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

```
Using TensorFlow backend.
```

In [2]:
```
!pip install PTable
from prettytable import PrettyTable
table = PrettyTable()
table.field_names = ["model_name", "Test Accuracy", "Description"]
```

```
Requirement already satisfied: PTable in /usr/local/lib/python3.6/dist-packages (0.9.2)
```

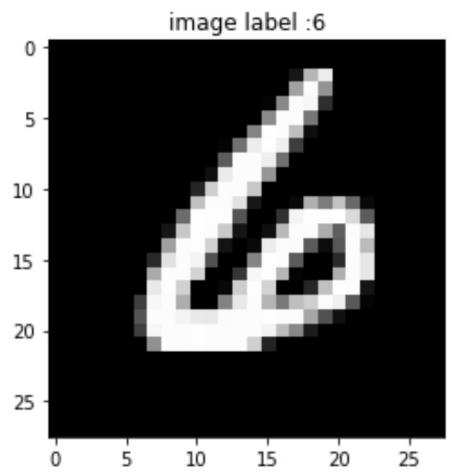
In [0]:
```
def plt_epoch_vs_loss(epochs, val_loss, train_loss, axis):
    axis.plot(epochs, val_loss, 'b', label="Validation Loss")
    axis.plot(epochs, train_loss, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
    fig.canvas.draw()
```

In [0]:
```
# the data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```

In [5]:
```
print("Number of train examples :", X_train.shape[0], "and each image is of shape (%d, %d)"%(X_train.shape[1], X_train.
shape[2]))
print("Number of test examples :", X_test.shape[0], "and each image is of shape (%d, %d)"%(X_test.shape[1], X_test.shap
e[2]))
```

```
Number of train examples : 60000 and each image is of shape (28, 28)
Number of test examples : 10000 and each image is of shape (28, 28)
```

In [6]:
```
#displaying random image and label
plt.imshow(X_train[2345], cmap='gray')
plt.title("image label :"+str(y_train[2345]))
plt.show()
```



In [0]:
```
#reshaping x_train and x_test tensor for input in model
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], X_train.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)
#converting class numbers in 10 dimensional vector
Y_train = keras.utils.np_utils.to_categorical(y_train, 10)
Y_test = keras.utils.np_utils.to_categorical(y_test, 10)
```

In [8]:
```
from keras.models import Sequential
model_3layer = Sequential()
model_3layer.add(keras.layers.Conv2D(filters=3,kernel_size=(5,5),strides=(1,1),padding='same',activation='relu',kernel_
initializer='he_normal',\
                                     input_shape = (X_train.shape[1],X_train.shape[2],1)))
model_3layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_3layer.add(keras.layers.Conv2D(filters=3,kernel_size=(3,3),kernel_initializer='he_normal',activation='relu'))
model_3layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_3layer.add(keras.layers.Conv2D(filters=3,kernel_size=(2,2),kernel_initializer='he_normal',activation='relu'))
model_3layer.add(keras.layers.Flatten())
model_3layer.add(keras.layers.Dense(128, activation='relu',kernel_initializer='he_normal'))
model_3layer.add(keras.layers.Dense(10,activation='softmax'))
```

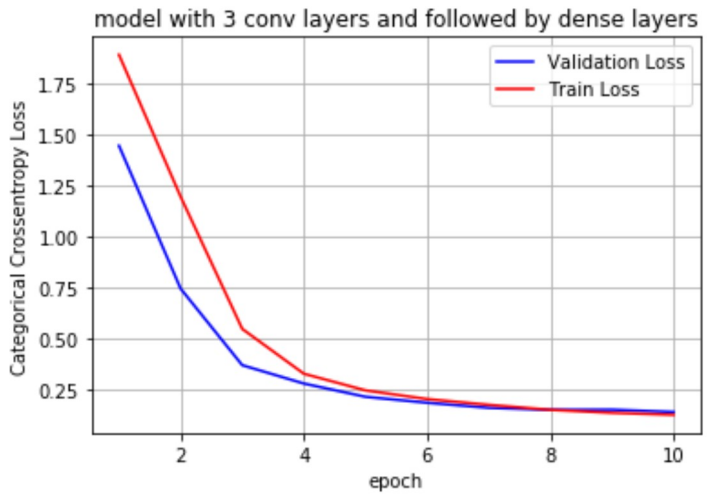
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The name tf.ge
t_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.p
laceholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4479: The name tf.
truncated_normal is deprecated. Please use tf.random.truncated_normal instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4267: The name tf.
nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432: The name tf.
random_uniform is deprecated. Please use tf.random.uniform instead.
```

In [9]:
```
model_3layer.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_3layer.fit(X_train, Y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, Y_test))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer i
s deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576: The name tf.
log is deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (f
rom tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.
assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.
assign is deprecated. Please use tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3005: The name tf.
Session is deprecated. Please use tf.compat.v1.Session instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/10
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.g
et_default_session is deprecated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.C
onfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.g
lobal_variables is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.i
s_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.v
ariables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

60000/60000 [==============================] - 30s 494us/step - loss: 1.8876 - acc: 0.4013 - val_loss: 1.4419 - val_a
cc: 0.4992
Epoch 2/10
60000/60000 [==============================] - 29s 491us/step - loss: 1.1926 - acc: 0.6024 - val_loss: 0.7421 - val_a
cc: 0.7670
Epoch 3/10
60000/60000 [==============================] - 30s 505us/step - loss: 0.5447 - acc: 0.8271 - val_loss: 0.3669 - val_a
cc: 0.8845
Epoch 4/10
60000/60000 [==============================] - 30s 498us/step - loss: 0.3250 - acc: 0.8956 - val_loss: 0.2769 - val_a
cc: 0.9103
Epoch 5/10
60000/60000 [==============================] - 30s 501us/step - loss: 0.2434 - acc: 0.9229 - val_loss: 0.2120 - val_a
cc: 0.9319
Epoch 6/10
60000/60000 [==============================] - 30s 497us/step - loss: 0.2009 - acc: 0.9360 - val_loss: 0.1829 - val_a
cc: 0.9413
Epoch 7/10
60000/60000 [==============================] - 30s 497us/step - loss: 0.1726 - acc: 0.9449 - val_loss: 0.1581 - val_a
cc: 0.9519
Epoch 8/10
60000/60000 [==============================] - 30s 493us/step - loss: 0.1482 - acc: 0.9536 - val_loss: 0.1482 - val_a
cc: 0.9532
Epoch 9/10
60000/60000 [==============================] - 30s 496us/step - loss: 0.1335 - acc: 0.9580 - val_loss: 0.1490 - val_a
cc: 0.9558
Epoch 10/10
60000/60000 [==============================] - 30s 497us/step - loss: 0.1238 - acc: 0.9610 - val_loss: 0.1377 - val_a
cc: 0.9567
```

In [10]:
```
score = model_3layer.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,axis = plt.subplots(1,1)
axis.set_title("model with 3 conv layers and followed by dense layers")
axis.set_xlabel('epoch') ; axis.set_ylabel('Categorical Crossentropy Loss')
plt_epoch_vs_loss(list(range(1,11)), history.history['val_loss'], history.history['loss'], axis)
table.add_row(["model_3convlayer",score[1],"kernel-5X5X3(relu, he_normal),\n maxpool-(2,2),\n \
kernel-3X3X3(relu, he_normal),\n maxpool-(2,2),\n \
kernel-2X2X3(relu, he_normal),\n \
flatten,\n Dense-128(relu, he_normal),\n softmax-10"])
```

```
Test score: 0.13773354607648217
Test accuracy: 0.9567
```

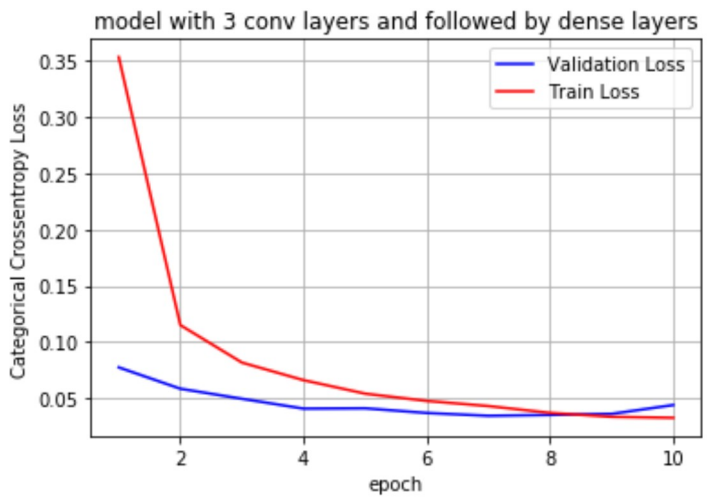


In [11]:
```
from keras.models import Sequential
model_3layer = Sequential()
model_3layer.add(keras.layers.Conv2D(filters=8,kernel_size=(5,5),strides=(1,1),padding='same',activation='relu',kernel_
initializer='he_normal',\
                                     input_shape = (X_train.shape[1],X_train.shape[2],1)))
model_3layer.add(keras.layers.BatchNormalization())
model_3layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_3layer.add(keras.layers.Conv2D(filters=16,kernel_size=(3,3),kernel_initializer='he_normal',activation='relu'))
model_3layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_3layer.add(keras.layers.Conv2D(filters=32,kernel_size=(2,2),kernel_initializer='he_normal',activation='relu'))
model_3layer.add(keras.layers.BatchNormalization())
model_3layer.add(keras.layers.Flatten())
model_3layer.add(keras.layers.Dense(128, activation='relu',kernel_initializer='he_normal'))
model_3layer.add(keras.layers.Dropout(0.4))
model_3layer.add(keras.layers.Dense(64, activation='relu',kernel_initializer='he_normal'))
model_3layer.add(keras.layers.Dropout(0.2))
model_3layer.add(keras.layers.Dense(10,activation='softmax'))

model_3layer.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_3layer.fit(X_train, Y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, Y_test))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:2041: The name tf.
nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:148: The name tf.p
laceholder_with_default is deprecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733: calling drop
out (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 47s 787us/step - loss: 0.3532 - acc: 0.8912 - val_loss: 0.0775 - val_a
cc: 0.9744
Epoch 2/10
60000/60000 [==============================] - 47s 777us/step - loss: 0.1150 - acc: 0.9670 - val_loss: 0.0585 - val_a
cc: 0.9832
Epoch 3/10
60000/60000 [==============================] - 47s 777us/step - loss: 0.0817 - acc: 0.9767 - val_loss: 0.0496 - val_a
cc: 0.9838
Epoch 4/10
60000/60000 [==============================] - 47s 782us/step - loss: 0.0661 - acc: 0.9807 - val_loss: 0.0408 - val_a
cc: 0.9888
Epoch 5/10
60000/60000 [==============================] - 47s 779us/step - loss: 0.0541 - acc: 0.9840 - val_loss: 0.0411 - val_a
cc: 0.9871
Epoch 6/10
60000/60000 [==============================] - 47s 779us/step - loss: 0.0477 - acc: 0.9863 - val_loss: 0.0370 - val_a
cc: 0.9896
Epoch 7/10
60000/60000 [==============================] - 47s 780us/step - loss: 0.0431 - acc: 0.9875 - val_loss: 0.0345 - val_a
cc: 0.9897
Epoch 8/10
60000/60000 [==============================] - 47s 777us/step - loss: 0.0371 - acc: 0.9887 - val_loss: 0.0352 - val_a
cc: 0.9889
Epoch 9/10
60000/60000 [==============================] - 47s 784us/step - loss: 0.0335 - acc: 0.9900 - val_loss: 0.0362 - val_a
cc: 0.9904
Epoch 10/10
60000/60000 [==============================] - 47s 780us/step - loss: 0.0327 - acc: 0.9902 - val_loss: 0.0440 - val_a
cc: 0.9882
```

