OBJECTIVE :- Try various CNN networks on MNIST dataset

In [1]:

```
# Importing libraries
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
from keras.initializers import he_normal
from keras.layers.normalization import BatchNormalization
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

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()
```

```
Using TensorFlow backend.
```

In [2]:

```
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)
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [0]:

```
# this function is used draw Categorical Crossentropy Loss VS No. of epochs plot
def plt_dynamic(x, vy, ty):
  plt.figure(figsize=(15,8))
  plt.plot(x, vy, 'g', label="Validation Loss")
  plt.plot(x, ty, 'r', label="Train Loss")
  plt.xlabel('Epochs')
  plt.ylabel('Categorical Crossentropy Loss')
  plt.title('\nCategorical Crossentropy Loss VS Epochs')
  plt.legend()
  plt.grid()
  plt.show()
```

(1). CNN with 3 Convolutional layers and kernel size - (3X3)

In [4]:

```
# Initialising the model
model_3 = Sequential()

# Adding first conv layer
model_3.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))

# Adding second conv layer
model_3.add(Conv2D(64, (3, 3), activation='relu'))

# Adding Maxpooling layer
model_3.add(MaxPooling2D(pool_size=(2, 2)))

# Adding Dropout
model_3.add(Dropout(0.3))

# Adding third conv layer
model_3.add(Conv2D(128, (3, 3), activation='relu'))

# Adding Maxpooling layer
model_3.add(MaxPooling2D(pool_size=(2, 2)))

# Adding Dropout
model_3.add(Dropout(0.3))

# Adding flatten layer
model_3.add(Flatten())

# Adding first hidden layer
model_3.add(Dense(256, activation='relu',kernel_initializer=he_normal(seed=None)))

# Adding Dropout
model_3.add(Dropout(0.3))

# Adding output layer
model_3.add(Dense(num_classes, activation='softmax'))

# Printing model Summary
print(model_3.summary())

# Compiling the model
model_3.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Fitting the data to the model
history_3 = model_3.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,vali
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/py
thon/framework/op_def_library.py:263: colocate_with (from tensorflow.python.
framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:3445: calling dropout (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`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

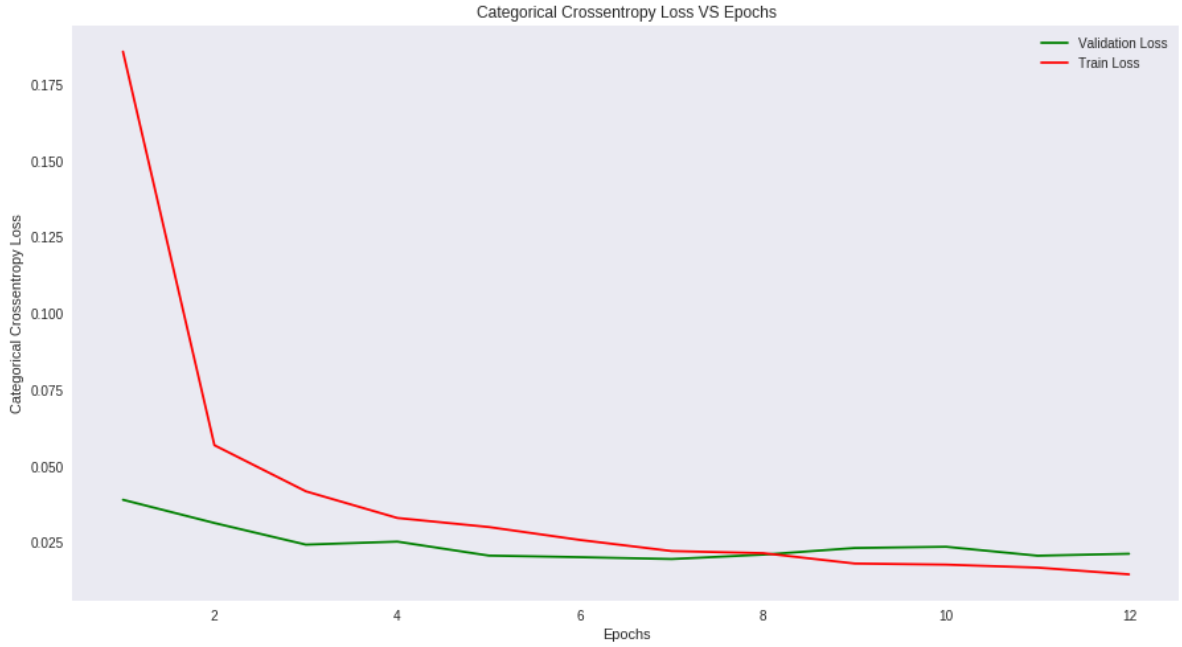
```
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               819456
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 914,698
Trainable params: 914,698
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/py
thon/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 230s 4ms/step - loss: 0.1859
- acc: 0.9406 - val_loss: 0.0391 - val_acc: 0.9872
Epoch 2/12
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0570
- acc: 0.9822 - val_loss: 0.0315 - val_acc: 0.9899
Epoch 3/12
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0419
- acc: 0.9867 - val_loss: 0.0244 - val_acc: 0.9921
Epoch 4/12
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0332
- acc: 0.9899 - val_loss: 0.0254 - val_acc: 0.9925
Epoch 5/12
60000/60000 [==============================] - 228s 4ms/step - loss: 0.0302
- acc: 0.9903 - val_loss: 0.0208 - val_acc: 0.9940
Epoch 6/12
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0260
- acc: 0.9919 - val_loss: 0.0203 - val_acc: 0.9940
Epoch 7/12
60000/60000 [==============================] - 228s 4ms/step - loss: 0.0223
- acc: 0.9929 - val_loss: 0.0197 - val_acc: 0.9938
Epoch 8/12
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0216
- acc: 0.9934 - val_loss: 0.0212 - val_acc: 0.9933
Epoch 9/12
60000/60000 [==============================] - 228s 4ms/step - loss: 0.0182
- acc: 0.9940 - val_loss: 0.0233 - val_acc: 0.9930
Epoch 10/12
```

```
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0179
- acc: 0.9943 - val_loss: 0.0237 - val_acc: 0.9921
Epoch 11/12
60000/60000 [==============================] - 228s 4ms/step - loss: 0.0169
- acc: 0.9941 - val_loss: 0.0208 - val_acc: 0.9936
Epoch 12/12
60000/60000 [==============================] - 227s 4ms/step - loss: 0.0147
- acc: 0.9950 - val_loss: 0.0215 - val_acc: 0.9920
```

