Importing libraries

In [0]:

```
# 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
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
from keras.layers import BatchNormalization

batch_size = 128
num_classes = 10
epochs = 15

# 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 (http
s://s3.amazonaws.com/img-datasets/mnist.npz)
11493376/11490434 [==============================] - 2s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [0]:

```
# plotting train and test error

import matplotlib.pyplot as plt
%matplotlib inline

def plot_loss(training, epochs):
    fig, ax = plt.subplots(1,1)
    ax.set_xlabel('epoch')
    ax.set_ylabel('Categorical Crossentropy Loss')
    x_values = list(range(1,epochs+1))

    validation_loss = training.history['val_loss']
    train_loss = training.history['loss']

    ax.plot(x_values, validation_loss, 'b', label="Validation Loss")
    ax.plot(x_values, train_loss, 'r', label="Train Loss")
    plt.legend()
    plt.grid()
```

2 conv layers, kernels =(3x3), Max pooling =(2x2)

In [0]:

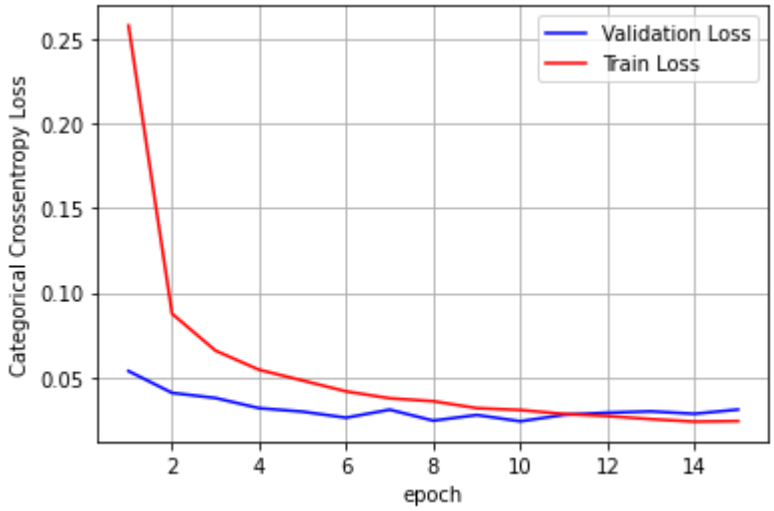
```
%%time

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
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.summary()

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

training = 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_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               1179776
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 147s 2ms/step - loss: 0.25
```

```
76 - accuracy: 0.9207 - val_loss: 0.0538 - val_accuracy: 0.9826
Epoch 2/15
60000/60000 [==============================] - 145s 2ms/step - loss: 0.08
76 - accuracy: 0.9737 - val_loss: 0.0409 - val_accuracy: 0.9856
Epoch 3/15
60000/60000 [==============================] - 144s 2ms/step - loss: 0.06
58 - accuracy: 0.9802 - val_loss: 0.0379 - val_accuracy: 0.9872
Epoch 4/15
60000/60000 [==============================] - 144s 2ms/step - loss: 0.05
46 - accuracy: 0.9835 - val_loss: 0.0319 - val_accuracy: 0.9890
Epoch 5/15
60000/60000 [==============================] - 145s 2ms/step - loss: 0.04
82 - accuracy: 0.9856 - val_loss: 0.0299 - val_accuracy: 0.9900
Epoch 6/15
60000/60000 [==============================] - 147s 2ms/step - loss: 0.04
18 - accuracy: 0.9872 - val_loss: 0.0263 - val_accuracy: 0.9908
Epoch 7/15
60000/60000 [==============================] - 147s 2ms/step - loss: 0.03
77 - accuracy: 0.9879 - val_loss: 0.0310 - val_accuracy: 0.9898
Epoch 8/15
60000/60000 [==============================] - 148s 2ms/step - loss: 0.03
60 - accuracy: 0.9892 - val_loss: 0.0246 - val_accuracy: 0.9917
Epoch 9/15
60000/60000 [==============================] - 148s 2ms/step - loss: 0.03
19 - accuracy: 0.9905 - val_loss: 0.0279 - val_accuracy: 0.9912
Epoch 10/15
60000/60000 [==============================] - 148s 2ms/step - loss: 0.03
08 - accuracy: 0.9905 - val_loss: 0.0242 - val_accuracy: 0.9921
Epoch 11/15
60000/60000 [==============================] - 151s 3ms/step - loss: 0.02
85 - accuracy: 0.9911 - val_loss: 0.0280 - val_accuracy: 0.9911
Epoch 12/15
60000/60000 [==============================] - 149s 2ms/step - loss: 0.02
73 - accuracy: 0.9913 - val_loss: 0.0292 - val_accuracy: 0.9914
Epoch 13/15
60000/60000 [==============================] - 145s 2ms/step - loss: 0.02
54 - accuracy: 0.9923 - val_loss: 0.0300 - val_accuracy: 0.9912
Epoch 14/15
60000/60000 [==============================] - 145s 2ms/step - loss: 0.02
40 - accuracy: 0.9927 - val_loss: 0.0286 - val_accuracy: 0.9915
Epoch 15/15
60000/60000 [==============================] - 150s 3ms/step - loss: 0.02
43 - accuracy: 0.9923 - val_loss: 0.0311 - val_accuracy: 0.9905
Test loss: 0.03108858480632407
Test accuracy: 0.9904999732971191
CPU times: user 1h 9min 25s, sys: 1min 15s, total: 1h 10min 40s
Wall time: 36min 50s
```

In [0]:

```
plot_loss(training, epochs)
```



from epoch 11 it starts overfitting. Best epoch is 10

2 conv layers, kernels =(5x5) max pooling = (2X2)

In [0]:

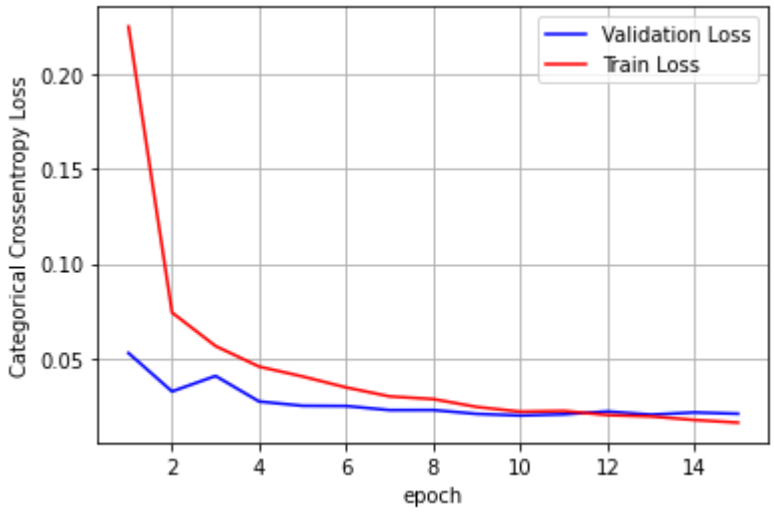
```
%%time
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

training = 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_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 24, 24, 32)        832
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
flatten_5 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               819328
_________________________________________________________________
dropout_10 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 10)                1290
=================================================================
Total params: 872,714
Trainable params: 872,714
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 232s 4ms/step - loss: 0.22
54 - accuracy: 0.9303 - val_loss: 0.0528 - val_accuracy: 0.9835
Epoch 2/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.07
```