In [12]:
```
score = model_3layer.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,axis = plt.subplots(1,1)
axis.set_title("model with 3 conv layers and followed by dense layers")
axis.set_xlabel('epoch') ; axis.set_ylabel('Categorical Crossentropy Loss')
plt_epoch_vs_loss(list(range(1,11)), history.history['val_loss'], history.history['loss'], axis)
table.add_row(["model_3convlayer",score[1],"kernel-5X5X8(padding-same, relu, he_normal),\n BatchNormalization(),\n maxp
ool-(2,2),\n \
kernel-3X3X16(relu, he_normal),\n maxpool-(2,2),\n \
kernel-2X2X32(relu, he_normal),\n BatchNormalization,\n \
flatten,\n  Dense-128,\n Dropout-0.4,\n Dense-64,\n Dropout-0.2,\n softmax-10"])
```

```
Test score: 0.04403250717676419
Test accuracy: 0.9882
```

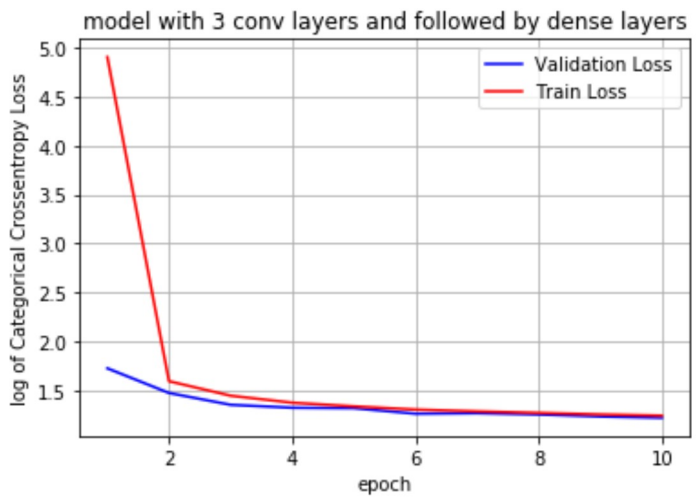


In [13]:
```
model_3layer = Sequential()
model_3layer.add(keras.layers.Conv2D(filters=32,kernel_size=(7,7),strides=(1,1),padding='same',activation='relu',kernel
_initializer='he_normal',\
                                     input_shape = (X_train.shape[1],X_train.shape[2],1)))
model_3layer.add(keras.layers.AvgPool2D(pool_size=(2,2)))
model_3layer.add(keras.layers.Conv2D(filters=8,kernel_size=(5,5),kernel_initializer='he_normal',activation='relu',kerne
l_regularizer=keras.regularizers.l2(0.001)))
model_3layer.add(keras.layers.AvgPool2D(pool_size=(2,2)))
model_3layer.add(keras.layers.Conv2D(filters=64,kernel_size=(3,3),kernel_initializer='he_normal',activation='relu',kern
el_regularizer=keras.regularizers.l2(0.0001)))
model_3layer.add(keras.layers.Flatten())
model_3layer.add(keras.layers.Dense(512, activation='relu',kernel_initializer='he_normal',kernel_regularizer=keras.regu
larizers.l1(0.1)))
model_3layer.add(keras.layers.Dense(32, activation='relu',kernel_initializer='he_normal'))
model_3layer.add(keras.layers.Dropout(0.2))
model_3layer.add(keras.layers.Dense(10,activation='softmax'))

model_3layer.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_3layer.fit(X_train, Y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, Y_test))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4271: The name tf.
nn.avg_pool is deprecated. Please use tf.nn.avg_pool2d instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 83s 1ms/step - loss: 134.6556 - acc: 0.4565 - val_loss: 5.6330 - val_a
cc: 0.8404
Epoch 2/10
60000/60000 [==============================] - 84s 1ms/step - loss: 4.9399 - acc: 0.8881 - val_loss: 4.3817 - val_ac
c: 0.9405
Epoch 3/10
60000/60000 [==============================] - 83s 1ms/step - loss: 4.2612 - acc: 0.9411 - val_loss: 3.8829 - val_ac
c: 0.9693
Epoch 4/10
60000/60000 [==============================] - 84s 1ms/step - loss: 3.9666 - acc: 0.9543 - val_loss: 3.7690 - val_ac
c: 0.9718
Epoch 5/10
60000/60000 [==============================] - 84s 1ms/step - loss: 3.8215 - acc: 0.9603 - val_loss: 3.7516 - val_ac
c: 0.9740
Epoch 6/10
60000/60000 [==============================] - 83s 1ms/step - loss: 3.6997 - acc: 0.9634 - val_loss: 3.5454 - val_ac
c: 0.9766
Epoch 7/10
60000/60000 [==============================] - 84s 1ms/step - loss: 3.6314 - acc: 0.9672 - val_loss: 3.5636 - val_ac
c: 0.9750
Epoch 8/10
60000/60000 [==============================] - 84s 1ms/step - loss: 3.5757 - acc: 0.9695 - val_loss: 3.5178 - val_ac
c: 0.9774
Epoch 9/10
60000/60000 [==============================] - 83s 1ms/step - loss: 3.5193 - acc: 0.9710 - val_loss: 3.4463 - val_ac
c: 0.9746
Epoch 10/10
60000/60000 [==============================] - 82s 1ms/step - loss: 3.4763 - acc: 0.9733 - val_loss: 3.3998 - val_ac
c: 0.9812
```