In [5]:

```
# Evaluating the model
score = model_3.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

# Test and train accuracy of the model
model_3_test = score[1]
model_3_train = max(history_3.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,epochs+1))

# Validation loss
vy = history_3.history['val_loss']
# Training loss
ty = history_3.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Test score: 0.021451755314664207
Test accuracy: 0.992
```



(2). CNN with 5 Convolutional layers and kernel size - (5X5)

In [6]:

```
# Initialising the model
model_5 = Sequential()

# Adding first conv layer
model_5.add(Conv2D(8, kernel_size=(5, 5),padding='same',activation='relu',input_shape=input

# Adding second conv layer
model_5.add(Conv2D(16, (5, 5), activation='relu'))

# Adding Maxpooling layer
model_5.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

# Adding Dropout
model_5.add(Dropout(0.3))

# Adding third conv layer
model_5.add(Conv2D(32, (5, 5),padding='same', activation='relu'))

# Adding Maxpooling layer
model_5.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

# Adding Dropout
model_5.add(Dropout(0.3))

# Adding fourth conv layer
model_5.add(Conv2D(64, (5, 5),padding='same',activation='relu'))

# Adding fifth conv layer
model_5.add(Conv2D(64, (5, 5), activation='relu'))

# Adding Maxpooling layer
model_5.add(MaxPooling2D(pool_size=(2, 2),padding='same'))

# Adding Dropout
model_5.add(Dropout(0.3))

# Adding flatten layer
model_5.add(Flatten())

# Adding first hidden layer
model_5.add(Dense(256, activation='relu',kernel_initializer=he_normal(seed=None)))

# Adding Batch Normalization
model_5.add(BatchNormalization())

# Adding Dropout
model_5.add(Dropout(0.3))

# Adding output layer
model_5.add(Dense(num_classes, activation='softmax'))

# Printing model Summary
print(model_5.summary())

# Compiling the model
model_5.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Fitting the data to the model
history_5 = model_5.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,vali
```

