In [7]:

```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, BatchNormalization
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)
```

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

1. Three Layers CovNets

In [0]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np

def plt_dynamic(x, vy, ty, colors=['b']):
    plt.plot(x, vy, color = 'b', label='Validation Loss')
    plt.plot(x, ty, color = 'r', label='Train Loss')
    plt.xlabel('epoch')
    plt.ylabel('Categorical Crossentropy Loss')
    plt.legend()
    plt.grid()
    plt.show();
```

In [4]:

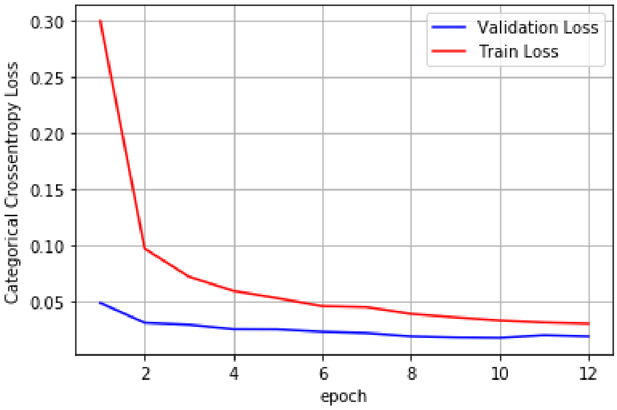
```
model = Sequential()
```

```
# Block 1
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))

# Block 2
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 3
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

history = 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 [==============================] - 173s 3ms/step - loss: 0.2998 - acc: 0.9050 - val_loss: 0
.0487 - val_acc: 0.9841
Epoch 2/12
60000/60000 [==============================] - 174s 3ms/step - loss: 0.0971 - acc: 0.9714 - val_loss: 0
.0310 - val_acc: 0.9902
Epoch 3/12
60000/60000 [==============================] - 173s 3ms/step - loss: 0.0720 - acc: 0.9786 - val_loss: 0
.0292 - val_acc: 0.9906
Epoch 4/12
60000/60000 [==============================] - 173s 3ms/step - loss: 0.0594 - acc: 0.9827 - val_loss: 0
.0253 - val_acc: 0.9908
Epoch 5/12
60000/60000 [==============================] - 174s 3ms/step - loss: 0.0528 - acc: 0.9842 - val_loss: 0
.0251 - val_acc: 0.9907
Epoch 6/12
60000/60000 [==============================] - 174s 3ms/step - loss: 0.0458 - acc: 0.9860 - val_loss: 0
.0230 - val_acc: 0.9928
Epoch 7/12
60000/60000 [==============================] - 172s 3ms/step - loss: 0.0449 - acc: 0.9868 - val_loss: 0
.0218 - val_acc: 0.9926
Epoch 8/12
60000/60000 [==============================] - 171s 3ms/step - loss: 0.0390 - acc: 0.9881 - val_loss: 0
.0188 - val_acc: 0.9940
Epoch 9/12
60000/60000 [==============================] - 170s 3ms/step - loss: 0.0358 - acc: 0.9890 - val_loss: 0
.0179 - val_acc: 0.9933
Epoch 10/12
60000/60000 [==============================] - 169s 3ms/step - loss: 0.0330 - acc: 0.9897 - val_loss: 0
.0175 - val_acc: 0.9937
Epoch 11/12
60000/60000 [==============================] - 169s 3ms/step - loss: 0.0312 - acc: 0.9903 - val_loss: 0
.0199 - val_acc: 0.9938
Epoch 12/12
60000/60000 [==============================] - 171s 3ms/step - loss: 0.0302 - acc: 0.9909 - val_loss: 0
.0188 - val_acc: 0.9935
Test loss: 0.018831969704075892
Test accuracy: 0.9935
```

In [5]:

```
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty)
```



In [8]:

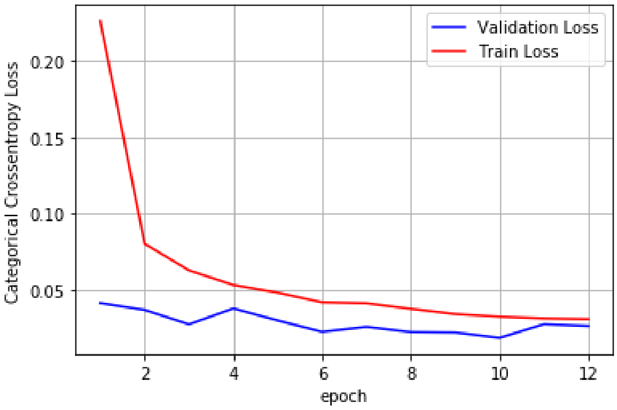
```
model = Sequential()
# Block 1
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))

# Block 2
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 3
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

history = 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 [==============================] - 180s 3ms/step - loss: 0.2264 - acc: 0.9291 - val_loss: 0
.0412 - val_acc: 0.9866
Epoch 2/12
60000/60000 [==============================] - 179s 3ms/step - loss: 0.0802 - acc: 0.9764 - val_loss: 0
.0367 - val_acc: 0.9882
Epoch 3/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0627 - acc: 0.9811 - val_loss: 0
.0273 - val_acc: 0.9914
Epoch 4/12
```

```
60000/60000 [==============================] - 179s 3ms/step - loss: 0.0531 - acc: 0.9839 - val_loss: 0
.0377 - val_acc: 0.9895
Epoch 5/12
60000/60000 [==============================] - 179s 3ms/step - loss: 0.0480 - acc: 0.9855 - val_loss: 0
.0298 - val_acc: 0.9909
Epoch 6/12
60000/60000 [==============================] - 179s 3ms/step - loss: 0.0417 - acc: 0.9872 - val_loss: 0
.0223 - val_acc: 0.9923
Epoch 7/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0411 - acc: 0.9874 - val_loss: 0
.0256 - val_acc: 0.9918
Epoch 8/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0374 - acc: 0.9884 - val_loss: 0
.0222 - val_acc: 0.9930
Epoch 9/12
60000/60000 [==============================] - 178s 3ms/step - loss: 0.0340 - acc: 0.9889 - val_loss: 0
.0219 - val_acc: 0.9944
Epoch 10/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0323 - acc: 0.9900 - val_loss: 0
.0184 - val_acc: 0.9939
Epoch 11/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0311 - acc: 0.9904 - val_loss: 0
.0274 - val_acc: 0.9929
Epoch 12/12
60000/60000 [==============================] - 182s 3ms/step - loss: 0.0307 - acc: 0.9902 - val_loss: 0
.0262 - val_acc: 0.9927
Test loss: 0.026163556197177797
Test accuracy: 0.9927
```

In [9]:

```
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty)
```



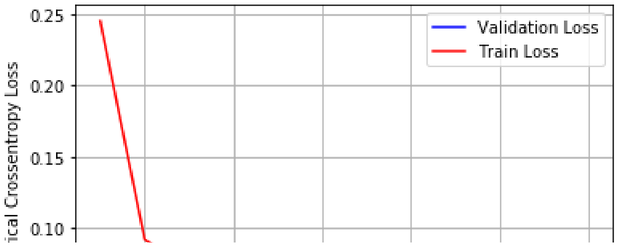
In [10]:

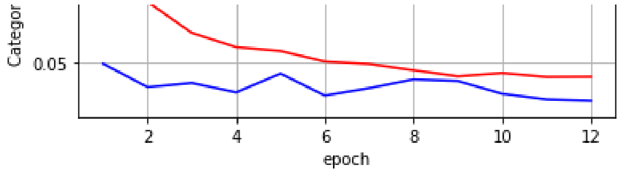
```
model = Sequential()
# Block 1
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))

# Block 2
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 3
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

history = 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 [==============================] - 246s 4ms/step - loss: 0.2451 - acc: 0.9220 - val_loss: 0
.0489 - val_acc: 0.9847
Epoch 2/12
60000/60000 [==============================] - 245s 4ms/step - loss: 0.0923 - acc: 0.9723 - val_loss: 0
.0326 - val_acc: 0.9891
Epoch 3/12
60000/60000 [==============================] - 243s 4ms/step - loss: 0.0705 - acc: 0.9787 - val_loss: 0
.0355 - val_acc: 0.9885
Epoch 4/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0606 - acc: 0.9819 - val_loss: 0
.0289 - val_acc: 0.9911
Epoch 5/12
60000/60000 [==============================] - 242s 4ms/step - loss: 0.0579 - acc: 0.9827 - val_loss: 0
.0421 - val_acc: 0.9863
Epoch 6/12
60000/60000 [==============================] - 241s 4ms/step - loss: 0.0505 - acc: 0.9842 - val_loss: 0
.0267 - val_acc: 0.9914
Epoch 7/12
60000/60000 [==============================] - 242s 4ms/step - loss: 0.0488 - acc: 0.9852 - val_loss: 0
.0319 - val_acc: 0.9905
Epoch 8/12
60000/60000 [==============================] - 244s 4ms/step - loss: 0.0444 - acc: 0.9863 - val_loss: 0
.0380 - val_acc: 0.9875
Epoch 9/12
60000/60000 [==============================] - 247s 4ms/step - loss: 0.0402 - acc: 0.9872 - val_loss: 0
.0368 - val_acc: 0.9882
Epoch 10/12
60000/60000 [==============================] - 244s 4ms/step - loss: 0.0424 - acc: 0.9868 - val_loss: 0
.0280 - val_acc: 0.9908
Epoch 11/12
60000/60000 [==============================] - 245s 4ms/step - loss: 0.0397 - acc: 0.9879 - val_loss: 0
.0240 - val_acc: 0.9913
Epoch 12/12
60000/60000 [==============================] - 243s 4ms/step - loss: 0.0399 - acc: 0.9878 - val_loss: 0
.0230 - val_acc: 0.9926
Test loss: 0.023043111748178 49
Test accuracy: 0.9926
```

In [11]:

```
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty)
```





2. 5 Layers

In [14]:

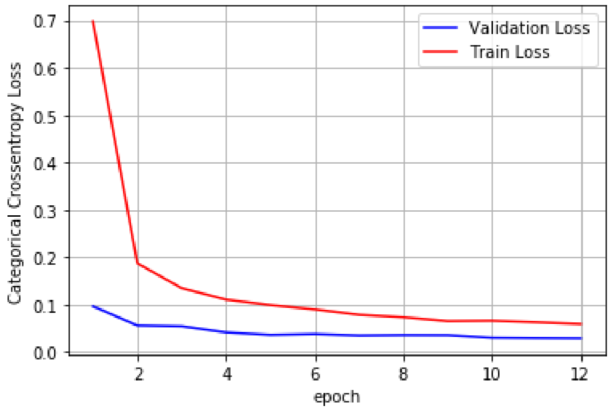
```
model = Sequential()
# Block 1
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))

# Block 2
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 3
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 4
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 5
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

model.summary()

history = 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])
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_27 (Conv2D)           (None, 26, 26, 32)        320       
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 24, 24, 64)        18496     
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 12, 12, 64)        0         
_________________________________________________________________
dropout_23 (Dropout)         (None, 12, 12, 64)        0         
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 10, 10, 64)        36928     
```

```
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 5, 5, 64)          0         
_________________________________________________________________
dropout_24 (Dropout)         (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 3, 3, 64)          36928     
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 1, 1, 64)          0         
_________________________________________________________________
dropout_25 (Dropout)         (None, 1, 1, 64)          0         
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 1, 1, 64)          4160      
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 1, 1, 64)          0         
_________________________________________________________________
dropout_26 (Dropout)         (None, 1, 1, 64)          0         
_________________________________________________________________
flatten_7 (Flatten)          (None, 64)                0         
_________________________________________________________________
dense_13 (Dense)             (None, 128)               8320      
_________________________________________________________________
dropout_27 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_14 (Dense)             (None, 10)                1290      
=================================================================
Total params: 106,442
Trainable params: 106,442
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.6978 - acc: 0.7677 - val_loss: 0
.0966 - val_acc: 0.9712
Epoch 2/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.1875 - acc: 0.9495 - val_loss: 0
.0560 - val_acc: 0.9841
Epoch 3/12
60000/60000 [==============================] - 182s 3ms/step - loss: 0.1349 - acc: 0.9650 - val_loss: 0
.0543 - val_acc: 0.9841
Epoch 4/12
60000/60000 [==============================] - 183s 3ms/step - loss: 0.1108 - acc: 0.9707 - val_loss: 0
.0415 - val_acc: 0.9882
Epoch 5/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.0992 - acc: 0.9746 - val_loss: 0
.0362 - val_acc: 0.9900
Epoch 6/12
60000/60000 [==============================] - 182s 3ms/step - loss: 0.0899 - acc: 0.9769 - val_loss: 0
.0379 - val_acc: 0.9891
Epoch 7/12
60000/60000 [==============================] - 182s 3ms/step - loss: 0.0788 - acc: 0.9790 - val_loss: 0
.0348 - val_acc: 0.9906
Epoch 8/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0734 - acc: 0.9809 - val_loss: 0
.0357 - val_acc: 0.9910
Epoch 9/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0653 - acc: 0.9831 - val_loss: 0
.0355 - val_acc: 0.9900
Epoch 10/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0660 - acc: 0.9832 - val_loss: 0
.0300 - val_acc: 0.9925
Epoch 11/12
60000/60000 [==============================] - 180s 3ms/step - loss: 0.0629 - acc: 0.9832 - val_loss: 0
.0293 - val_acc: 0.9923
Epoch 12/12
60000/60000 [==============================] - 181s 3ms/step - loss: 0.0592 - acc: 0.9851 - val_loss: 0
.0288 - val_acc: 0.9927
Test loss: 0.028791089874924364
Test accuracy: 0.9927
```

In [15]:

```
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty)
```



In [16]:

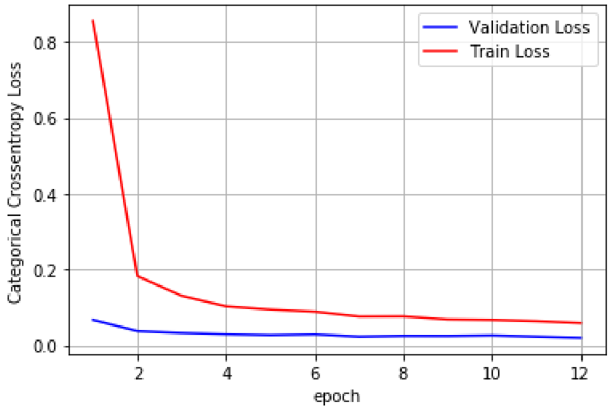
```
model = Sequential()
# Block 1
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))

# Block 2
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 3
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))

# Block 4
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Block 5
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(1, 1)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])

model.summary()

history = 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])
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_32 (Conv2D)           (None, 26, 26, 32)        320       
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 24, 24, 64)        18496     
```

```
conv2d_33 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_28 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 8, 8, 64)          102464
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 4, 4, 64)          0
_________________________________________________________________
dropout_29 (Dropout)         (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 2, 2, 64)          36928
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 1, 1, 64)          0
_________________________________________________________________
dropout_30 (Dropout)         (None, 1, 1, 64)          0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 1, 1, 64)          4160
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 1, 1, 64)          0
_________________________________________________________________
dropout_31 (Dropout)         (None, 1, 1, 64)          0
_________________________________________________________________
flatten_8 (Flatten)          (None, 64)                0
_________________________________________________________________
batch_normalization_3 (Batch (None, 64)                256
_________________________________________________________________
dense_15 (Dense)             (None, 128)               8320
_________________________________________________________________
dropout_32 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_16 (Dense)             (None, 10)                1290
=================================================================
Total params: 172,234
Trainable params: 172,106
Non-trainable params: 128
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 226s 4ms/step - loss: 0.8549 - acc: 0.7155 - val_loss: 0
.0674 - val_acc: 0.9797
Epoch 2/12
60000/60000 [==============================] - 225s 4ms/step - loss: 0.1830 - acc: 0.9500 - val_loss: 0
.0378 - val_acc: 0.9891
Epoch 3/12
60000/60000 [==============================] - 225s 4ms/step - loss: 0.1309 - acc: 0.9655 - val_loss: 0
.0328 - val_acc: 0.9912
Epoch 4/12
60000/60000 [==============================] - 223s 4ms/step - loss: 0.1032 - acc: 0.9727 - val_loss: 0
.0295 - val_acc: 0.9908
Epoch 5/12
60000/60000 [==============================] - 222s 4ms/step - loss: 0.0946 - acc: 0.9751 - val_loss: 0
.0272 - val_acc: 0.9923
Epoch 6/12
60000/60000 [==============================] - 222s 4ms/step - loss: 0.0890 - acc: 0.9768 - val_loss: 0
.0291 - val_acc: 0.9919
Epoch 7/12
60000/60000 [==============================] - 222s 4ms/step - loss: 0.0765 - acc: 0.9805 - val_loss: 0
.0230 - val_acc: 0.9931
Epoch 8/12
60000/60000 [==============================] - 223s 4ms/step - loss: 0.0768 - acc: 0.9813 - val_loss: 0
.0249 - val_acc: 0.9932
Epoch 9/12
60000/60000 [==============================] - 224s 4ms/step - loss: 0.0685 - acc: 0.9820 - val_loss: 0
.0248 - val_acc: 0.9934
Epoch 10/12
60000/60000 [==============================] - 221s 4ms/step - loss: 0.0672 - acc: 0.9831 - val_loss: 0
.0262 - val_acc: 0.9928
Epoch 11/12
60000/60000 [==============================] - 222s 4ms/step - loss: 0.0639 - acc: 0.9834 - val_loss: 0
.0229 - val_acc: 0.9929
Epoch 12/12
60000/60000 [==============================] - 220s 4ms/step - loss: 0.0591 - acc: 0.9851 - val_loss: 0
.0199 - val_acc: 0.9948
Test loss: 0.019945289859417244
Test accuracy: 0.9948
```

In [20]:

```
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
plt_dynamic(x, vy, ty)
```



In [0]:

```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["#layers", "Train Loss/Acc","Test Loss/Acc"]

x.add_row(['3 layers (Conv2D: 32 3x3+[Conv2D: 64 3x3+MaxPooling+Dropout(25%)]x2+Dense(128)+Dropout(50%)
+Output)','3% / 99.1%','1.88% / 99.35%'])
x.add_row(['3 layers (Conv2D: 32 3x3+[Conv2D: 64 3x3+MaxPooling+Dropout(25%)]x2+Dense(128)+BN+Dropout(5
0%)+Output)','3.1% / 99.02%','2.64% / 99.27%'])
x.add_row(['3 layers (Conv2D: 32 5x5+Conv2D: 64 5x5+MaxPooling+Dropout(25%)+Conv2D: 64 1x1+MaxPooling+D
ropout(25%)+Dense(128)+BN+Dropout(50%)+Output)','4% / 98.78%','2.3% / 99.26%'])
x.add_row(['5 layers (Conv2D: 32 3x3+[Conv2D: 64 3x3+MaxPooling+Dropout(25%)]x4+Dense(128)+Dropout(50%)
+Output)','6% / 98.51%','2.88% / 99.27%'])
x.add_row(['5 layers (Conv2D: 32 3x3+Conv2D: 64 3x3+MaxPooling+Dropout(25%)+Conv2D: 64 5x5+MaxPooling+D
ropout(50%)+\
Conv2D: 64 3x3+MaxPooling+Dropout(25%)+Conv2D: 64 1x1+MaxPooling+Dropout(25%)+Dense(128)+Dropout(50%)+O
utput)','5.91% / 98.51%','1.99% / 99.48%'])
```

In [19]:

```
print(x)
```

```
+---------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------
---------+----------------+----------------+
|
#layers
| Train Loss/Acc | Test Loss/Acc  |
+---------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------
---------+----------------+----------------+
|                                                            3 layers (Conv2D: 32 3x3+[Conv2D: 64 3x3+MaxP
ooling+Dropout(25%)]x2+Dense(128)+Dropout(50%)+Output)
|   3% / 99.1%   | 1.88% / 99.35% |
|                                                           3 layers (Conv2D: 32 3x3+[Conv2D: 64 3x3+MaxPo
oling+Dropout(25%)]x2+Dense(128)+BN+Dropout(50%)+Output)
| 3.1% / 99.02%  | 2.64% / 99.27% |
|                                            3 layers (Conv2D: 32 5x5+Conv2D: 64 5x5+MaxPooling+Dropout(25%)+
Conv2D: 64 1x1+MaxPooling+Dropout(25%)+Dense(128)+BN+Dropout(50%)+Output)
|  4% / 98.78%   | 2.3% / 99.26%  |
|                                                            5 layers (Conv2D: 32 3x3+[Conv2D: 64 3x3+MaxP
ooling+Dropout(25%)]x4+Dense(128)+Dropout(50%)+Output)
|  6% / 98.51%   | 2.88% / 99.27% |
| 5 layers (Conv2D: 32 3x3+Conv2D: 64 3x3+MaxPooling+Dropout(25%)+Conv2D: 64 5x5+MaxPooling+Dropout(50%
)+Conv2D: 64 3x3+MaxPooling+Dropout(25%)+Conv2D: 64 1x1+MaxPooling+Dropout(25%)+Dense(128)+Dropout(50%)
+Output) | 5.91% / 98.51% | 1.99% / 99.48% |
+Output) | 5.91% / 98.51% | 1.99% / 99.48% |
+-------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------
---------+-----------------+-----------------+
```

In [0]:

```

```