In [20]:
```
score = model_3layer.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,axis = plt.subplots(1,1)
axis.set_title("model with 3 conv layers and followed by dense layers")
axis.set_xlabel('epoch') ; axis.set_ylabel('log of Categorical Crossentropy Loss')
plt_epoch_vs_loss(list(range(1,11)), np.log(history.history['val_loss']), np.log(history.history['loss']), axis)
table.add_row(["model_3convlayer",score[1],"kernel-7X7X32(padding-same, relu, he_normal),\n AvgPool-(2,2),\n \
kernel-5X5X8(relu, he_normal, L2_regularizer(0.001)),\n AvgPool-(2,2),\n \
kernel-3X3X64(relu, he_normal, L2_regularizer(0.0001)),\n \
flatten,\n  Dense-512(relu, he_normal, L1_regularizer(0.1)),\n Dense-32(relu, he_normal),\n Dropout-0.2,\n softmax-1
0"])
```

```
Test score: 3.3998314212799072
Test accuracy: 0.9812
```

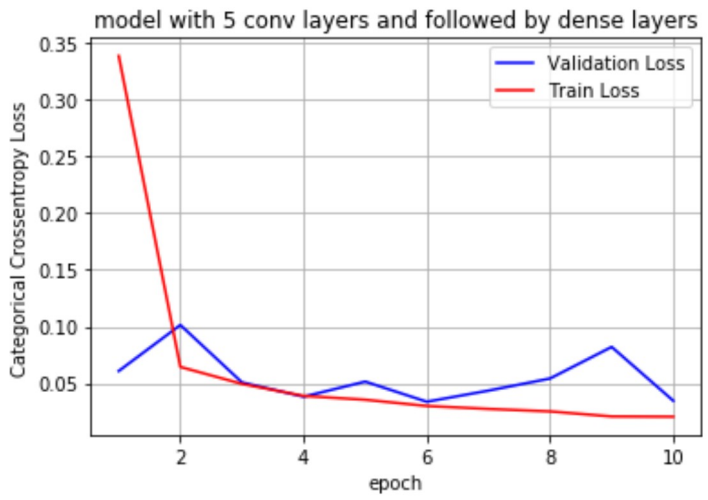


In [21]:
```
from keras.models import Sequential
model_5layer = Sequential()
model_5layer.add(keras.layers.Conv2D(filters=8,kernel_size=(7,7),strides=(1,1),padding='same',activation='relu',kernel_
initializer='he_normal',\
                                     input_shape = (X_train.shape[1],X_train.shape[2],1)))
model_5layer.add(keras.layers.BatchNormalization())
model_5layer.add(keras.layers.Conv2D(filters=16,kernel_size=(5,5), strides=(2,2), padding='same', activation='relu', ke
rnel_initializer='he_normal'))
model_5layer.add(keras.layers.Conv2D(filters=32,kernel_size=(3,3), kernel_initializer='he_normal', activation='relu'))
model_5layer.add(keras.layers.AvgPool2D(pool_size=(2,2)))
model_5layer.add(keras.layers.Conv2D(filters=64,kernel_size=(1,1), kernel_initializer='he_normal', activation='relu'))
model_5layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_5layer.add(keras.layers.Conv2D(filters=128,kernel_size=(3,3),kernel_initializer='he_normal',activation='relu'))
model_5layer.add(keras.layers.BatchNormalization())
model_5layer.add(keras.layers.Flatten())
model_5layer.add(keras.layers.Dense(128, activation='sigmoid',kernel_initializer=keras.initializers.RandomNormal()))
model_5layer.add(keras.layers.Dropout(0.5))
model_5layer.add(keras.layers.Dense(64, activation='tanh',kernel_initializer='glorot_normal'))
model_5layer.add(keras.layers.Dense(10,activation='softmax'))

model_5layer.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_5layer.fit(X_train, Y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, Y_test))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4409: The name tf.
random_normal is deprecated. Please use tf.random.normal instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 88s 1ms/step - loss: 0.3387 - acc: 0.9091 - val_loss: 0.0608 - val_ac
c: 0.9806
Epoch 2/10
60000/60000 [==============================] - 89s 1ms/step - loss: 0.0644 - acc: 0.9824 - val_loss: 0.1014 - val_ac
c: 0.9706
Epoch 3/10
60000/60000 [==============================] - 90s 1ms/step - loss: 0.0494 - acc: 0.9859 - val_loss: 0.0507 - val_ac
c: 0.9841
Epoch 4/10
60000/60000 [==============================] - 88s 1ms/step - loss: 0.0388 - acc: 0.9881 - val_loss: 0.0381 - val_ac
c: 0.9879
Epoch 5/10
60000/60000 [==============================] - 89s 1ms/step - loss: 0.0355 - acc: 0.9898 - val_loss: 0.0514 - val_ac
c: 0.9838
Epoch 6/10
60000/60000 [==============================] - 89s 1ms/step - loss: 0.0299 - acc: 0.9912 - val_loss: 0.0337 - val_ac
c: 0.9898
Epoch 7/10
60000/60000 [==============================] - 88s 1ms/step - loss: 0.0274 - acc: 0.9913 - val_loss: 0.0435 - val_ac
c: 0.9859
Epoch 8/10
60000/60000 [==============================] - 90s 1ms/step - loss: 0.0251 - acc: 0.9921 - val_loss: 0.0542 - val_ac
c: 0.9865
Epoch 9/10
60000/60000 [==============================] - 88s 1ms/step - loss: 0.0207 - acc: 0.9938 - val_loss: 0.0820 - val_ac
c: 0.9781
Epoch 10/10
60000/60000 [==============================] - 87s 1ms/step - loss: 0.0205 - acc: 0.9936 - val_loss: 0.0345 - val_ac
c: 0.9891
```

In [22]:
```
score = model_5layer.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,axis = plt.subplots(1,1)
axis.set_title("model with 5 conv layers and followed by dense layers")
axis.set_xlabel('epoch') ; axis.set_ylabel('Categorical Crossentropy Loss')
plt_epoch_vs_loss(list(range(1,11)), history.history['val_loss'], history.history['loss'], axis)
table.add_row(["model_5convlayer",score[1],"kernel-7X7X8(padding-same, relu, he_normal),\n BatchNormalization(),\n \
kernel-5X5X16(stride=(2,2), padding-same, relu, he_normal),\n \
kernel-3X3X32(relu, he_normal),\n AvgPool-(2,2),\n \
kernel-1X1X64(relu, he_normal),\n MaxPool-(2,2),\n \
kernel-3X3X128(relu, he_normal),\n BatchNormalization(),\n \
flatten,\n  Dense-128(sigmoid, random_normal),\n Dropout-0.5,\n Dense-64(tanh, glorot_normal),\n softmax-10"])
```

```
Test score: 0.03454381320771354
Test accuracy: 0.9891
```

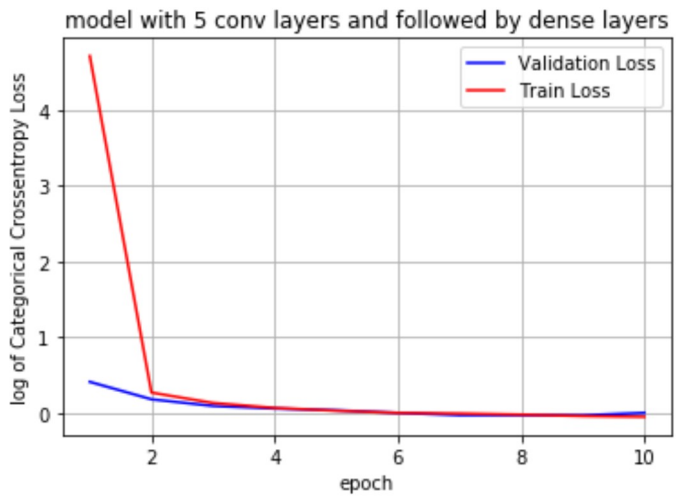


In [23]:
```
model_5layer = Sequential()
model_5layer.add(keras.layers.Conv2D(filters=8,kernel_size=(11,11),strides=(2,2),padding='same',activation='sigmoid',ke
rnel_initializer='glorot_normal',\
                                     input_shape = (X_train.shape[1],X_train.shape[2],1)))
model_5layer.add(keras.layers.BatchNormalization())
model_5layer.add(keras.layers.Conv2D(filters=16,kernel_size=(7,7), strides=(3,3), padding='same', activation='tanh', ke
rnel_initializer='glorot_normal'))
model_5layer.add(keras.layers.BatchNormalization())
model_5layer.add(keras.layers.Conv2D(filters=32,kernel_size=(5,5), kernel_initializer='glorot_normal', activation='sigm
oid'))
model_5layer.add(keras.layers.BatchNormalization())
model_5layer.add(keras.layers.Conv2D(filters=64,kernel_size=(1,1), kernel_initializer='glorot_normal', activation='sigm
oid'))
model_5layer.add(keras.layers.BatchNormalization())
model_5layer.add(keras.layers.Conv2D(filters=128,kernel_size=(1,1),kernel_initializer='glorot_normal',activation='sigmo
id'))
model_5layer.add(keras.layers.BatchNormalization())
model_5layer.add(keras.layers.Flatten())
model_5layer.add(keras.layers.Dense(512, activation=keras.layers.LeakyReLU(alpha=0.3),kernel_initializer='he_normal',ke
rnel_regularizer=keras.regularizers.l1(0.1)))
model_5layer.add(keras.layers.Dense(32, activation='relu',kernel_initializer='he_normal'))
model_5layer.add(keras.layers.Dense(10,activation='softmax'))

model_5layer.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_5layer.fit(X_train, Y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, Y_test))
```

```
/usr/local/lib/python3.6/dist-packages/keras/activations.py:235: UserWarning: Do not pass a layer instance (such as L
eakyReLU) as the activation argument of another layer. Instead, advanced activation layers should be used just like a
ny other layer in a model.
  identifier=identifier.__class__.__name__))
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 37s 612us/step - loss: 110.9730 - acc: 0.9134 - val_loss: 1.5035 - val
_acc: 0.9580
Epoch 2/10
60000/60000 [==============================] - 35s 584us/step - loss: 1.3051 - acc: 0.9633 - val_loss: 1.1942 - val_a
cc: 0.9679
Epoch 3/10
60000/60000 [==============================] - 36s 593us/step - loss: 1.1374 - acc: 0.9708 - val_loss: 1.0935 - val_a
cc: 0.9696
Epoch 4/10
60000/60000 [==============================] - 36s 596us/step - loss: 1.0653 - acc: 0.9754 - val_loss: 1.0569 - val_a
cc: 0.9733
Epoch 5/10
60000/60000 [==============================] - 36s 598us/step - loss: 1.0273 - acc: 0.9788 - val_loss: 1.0323 - val_a
cc: 0.9721
Epoch 6/10
60000/60000 [==============================] - 36s 594us/step - loss: 0.9978 - acc: 0.9802 - val_loss: 0.9983 - val_a
cc: 0.9766
Epoch 7/10
60000/60000 [==============================] - 35s 581us/step - loss: 0.9895 - acc: 0.9800 - val_loss: 0.9678 - val_a
cc: 0.9794
Epoch 8/10
60000/60000 [==============================] - 35s 589us/step - loss: 0.9754 - acc: 0.9810 - val_loss: 0.9686 - val_a
cc: 0.9777
Epoch 9/10
60000/60000 [==============================] - 36s 597us/step - loss: 0.9571 - acc: 0.9835 - val_loss: 0.9660 - val_a
cc: 0.9775
Epoch 10/10
60000/60000 [==============================] - 36s 592us/step - loss: 0.9478 - acc: 0.9842 - val_loss: 0.9983 - val_a
cc: 0.9735
```