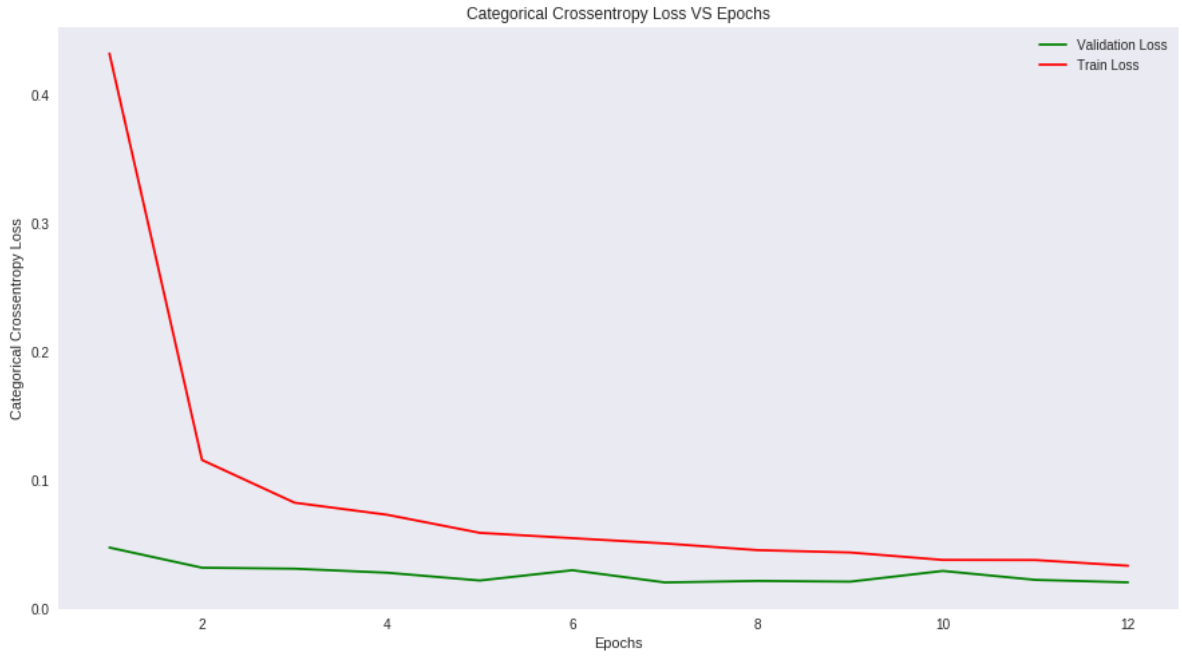
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 28, 28, 8)         208
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 24, 24, 16)        3216
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 16)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 12, 12, 16)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 32)        12832
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 6, 6, 32)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 6, 6, 64)          51264
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 2, 2, 64)          102464
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 1, 1, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               16640
_________________________________________________________________
batch_normalization_1 (Batch (None, 256)               1024
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 190,218
Trainable params: 189,706
Non-trainable params: 512
_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.4323
- acc: 0.8616 - val_loss: 0.0476 - val_acc: 0.9850
Epoch 2/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.1157
- acc: 0.9675 - val_loss: 0.0319 - val_acc: 0.9911
Epoch 3/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0825
- acc: 0.9764 - val_loss: 0.0311 - val_acc: 0.9913
Epoch 4/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0731
- acc: 0.9795 - val_loss: 0.0279 - val_acc: 0.9904
Epoch 5/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0590
- acc: 0.9837 - val_loss: 0.0219 - val_acc: 0.9929
Epoch 6/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0548
- acc: 0.9841 - val_loss: 0.0299 - val_acc: 0.9915
Epoch 7/12
```

```
60000/60000 [==============================] - 136s 2ms/step - loss: 0.0507
- acc: 0.9862 - val_loss: 0.0204 - val_acc: 0.9939
Epoch 8/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0455
- acc: 0.9871 - val_loss: 0.0216 - val_acc: 0.9945
Epoch 9/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0437
- acc: 0.9880 - val_loss: 0.0210 - val_acc: 0.9941
Epoch 10/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0379
- acc: 0.9892 - val_loss: 0.0293 - val_acc: 0.9894
Epoch 11/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0378
- acc: 0.9895 - val_loss: 0.0224 - val_acc: 0.9944
Epoch 12/12
60000/60000 [==============================] - 135s 2ms/step - loss: 0.0334
- acc: 0.9905 - val_loss: 0.0205 - val_acc: 0.9944
```

In [7]:

```
# Evaluating the model
score = model_5.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

# Test and train accuracy of the model
model_5_test = score[1]
model_5_train = max(history_5.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,epochs+1))

# Validation loss
vy = history_5.history['val_loss']
# Training loss
ty = history_5.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Test score: 0.02045143669675963
Test accuracy: 0.9944
```



(3). CNN with 7 Convolutional layers and kernel size - (2X2)

In [8]:

```
# Initialising the model
model_7 = Sequential()