```
42 - accuracy: 0.9779 - val_loss: 0.0325 - val_accuracy: 0.9886
Epoch 3/15
60000/60000 [==============================] - 229s 4ms/step - loss: 0.05
65 - accuracy: 0.9836 - val_loss: 0.0407 - val_accuracy: 0.9874
Epoch 4/15
60000/60000 [==============================] - 228s 4ms/step - loss: 0.04
56 - accuracy: 0.9867 - val_loss: 0.0272 - val_accuracy: 0.9906
Epoch 5/15
60000/60000 [==============================] - 229s 4ms/step - loss: 0.04
04 - accuracy: 0.9881 - val_loss: 0.0249 - val_accuracy: 0.9913
Epoch 6/15
60000/60000 [==============================] - 229s 4ms/step - loss: 0.03
46 - accuracy: 0.9894 - val_loss: 0.0248 - val_accuracy: 0.9916
Epoch 7/15
60000/60000 [==============================] - 234s 4ms/step - loss: 0.02
99 - accuracy: 0.9910 - val_loss: 0.0226 - val_accuracy: 0.9918
Epoch 8/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.02
85 - accuracy: 0.9916 - val_loss: 0.0227 - val_accuracy: 0.9920
Epoch 9/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.02
43 - accuracy: 0.9923 - val_loss: 0.0206 - val_accuracy: 0.9929
Epoch 10/15
60000/60000 [==============================] - 236s 4ms/step - loss: 0.02
18 - accuracy: 0.9934 - val_loss: 0.0198 - val_accuracy: 0.9932
Epoch 11/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.02
22 - accuracy: 0.9932 - val_loss: 0.0204 - val_accuracy: 0.9937
Epoch 12/15
60000/60000 [==============================] - 235s 4ms/step - loss: 0.02
00 - accuracy: 0.9936 - val_loss: 0.0219 - val_accuracy: 0.9930
Epoch 13/15
60000/60000 [==============================] - 237s 4ms/step - loss: 0.01
93 - accuracy: 0.9939 - val_loss: 0.0203 - val_accuracy: 0.9936
Epoch 14/15
60000/60000 [==============================] - 237s 4ms/step - loss: 0.01
74 - accuracy: 0.9945 - val_loss: 0.0214 - val_accuracy: 0.9929
Epoch 15/15
60000/60000 [==============================] - 236s 4ms/step - loss: 0.01
60 - accuracy: 0.9948 - val_loss: 0.0207 - val_accuracy: 0.9934
Test loss: 0.020748127230865065
Test accuracy: 0.993399977684021
CPU times: user 1h 50min, sys: 1min 20s, total: 1h 51min 21s
Wall time: 58min 12s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 11. It starts overfitting after 11 epochs. Here best accuracy is 99.37

3 conv layers, kernels = (7x7),(5x5),(3x3), max pooling =(2x2)

In [0]:

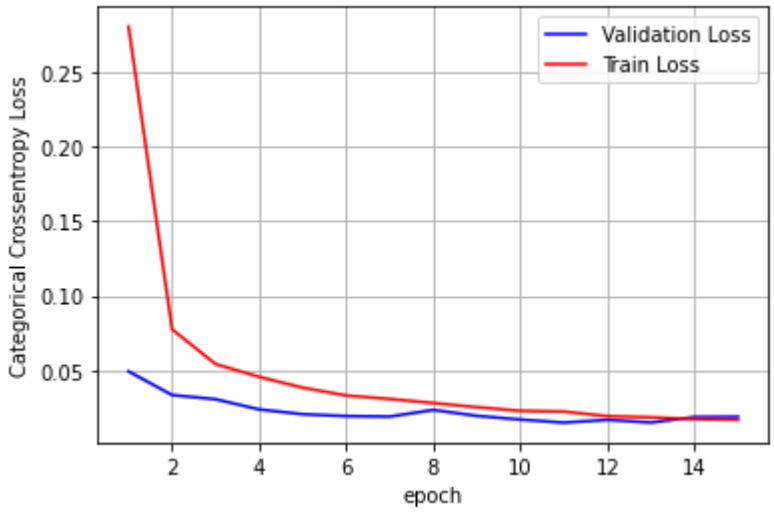
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

training = 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_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 22, 22, 32)        1600
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 18, 18, 64)        51264
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 9, 9, 64)          0
_________________________________________________________________
dropout_11 (Dropout)         (None, 9, 9, 64)          0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 7, 7, 128)         73856
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
dropout_12 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_11 (Dense)             (None, 512)               590336
_________________________________________________________________
dropout_13 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                5130
=================================================================
```

```
Total params: 722,186
Trainable params: 722,186
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.28
04 - accuracy: 0.9097 - val_loss: 0.0493 - val_accuracy: 0.9831
Epoch 2/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.07
75 - accuracy: 0.9762 - val_loss: 0.0334 - val_accuracy: 0.9890
Epoch 3/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.05
42 - accuracy: 0.9834 - val_loss: 0.0306 - val_accuracy: 0.9898
Epoch 4/15
60000/60000 [==============================] - 226s 4ms/step - loss: 0.04
56 - accuracy: 0.9862 - val_loss: 0.0238 - val_accuracy: 0.9922
Epoch 5/15
60000/60000 [==============================] - 227s 4ms/step - loss: 0.03
84 - accuracy: 0.9882 - val_loss: 0.0206 - val_accuracy: 0.9932
Epoch 6/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.03
31 - accuracy: 0.9899 - val_loss: 0.0193 - val_accuracy: 0.9941
Epoch 7/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.03
08 - accuracy: 0.9904 - val_loss: 0.0190 - val_accuracy: 0.9939
Epoch 8/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.02
81 - accuracy: 0.9912 - val_loss: 0.0234 - val_accuracy: 0.9922
Epoch 9/15
60000/60000 [==============================] - 237s 4ms/step - loss: 0.02
53 - accuracy: 0.9922 - val_loss: 0.0195 - val_accuracy: 0.9941
Epoch 10/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.02
29 - accuracy: 0.9928 - val_loss: 0.0170 - val_accuracy: 0.9939
Epoch 11/15
60000/60000 [==============================] - 227s 4ms/step - loss: 0.02
24 - accuracy: 0.9926 - val_loss: 0.0149 - val_accuracy: 0.9953
Epoch 12/15
60000/60000 [==============================] - 234s 4ms/step - loss: 0.01
92 - accuracy: 0.9938 - val_loss: 0.0168 - val_accuracy: 0.9950
Epoch 13/15
60000/60000 [==============================] - 229s 4ms/step - loss: 0.01
85 - accuracy: 0.9941 - val_loss: 0.0150 - val_accuracy: 0.9955
Epoch 14/15
60000/60000 [==============================] - 230s 4ms/step - loss: 0.01
73 - accuracy: 0.9948 - val_loss: 0.0187 - val_accuracy: 0.9941
Epoch 15/15
60000/60000 [==============================] - 233s 4ms/step - loss: 0.01
69 - accuracy: 0.9947 - val_loss: 0.0188 - val_accuracy: 0.9951
Test loss: 0.01883504462489873
Test accuracy: 0.9951000213623047
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 13. Its overfitting from epoch 14 onwards.Best accuracy is 99.55%

2 conv layers, kernels =(3x3),(3x3),Max pooling = (2x2), padding = same

In [0]:

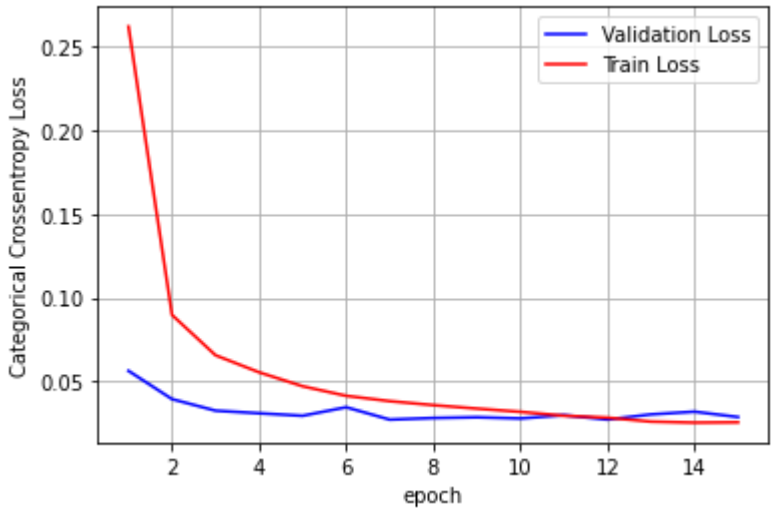
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu', padding ='same',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

training = 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_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_14 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
flatten_7 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               1605760
_________________________________________________________________
dropout_15 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                1290
=================================================================
Total params: 1,625,866
Trainable params: 1,625,866
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 193s 3ms/step - loss: 0.2618
- accuracy: 0.9196 - val_loss: 0.0563 - val_accuracy: 0.9819
Epoch 2/15
60000/60000 [==============================] - 190s 3ms/step - loss: 0.0897
- accuracy: 0.9735 - val_loss: 0.0394 - val_accuracy: 0.9866
Epoch 3/15
```

```
60000/60000 [==============================] - 190s 3ms/step - loss: 0.0656
- accuracy: 0.9803 - val_loss: 0.0325 - val_accuracy: 0.9885
Epoch 4/15
60000/60000 [==============================] - 190s 3ms/step - loss: 0.0554
- accuracy: 0.9833 - val_loss: 0.0310 - val_accuracy: 0.9894
Epoch 5/15
60000/60000 [==============================] - 190s 3ms/step - loss: 0.0471
- accuracy: 0.9860 - val_loss: 0.0295 - val_accuracy: 0.9900
Epoch 6/15
60000/60000 [==============================] - 194s 3ms/step - loss: 0.0414
- accuracy: 0.9875 - val_loss: 0.0346 - val_accuracy: 0.9887
Epoch 7/15
60000/60000 [==============================] - 194s 3ms/step - loss: 0.0382
- accuracy: 0.9885 - val_loss: 0.0272 - val_accuracy: 0.9905
Epoch 8/15
60000/60000 [==============================] - 197s 3ms/step - loss: 0.0358
- accuracy: 0.9890 - val_loss: 0.0281 - val_accuracy: 0.9910
Epoch 9/15
60000/60000 [==============================] - 199s 3ms/step - loss: 0.0338
- accuracy: 0.9897 - val_loss: 0.0286 - val_accuracy: 0.9912
Epoch 10/15
60000/60000 [==============================] - 203s 3ms/step - loss: 0.0318
- accuracy: 0.9903 - val_loss: 0.0279 - val_accuracy: 0.9917
Epoch 11/15
60000/60000 [==============================] - 210s 3ms/step - loss: 0.0295
- accuracy: 0.9912 - val_loss: 0.0298 - val_accuracy: 0.9908
Epoch 12/15
60000/60000 [==============================] - 206s 3ms/step - loss: 0.0283
- accuracy: 0.9917 - val_loss: 0.0272 - val_accuracy: 0.9921
Epoch 13/15
60000/60000 [==============================] - 197s 3ms/step - loss: 0.0260
- accuracy: 0.9918 - val_loss: 0.0303 - val_accuracy: 0.9904
Epoch 14/15
60000/60000 [==============================] - 200s 3ms/step - loss: 0.0254
- accuracy: 0.9919 - val_loss: 0.0319 - val_accuracy: 0.9920
Epoch 15/15
60000/60000 [==============================] - 196s 3ms/step - loss: 0.0256
- accuracy: 0.9923 - val_loss: 0.0287 - val_accuracy: 0.9917
Test loss: 0.028727352098264054
Test accuracy: 0.9916999936103821
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 10. Its overfitting from epoch 11. Best accuracy is 99.17%

3 conv layers, kernels = (7x7),(5x5),(3x3), padding =same

In [0]:

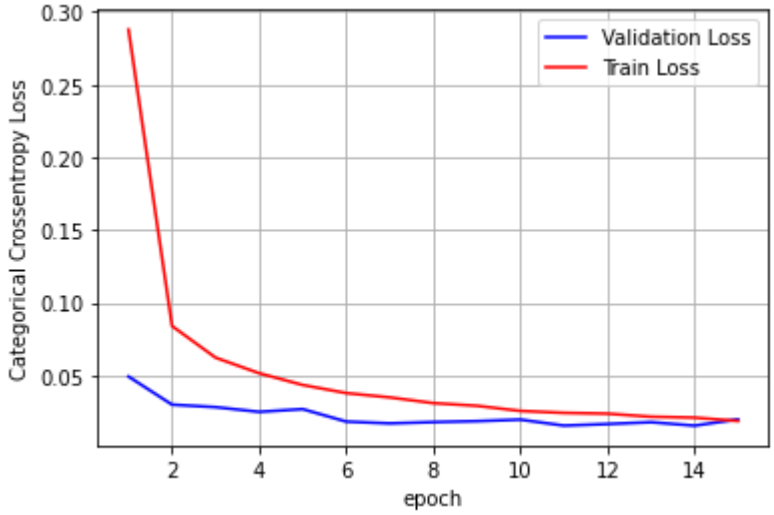
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu', padding ='same',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), activation = 'relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.3))
model.add(Dense(128, activation='relu'))

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

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

training = 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_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 28, 28, 32)        1600
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 28, 28, 64)        51264
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_16 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 14, 14, 128)       73856
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_17 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
dense_15 (Dense)             (None, 7, 7, 128)         16512
_________________________________________________________________
flatten_8 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               802944
_________________________________________________________________
dropout_18 (Dropout)         (None, 128)               0
_________________________________________________________________
```

```
dense_17 (Dense)             (None, 10)                1290
=================================================================
Total params: 947,466
Trainable params: 947,466
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 568s 9ms/step - loss: 0.28
78 - accuracy: 0.9090 - val_loss: 0.0494 - val_accuracy: 0.9842
Epoch 2/15
60000/60000 [==============================] - 575s 10ms/step - loss: 0.0
841 - accuracy: 0.9751 - val_loss: 0.0300 - val_accuracy: 0.9901
Epoch 3/15
60000/60000 [==============================] - 573s 10ms/step - loss: 0.0
623 - accuracy: 0.9818 - val_loss: 0.0283 - val_accuracy: 0.9905
Epoch 4/15
60000/60000 [==============================] - 563s 9ms/step - loss: 0.05
16 - accuracy: 0.9850 - val_loss: 0.0251 - val_accuracy: 0.9915
Epoch 5/15
60000/60000 [==============================] - 561s 9ms/step - loss: 0.04
36 - accuracy: 0.9873 - val_loss: 0.0270 - val_accuracy: 0.9907
Epoch 6/15
60000/60000 [==============================] - 567s 9ms/step - loss: 0.03
80 - accuracy: 0.9891 - val_loss: 0.0184 - val_accuracy: 0.9934
Epoch 7/15
60000/60000 [==============================] - 567s 9ms/step - loss: 0.03
50 - accuracy: 0.9897 - val_loss: 0.0172 - val_accuracy: 0.9943
Epoch 8/15
60000/60000 [==============================] - 571s 10ms/step - loss: 0.0
311 - accuracy: 0.9904 - val_loss: 0.0180 - val_accuracy: 0.9940
Epoch 9/15
60000/60000 [==============================] - 582s 10ms/step - loss: 0.0
293 - accuracy: 0.9913 - val_loss: 0.0187 - val_accuracy: 0.9936
Epoch 10/15
60000/60000 [==============================] - 635s 11ms/step - loss: 0.0
257 - accuracy: 0.9924 - val_loss: 0.0197 - val_accuracy: 0.9940
Epoch 11/15
60000/60000 [==============================] - 596s 10ms/step - loss: 0.0
244 - accuracy: 0.9926 - val_loss: 0.0156 - val_accuracy: 0.9948
Epoch 12/15
60000/60000 [==============================] - 569s 9ms/step - loss: 0.02
38 - accuracy: 0.9930 - val_loss: 0.0167 - val_accuracy: 0.9940
Epoch 13/15
60000/60000 [==============================] - 573s 10ms/step - loss: 0.0
217 - accuracy: 0.9937 - val_loss: 0.0180 - val_accuracy: 0.9942
Epoch 14/15
60000/60000 [==============================] - 577s 10ms/step - loss: 0.0
211 - accuracy: 0.9938 - val_loss: 0.0156 - val_accuracy: 0.9946
Epoch 15/15
60000/60000 [==============================] - 591s 10ms/step - loss: 0.0
188 - accuracy: 0.9946 - val_loss: 0.0199 - val_accuracy: 0.9931
Test loss: 0.01988242387709852
Test accuracy: 0.9930999875068665
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 7. Best accuracy 99.43%

3 conv layers, kernels=(5x5), padding = same, strides= 1

In [0]:

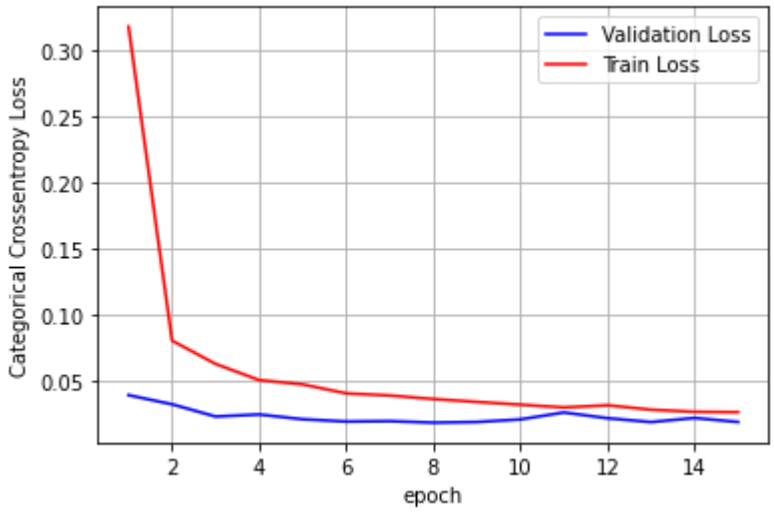
```
%%time
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5),strides=1, activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(80, (5,5), activation= 'relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding = 'same',strides=1))
model.add(Dropout(0.75))

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

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

training = 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_19 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
dropout_19 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 6, 6, 80)          128080
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 6, 6, 80)          0
_________________________________________________________________
dropout_20 (Dropout)         (None, 6, 6, 80)          0
_________________________________________________________________
flatten_9 (Flatten)          (None, 2880)              0
_________________________________________________________________
dense_18 (Dense)             (None, 128)               368768
_________________________________________________________________
dropout_21 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_19 (Dense)             (None, 10)                1290
=================================================================
```

```
Total params: 550,234
Trainable params: 550,234
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 286s 5ms/step - loss: 0.3174
- accuracy: 0.8973 - val_loss: 0.0386 - val_accuracy: 0.9877
Epoch 2/15
60000/60000 [==============================] - 281s 5ms/step - loss: 0.0798
- accuracy: 0.9762 - val_loss: 0.0317 - val_accuracy: 0.9899
Epoch 3/15
60000/60000 [==============================] - 275s 5ms/step - loss: 0.0622
- accuracy: 0.9821 - val_loss: 0.0224 - val_accuracy: 0.9926
Epoch 4/15
60000/60000 [==============================] - 275s 5ms/step - loss: 0.0500
- accuracy: 0.9855 - val_loss: 0.0240 - val_accuracy: 0.9919
Epoch 5/15
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0467
- accuracy: 0.9860 - val_loss: 0.0204 - val_accuracy: 0.9935
Epoch 6/15
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0399
- accuracy: 0.9884 - val_loss: 0.0186 - val_accuracy: 0.9941
Epoch 7/15
60000/60000 [==============================] - 277s 5ms/step - loss: 0.0383
- accuracy: 0.9891 - val_loss: 0.0189 - val_accuracy: 0.9943
Epoch 8/15
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0357
- accuracy: 0.9895 - val_loss: 0.0178 - val_accuracy: 0.9939
Epoch 9/15
60000/60000 [==============================] - 275s 5ms/step - loss: 0.0335
- accuracy: 0.9904 - val_loss: 0.0183 - val_accuracy: 0.9942
Epoch 10/15
60000/60000 [==============================] - 281s 5ms/step - loss: 0.0314
- accuracy: 0.9908 - val_loss: 0.0202 - val_accuracy: 0.9940
Epoch 11/15
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0293
- accuracy: 0.9911 - val_loss: 0.0255 - val_accuracy: 0.9930
Epoch 12/15
60000/60000 [==============================] - 283s 5ms/step - loss: 0.0309
- accuracy: 0.9909 - val_loss: 0.0212 - val_accuracy: 0.9929
Epoch 13/15
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0276
- accuracy: 0.9919 - val_loss: 0.0182 - val_accuracy: 0.9944
Epoch 14/15
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0260
- accuracy: 0.9925 - val_loss: 0.0213 - val_accuracy: 0.9941
Epoch 15/15
60000/60000 [==============================] - 279s 5ms/step - loss: 0.0257
- accuracy: 0.9931 - val_loss: 0.0183 - val_accuracy: 0.9944
Test loss: 0.018290850276246533
Test accuracy: 0.9944000244140625
CPU times: user 2h 11min 45s, sys: 1min 44s, total: 2h 13min 29s
Wall time: 1h 9min 40s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 9. Best accuracy 99.42%

3 conv layers, kernels =(7x7), padding = same, strides =1,2

In [0]:

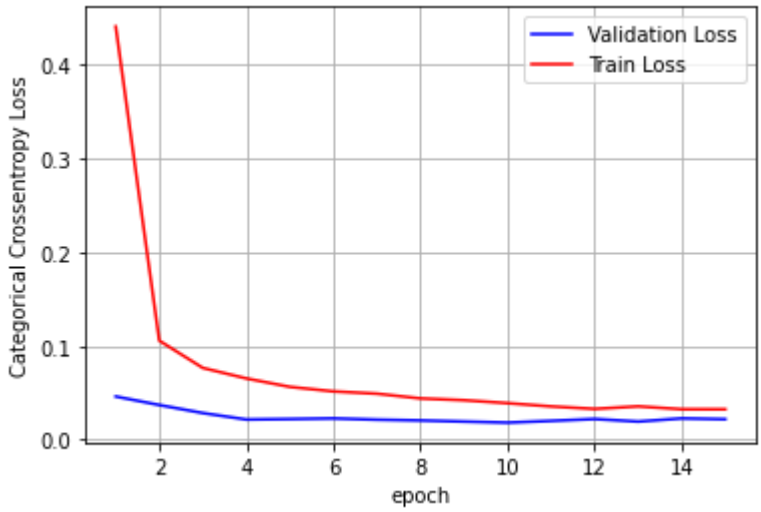
```
%%time
model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu', padding ='same',strides=1,
                 input_shape=input_shape))
model.add(Conv2D(64, (7, 7), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(80, (7,7), activation= 'relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding = 'same',strides=2))
model.add(Dropout(0.75))

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

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

training = 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_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        1600
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 22, 22, 64)        100416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 11, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 11, 11, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 5, 80)          250960
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 80)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 3, 3, 80)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 720)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               92288
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
```

```
Total params: 446,554
Trainable params: 446,554
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 568s 9ms/step - loss: 0.44
05 - accuracy: 0.8557 - val_loss: 0.0462 - val_accuracy: 0.9856
Epoch 2/15
60000/60000 [==============================] - 566s 9ms/step - loss: 0.10
57 - accuracy: 0.9701 - val_loss: 0.0371 - val_accuracy: 0.9880
Epoch 3/15
60000/60000 [==============================] - 566s 9ms/step - loss: 0.07
66 - accuracy: 0.9791 - val_loss: 0.0285 - val_accuracy: 0.9914
Epoch 4/15
60000/60000 [==============================] - 572s 10ms/step - loss: 0.0
654 - accuracy: 0.9825 - val_loss: 0.0217 - val_accuracy: 0.9932
Epoch 5/15
60000/60000 [==============================] - 573s 10ms/step - loss: 0.0
564 - accuracy: 0.9847 - val_loss: 0.0222 - val_accuracy: 0.9931
Epoch 6/15
60000/60000 [==============================] - 568s 9ms/step - loss: 0.05
16 - accuracy: 0.9860 - val_loss: 0.0228 - val_accuracy: 0.9927
Epoch 7/15
60000/60000 [==============================] - 566s 9ms/step - loss: 0.04
92 - accuracy: 0.9868 - val_loss: 0.0215 - val_accuracy: 0.9933
Epoch 8/15
60000/60000 [==============================] - 566s 9ms/step - loss: 0.04
41 - accuracy: 0.9880 - val_loss: 0.0206 - val_accuracy: 0.9932
Epoch 9/15
60000/60000 [==============================] - 569s 9ms/step - loss: 0.04
22 - accuracy: 0.9885 - val_loss: 0.0195 - val_accuracy: 0.9942
Epoch 10/15
60000/60000 [==============================] - 574s 10ms/step - loss: 0.0
391 - accuracy: 0.9886 - val_loss: 0.0183 - val_accuracy: 0.9942
Epoch 11/15
60000/60000 [==============================] - 574s 10ms/step - loss: 0.0
356 - accuracy: 0.9902 - val_loss: 0.0202 - val_accuracy: 0.9937
Epoch 12/15
60000/60000 [==============================] - 566s 9ms/step - loss: 0.03
30 - accuracy: 0.9909 - val_loss: 0.0222 - val_accuracy: 0.9938
Epoch 13/15
60000/60000 [==============================] - 564s 9ms/step - loss: 0.03
57 - accuracy: 0.9895 - val_loss: 0.0195 - val_accuracy: 0.9940
Epoch 14/15
60000/60000 [==============================] - 567s 9ms/step - loss: 0.03
26 - accuracy: 0.9905 - val_loss: 0.0228 - val_accuracy: 0.9939
Epoch 15/15
60000/60000 [==============================] - 566s 9ms/step - loss: 0.03
26 - accuracy: 0.9908 - val_loss: 0.0220 - val_accuracy: 0.9930
Test loss: 0.02198781901857328
Test accuracy: 0.9929999709129333
CPU times: user 4h 33min 20s, sys: 3min 5s, total: 4h 36min 25s
Wall time: 2h 22min 26s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 5. best accuracy 99.31%

3 conv layers, kernels = (7x7), max pooling =(2x2), strides=2,1

In [0]:

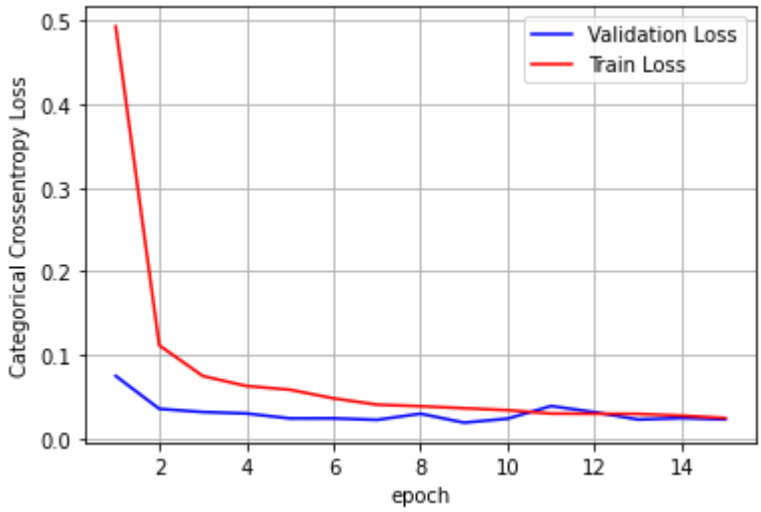
```
%%time
model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (7, 7), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2),strides=2))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(80, (7,7), activation= 'relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding = 'same',strides=1))
model.add(BatchNormalization())
model.add(Dropout(0.75))

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

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

training = 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_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 22, 22, 32)        1600
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 64)        100416
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
dropout_4 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 2, 2, 80)          250960
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 80)          0
_________________________________________________________________
batch_normalization_2 (Batch (None, 2, 2, 80)          320
_________________________________________________________________
dropout_5 (Dropout)          (None, 2, 2, 80)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 320)               0
_________________________________________________________________
```

```
dense_3 (Dense)              (None, 128)               41088
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 395,930
Trainable params: 395,642
Non-trainable params: 288
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 323s 5ms/step - loss: 0.4930
- accuracy: 0.8647 - val_loss: 0.0750 - val_accuracy: 0.9799
Epoch 2/15
60000/60000 [==============================] - 318s 5ms/step - loss: 0.1116
- accuracy: 0.9691 - val_loss: 0.0356 - val_accuracy: 0.9884
Epoch 3/15
60000/60000 [==============================] - 320s 5ms/step - loss: 0.0749
- accuracy: 0.9794 - val_loss: 0.0318 - val_accuracy: 0.9908
Epoch 4/15
60000/60000 [==============================] - 316s 5ms/step - loss: 0.0630
- accuracy: 0.9823 - val_loss: 0.0301 - val_accuracy: 0.9914
Epoch 5/15
60000/60000 [==============================] - 315s 5ms/step - loss: 0.0585
- accuracy: 0.9840 - val_loss: 0.0242 - val_accuracy: 0.9926
Epoch 6/15
60000/60000 [==============================] - 316s 5ms/step - loss: 0.0482
- accuracy: 0.9866 - val_loss: 0.0243 - val_accuracy: 0.9931
Epoch 7/15
60000/60000 [==============================] - 317s 5ms/step - loss: 0.0407
- accuracy: 0.9885 - val_loss: 0.0225 - val_accuracy: 0.9934
Epoch 8/15
60000/60000 [==============================] - 317s 5ms/step - loss: 0.0387
- accuracy: 0.9891 - val_loss: 0.0298 - val_accuracy: 0.9920
Epoch 9/15
60000/60000 [==============================] - 318s 5ms/step - loss: 0.0364
- accuracy: 0.9897 - val_loss: 0.0191 - val_accuracy: 0.9939
Epoch 10/15
60000/60000 [==============================] - 318s 5ms/step - loss: 0.0340
- accuracy: 0.9905 - val_loss: 0.0239 - val_accuracy: 0.9931
Epoch 11/15
60000/60000 [==============================] - 318s 5ms/step - loss: 0.0300
- accuracy: 0.9914 - val_loss: 0.0390 - val_accuracy: 0.9888
Epoch 12/15
60000/60000 [==============================] - 323s 5ms/step - loss: 0.0297
- accuracy: 0.9918 - val_loss: 0.0316 - val_accuracy: 0.9919
Epoch 13/15
60000/60000 [==============================] - 317s 5ms/step - loss: 0.0295
- accuracy: 0.9919 - val_loss: 0.0229 - val_accuracy: 0.9920
Epoch 14/15
60000/60000 [==============================] - 320s 5ms/step - loss: 0.0274
- accuracy: 0.9924 - val_loss: 0.0245 - val_accuracy: 0.9915
Epoch 15/15
60000/60000 [==============================] - 312s 5ms/step - loss: 0.0246
- accuracy: 0.9930 - val_loss: 0.0230 - val_accuracy: 0.9934
Test loss: 0.023033796002792678
Test accuracy: 0.993399977684021
CPU times: user 2h 26min 53s, sys: 2min 41s, total: 2h 29min 35s
Wall time: 1h 19min 39s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 7. Best accuracy 99.34%

3 conv layers, kernels = (5x5), strides =1, max pooling = (2x2), padding = same

In [0]:

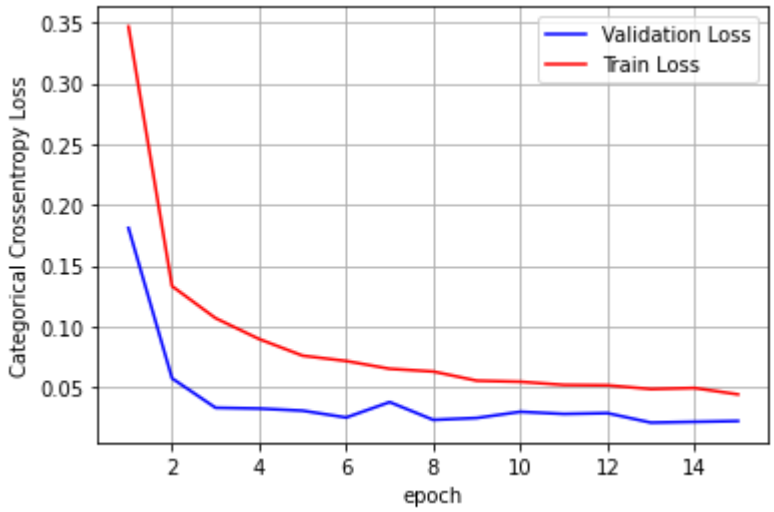
```
%%time

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu', strides =1,
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5),strides=1, activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(80, (5,5), activation= 'relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding = 'same',strides=1))
model.add(BatchNormalization())
model.add(Dropout(0.75))

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

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

training = 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_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 24, 24, 32)        832
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
dropout_7 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 6, 6, 80)          128080
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 80)          0
_________________________________________________________________
batch_normalization_4 (Batch (None, 6, 6, 80)          320
_________________________________________________________________
dropout_8 (Dropout)          (None, 6, 6, 80)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 2880)              0
```

```
_________________________________________________________________
dense_5 (Dense)              (None, 128)               368768    
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290      
=================================================================
Total params: 550,810
Trainable params: 550,522
Non-trainable params: 288
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 292s 5ms/step - loss: 0.3540
- accuracy: 0.9021 - val_loss: 0.2448 - val_accuracy: 0.9415
Epoch 2/15
60000/60000 [==============================] - 291s 5ms/step - loss: 0.1071
- accuracy: 0.9697 - val_loss: 0.0404 - val_accuracy: 0.9883
Epoch 3/15
60000/60000 [==============================] - 290s 5ms/step - loss: 0.0823
- accuracy: 0.9772 - val_loss: 0.0255 - val_accuracy: 0.9911
Epoch 4/15
60000/60000 [==============================] - 288s 5ms/step - loss: 0.0699
- accuracy: 0.9796 - val_loss: 0.0366 - val_accuracy: 0.9885
Epoch 5/15
60000/60000 [==============================] - 288s 5ms/step - loss: 0.0613
- accuracy: 0.9823 - val_loss: 0.0364 - val_accuracy: 0.9886
Epoch 6/15
60000/60000 [==============================] - 288s 5ms/step - loss: 0.0561
- accuracy: 0.9842 - val_loss: 0.0276 - val_accuracy: 0.9921
Epoch 7/15
60000/60000 [==============================] - 294s 5ms/step - loss: 0.0512
- accuracy: 0.9853 - val_loss: 0.0356 - val_accuracy: 0.9900
Epoch 8/15
60000/60000 [==============================] - 289s 5ms/step - loss: 0.0472
- accuracy: 0.9869 - val_loss: 0.0209 - val_accuracy: 0.9931
Epoch 9/15
60000/60000 [==============================] - 293s 5ms/step - loss: 0.0467
- accuracy: 0.9873 - val_loss: 0.0194 - val_accuracy: 0.9940
Epoch 10/15
60000/60000 [==============================] - 287s 5ms/step - loss: 0.0436
- accuracy: 0.9877 - val_loss: 0.0295 - val_accuracy: 0.9917
Epoch 11/15
60000/60000 [==============================] - 291s 5ms/step - loss: 0.0429
- accuracy: 0.9876 - val_loss: 0.0171 - val_accuracy: 0.9948
Epoch 12/15
60000/60000 [==============================] - 287s 5ms/step - loss: 0.0398
- accuracy: 0.9884 - val_loss: 0.0234 - val_accuracy: 0.9942
Epoch 13/15
60000/60000 [==============================] - 294s 5ms/step - loss: 0.0358
- accuracy: 0.9901 - val_loss: 0.0196 - val_accuracy: 0.9943
Epoch 14/15
60000/60000 [==============================] - 289s 5ms/step - loss: 0.0355
- accuracy: 0.9900 - val_loss: 0.0247 - val_accuracy: 0.9928
Epoch 15/15
60000/60000 [==============================] - 293s 5ms/step - loss: 0.0344
- accuracy: 0.9903 - val_loss: 0.0252 - val_accuracy: 0.9930
Test loss: 0.02517386771799874
Test accuracy: 0.9929999709129333
CPU times: user 2h 16min 3s, sys: 2min 40s, total: 2h 18min 43s
Wall time: 1h 12min 44s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 9. best accuracy 99.40%

3 conv layers, kernels=(3x3), max pooling =(2x2), padding =same, strides = 1

In [0]:

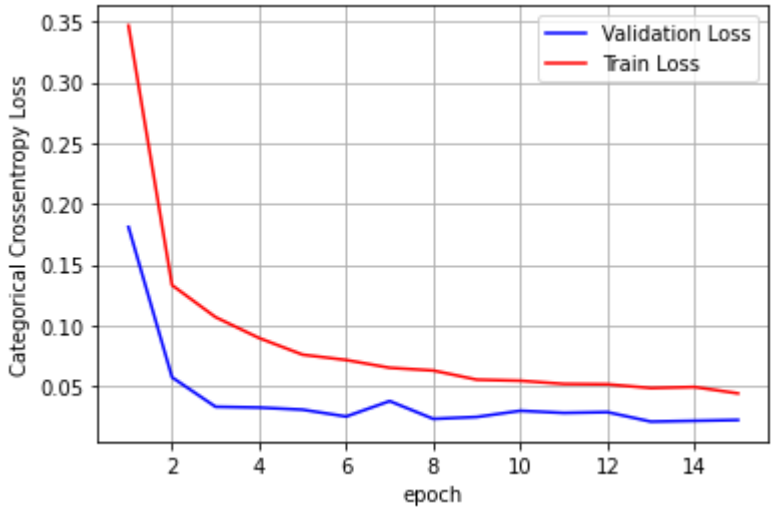
```
%%time

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3),strides=1, activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Conv2D(80, (3,3), activation= 'relu'))
model.add(MaxPooling2D(pool_size=(3,3),padding = 'same',strides=1))
model.add(BatchNormalization())
model.add(Dropout(0.75))

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

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

training = 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_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
batch_normalization_5 (Batch (None, 12, 12, 64)        256
_________________________________________________________________
dropout_10 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 10, 10, 80)        46160
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 10, 10, 80)        0
_________________________________________________________________
batch_normalization_6 (Batch (None, 10, 10, 80)        320
_________________________________________________________________
dropout_11 (Dropout)         (None, 10, 10, 80)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 8000)              0
```

```
_________________________________________________________________
dense_7 (Dense)              (None, 128)               1024128
_________________________________________________________________
dropout_12 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,090,970
Trainable params: 1,090,682
Non-trainable params: 288
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 209s 3ms/step - loss: 0.3466
- accuracy: 0.9030 - val_loss: 0.1810 - val_accuracy: 0.9499
Epoch 2/15
60000/60000 [==============================] - 209s 3ms/step - loss: 0.1333
- accuracy: 0.9634 - val_loss: 0.0575 - val_accuracy: 0.9825
Epoch 3/15
60000/60000 [==============================] - 210s 3ms/step - loss: 0.1070
- accuracy: 0.9704 - val_loss: 0.0333 - val_accuracy: 0.9899
Epoch 4/15
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0898
- accuracy: 0.9760 - val_loss: 0.0327 - val_accuracy: 0.9907
Epoch 5/15
60000/60000 [==============================] - 211s 4ms/step - loss: 0.0761
- accuracy: 0.9781 - val_loss: 0.0310 - val_accuracy: 0.9913
Epoch 6/15
60000/60000 [==============================] - 215s 4ms/step - loss: 0.0718
- accuracy: 0.9803 - val_loss: 0.0253 - val_accuracy: 0.9932
Epoch 7/15
60000/60000 [==============================] - 209s 3ms/step - loss: 0.0654
- accuracy: 0.9818 - val_loss: 0.0380 - val_accuracy: 0.9897
Epoch 8/15
60000/60000 [==============================] - 209s 3ms/step - loss: 0.0631
- accuracy: 0.9822 - val_loss: 0.0234 - val_accuracy: 0.9937
Epoch 9/15
60000/60000 [==============================] - 214s 4ms/step - loss: 0.0556
- accuracy: 0.9833 - val_loss: 0.0250 - val_accuracy: 0.9928
Epoch 10/15
60000/60000 [==============================] - 208s 3ms/step - loss: 0.0547
- accuracy: 0.9844 - val_loss: 0.0300 - val_accuracy: 0.9927
Epoch 11/15
60000/60000 [==============================] - 209s 3ms/step - loss: 0.0521
- accuracy: 0.9851 - val_loss: 0.0283 - val_accuracy: 0.9928
Epoch 12/15
60000/60000 [==============================] - 210s 3ms/step - loss: 0.0518
- accuracy: 0.9856 - val_loss: 0.0289 - val_accuracy: 0.9920
Epoch 13/15
60000/60000 [==============================] - 209s 3ms/step - loss: 0.0487
- accuracy: 0.9862 - val_loss: 0.0211 - val_accuracy: 0.9932
Epoch 14/15
60000/60000 [==============================] - 210s 3ms/step - loss: 0.0496
- accuracy: 0.9866 - val_loss: 0.0218 - val_accuracy: 0.9930
Epoch 15/15
60000/60000 [==============================] - 212s 4ms/step - loss: 0.0443
- accuracy: 0.9871 - val_loss: 0.0225 - val_accuracy: 0.9939
Test loss: 0.022474063225078576
Test accuracy: 0.9939000010490417
CPU times: user 1h 39min 13s, sys: 2min 8s, total: 1h 41min 21s
Wall time: 52min 45s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 8. best accuracy 99.37%

Summary

In [1]:

```
from prettytable import PrettyTable
pt = PrettyTable()
pt.field_names = ["Layers","Kernels", "Test Accuracy","epoch"]
pt.add_row(["2","(3x3),(3x3)","99.21","10"])
pt.add_row(["2","(5x5),(5x5)","99.37","11"])
pt.add_row(["3","(7x7),(5x5),(3x3)","99.55","13"])
print(pt)
```

```
+--------+------------------+---------------+-------+
| Layers |      Kernels     | Test Accuracy | epoch |
+--------+------------------+---------------+-------+
|   2    |    (3x3),(3x3)   |     99.21     |   10  |
|   2    |    (5x5),(5x5)   |     99.37     |   11  |
|   3    | (7x7),(5x5),(3x3) |     99.55     |   13  |
+--------+------------------+---------------+-------+
```

In [2]:

```
pt = PrettyTable()
pt.field_names = ["Layers","Kernels", "Test Accuracy","epoch", "padding"]
pt.add_row(["2","(3x3),(3x3)","99.17","10","same"])
pt.add_row(["3","(7x7),(5x5),(3x3)","99.43","7","same"])
print(pt)
```

```
+--------+-------------------+---------------+-------+---------+
| Layers |      Kernels      | Test Accuracy | epoch | padding |
+--------+-------------------+---------------+-------+---------+
|   2    |    (3x3),(3x3)    |     99.17     |   10  |   same  |
|   3    | (7x7),(5x5),(3x3) |     99.43     |   7   |   same  |
+--------+-------------------+---------------+-------+---------+
```

In [3]:

```
pt = PrettyTable()
pt.field_names = ["Layers","Kernels", "Test Accuracy", "epoch", "padding","strides"]
pt.add_row(["3","(3x3),(3x3),(3x3)","99.42","9","same","1"])
pt.add_row(["3","(5x5),(5x5),(5x5)","99.34","5","same","1"])
pt.add_row(["3","(7x7),(7x7),(7x7)","99.40","9","same","2"])
pt.add_row(["3","(7x7),(7x7),(7x7)","99.37","8","same","1,2"])
print(pt)
```

```
+--------+-------------------+---------------+-------+---------+---------+
| Layers |      Kernels      | Test Accuracy | epoch | padding | strides |
+--------+-------------------+---------------+-------+---------+---------+
|   3    | (3x3),(3x3),(3x3) |     99.42     |   9   |   same  |    1    |
|   3    | (5x5),(5x5),(5x5) |     99.34     |   5   |   same  |    1    |
|   3    | (7x7),(7x7),(7x7) |     99.40     |   9   |   same  |    2    |
|   3    | (7x7),(7x7),(7x7) |     99.37     |   8   |   same  |   1,2   |
+--------+-------------------+---------------+-------+---------+---------+
```

3 conv layers, kernels = (7x7),(5x5),(3x3), Using sigmoid

In [0]:

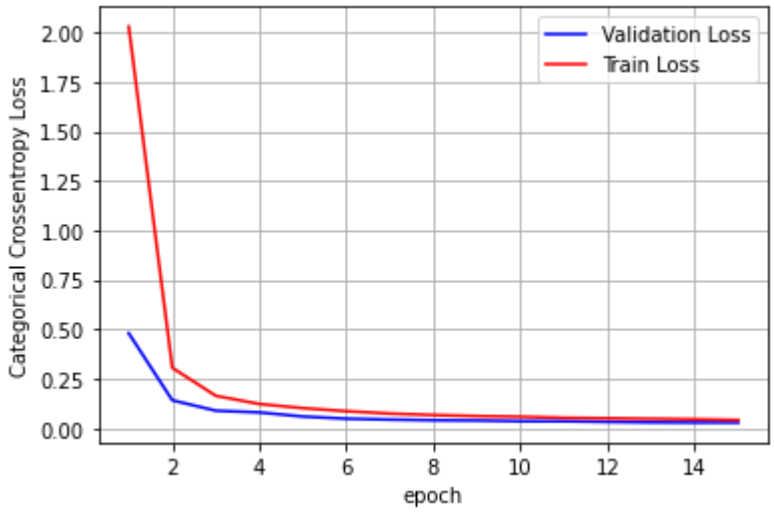
```
%%time

model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), activation='sigmoid'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(512, activation='sigmoid'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

training = 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_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 22, 22, 32)        1600
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 18, 18, 64)        51264
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 9, 9, 64)          0
_________________________________________________________________
dropout_7 (Dropout)          (None, 9, 9, 64)          0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 7, 7, 128)         73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
dropout_8 (Dropout)          (None, 3, 3, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               590336
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
```

```
dense_6 (Dense)              (None, 10)                5130
=================================================================
Total params: 722,186
Trainable params: 722,186
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 226s 4ms/step - loss: 2.02
95 - accuracy: 0.2398 - val_loss: 0.4800 - val_accuracy: 0.8586
Epoch 2/15
60000/60000 [==============================] - 226s 4ms/step - loss: 0.30
56 - accuracy: 0.9069 - val_loss: 0.1415 - val_accuracy: 0.9534
Epoch 3/15
60000/60000 [==============================] - 227s 4ms/step - loss: 0.16
39 - accuracy: 0.9489 - val_loss: 0.0888 - val_accuracy: 0.9737
Epoch 4/15
60000/60000 [==============================] - 227s 4ms/step - loss: 0.12
28 - accuracy: 0.9620 - val_loss: 0.0801 - val_accuracy: 0.9747
Epoch 5/15
60000/60000 [==============================] - 226s 4ms/step - loss: 0.10
18 - accuracy: 0.9679 - val_loss: 0.0593 - val_accuracy: 0.9810
Epoch 6/15
60000/60000 [==============================] - 223s 4ms/step - loss: 0.08
63 - accuracy: 0.9727 - val_loss: 0.0484 - val_accuracy: 0.9826
Epoch 7/15
60000/60000 [==============================] - 223s 4ms/step - loss: 0.07
43 - accuracy: 0.9766 - val_loss: 0.0439 - val_accuracy: 0.9850
Epoch 8/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.06
68 - accuracy: 0.9791 - val_loss: 0.0404 - val_accuracy: 0.9865
Epoch 9/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.06
15 - accuracy: 0.9810 - val_loss: 0.0391 - val_accuracy: 0.9871
Epoch 10/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.05
83 - accuracy: 0.9818 - val_loss: 0.0358 - val_accuracy: 0.9880
Epoch 11/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.05
27 - accuracy: 0.9834 - val_loss: 0.0355 - val_accuracy: 0.9880
Epoch 12/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.05
05 - accuracy: 0.9837 - val_loss: 0.0324 - val_accuracy: 0.9884
Epoch 13/15
60000/60000 [==============================] - 223s 4ms/step - loss: 0.04
79 - accuracy: 0.9847 - val_loss: 0.0299 - val_accuracy: 0.9903
Epoch 14/15
60000/60000 [==============================] - 223s 4ms/step - loss: 0.04
53 - accuracy: 0.9850 - val_loss: 0.0287 - val_accuracy: 0.9901
Epoch 15/15
60000/60000 [==============================] - 223s 4ms/step - loss: 0.04
19 - accuracy: 0.9872 - val_loss: 0.0285 - val_accuracy: 0.9902
Test loss: 0.028495841736346485
Test accuracy: 0.9901999831199646
CPU times: user 1h 46min 58s, sys: 1min 31s, total: 1h 48min 29s
Wall time: 56min 17s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 15. best accuracy is 99.02%

3 conv layers, kernels = (7x7),(5x5),(3x3), Using tanh

In [0]:

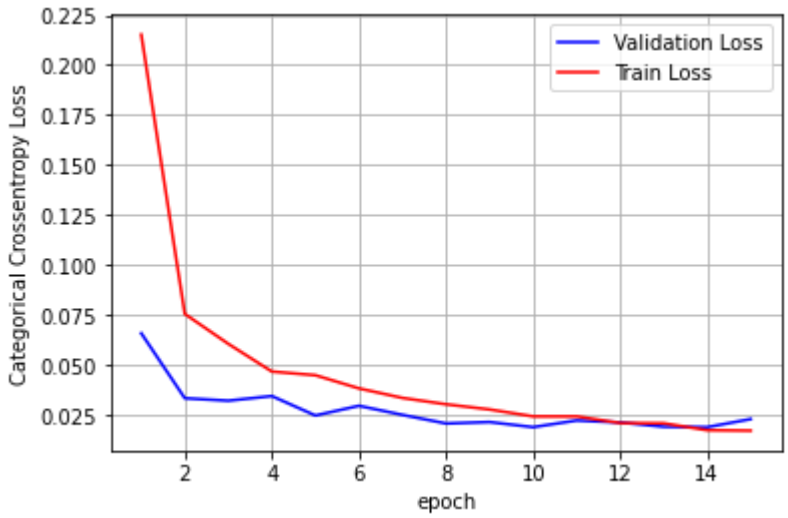
```
%%time

model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), activation='tanh'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(512, activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

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

training = 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_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 22, 22, 32)        1600
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 18, 18, 64)        51264
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 9, 9, 64)          0
_________________________________________________________________
dropout_10 (Dropout)         (None, 9, 9, 64)          0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 7, 7, 128)         73856
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
dropout_11 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_7 (Dense)              (None, 512)               590336
_________________________________________________________________
dropout_12 (Dropout)         (None, 512)               0
_________________________________________________________________
```

```
dense_8 (Dense)              (None, 10)                5130
=================================================================
Total params: 722,186
Trainable params: 722,186
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.21
53 - accuracy: 0.9316 - val_loss: 0.0658 - val_accuracy: 0.9805
Epoch 2/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.07
55 - accuracy: 0.9766 - val_loss: 0.0332 - val_accuracy: 0.9889
Epoch 3/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.06
05 - accuracy: 0.9811 - val_loss: 0.0320 - val_accuracy: 0.9889
Epoch 4/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.04
66 - accuracy: 0.9855 - val_loss: 0.0343 - val_accuracy: 0.9895
Epoch 5/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.04
48 - accuracy: 0.9862 - val_loss: 0.0246 - val_accuracy: 0.9930
Epoch 6/15
60000/60000 [==============================] - 225s 4ms/step - loss: 0.03
82 - accuracy: 0.9878 - val_loss: 0.0295 - val_accuracy: 0.9909
Epoch 7/15
60000/60000 [==============================] - 226s 4ms/step - loss: 0.03
34 - accuracy: 0.9896 - val_loss: 0.0249 - val_accuracy: 0.9922
Epoch 8/15
60000/60000 [==============================] - 226s 4ms/step - loss: 0.03
02 - accuracy: 0.9901 - val_loss: 0.0206 - val_accuracy: 0.9932
Epoch 9/15
60000/60000 [==============================] - 227s 4ms/step - loss: 0.02
76 - accuracy: 0.9911 - val_loss: 0.0213 - val_accuracy: 0.9937
Epoch 10/15
60000/60000 [==============================] - 226s 4ms/step - loss: 0.02
42 - accuracy: 0.9924 - val_loss: 0.0188 - val_accuracy: 0.9945
Epoch 11/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.02
41 - accuracy: 0.9921 - val_loss: 0.0221 - val_accuracy: 0.9932
Epoch 12/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.02
08 - accuracy: 0.9930 - val_loss: 0.0211 - val_accuracy: 0.9935
Epoch 13/15
60000/60000 [==============================] - 223s 4ms/step - loss: 0.02
07 - accuracy: 0.9934 - val_loss: 0.0190 - val_accuracy: 0.9936
Epoch 14/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.01
72 - accuracy: 0.9941 - val_loss: 0.0188 - val_accuracy: 0.9947
Epoch 15/15
60000/60000 [==============================] - 224s 4ms/step - loss: 0.01
70 - accuracy: 0.9944 - val_loss: 0.0228 - val_accuracy: 0.9925
Test loss: 0.02281436841523082
Test accuracy: 0.9925000071525574
CPU times: user 1h 46min 56s, sys: 1min 34s, total: 1h 48min 30s
Wall time: 56min 25s
```

In [0]:

```
# plotting train and test error
plot_loss(training, epochs)
```



best epoch is 5. best accuracy is 99.30%

In [4]:

```
pt = PrettyTable()
pt.field_names = ["Layers","Kernels", "Test Accuracy", "epoch","activation"]
pt.add_row(["2","(7x7),(5x5),(3x3)","99.02","15","sigmoid"])
pt.add_row(["3","(7x7),(5x5),(3x3)","99.30","5","tanh"])
print(pt)
```

```
+--------+-------------------+---------------+-------+------------+
| Layers |      Kernels      | Test Accuracy | epoch | activation |
+--------+-------------------+---------------+-------+------------+
|   2    | (7x7),(5x5),(3x3) |     99.02     |   15  |  sigmoid   |
|   3    | (7x7),(5x5),(3x3) |     99.30     |   5   |    tanh    |
+--------+-------------------+---------------+-------+------------+
```

As number of convolution layers increase test accuracy also increased.