In [24]:
```
score = model_5layer.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,axis = plt.subplots(1,1)
axis.set_title("model with 5 conv layers and followed by dense layers")
axis.set_xlabel('epoch') ; axis.set_ylabel('log of Categorical Crossentropy Loss')
plt_epoch_vs_loss(list(range(1,11)), np.log(history.history['val_loss']), np.log(history.history['loss']), axis)
table.add_row(["model_5convlayer",score[1],"kernel-11X11X8(strides-(2,2), padding-same, sigmoid, glorot_normal),\n Batc
hNormalization(),\n \
kernel-7X7X16(stride=(3,3), padding-same, tanh, glorot_normal),\n BatchNormalization(),\n \
kernel-5X5X32(sigmoid, glorot_normal),\n BatchNormalization(),\n \
kernel-1X1X64(sigmoid, glorot_normal),\n BatchNormalization(),\n \
kernel-1X1X128(sigmoid, glorot_normal),\n BatchNormalization(),\n \
flatten,\n  Dense-512(LeakyRelu(alpha=0.3), he_normal, L1_regularizer(0.1)),\n Dense-32(relu, he_normal),\n softmax-1
0"])
```

```
Test score: 0.9983397294044495
Test accuracy: 0.9735
```

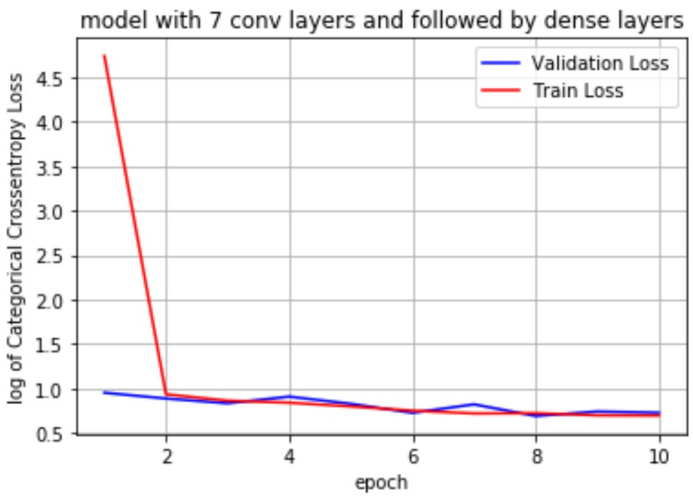


In [25]:
```
model_7layer = Sequential()
model_7layer.add(keras.layers.Conv2D(filters=4,kernel_size=(7,7),strides=(2,2),padding='same',activation='relu',kernel_
initializer='he_normal',\
                                     input_shape = (X_train.shape[1],X_train.shape[2],1)))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_7layer.add(keras.layers.Conv2D(filters=8,kernel_size=(5,5),padding='same',activation='relu',kernel_initializer='h
e_normal'))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.Conv2D(filters=16,kernel_size=(5,5),padding='same',kernel_initializer='he_normal',activat
ion='relu'))
model_7layer.add(keras.layers.AvgPool2D(pool_size=(2,2)))
model_7layer.add(keras.layers.Conv2D(filters=32,kernel_size=(3,3),padding='same', activation='relu', kernel_initializer
='he_normal'))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.Conv2D(filters=64,kernel_size=(3,3),padding='same',kernel_initializer='he_normal', activa
tion='relu'))
model_7layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_7layer.add(keras.layers.Conv2D(filters=128,kernel_size=(1,1),padding='same',kernel_initializer='glorot_normal', a
ctivation='sigmoid'))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.Conv2D(filters=256,kernel_size=(1,1),padding='same',kernel_initializer='glorot_normal',ac
tivation='sigmoid'))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.Flatten())
model_7layer.add(keras.layers.Dense(512, activation=keras.layers.LeakyReLU(alpha=0.3),kernel_initializer='he_normal',ke
rnel_regularizer=keras.regularizers.l1(0.1)))
model_7layer.add(keras.layers.Dropout(0.5))
model_7layer.add(keras.layers.Dense(128, activation='tanh',kernel_initializer='glorot_normal'))
model_7layer.add(keras.layers.Dropout(0.5))
model_7layer.add(keras.layers.Dense(10,activation='softmax'))

model_7layer.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_7layer.fit(X_train, Y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, Y_test))
```

```
/usr/local/lib/python3.6/dist-packages/keras/activations.py:235: UserWarning: Do not pass a layer instance (such as L
eakyReLU) as the activation argument of another layer. Instead, advanced activation layers should be used just like a
ny other layer in a model.
  identifier=identifier.__class__.__name__))

Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 39s 658us/step - loss: 114.3326 - acc: 0.8124 - val_loss: 2.5967 - val
_acc: 0.9369
Epoch 2/10
60000/60000 [==============================] - 37s 618us/step - loss: 2.5513 - acc: 0.9288 - val_loss: 2.4331 - val_a
cc: 0.9416
Epoch 3/10
60000/60000 [==============================] - 37s 613us/step - loss: 2.3752 - acc: 0.9521 - val_loss: 2.3065 - val_a
cc: 0.9691
Epoch 4/10
60000/60000 [==============================] - 37s 613us/step - loss: 2.3214 - acc: 0.9614 - val_loss: 2.4855 - val_a
cc: 0.9742
Epoch 5/10
60000/60000 [==============================] - 36s 606us/step - loss: 2.2267 - acc: 0.9671 - val_loss: 2.2931 - val_a
cc: 0.9718
Epoch 6/10
60000/60000 [==============================] - 36s 607us/step - loss: 2.1223 - acc: 0.9729 - val_loss: 2.0710 - val_a
cc: 0.9813
Epoch 7/10
60000/60000 [==============================] - 37s 622us/step - loss: 2.0574 - acc: 0.9772 - val_loss: 2.2790 - val_a
cc: 0.8893
Epoch 8/10
60000/60000 [==============================] - 37s 624us/step - loss: 2.0629 - acc: 0.9787 - val_loss: 1.9981 - val_a
cc: 0.9771
Epoch 9/10
60000/60000 [==============================] - 37s 621us/step - loss: 2.0145 - acc: 0.9806 - val_loss: 2.1026 - val_a
cc: 0.9682
Epoch 10/10
60000/60000 [==============================] - 38s 631us/step - loss: 2.0093 - acc: 0.9819 - val_loss: 2.0750 - val_a
cc: 0.9734
```

In [26]:
```
score = model_7layer.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,axis = plt.subplots(1,1)
axis.set_title("model with 7 conv layers and followed by dense layers")
axis.set_xlabel('epoch') ; axis.set_ylabel('log of Categorical Crossentropy Loss')
plt_epoch_vs_loss(list(range(1,11)), np.log(history.history['val_loss']), np.log(history.history['loss']), axis)
table.add_row(["model_7convlayer",score[1],"kernel-7X7X4(strides-(2,2), padding-same, relu, he_normal),\n BatchNormaliz
ation(),\n MaxPool-(2,2)\
kernel-5X5X8(padding-same, relu, he_normal),\n BatchNormalization(),\n \
kernel-5X5X16(padding-same, relu, he_normal),\n AvgPool-(2,2),\n \
kernel-3X3X32(padding-same, relu, he_normal),\n BatchNormalization(),\n \
kernel-3X3X64(padding-same, relu, he_normal),\n MaxPool-(2,2),\n \
kernel-1X1X128(padding-same, sigmoid, glorot_normal),\n BatchNormalization(),\n \
kernel-1X1X256(padding-same, sigmoid, glorot_normal),\n BatchNormalization(),\n \
flatten,\n  Dense-512(LeakyRelu(alpha=0.3), he_normal, L1_regularizer(0.1)),\n Dropout-0.5,\n Dense-128(tanh, glorot_no
rmal),\n Dropout-0.5,\n softmax-10"])
```

```
Test score: 2.0750079093933107
Test accuracy: 0.9734
```

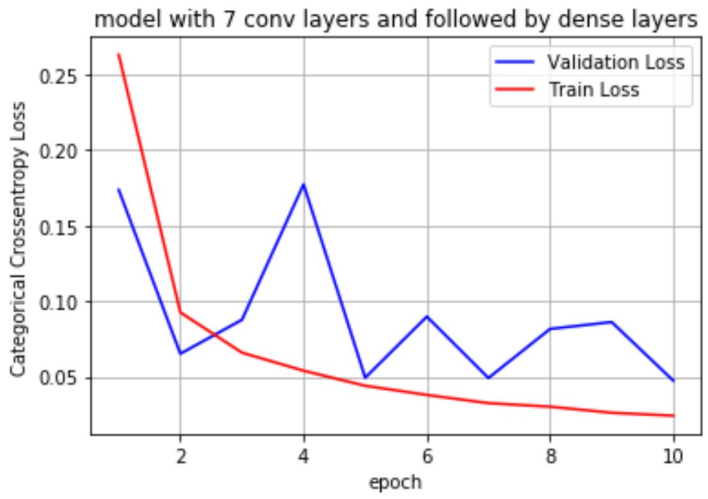


In [27]:
```
model_7layer = Sequential()
model_7layer.add(keras.layers.Conv2D(filters=4,kernel_size=(7,7),activation='relu',kernel_initializer='he_normal',\
                                     input_shape = (X_train.shape[1],X_train.shape[2],1)))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_7layer.add(keras.layers.Conv2D(filters=8,kernel_size=(5,5),padding='same',activation='relu',kernel_initializer='h
e_normal'))
model_7layer.add(keras.layers.Conv2D(filters=16,kernel_size=(5,5),padding='same',kernel_initializer='he_normal',activat
ion='relu'))
model_7layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_7layer.add(keras.layers.Conv2D(filters=32,kernel_size=(3,3),padding='same', activation='relu', kernel_initializer
='he_normal'))
model_7layer.add(keras.layers.Conv2D(filters=64,kernel_size=(3,3),padding='same',kernel_initializer='he_normal', activa
tion='relu'))
model_7layer.add(keras.layers.MaxPool2D(pool_size=(2,2)))
model_7layer.add(keras.layers.Conv2D(filters=128,kernel_size=(1,1),padding='same',kernel_initializer='he_normal', activ
ation='relu'))
model_7layer.add(keras.layers.Conv2D(filters=256,kernel_size=(1,1),padding='same',kernel_initializer='he_normal',activa
tion='relu'))
model_7layer.add(keras.layers.Flatten())
model_7layer.add(keras.layers.Dense(512, activation='relu',kernel_initializer='he_normal'))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.Dropout(0.4))
model_7layer.add(keras.layers.Dense(128, activation='relu',kernel_initializer='he_normal'))
model_7layer.add(keras.layers.BatchNormalization())
model_7layer.add(keras.layers.Dropout(0.2))
model_7layer.add(keras.layers.Dense(10,activation='softmax'))

model_7layer.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model_7layer.fit(X_train, Y_train, batch_size=128, epochs=10, verbose=1, validation_data=(X_test, Y_test))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 77s 1ms/step - loss: 0.2631 - acc: 0.9191 - val_loss: 0.1737 - val_ac
c: 0.9451
Epoch 2/10
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0926 - acc: 0.9711 - val_loss: 0.0653 - val_ac
c: 0.9787
Epoch 3/10
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0660 - acc: 0.9798 - val_loss: 0.0878 - val_ac
c: 0.9715
Epoch 4/10
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0540 - acc: 0.9835 - val_loss: 0.1774 - val_ac
c: 0.9446
Epoch 5/10
60000/60000 [==============================] - 74s 1ms/step - loss: 0.0441 - acc: 0.9861 - val_loss: 0.0494 - val_ac
c: 0.9845
Epoch 6/10
60000/60000 [==============================] - 74s 1ms/step - loss: 0.0380 - acc: 0.9878 - val_loss: 0.0898 - val_ac
c: 0.9733
Epoch 7/10
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0326 - acc: 0.9899 - val_loss: 0.0492 - val_ac
c: 0.9851
Epoch 8/10
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0302 - acc: 0.9904 - val_loss: 0.0817 - val_ac
c: 0.9751
Epoch 9/10
60000/60000 [==============================] - 76s 1ms/step - loss: 0.0262 - acc: 0.9914 - val_loss: 0.0863 - val_ac
c: 0.9725
Epoch 10/10
60000/60000 [==============================] - 75s 1ms/step - loss: 0.0243 - acc: 0.9923 - val_loss: 0.0474 - val_ac
c: 0.9853
```

In [28]:
```
score = model_7layer.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

fig,axis = plt.subplots(1,1)
axis.set_title("model with 7 conv layers and followed by dense layers")
axis.set_xlabel('epoch') ; axis.set_ylabel('Categorical Crossentropy Loss')
plt_epoch_vs_loss(list(range(1,11)), history.history['val_loss'], history.history['loss'], axis)
table.add_row(["model_7convlayer",score[1],"kernel-7X7X4(relu, he_normal),\n BatchNormalization(),\n MaxPool-(2,2)\
kernel-5X5X8(padding-same, relu, he_normal),\n \
kernel-5X5X16(padding-same, relu, he_normal),\n MaxPool-(2,2),\n \
kernel-3X3X32(padding-same, relu, he_normal),\n \
kernel-3X3X64(padding-same, relu, he_normal),\n MaxPool-(2,2),\n \
kernel-1X1X128(padding-same, relu, he_normal),\n \
kernel-1X1X256(padding-same, relu, he_normal),\n \
flatten,\n  Dense-512(relu, he_normal),\n BatchNormalization(),\n Dropout-0.4,\n Dense-128(relu, he_normal),\n BatchNor
malization(),\n Dropout-0.2,\n softmax-10"])
```

```
Test score: 0.04743871111291737
Test accuracy: 0.9853
```



In [29]:
```
print(table)
```

model_name	Test Accuracy	Description
model_3convlayer	0.9567	kernel-5X5X3(relu, he_normal), maxpool-(2,2), kernel-3X3X3(relu, he_normal), maxpool-(2,2), kernel-2X2X3(relu, he_normal), flatten, Dense-128(relu, he_normal), softmax-10
model_3convlayer	0.9882	kernel-5X5X8(padding-same, relu, he_normal), BatchNormalization(), maxpool-(2,2), kernel-3X3X16(relu, he_normal), maxpool-(2,2), kernel-2X2X32(relu, he_normal), BatchNormalization, flatten, Dense-128, Dropout-0.4, Dense-64, Dropout-0.2, softmax-10
model_3convlayer	0.9812	kernel-7X7X32(padding-same, relu, he_normal), AvgPool-(2,2), kernel-5X5X8(relu, he_normal, L2_regularizer(0.001)), AvgPool-(2,2), kernel-3X3X64(relu, he_normal, L2_regularizer(0.0001)), flatten, Dense-512(relu, he_normal, L1_regularizer(0.1)), Dense-32(relu, he_normal), Dropout-0.2, softmax-10
model_5convlayer	0.9891	kernel-7X7X8(padding-same, relu, he_normal), BatchNormalization(), kernel-5X5X16(stride=(2,2), padding-same, relu, he_normal), kernel-3X3X32(relu, he_normal), AvgPool-(2,2), kernel-1X1X64(relu, he_normal), MaxPool-(2,2), kernel-3X3X128(relu, he_normal), BatchNormalization(), flatten, Dense-128(sigmoid, random_normal), Dropout-0.5, Dense-64(tanh, glorot_normal), softmax-10
model_5convlayer	0.9735	kernel-11X11X8(strides-(2,2), padding-same, sigmoid, glorot_normal), BatchNormalization(), kernel-7X7X16(stride=(3,3), padding-same, tanh, glorot_normal), BatchNormalization(), kernel-5X5X32(sigmoid, glorot_normal), BatchNormalization(), kernel-1X1X64(sigmoid, glorot_normal), BatchNormalization(), kernel-1X1X128(sigmoid, glorot_normal), BatchNormalization(), flatten, Dense-512(LeakyRelu(alpha=0.3), he_normal, L1_regularizer(0.1)), Dense-32(relu, he_normal), softmax-10
model_7convlayer	0.9734	kernel-7X7X4(strides-(2,2), padding-same, relu, he_normal), BatchNormalization(), MaxPool-(2,2)kernel-5X5X8(padding-same, relu, he_normal), BatchNormalization(), kernel-5X5X16(padding-same, relu, he_normal), AvgPool-(2,2), kernel-3X3X32(padding-same, relu, he_normal), BatchNormalization(), kernel-3X3X64(padding-same, relu, he_normal), MaxPool-(2,2), kernel-1X1X128(padding-same, sigmoid, glorot_normal), BatchNormalization(), kernel-1X1X256(padding-same, sigmoid, glorot_normal), BatchNormalization(), flatten, Dense-512(LeakyRelu(alpha=0.3), he_normal, L1_regularizer(0.1)), Dropout-0.5, Dense-128(tanh, glorot_normal), Dropout-0.5, softmax-10
model_7convlayer	0.9853	kernel-7X7X4(relu, he_normal), BatchNormalization(), MaxPool-(2,2)kernel-5X5X8(padding-same, relu, he_normal), kernel-5X5X16(padding-same, relu, he_normal), MaxPool-(2,2), kernel-3X3X32(padding-same, relu, he_normal), kernel-3X3X64(padding-same, relu, he_normal), MaxPool-(2,2), kernel-1X1X128(padding-same, relu, he_normal), kernel-1X1X256(padding-same, relu, he_normal), flatten, Dense-512(relu, he_normal), BatchNormalization(), Dropout-0.4,

In [0]: