In [2]:

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
Using TensorFlow backend.
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

Model 1

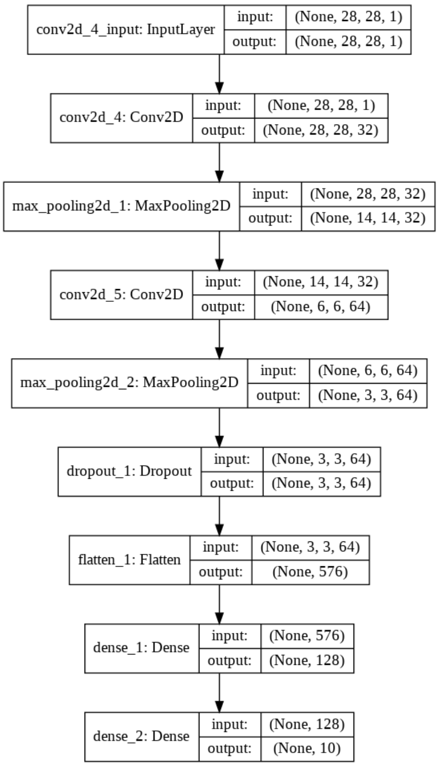
In [0]:

```
Adam=keras.optimizers.Adam()
model=Sequential()
model.add(Conv2D(32,kernel_size=(5,5),padding='same',activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,kernel_size=(3,3),strides=(2,2),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(Dense(num_classes,activation='softmax'))
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=Adam,
              metrics=['accuracy'])
```

In [0]:

```
from keras.utils import plot_model
from keras.utils import plot_model
from IPython.display import Image
plot_model(model, show_shapes=True, show_layer_names=True, to_file='model1.png')
Image(retina=True, filename='model1.png')
```

Out[0]:



In [4]:

```
history1=model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 41s 686us/step - loss: 0.3621 - accuracy: 0.8866 -
val_loss: 0.0748 - val_accuracy: 0.9764
Epoch 2/12
60000/60000 [==============================] - 40s 670us/step - loss: 0.1212 - accuracy: 0.9620 -
val_loss: 0.0495 - val_accuracy: 0.9857
Epoch 3/12
60000/60000 [==============================] - 40s 672us/step - loss: 0.0927 - accuracy: 0.9714 -
val_loss: 0.0447 - val_accuracy: 0.9856
Epoch 4/12
60000/60000 [==============================] - 40s 669us/step - loss: 0.0797 - accuracy: 0.9758 -
val_loss: 0.0362 - val_accuracy: 0.9872
Epoch 5/12
60000/60000 [==============================] - 40s 668us/step - loss: 0.0691 - accuracy: 0.9778 -
val_loss: 0.0327 - val_accuracy: 0.9893
Epoch 6/12
60000/60000 [==============================] - 40s 667us/step - loss: 0.0632 - accuracy: 0.9796 -
val_loss: 0.0296 - val_accuracy: 0.9899
Epoch 7/12
60000/60000 [==============================] - 40s 667us/step - loss: 0.0586 - accuracy: 0.9809 -
val_loss: 0.0286 - val_accuracy: 0.9903
Epoch 8/12
60000/60000 [==============================] - 40s 664us/step - loss: 0.0547 - accuracy: 0.9825 -
val_loss: 0.0268 - val_accuracy: 0.9901
Epoch 9/12
60000/60000 [==============================] - 40s 666us/step - loss: 0.0514 - accuracy: 0.9839 -
val_loss: 0.0255 - val_accuracy: 0.9917
Epoch 10/12
Epoch 10/12
60000/60000 [==============================] - 40s 667us/step - loss: 0.0463 - accuracy: 0.9849 -
val_loss: 0.0251 - val_accuracy: 0.9919
Epoch 11/12
60000/60000 [==============================] - 40s 668us/step - loss: 0.0458 - accuracy: 0.9852 -
val_loss: 0.0233 - val_accuracy: 0.9921
Epoch 12/12
60000/60000 [==============================] - 40s 666us/step - loss: 0.0415 - accuracy: 0.9863 -
val_loss: 0.0232 - val_accuracy: 0.9918
Test loss: 0.02322798852327833
Test accuracy: 0.9918000102043152
```

Model 2

In [0]:

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

inputshape = Input(shape=input_shape)
n01=Conv2D(64,kernel_size=(1,1),activation='relu')(inputshape)
n11=Conv2D(128,kernel_size=(3,3),activation='relu')(n01)
n11f=Flatten()(n11)
n02=MaxPooling2D(pool_size=(2,2))(inputshape)
n12=Conv2D(128,kernel_size=(1,1),activation='relu')(n02)
n12f=Flatten()(n12)
n13=Conv2D(32,kernel_size=(5,5),strides=(2,2),activation='relu')(inputshape)
n13f=Flatten()(n13)
final= concatenate([n11f,n12f,n13f])
l1=Dense(256, activation='relu')(final)

l2=Dropout(0.25)(l1)
l3=Dense(64, activation='relu')(l2)
output=Dense(num_classes, activation='softmax')(l3)
```

In [6]:

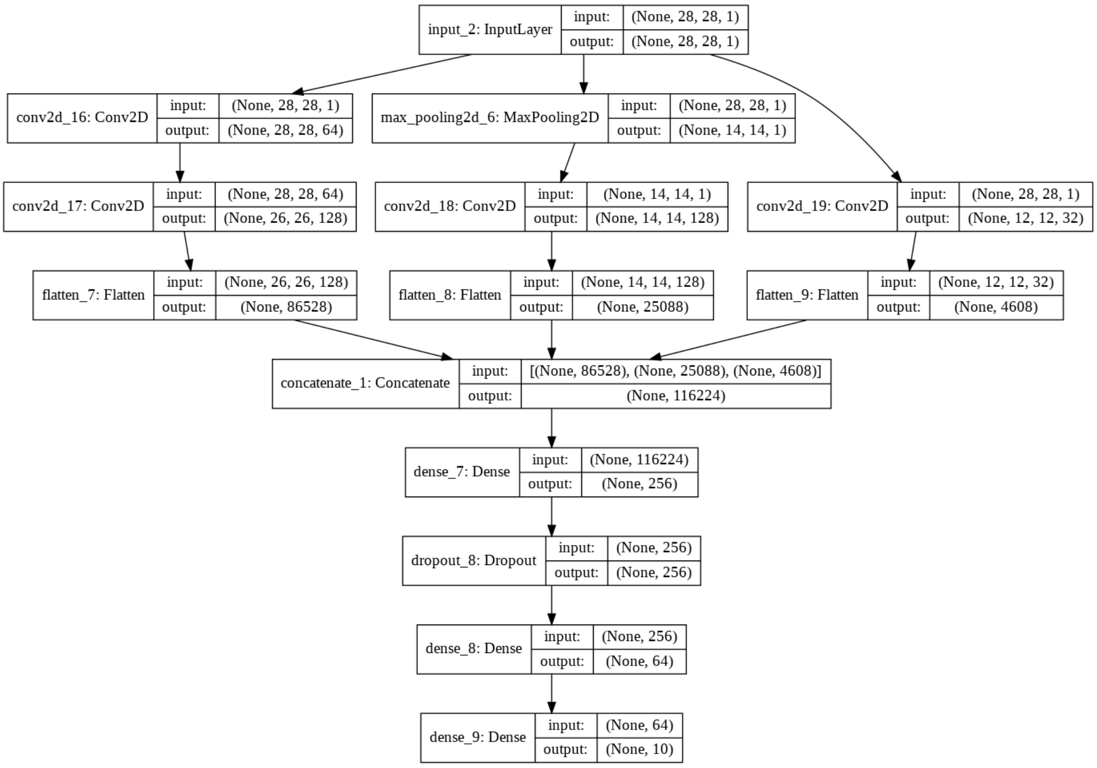
```python
import tensorflow as tf
from keras.models import Model


from time import time
#from tf.keras import metrics

model=Model(inputs=[inputshape],outputs=output)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
print(model.summary())
```

```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 28, 28, 1)    0
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 28, 28, 64)   128         input_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 14, 14, 1)    0           input_1[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)               (None, 26, 26, 128)  73856       conv2d_3[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)               (None, 14, 14, 128)  256         max_pooling2d_3[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, 12, 12, 32)   832         input_1[0][0]
__________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 86528)        0           conv2d_4[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 25088)        0           conv2d_5[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 4608)         0           conv2d_6[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 116224)       0           flatten_2[0][0]
                                                                 flatten_3[0][0]
                                                                 flatten_4[0][0]
                                                                 flatten_4[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 256)          29753600    concatenate_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 256)          0           dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 64)           16448       dropout_2[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 10)           650         dense_4[0][0]
==================================================================================================
Total params: 29,845,770
Trainable params: 29,845,770
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

In [0]:

```python
from keras.utils import plot_model
from IPython.display import Image
plot_model(model, show_shapes=True, show_layer_names=True, to_file='model2.png')
Image(retina=True, filename='model2.png')
```

Out[0]:



In [7]:

```python
history2 =model.fit(x_train,y_train, batch_size=batch_size, epochs=5, verbose=1, validation_data=(x
_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 659s 11ms/step - loss: 0.1679 - accuracy: 0.9488 -
val_loss: 0.0590 - val_accuracy: 0.9813
Epoch 2/5
60000/60000 [==============================] - 657s 11ms/step - loss: 0.0541 - accuracy: 0.9829 -
val_loss: 0.0470 - val_accuracy: 0.9854
Epoch 3/5
60000/60000 [==============================] - 658s 11ms/step - loss: 0.0317 - accuracy: 0.9894 -
val_loss: 0.0484 - val_accuracy: 0.9843
Epoch 4/5
60000/60000 [==============================] - 662s 11ms/step - loss: 0.0228 - accuracy: 0.9928 -
val_loss: 0.0481 - val_accuracy: 0.9843
```

```
val_loss: 0.0481 - val_accuracy: 0.9843
Epoch 5/5
60000/60000 [==============================] - 666s 11ms/step - loss: 0.0172 - accuracy: 0.9942 -
val_loss: 0.0533 - val_accuracy: 0.9856
Test loss: 0.05329901970065039
Test accuracy: 0.9855999946594238
```

Model 3

In [0]:

```
from keras.models import Model
from keras.layers import Input , concatenate
from keras.layers import BatchNormalization

inputshape = Input(shape=input_shape)
n01=Conv2D(128,kernel_size=(3,3),activation='relu')(inputshape)
nf1=Flatten()(n01)
n12=Conv2D(64,kernel_size=(3,3),activation='relu')(n01)
n13=Conv2D(32,kernel_size=(3,3),strides=(2,2),activation='relu')(n12)
nf2=Flatten()(n13)
final= concatenate([nf1,nf2])

l1=Dense(128, activation='relu')(final)

l2=Dropout(0.25)(l1)

output=Dense(num_classes, activation='softmax')(l2)
```

In [9]:

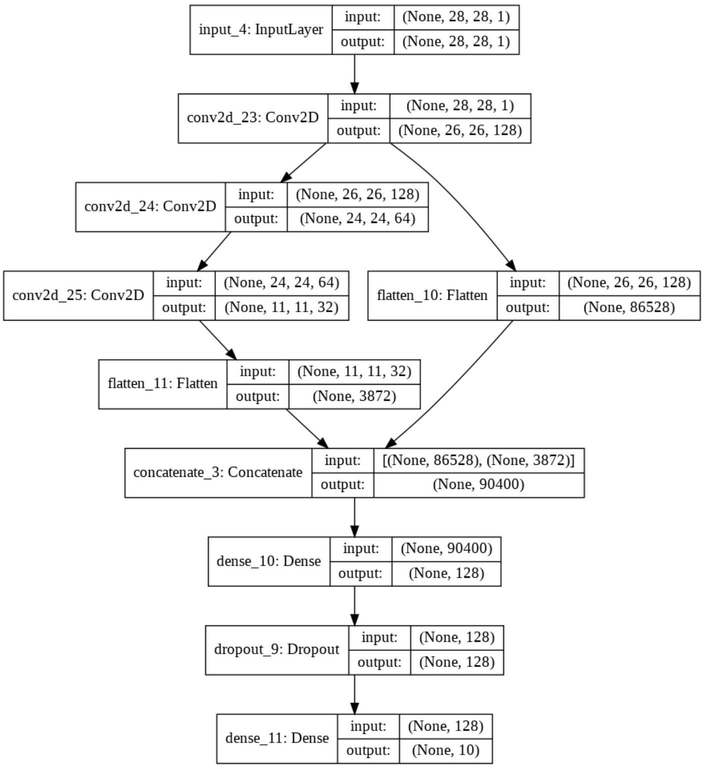
```
model=Model(inputs=[inputshape],outputs=output)
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
print(model.summary())
```

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 28, 28, 1)    0
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, 26, 26, 128)  1280        input_2[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, 24, 24, 64)   73792       conv2d_7[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, 11, 11, 32)   18464       conv2d_8[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 86528)        0           conv2d_7[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 3872)         0           conv2d_9[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 90400)        0           flatten_5[0][0]
                                                                 flatten_6[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 128)          11571328    concatenate_2[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 128)          0           dense_6[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 10)           1290        dropout_3[0][0]
==================================================================================================
Total params: 11,666,154
Trainable params: 11,666,154
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

In [0]:

```
from keras.utils import plot_model
from IPython.display import Image
plot_model(model, show_shapes=True, show_layer_names=True, to_file='model3.png')
Image(retina=True, filename='model3.png')
```

Out[0]:


input_4: InputLayer | input: (None, 28, 28, 1) | output: (None, 28, 28, 1)
conv2d_23: Conv2D | input: (None, 28, 28, 1) | output: (None, 26, 26, 128)
conv2d_24: Conv2D | input: (None, 26, 26, 128) | output: (None, 24, 24, 64)
conv2d_25: Conv2D | input: (None, 24, 24, 64) | output: (None, 11, 11, 32)
flatten_10: Flatten | input: (None, 26, 26, 128) | output: (None, 86528)
flatten_11: Flatten | input: (None, 11, 11, 32) | output: (None, 3872)
concatenate_3: Concatenate | input: [(None, 86528), (None, 3872)] | output: (None, 90400)
dense_10: Dense | input: (None, 90400) | output: (None, 128)
dropout_9: Dropout | input: (None, 128) | output: (None, 128)
dense_11: Dense | input: (None, 128) | output: (None, 10)

In [10]:

```
history3 =model.fit(x_train,y_train, batch_size=batch_size, epochs=5, verbose=1, validation_data=(x
_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 528s 9ms/step - loss: 0.1611 - accuracy: 0.9497 - v
al_loss: 0.0459 - val_accuracy: 0.9834
Epoch 2/5
60000/60000 [==============================] - 528s 9ms/step - loss: 0.0497 - accuracy: 0.9846 - v
al_loss: 0.0480 - val_accuracy: 0.9836
Epoch 3/5
60000/60000 [==============================] - 527s 9ms/step - loss: 0.0307 - accuracy: 0.9904 - v
al_loss: 0.0350 - val_accuracy: 0.9880
Epoch 4/5
60000/60000 [==============================] - 528s 9ms/step - loss: 0.0227 - accuracy: 0.9933 - v
al_loss: 0.0359 - val_accuracy: 0.9891
Epoch 5/5
60000/60000 [==============================] - 528s 9ms/step - loss: 0.0154 - accuracy: 0.9949 - v
al_loss: 0.0390 - val_accuracy: 0.9893
Test loss: 0.03904903102653225
Test accuracy: 0.989300012588501
```

In [0]:

```
model_json = model.to_json()
with open("drive/My Drive/model3.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("drive/My Drive/model3.h5")
print("Saved model to disk")
```

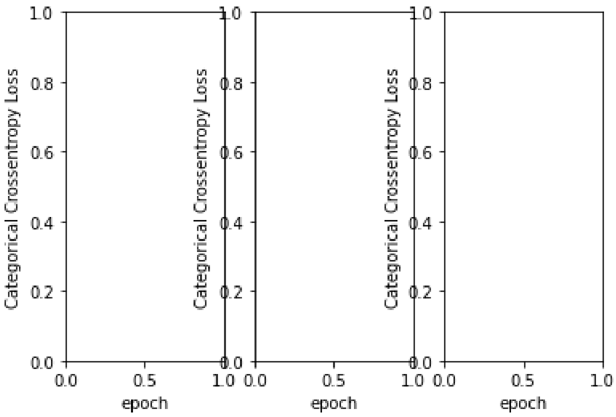
In [11]:

```
import matplotlib.pyplot as plt
fig,(ax1,ax2,ax3) = plt.subplots(1,3)
ax1.set_xlabel('epoch') ; ax1.set_ylabel('Categorical Crossentropy Loss')
ax2.set_xlabel('epoch') ; ax2.set_ylabel('Categorical Crossentropy Loss')
```

```
ax2.set_xlabel('epoch') ; ax2.set_ylabel('Categorical Crossentropy Loss')
ax3.set_xlabel('epoch') ; ax3.set_ylabel('Categorical Crossentropy Loss')
```

Out[11]:

```
Text(0, 0.5, 'Categorical Crossentropy Loss')
```



In [0]:

```
def plt_dynamic1(x, y, y_1, ax, title, colors=['b']):
    #fig = ax.figure()
    ax.plot(x, y, 'b', label="Train Loss")
    ax.plot(x, y_1, 'r', label="Test Loss")
    ax.set_title(title)

    #plt.legend()
    fig.canvas.draw()
    fig.savefig('my_figure.png')
```

In [0]:

```
x = list(range(1,epochs+1))
vy = history1.history['val_loss']
ty = history1.history['loss']
plt_dynamic1(x, vy, ty, ax1, "Model1", colors=['b'])
```

In [0]:

```
x = list(range(1,6))
vy = history2.history['val_loss']
ty = history2.history['loss']
plt_dynamic1(x, vy, ty, ax2, "Model2", colors=['b'])
```

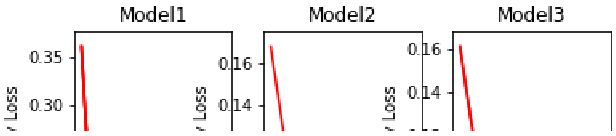
In [0]:

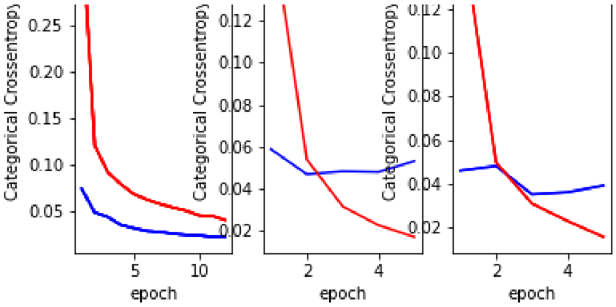
```
x = list(range(1,6))
vy = history3.history['val_loss']
ty = history3.history['loss']
plt_dynamic1(x, vy, ty, ax3, "Model3", colors=['b'])
```

In [26]:

```
from IPython.display import Image
Image('my_figure.png')
```

Out[26]:





In [29]:

```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names=['Model','Test loss','Test Accuracy']
x.add_row(["Model1","0.023","0.991"])
x.add_row(["Model2","0.053","0.985"])
x.add_row(["Model3","0.039","0.989"])
print(x)
```

```
+---------+-----------+---------------+
| Model   | Test loss | Test Accuracy |
+---------+-----------+---------------+
| Model1  |   0.023   |     0.991     |
| Model2  |   0.053   |     0.985     |
| Model3  |   0.039   |     0.989     |
+---------+-----------+---------------+
```