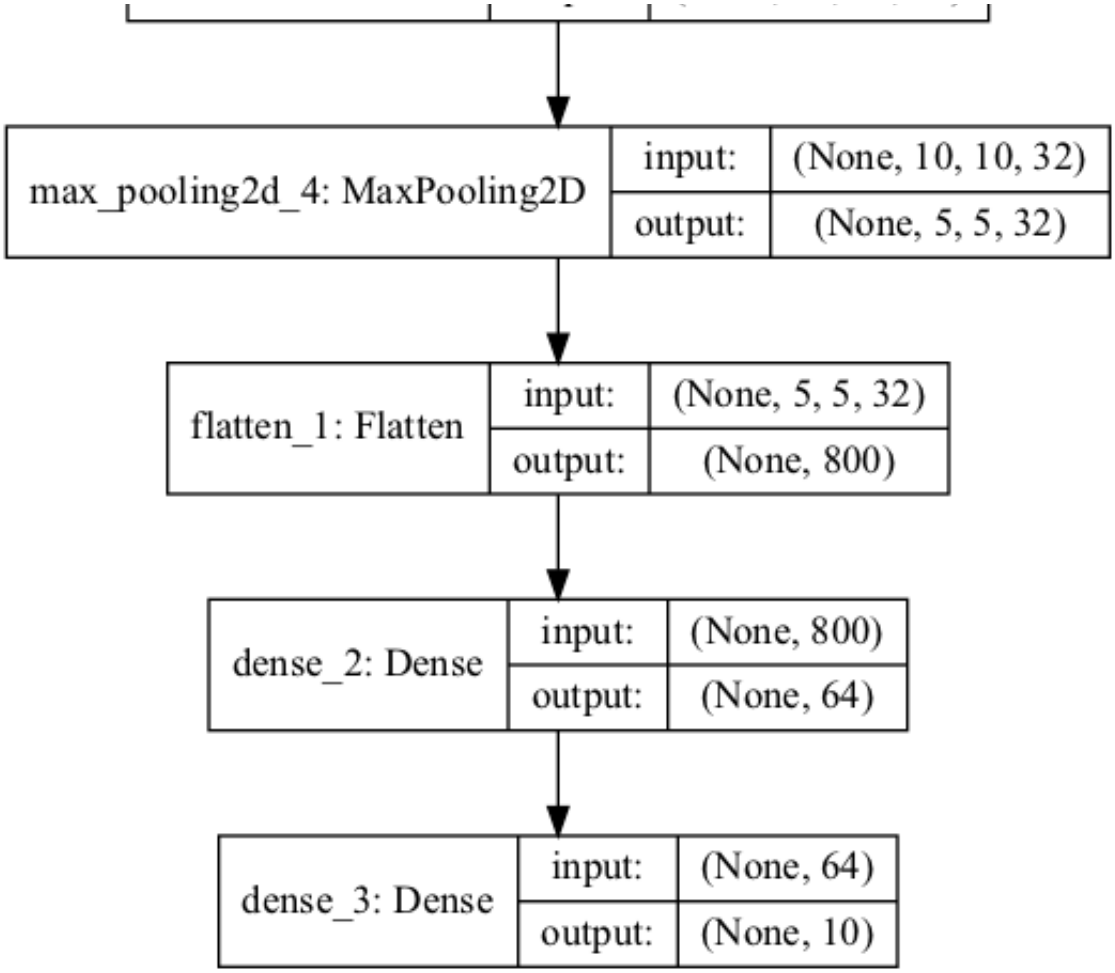
```
B_net=Sequential()
B_net.add(Conv2D(16,(3,3),activation='relu',input_shape=(28,28,1),padding=
B_net.add(Conv2D(16,(3,3),activation='relu',padding='same'))
B_net.add(MaxPooling2D((2,2), strides=2))
B_net.add(Conv2D(32,(3,3),activation='relu'))
B_net.add(Conv2D(32,(3,3),activation='relu'))
B_net.add(MaxPooling2D((2,2), strides=2))
B_net.add(Flatten())
B_net.add(Dense(64,activation='relu'))
B_net.add(Dense(10,activation='softmax'))

B_net.summary()
plot_model(B_net, show_shapes=True)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 16)        160
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 16)        2320
```

```
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 32)        4640
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 10, 10, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 800)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                51264
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 68,282
Trainable params: 68,282
Non-trainable params: 0
_________________________________________________________________
```

Out[7]:





3. Training and Evaluation

3.a)

In [8]:
```
A_net.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['
A_result=A_net.fit(X_train,y_train,batch_size=32,epochs=10,validation_data=
```

```
2021-12-02 00:25:29.067786: I tensorflow/compiler/mlir/mlir_graph_optimizat
ion_pass.cc:176] None of the MLIR Optimization Passes are enabled (register
ed 2)
2021-12-02 00:25:29.071482: W tensorflow/core/platform/profile_utils/cpu_ut
ils.cc:128] Failed to get CPU frequency: 0 Hz
2021-12-02 00:25:29.211361: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
Epoch 1/10
1688/1688 [==============================] - ETA: 0s - loss: 0.2151 - acc:
0.9339
2021-12-02 00:25:43.756813: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
1688/1688 [==============================] - 15s 9ms/step - loss: 0.2151 -
acc: 0.9339 - val_loss: 0.0941 - val_acc: 0.9738
Epoch 2/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0747 -
acc: 0.9763 - val_loss: 0.1026 - val_acc: 0.9693
Epoch 3/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0516 -
acc: 0.9840 - val_loss: 0.0633 - val_acc: 0.9807
Epoch 4/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0408 -
acc: 0.9875 - val_loss: 0.0595 - val_acc: 0.9828
Epoch 5/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0342 -
acc: 0.9890 - val_loss: 0.0537 - val_acc: 0.9813
Epoch 6/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0259 -
acc: 0.9916 - val_loss: 0.0406 - val_acc: 0.9890
Epoch 7/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0217 -
acc: 0.9929 - val_loss: 0.0635 - val_acc: 0.9827
Epoch 8/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0182 -
acc: 0.9941 - val_loss: 0.0447 - val_acc: 0.9877
Epoch 9/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0148 -
acc: 0.9951 - val_loss: 0.0606 - val_acc: 0.9845
Epoch 10/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0135 -
acc: 0.9956 - val_loss: 0.0579 - val_acc: 0.9860
```

In [9]:

```
A_val_loss, A_val_acc = A_net.evaluate(X_val,y_val)
print(A_val_acc)
```

```
188/188 [==============================] - 1s 5ms/step - loss: 0.0579 - acc
: 0.9860
0.9860000014305115
```

In [10]:

```
B_net.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['a
B_result=B_net.fit(X_train,y_train,batch_size=32,epochs=10,validation_data=
```

```
Epoch 1/10
   6/1688 [..............................] - ETA: 17s - loss: 2.2830 - acc:
0.1823
2021-12-02 00:28:13.572537: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
1687/1688 [============================>.] - ETA: 0s - loss: 0.1448 - acc:
0.9538
2021-12-02 00:28:28.066847: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
1688/1688 [==============================] - 15s 9ms/step - loss: 0.1448 -
acc: 0.9539 - val_loss: 0.0689 - val_acc: 0.9810
Epoch 2/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0461 -
acc: 0.9856 - val_loss: 0.0504 - val_acc: 0.9855
Epoch 3/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0329 -
acc: 0.9893 - val_loss: 0.0398 - val_acc: 0.9882
Epoch 4/10
1688/1688 [==============================] - 16s 10ms/step - loss: 0.0248 -
acc: 0.9921 - val_loss: 0.0441 - val_acc: 0.9883
Epoch 5/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0191 -
acc: 0.9940 - val_loss: 0.0296 - val_acc: 0.9913
Epoch 6/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0172 -
acc: 0.9945 - val_loss: 0.0350 - val_acc: 0.9908
Epoch 7/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0138 -
acc: 0.9956 - val_loss: 0.0399 - val_acc: 0.9890
Epoch 8/10
1688/1688 [==============================] - 16s 9ms/step - loss: 0.0122 -
acc: 0.9960 - val_loss: 0.0374 - val_acc: 0.9897
Epoch 9/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0099 -
acc: 0.9966 - val_loss: 0.0467 - val_acc: 0.9895
Epoch 10/10
1688/1688 [==============================] - 15s 9ms/step - loss: 0.0089 -
acc: 0.9971 - val_loss: 0.0381 - val_acc: 0.9903
```

In [11]:

```
B_val_loss, B_val_acc = B_net.evaluate(X_val,y_val)
print(B_val_acc)
```

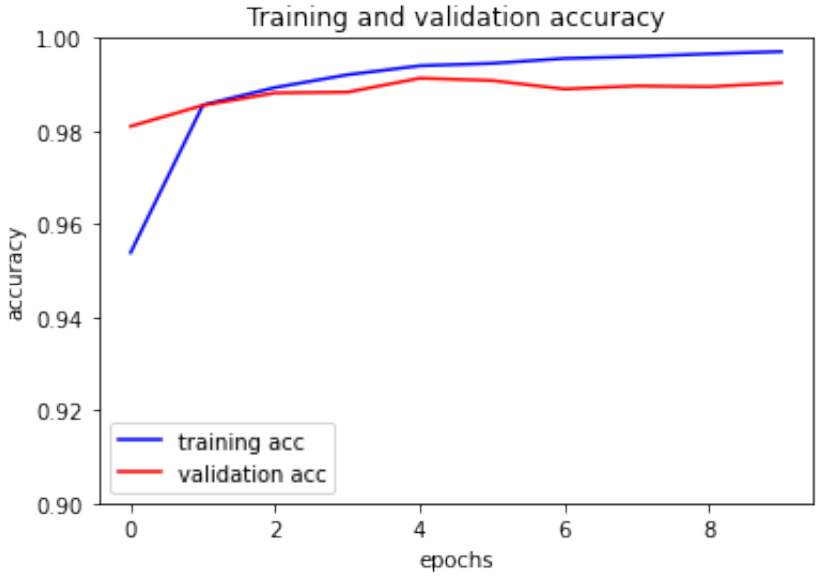
```
188/188 [==============================] - 1s 5ms/step - loss: 0.0381 - acc
: 0.9903
0.9903333187103271
```

Since validation accuracy for A network is 0.986 & validation accuracy for B network is 0.9903, network B works slightly better than network A. But there could have some bias with the validation set so in order to precise the result, we have to consider doing cross validation or we have to analyze from test set. But in terms of the validation accuracy for this certain validation set, we can say network B is better.

3.b)

In [14]:
```
import matplotlib.pyplot as plt

accuracy=B_result.history['acc']
accuracy_val=B_result.history['val_acc']

plt.clf()
plt.plot(accuracy,'b',label='training acc')
plt.plot(accuracy_val,'b', color='r', label='validation acc' )

plt.title('Training and validation accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.ylim(0.9,1)
plt.legend()
plt.show()
```

```
/var/folders/9r/7f1bw3q15yz7435178lsgb5w0000gp/T/ipykernel_3857/2743705492.
py:8: UserWarning: color is redundantly defined by the 'color' keyword argu
ment and the fmt string "b" (-> color=(0.0, 0.0, 1.0, 1)). The keyword argu
ment will take precedence.
  plt.plot(accuracy_val,'b', color='r', label='validation acc' )
```



In first epoch, train accuracy is about 0.95 and validation accuracy is 0.981. When the epoch goes on, train accuracy and validation accuracy converges to 1 & about 0.99 respectively. We have to evaluate the model to test set to find whether the model is overfitted or not but in this graph with the validation accuracy, the model seems to be not overfitted or very slightly overfitted and it seems to converge well.

3.c)

In [15]:
```
B_net.evaluate(X_test, y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 0.0306 - acc
: 0.9915
```

Out[15]:
```
[0.030552800744771957, 0.9915000200271606]
```

Test accuracy is about 0.9915. Since the final train accuracy is 0.9971, we can say the model is very slightly overfitted or not overfitted to train set.

3.d)

First, add dropout function to the model between the layers to prevent overfitting. Second, add regularizer in Convolution layer and it will prevent overfitting problem. Also, Callback that stops learning when validation accuracy was not improved for several epochs would be helpful to prevent overfitting. Lastly, when we take a look at the B_network, there are many parameters in Dense layer after Flatten function. So, we can use GlobalAveragePooling2D or GlobalMaxPooling2D instead of Flatten function and it makes the number of parameter much smaller than before.

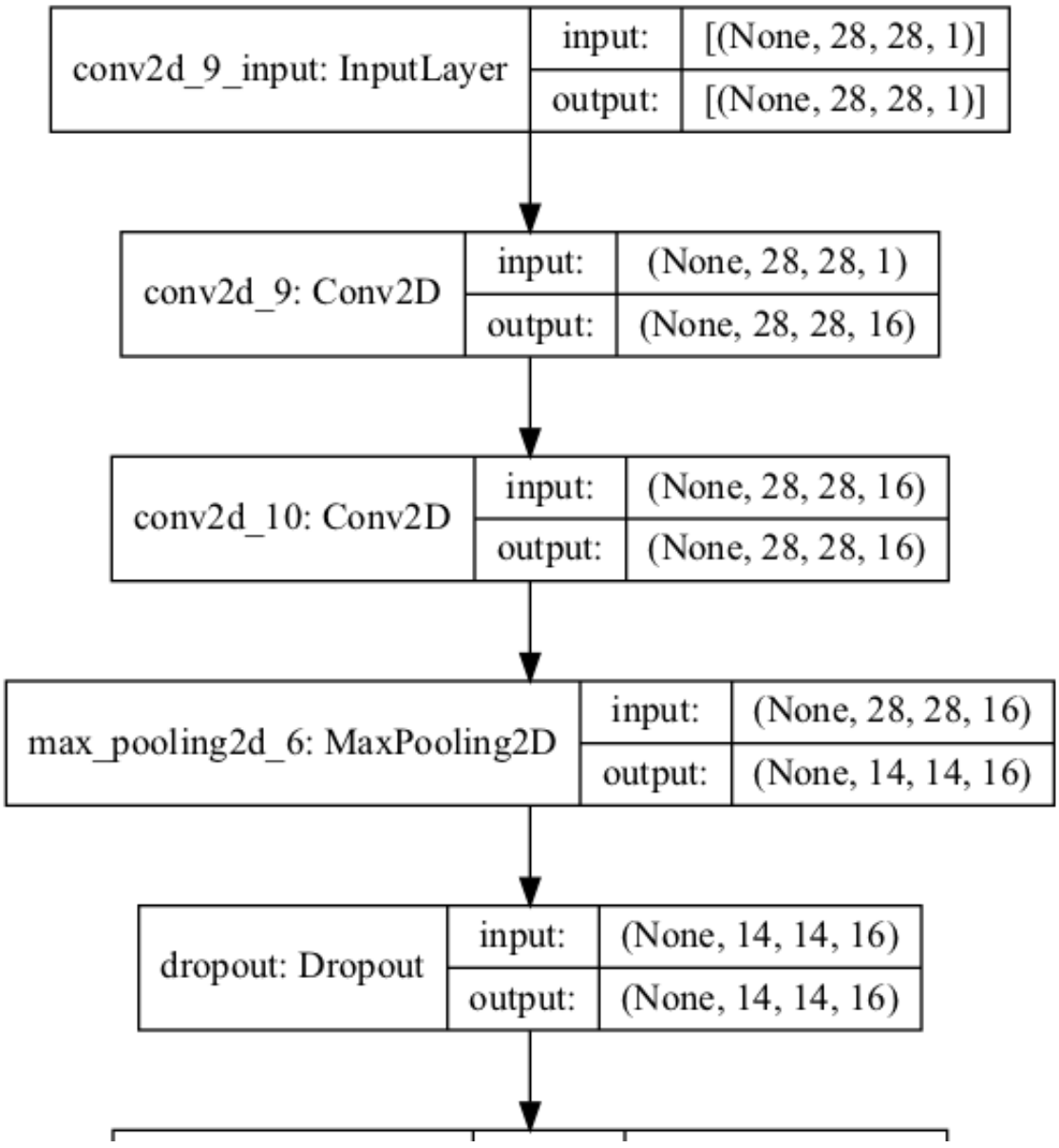
For example in below, I add some Dropout layer between the Convolution layers and everything else is same with B_net.

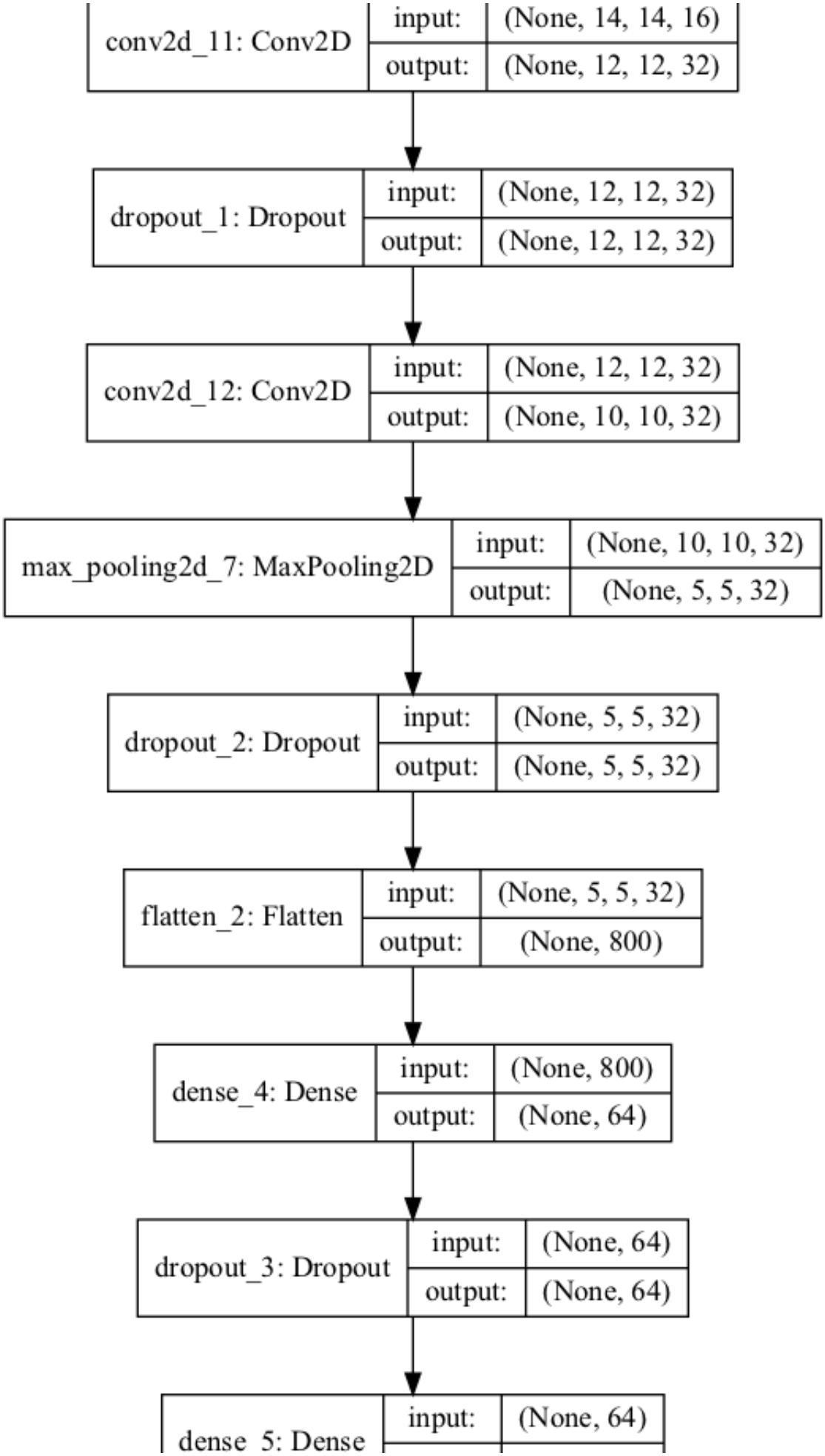
In [17]:
```
from tensorflow.keras.layers import Dropout

B_net_adj=Sequential()
B_net_adj.add(Conv2D(16,(3,3),activation='relu',input_shape=(28,28,1),padd
B_net_adj.add(Conv2D(16,(3,3),activation='relu',padding='same'))
B_net_adj.add(MaxPooling2D((2,2), strides=2))
B_net_adj.add(Dropout(0.2))
B_net_adj.add(Conv2D(32,(3,3),activation='relu'))
B_net_adj.add(Dropout(0.2))
B_net_adj.add(Conv2D(32,(3,3),activation='relu'))
B_net_adj.add(MaxPooling2D((2,2), strides=2))
B_net_adj.add(Dropout(0.3))
B_net_adj.add(Flatten())
B_net_adj.add(Dense(64,activation='relu'))
B_net_adj.add(Dropout(0.4))
B_net_adj.add(Dense(10,activation='softmax'))

B_net_adj.summary()
plot_model(B_net_adj, show_shapes=True)
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 28, 28, 16)        160
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 28, 16)        2320
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 16)        0
_________________________________________________________________
```

```
conv2d_11 (Conv2D)           (None, 12, 12, 32)        4640
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 10, 10, 32)        9248
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 5, 5, 32)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 800)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                51264
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650
=================================================================
Total params: 68,282
Trainable params: 68,282
Non-trainable params: 0
_________________________________________________________________
```

Out[17]:




conv2d_11: Conv2D
input: (None, 14, 14, 16)
output: (None, 12, 12, 32)
dropout_1: Dropout
input: (None, 12, 12, 32)
output: (None, 12, 12, 32)
conv2d_12: Conv2D
input: (None, 12, 12, 32)
output: (None, 10, 10, 32)
max_pooling2d_7: MaxPooling2D
input: (None, 10, 10, 32)
output: (None, 5, 5, 32)
dropout_2: Dropout
input: (None, 5, 5, 32)
output: (None, 5, 5, 32)
flatten_2: Flatten
input: (None, 5, 5, 32)
output: (None, 800)
dense_4: Dense
input: (None, 800)
output: (None, 64)
dropout_3: Dropout
input: (None, 64)
output: (None, 64)
dense_5: Dense
input: (None, 64)

	output:	(None, 10)

In [18]:
```
B_net_adj.compile(loss='categorical_crossentropy',optimizer='adam', metrics
B_result_adj=B_net_adj.fit(X_train,y_train,batch_size=32,epochs=10,validat
```

```
Epoch 1/10
2021-12-02 00:54:11.151232: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
1688/1688 [==============================] - ETA: 0s - loss: 0.3370 - acc:
0.8929
2021-12-02 00:54:27.624037: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
1688/1688 [==============================] - 18s 10ms/step - loss: 0.3370 -
acc: 0.8929 - val_loss: 0.0668 - val_acc: 0.9797
Epoch 2/10
1688/1688 [==============================] - 17s 10ms/step - loss: 0.1221 -
acc: 0.9641 - val_loss: 0.0542 - val_acc: 0.9830
Epoch 3/10
1688/1688 [==============================] - 17s 10ms/step - loss: 0.0962 -
acc: 0.9716 - val_loss: 0.0419 - val_acc: 0.9870
Epoch 4/10
1688/1688 [==============================] - 17s 10ms/step - loss: 0.0791 -
acc: 0.9774 - val_loss: 0.0385 - val_acc: 0.9892
Epoch 5/10
1688/1688 [==============================] - 18s 10ms/step - loss: 0.0717 -
acc: 0.9788 - val_loss: 0.0404 - val_acc: 0.9883
Epoch 6/10
1688/1688 [==============================] - 18s 10ms/step - loss: 0.0656 -
acc: 0.9810 - val_loss: 0.0386 - val_acc: 0.9875
Epoch 7/10
1688/1688 [==============================] - 17s 10ms/step - loss: 0.0598 -
acc: 0.9824 - val_loss: 0.0350 - val_acc: 0.9888
Epoch 8/10
1688/1688 [==============================] - 17s 10ms/step - loss: 0.0578 -
acc: 0.9831 - val_loss: 0.0321 - val_acc: 0.9892
Epoch 9/10
1688/1688 [==============================] - 17s 10ms/step - loss: 0.0541 -
acc: 0.9847 - val_loss: 0.0381 - val_acc: 0.9883
Epoch 10/10
1688/1688 [==============================] - 17s 10ms/step - loss: 0.0534 -
acc: 0.9842 - val_loss: 0.0361 - val_acc: 0.9880
```

In [20]:
```
B_net_adj.evaluate(X_test, y_test)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 0.0219 - acc
: 0.9931
```

Out[20]:
```
[0.021898901090025902, 0.9931000471115112]
```

The train accuracy in last epoch is about 0.984 but test accuracy is about 0.9931. It is better than B_net in above! The Dropout function prevents overfitting and results the best output.

Lastly, I add some l2 regularizer in Dense layer after Flatten. Also I add Callback that stops learning if validation acc was not improved for 4 epochs.

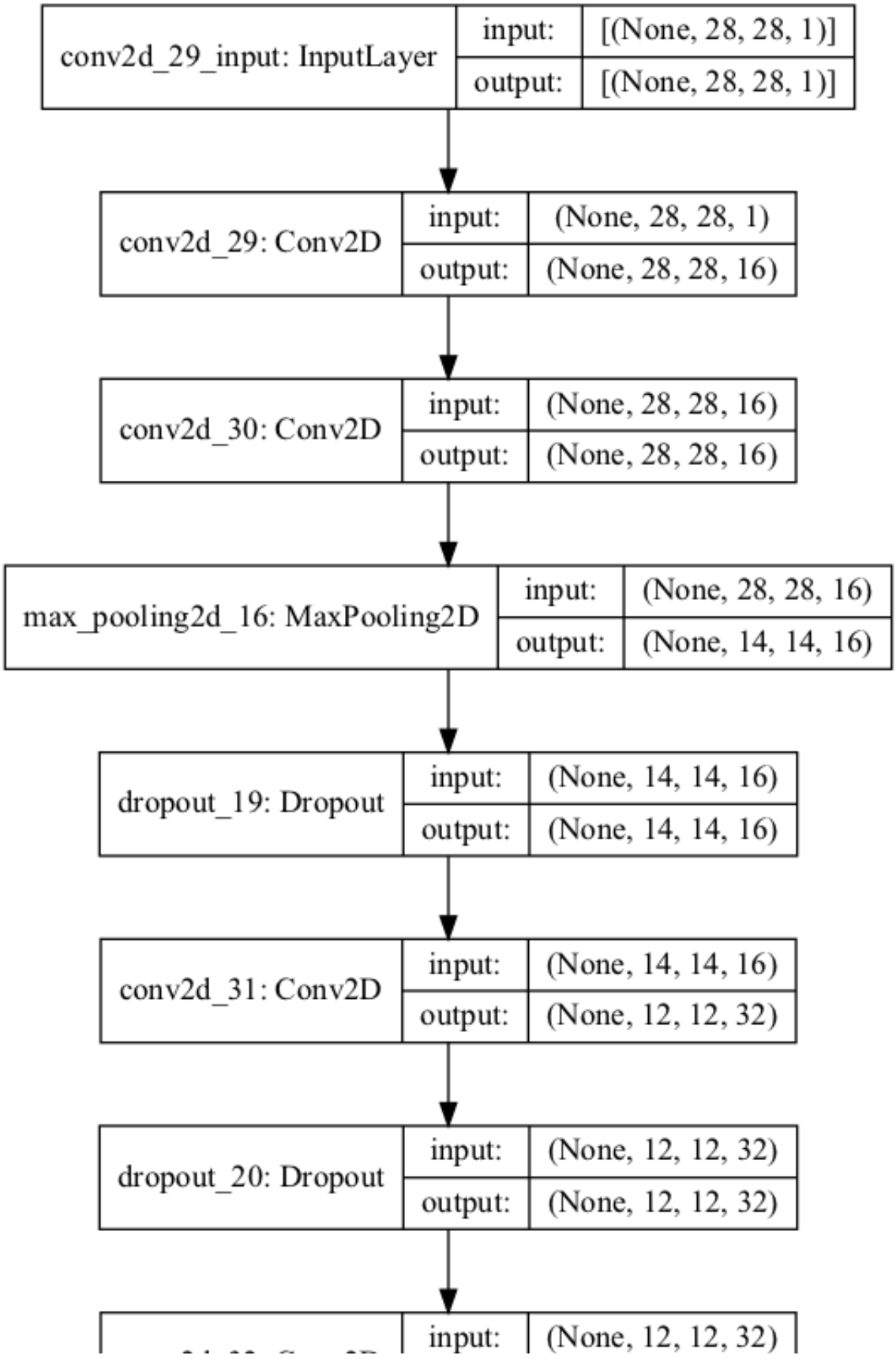
In [33]:

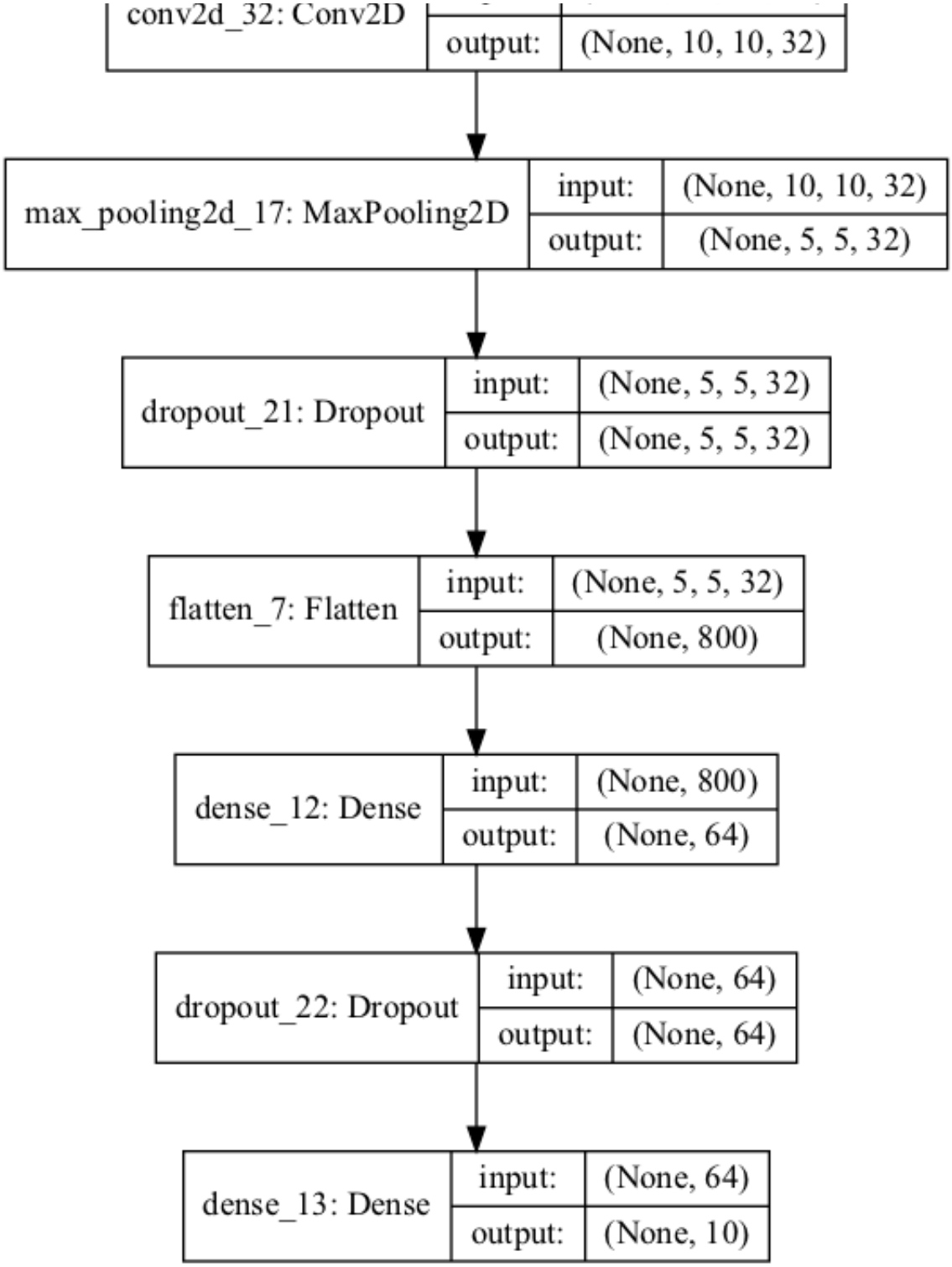
```
import keras

B_net_adj2=Sequential()
B_net_adj2.add(Conv2D(16,(3,3),activation='relu',input_shape=(28,28,1),pad
B_net_adj2.add(Conv2D(16,(3,3),activation='relu',padding='same'))
B_net_adj2.add(MaxPooling2D((2,2), strides=2))
B_net_adj2.add(Dropout(0.2))
B_net_adj2.add(Conv2D(32,(3,3),activation='relu'))
B_net_adj2.add(Dropout(0.2))
B_net_adj2.add(Conv2D(32,(3,3),activation='relu'))
B_net_adj2.add(MaxPooling2D((2,2), strides=2))
B_net_adj2.add(Dropout(0.3))
B_net_adj2.add(Flatten())
B_net_adj2.add(Dense(64,activation='relu',kernel_regularizer=keras.regulari
B_net_adj2.add(Dropout(0.4))
B_net_adj2.add(Dense(10,activation='softmax'))

B_net_adj2.summary()
plot_model(B_net_adj2, show_shapes=True)
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 28, 28, 16)        160
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 28, 28, 16)        2320
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 14, 14, 16)        0
_________________________________________________________________
dropout_19 (Dropout)         (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 12, 12, 32)        4640
_________________________________________________________________
dropout_20 (Dropout)         (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 10, 10, 32)        9248
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 5, 5, 32)          0
_________________________________________________________________
dropout_21 (Dropout)         (None, 5, 5, 32)          0
_________________________________________________________________
flatten_7 (Flatten)          (None, 800)               0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                51264
_________________________________________________________________
dropout_22 (Dropout)         (None, 64)                0
```

```
_________________________________________________________________
dense_13 (Dense)             (None, 10)                650       
=================================================================
Total params: 68,282
Trainable params: 68,282
Non-trainable params: 0
_________________________________________________________________
```

Out[33]:





In [36]:
```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
callback_list=[EarlyStopping(monitor='val_acc',patience=4),ModelCheckpoint
```

In [37]:
```
B_net_adj2.compile(loss='categorical_crossentropy',optimizer='adam', metri
B_result_adj2=B_net_adj2.fit(X_train,y_train,batch_size=32,epochs=20,callb
```

```
Epoch 1/20
   3/1688 [..............................] - ETA: 42s - loss: 0.3177 - acc:
0.9479
```

```
2021-12-02 01:17:05.890591: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
1688/1688 [==============================] - ETA: 0s - loss: 0.2544 - acc:
0.9587
2021-12-02 01:17:22.926795: I tensorflow/core/grappler/optimizers/custom_gr
aph_optimizer_registry.cc:112] Plugin optimizer for device_type GPU is enab
led.
```

```
1688/1688 [==============================] - 18s 11ms/step - loss: 0.2544 -
acc: 0.9587 - val_loss: 0.1750 - val_acc: 0.9837
Epoch 2/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.2374 -
acc: 0.9642 - val_loss: 0.1692 - val_acc: 0.9832
Epoch 3/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.2246 -
acc: 0.9663 - val_loss: 0.1648 - val_acc: 0.9845
Epoch 4/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.2106 -
acc: 0.9703 - val_loss: 0.1588 - val_acc: 0.9848
Epoch 5/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.2004 -
acc: 0.9718 - val_loss: 0.1566 - val_acc: 0.9850
Epoch 6/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1967 -
acc: 0.9726 - val_loss: 0.1439 - val_acc: 0.9878
Epoch 7/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1920 -
acc: 0.9741 - val_loss: 0.1450 - val_acc: 0.9878
Epoch 8/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1882 -
acc: 0.9750 - val_loss: 0.1448 - val_acc: 0.9860
Epoch 9/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1862 -
acc: 0.9748 - val_loss: 0.1433 - val_acc: 0.9872
Epoch 10/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1797 -
acc: 0.9761 - val_loss: 0.1312 - val_acc: 0.9888
Epoch 11/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1786 -
acc: 0.9768 - val_loss: 0.1377 - val_acc: 0.9887
Epoch 12/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1771 -
acc: 0.9771 - val_loss: 0.1383 - val_acc: 0.9878
Epoch 13/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1729 -
acc: 0.9775 - val_loss: 0.1237 - val_acc: 0.9905
Epoch 14/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1730 -
acc: 0.9786 - val_loss: 0.1326 - val_acc: 0.9885
Epoch 15/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1712 -
acc: 0.9777 - val_loss: 0.1333 - val_acc: 0.9885
Epoch 16/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1694 -
acc: 0.9778 - val_loss: 0.1222 - val_acc: 0.9897
Epoch 17/20
1688/1688 [==============================] - 18s 11ms/step - loss: 0.1687 -
acc: 0.9787 - val_loss: 0.1228 - val_acc: 0.9898
```

In [38]:

```
B_net_adj2.evaluate(X_test, y_test)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.1113 - acc
: 0.9934
```

Out[38]:
```
[0.11134836822748184, 0.9934000372886658]
```

The train accuracy for last epoch is 0.979 but test accuracy is 0.9934. It is slightly better than above model!