# Adding first conv layer
model_7.add(Conv2D(32, kernel_size=(2, 2),padding='same',activation='relu',input_shape=inpu

# Adding second conv layer
model_7.add(Conv2D(32, (2, 2), activation='relu'))

# Adding Maxpooling layer
model_7.add(MaxPooling2D(pool_size=(3, 3), strides=(1,1)))

# Adding Dropout
model_7.add(Dropout(0.3))

# Adding third conv layer
model_7.add(Conv2D(64, (2, 2), activation='relu'))

# Adding Maxpooling layer
model_7.add(MaxPooling2D(pool_size=(2, 2),padding='same'))


# Adding fourth conv layer
model_7.add(Conv2D(64, (2, 2),padding='same',activation='relu'))

# Adding fifth conv layer
model_7.add(Conv2D(128, (2, 2), activation='relu'))

# Adding Maxpooling layer
model_7.add(MaxPooling2D(pool_size=(3, 3),padding='same'))

# Adding Dropout
model_7.add(Dropout(0.3))

# Adding sixth conv layer
model_7.add(Conv2D(128, (2, 2),padding='same',activation='relu'))

# Adding seventh conv layer
model_7.add(Conv2D(256, (2, 2), activation='relu'))

# Adding Maxpooling layer
model_7.add(MaxPooling2D(pool_size=(2, 2), strides=(1,1)))

# Adding Dropout
model_7.add(Dropout(0.3))

# Adding flatten layer
model_7.add(Flatten())

# Adding first hidden layer
model_7.add(Dense(256, activation='relu',kernel_initializer=he_normal(seed=None)))

# Adding Batch Normalization
model_7.add(BatchNormalization())

# Adding Dropout
model_7.add(Dropout(0.3))

# Adding second hidden layer
```

```
model_7.add(Dense(128, activation='relu',kernel_initializer=he_normal(seed=None)))

# Adding Dropout
model_7.add(Dropout(0.3))

# Adding output layer
model_7.add(Dense(num_classes, activation='softmax'))

# Printing model Summary
print(model_7.summary())

# Compiling the model
model_7.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Fitting the data to the model
history_7 = model_7.fit(x_train, y_train,batch_size=batch_size,epochs=epochs,verbose=1,vali
```

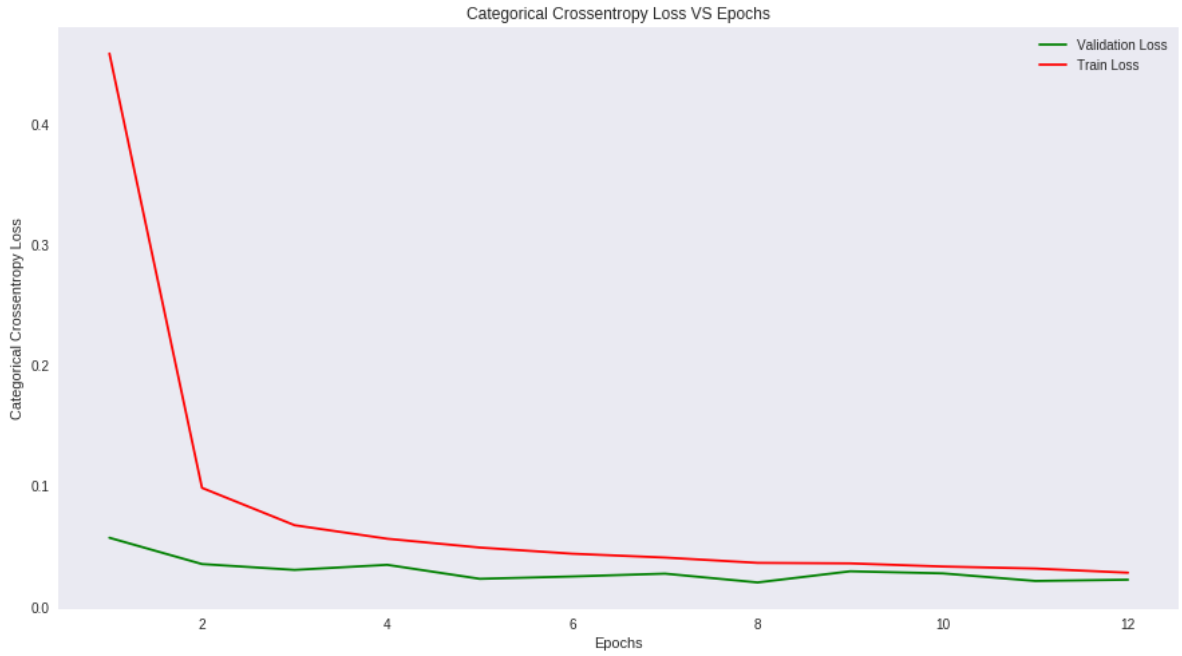
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 28, 28, 32)        160
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 27, 27, 32)        4128
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 25, 25, 32)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 64)        8256
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 12, 12, 64)        16448
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 11, 11, 128)       32896
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_9 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 4, 4, 128)         65664
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 3, 3, 256)         131328
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
dropout_10 (Dropout)         (None, 2, 2, 256)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 256)               262400
_________________________________________________________________
batch_normalization_2 (Batch (None, 256)               1024
_________________________________________________________________
dropout_11 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_12 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
```

```
=================================================================
Total params: 556,490
Trainable params: 555,978
Non-trainable params: 512

_________________________________________________________________
None
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 295s 5ms/step - loss: 0.4589
- acc: 0.8478 - val_loss: 0.0576 - val_acc: 0.9824
Epoch 2/12
60000/60000 [==============================] - 295s 5ms/step - loss: 0.0989
- acc: 0.9706 - val_loss: 0.0358 - val_acc: 0.9886
Epoch 3/12
60000/60000 [==============================] - 294s 5ms/step - loss: 0.0679
- acc: 0.9798 - val_loss: 0.0309 - val_acc: 0.9905
Epoch 4/12
60000/60000 [==============================] - 294s 5ms/step - loss: 0.0567
- acc: 0.9831 - val_loss: 0.0351 - val_acc: 0.9897
Epoch 5/12
60000/60000 [==============================] - 293s 5ms/step - loss: 0.0494
- acc: 0.9854 - val_loss: 0.0235 - val_acc: 0.9933
Epoch 6/12
60000/60000 [==============================] - 293s 5ms/step - loss: 0.0443
- acc: 0.9869 - val_loss: 0.0255 - val_acc: 0.9927
Epoch 7/12
60000/60000 [==============================] - 297s 5ms/step - loss: 0.0411
- acc: 0.9882 - val_loss: 0.0278 - val_acc: 0.9927
Epoch 8/12
60000/60000 [==============================] - 296s 5ms/step - loss: 0.0367
- acc: 0.9891 - val_loss: 0.0205 - val_acc: 0.9936
Epoch 9/12
60000/60000 [==============================] - 297s 5ms/step - loss: 0.0363
- acc: 0.9895 - val_loss: 0.0298 - val_acc: 0.9910
Epoch 10/12
60000/60000 [==============================] - 297s 5ms/step - loss: 0.0337
- acc: 0.9898 - val_loss: 0.0280 - val_acc: 0.9913
Epoch 11/12
60000/60000 [==============================] - 294s 5ms/step - loss: 0.0320
- acc: 0.9905 - val_loss: 0.0217 - val_acc: 0.9928
Epoch 12/12
60000/60000 [==============================] - 293s 5ms/step - loss: 0.0286
- acc: 0.9916 - val_loss: 0.0228 - val_acc: 0.9930
```

In [9]:

```
# Evaluating the model
score = model_7.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])

# Test and train accuracy of the model
model_7_test = score[1]
model_7_train = max(history_7.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,epochs+1))

# Validation loss
vy = history_7.history['val_loss']
# Training loss
ty = history_7.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Test score: 0.022784425553405162
Test accuracy: 0.993
```



CONCLUSION

(a). Procedure Followed :

1. Load MNIST dataset.
2. Split the dataset into train and test.
3. Normalize the train and test data.
4. Convert class variable into categorical data vector.
5. Implement Softmax classifier with 3, 5 and 7 conv layers .

6. Use kernel -size (3X3) , (5X5) and (2,2) .
7. Draw Categorical Crossentropy Loss VS No.of Epochs plot .

(b) Table (Different models with their train and test accuracies):

In [10]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["N0__Conv_Layers", "Kernel_size", "Training Accuracy", "Test Accuracy"]

x.add_row([3, "(3x3)",0.9929,0.9938])
x.add_row([5, "(5x5)",0.9892,0.9894])
x.add_row([7, "(2x2)",0.9898,0.9913])
print(x)
```

```
+-----------------+-------------+-------------------+---------------+
| N0__Conv_Layers | Kernel_size | Training Accuracy | Test Accuracy |
+-----------------+-------------+-------------------+---------------+
|        3        |    (3x3)    |       0.9929      |     0.9938    |
|        5        |    (5x5)    |       0.9892      |     0.9894    |
|        7        |    (2x2)    |       0.9898      |     0.9913    |
+-----------------+-------------+-------------------+---------------+
```

In [0]: