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

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)

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

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])
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic:
more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated.
Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Pleas
e use tf.compat.v1.placeholder instead.

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

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:148: The name tf.placeholder_with_default is depr
ecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:3733: calling dropout (from tensorflow.python.op
s.nn_ops) with keep_prob is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optim
izers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.com
pat.v1.train.Optimizer instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_
core/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array
_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Pleas
e use tf.compat.v1.assign_add instead.

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

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

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:190: The name tf.get_default_session is deprecate
```

```
d. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Pleas
e use tf.compat.v1.ConfigProto instead.

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

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:216: The name tf.is_variable_initialized is depre
cated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backe
nd/tensorflow_backend.py:223: The name tf.variables_initializer is depreca
ted. Please use tf.compat.v1.variables_initializer instead.

60000/60000 [==============================] - 163s 3ms/step - loss: 0.266
0 - acc: 0.9183 - val_loss: 0.0559 - val_acc: 0.9824
Epoch 2/12
60000/60000 [==============================] - 162s 3ms/step - loss: 0.087
4 - acc: 0.9742 - val_loss: 0.0422 - val_acc: 0.9865
Epoch 3/12
60000/60000 [==============================] - 162s 3ms/step - loss: 0.067
7 - acc: 0.9798 - val_loss: 0.0347 - val_acc: 0.9887
Epoch 4/12
60000/60000 [==============================] - 161s 3ms/step - loss: 0.055
7 - acc: 0.9833 - val_loss: 0.0327 - val_acc: 0.9890
Epoch 5/12
60000/60000 [==============================] - 161s 3ms/step - loss: 0.048
7 - acc: 0.9853 - val_loss: 0.0303 - val_acc: 0.9899
Epoch 6/12
60000/60000 [==============================] - 161s 3ms/step - loss: 0.041
7 - acc: 0.9875 - val_loss: 0.0273 - val_acc: 0.9915
Epoch 7/12
60000/60000 [==============================] - 161s 3ms/step - loss: 0.038
9 - acc: 0.9885 - val_loss: 0.0301 - val_acc: 0.9905
Epoch 8/12
60000/60000 [==============================] - 161s 3ms/step - loss: 0.037
4 - acc: 0.9884 - val_loss: 0.0257 - val_acc: 0.9924
Epoch 9/12
60000/60000 [==============================] - 161s 3ms/step - loss: 0.032
3 - acc: 0.9900 - val_loss: 0.0295 - val_acc: 0.9899
Epoch 10/12
60000/60000 [==============================] - 160s 3ms/step - loss: 0.031
4 - acc: 0.9902 - val_loss: 0.0258 - val_acc: 0.9913
Epoch 11/12
60000/60000 [==============================] - 160s 3ms/step - loss: 0.026
6 - acc: 0.9916 - val_loss: 0.0281 - val_acc: 0.9916
Epoch 12/12
60000/60000 [==============================] - 160s 3ms/step - loss: 0.027
6 - acc: 0.9916 - val_loss: 0.0260 - val_acc: 0.9927
Test loss: 0.025978640862212615
Test accuracy: 0.9927
```

ASSIGNMENT **

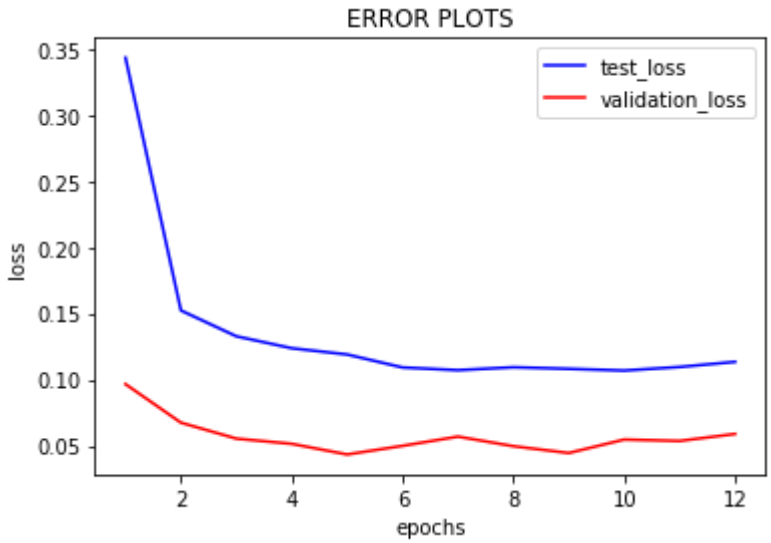
3 LAYERED ARCHITECTURE WITH KERNEL SIZE 3*3

In [0]:

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

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

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

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

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

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

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(Conv2D(128, (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.Adadelta(),
              metrics=['accuracy'])

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])

x=list(range(1,(epochs)+1))

plt.plot(x,model.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 224s 4ms/step - loss: 0.343
3 - acc: 0.8934 - val_loss: 0.0965 - val_acc: 0.9692
Epoch 2/12
60000/60000 [==============================] - 222s 4ms/step - loss: 0.152
2 - acc: 0.9572 - val_loss: 0.0674 - val_acc: 0.9787
Epoch 3/12
60000/60000 [==============================] - 221s 4ms/step - loss: 0.132
7 - acc: 0.9635 - val_loss: 0.0553 - val_acc: 0.9814
Epoch 4/12
60000/60000 [==============================] - 222s 4ms/step - loss: 0.123
7 - acc: 0.9664 - val_loss: 0.0514 - val_acc: 0.9845
Epoch 5/12
60000/60000 [==============================] - 221s 4ms/step - loss: 0.118
9 - acc: 0.9691 - val_loss: 0.0433 - val_acc: 0.9859
Epoch 6/12
60000/60000 [==============================] - 222s 4ms/step - loss: 0.109
1 - acc: 0.9720 - val_loss: 0.0499 - val_acc: 0.9855
Epoch 7/12
60000/60000 [==============================] - 224s 4ms/step - loss: 0.107
1 - acc: 0.9727 - val_loss: 0.0569 - val_acc: 0.9834
Epoch 8/12
60000/60000 [==============================] - 223s 4ms/step - loss: 0.109
3 - acc: 0.9725 - val_loss: 0.0496 - val_acc: 0.9873
Epoch 9/12
60000/60000 [==============================] - 218s 4ms/step - loss: 0.108
1 - acc: 0.9729 - val_loss: 0.0444 - val_acc: 0.9876
Epoch 10/12
60000/60000 [==============================] - 216s 4ms/step - loss: 0.106
8 - acc: 0.9731 - val_loss: 0.0546 - val_acc: 0.9852
Epoch 11/12
60000/60000 [==============================] - 218s 4ms/step - loss: 0.109
5 - acc: 0.9733 - val_loss: 0.0536 - val_acc: 0.9845
Epoch 12/12
60000/60000 [==============================] - 219s 4ms/step - loss: 0.113
4 - acc: 0.9740 - val_loss: 0.0588 - val_acc: 0.9841
Test loss: 0.05884759916906214
Test accuracy: 0.9841
```

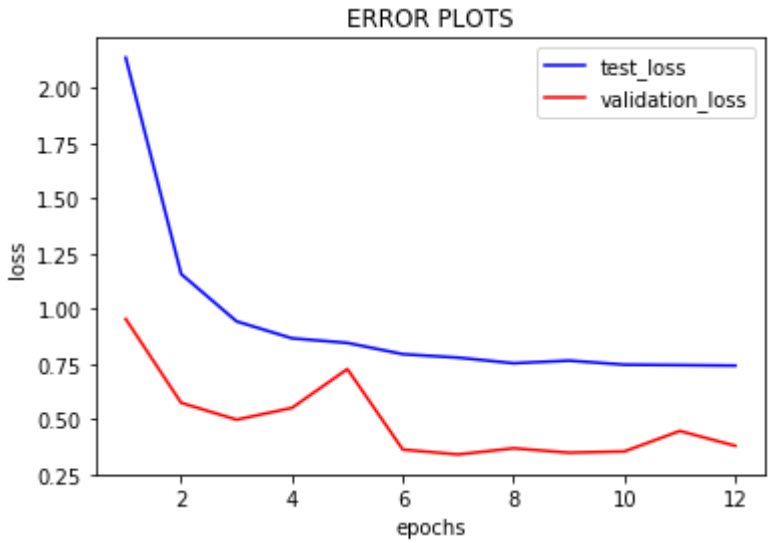


In [0]:

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

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

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

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

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

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

model_2 = Sequential()
model_2.add(Conv2D(32, kernel_size=(5,5),
                 activation='relu',
                 input_shape=input_shape))
model_2.add(Conv2D(64, (5,5), activation='relu'))
model_2.add(MaxPooling2D(pool_size=(2, 2)))
model_2.add(Dropout(0.25))
model_2.add(Conv2D(128, (5,5), activation='relu'))
model_2.add(MaxPooling2D(pool_size=(2, 2)))
model_2.add(Dropout(0.25))
model_2.add(Flatten())
model_2.add(Dense(128, activation='relu'))
model_2.add(Dropout(0.5))
model_2.add(Dense(num_classes, activation='softmax'))

model_2.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

```
model_2.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model_2.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

x=list(range(1,(epochs)+1))

plt.plot(x,model_2.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model_2.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 313s 5ms/step - loss: 2.136
9 - acc: 0.3243 - val_loss: 0.9534 - val_acc: 0.6856
Epoch 2/12
60000/60000 [==============================] - 314s 5ms/step - loss: 1.157
2 - acc: 0.6217 - val_loss: 0.5736 - val_acc: 0.8197
Epoch 3/12
60000/60000 [==============================] - 313s 5ms/step - loss: 0.942
6 - acc: 0.7003 - val_loss: 0.4973 - val_acc: 0.8397
Epoch 4/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.866
0 - acc: 0.7335 - val_loss: 0.5507 - val_acc: 0.8076
Epoch 5/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.845
1 - acc: 0.7478 - val_loss: 0.7266 - val_acc: 0.7867
Epoch 6/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.794
1 - acc: 0.7720 - val_loss: 0.3618 - val_acc: 0.8802
Epoch 7/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.778
4 - acc: 0.7799 - val_loss: 0.3397 - val_acc: 0.8991
Epoch 8/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.753
0 - acc: 0.7897 - val_loss: 0.3676 - val_acc: 0.9041
Epoch 9/12
60000/60000 [==============================] - 315s 5ms/step - loss: 0.764
9 - acc: 0.7922 - val_loss: 0.3477 - val_acc: 0.9012
Epoch 10/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.746
4 - acc: 0.8021 - val_loss: 0.3535 - val_acc: 0.8972
Epoch 11/12
60000/60000 [==============================] - 314s 5ms/step - loss: 0.744
6 - acc: 0.8019 - val_loss: 0.4460 - val_acc: 0.8577
Epoch 12/12
60000/60000 [==============================] - 312s 5ms/step - loss: 0.742
2 - acc: 0.8034 - val_loss: 0.3789 - val_acc: 0.9011
Test loss: 0.37893825750648974
Test accuracy: 0.9011
```



In [0]:

```
```

5 LAYER ARCHITECTURE

In [0]:

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

batch_size = 128
num_classes = 10
epochs = 10

# 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)

model_3 = Sequential()
model_3.add(Conv2D(32, kernel_size=(3,3),
                 activation='relu',
                 input_shape=input_shape))
model_3.add(Conv2D(64, (3,3), activation='relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))
model_3.add(Conv2D(128, (3,3), activation='relu'))
model_3.add(Dropout(0.25))
model_3.add(Conv2D(256, (3,3), activation='relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))
model_3.add(Conv2D(356, (3,3), activation='relu'))
model_3.add(MaxPooling2D(pool_size=(2, 2)))
model_3.add(Dropout(0.25))
model_3.add(Flatten())
model_3.add(Dense(128, activation='relu'))
model_3.add(Dropout(0.5))
model_3.add(Dense(num_classes, activation='softmax'))
```

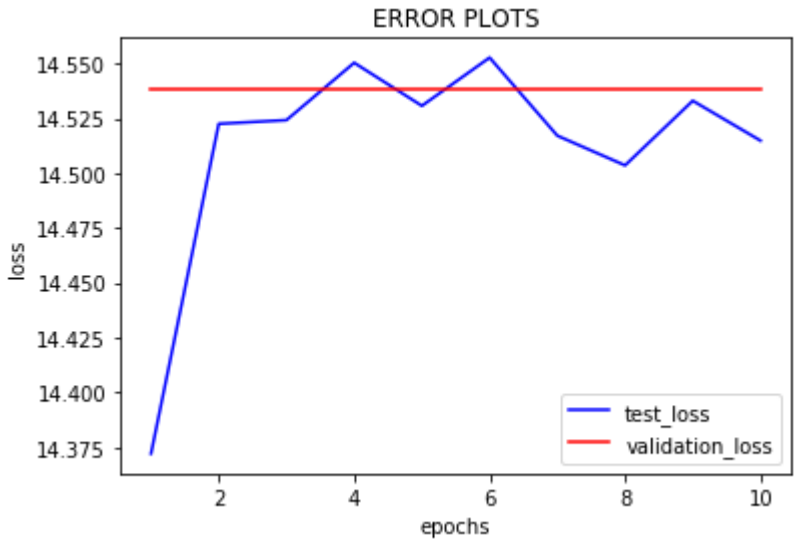
```
model_3.summary()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_18 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 10, 10, 128)       73856
_________________________________________________________________
dropout_19 (Dropout)         (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 8, 8, 256)         295168
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 4, 4, 256)         0
_________________________________________________________________
dropout_20 (Dropout)         (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 2, 2, 356)         820580
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 1, 1, 356)         0
_________________________________________________________________
dropout_21 (Dropout)         (None, 1, 1, 356)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 356)               0
_________________________________________________________________
dense_13 (Dense)             (None, 128)               45696
_________________________________________________________________
dropout_22 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_14 (Dense)             (None, 10)                1290
=================================================================
Total params: 1,255,406
Trainable params: 1,255,406
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
model_3.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model_3.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model_3.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

x=list(range(1,(epochs)+1))

plt.plot(x,model_3.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model_3.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 493s 8ms/step - loss: 14.37
21 - acc: 0.0998 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 2/10
60000/60000 [==============================] - 490s 8ms/step - loss: 14.52
26 - acc: 0.0990 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 3/10
60000/60000 [==============================] - 496s 8ms/step - loss: 14.52
42 - acc: 0.0989 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 4/10
60000/60000 [==============================] - 499s 8ms/step - loss: 14.55
04 - acc: 0.0972 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 5/10
60000/60000 [==============================] - 499s 8ms/step - loss: 14.53
08 - acc: 0.0984 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 6/10
60000/60000 [==============================] - 496s 8ms/step - loss: 14.55
28 - acc: 0.0971 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 7/10
60000/60000 [==============================] - 492s 8ms/step - loss: 14.51
71 - acc: 0.0993 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 8/10
60000/60000 [==============================] - 491s 8ms/step - loss: 14.50
36 - acc: 0.1002 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 9/10
60000/60000 [==============================] - 492s 8ms/step - loss: 14.53
31 - acc: 0.0983 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 10/10
60000/60000 [==============================] - 490s 8ms/step - loss: 14.51
49 - acc: 0.0995 - val_loss: 14.5385 - val_acc: 0.0980
Test loss: 14.538521841430665
Test accuracy: 0.098
```


ERROR PLOTS
14.550
14.525
14.500
14.475
14.450
14.425
14.400
14.375
loss
2
4
6
8
10
epochs
test_loss
validation_loss

In [0]:

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

batch_size = 128
num_classes = 10
epochs = 10

# 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)

model_4 = Sequential()
model_4.add(Conv2D(32, kernel_size=(2,2),
                 activation='relu',
                 input_shape=input_shape))
model_4.add(Conv2D(64, (2,2), activation='relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))
model_4.add(Dropout(0.25))
model_4.add(Conv2D(128, (2,2), activation='relu'))
model_4.add(Dropout(0.25))
model_4.add(Conv2D(256, (2,2), activation='relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))
model_4.add(Dropout(0.25))
model_4.add(Conv2D(356, (2,2), activation='relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))
model_4.add(Dropout(0.25))
model_4.add(Flatten())
model_4.add(Dense(128, activation='relu'))
model_4.add(Dropout(0.5))
model_4.add(Dense(num_classes, activation='softmax'))
```

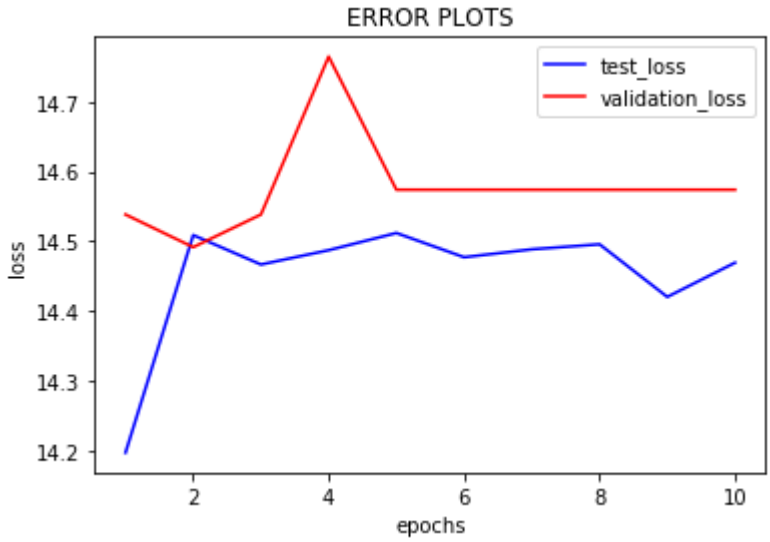
```
model_4.summary()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 27, 27, 32)        160
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 26, 26, 64)        8256
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 13, 13, 64)        0
_________________________________________________________________
dropout_23 (Dropout)         (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 12, 12, 128)       32896
_________________________________________________________________
dropout_24 (Dropout)         (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 11, 11, 256)       131328
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 5, 5, 256)         0
_________________________________________________________________
dropout_25 (Dropout)         (None, 5, 5, 256)         0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 4, 4, 356)         364900
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 2, 2, 356)         0
_________________________________________________________________
dropout_26 (Dropout)         (None, 2, 2, 356)         0
_________________________________________________________________
flatten_8 (Flatten)          (None, 1424)              0
_________________________________________________________________
dense_15 (Dense)             (None, 128)               182400
_________________________________________________________________
dropout_27 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_16 (Dense)             (None, 10)                1290
=================================================================
Total params: 721,230
Trainable params: 721,230
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
model_4.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model_4.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model_4.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

x=list(range(1,(epochs)+1))

plt.plot(x,model_4.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model_4.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 369s 6ms/step - loss: 14.19
79 - acc: 0.0990 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 2/10
60000/60000 [==============================] - 367s 6ms/step - loss: 14.50
91 - acc: 0.0996 - val_loss: 14.4918 - val_acc: 0.1009
Epoch 3/10
60000/60000 [==============================] - 367s 6ms/step - loss: 14.46
70 - acc: 0.1024 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 4/10
60000/60000 [==============================] - 367s 6ms/step - loss: 14.48
75 - acc: 0.1011 - val_loss: 14.7644 - val_acc: 0.0827
Epoch 5/10
60000/60000 [==============================] - 367s 6ms/step - loss: 14.51
21 - acc: 0.0996 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 6/10
60000/60000 [==============================] - 367s 6ms/step - loss: 14.47
74 - acc: 0.1017 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 7/10
60000/60000 [==============================] - 368s 6ms/step - loss: 14.48
88 - acc: 0.1011 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 8/10
60000/60000 [==============================] - 368s 6ms/step - loss: 14.49
59 - acc: 0.1006 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 9/10
60000/60000 [==============================] - 368s 6ms/step - loss: 14.42
06 - acc: 0.1053 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 10/10
60000/60000 [==============================] - 368s 6ms/step - loss: 14.46
96 - acc: 0.1023 - val_loss: 14.5740 - val_acc: 0.0958
Test loss: 14.573981651306152
Test accuracy: 0.0958
```



In [0]:

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

batch_size = 128
num_classes = 10
epochs = 5

# 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)

model_5 = Sequential()
model_5.add(Conv2D(32, kernel_size=(5,5),padding='same',
                 activation='relu',
                 input_shape=input_shape))
model_5.add(Conv2D(64, (5,5),padding='same', activation='relu'))
model_5.add(MaxPooling2D(pool_size=(2, 2)))
model_5.add(Dropout(0.5))
model_5.add(Conv2D(128, (5,5), padding='same',activation='relu'))
model_5.add(MaxPooling2D(pool_size=(2, 2)))
model_5.add(Dropout(0.5))
model_5.add(Conv2D(128, (5,5), padding='same',activation='relu'))
model_5.add(MaxPooling2D(pool_size=(2, 2)))
model_5.add(Dropout(0.5))
model_5.add(Conv2D(128, (5,5),padding='same', activation='relu'))
model_5.add(MaxPooling2D(pool_size=(2, 2)))
model_5.add(Dropout(0.5))
model_5.add(Flatten())
model_5.add(Dense(128, activation='relu'))
model_5.add(Dropout(0.5))
model_5.add(Dense(num_classes, activation='softmax'))
```

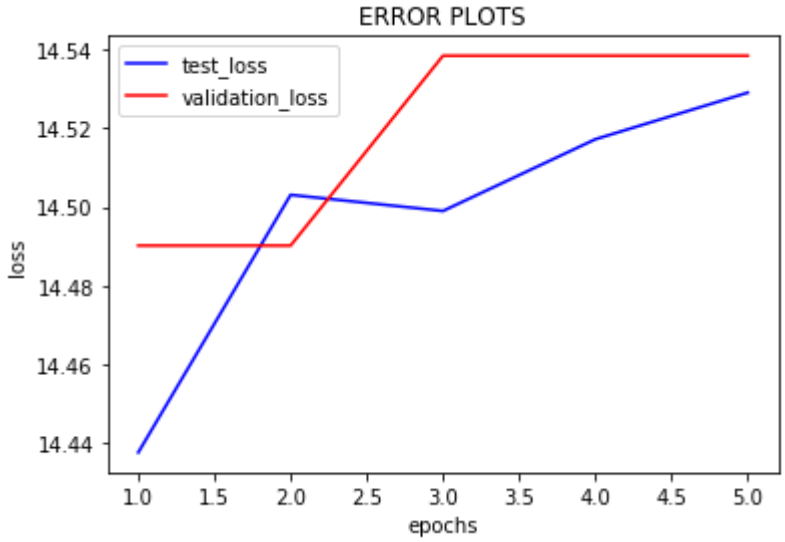
```
model_5.summary()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_28 (Conv2D)           (None, 28, 28, 32)        832
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 28, 28, 64)        51264
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_28 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 14, 14, 128)       204928
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_29 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 7, 7, 128)         409728
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
dropout_30 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 3, 3, 128)         409728
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
dropout_31 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
flatten_9 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_17 (Dense)             (None, 128)               16512
_________________________________________________________________
dropout_32 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_18 (Dense)             (None, 10)                1290
=================================================================
Total params: 1,094,282
Trainable params: 1,094,282
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
model_5.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model_5.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model_5.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

x=list(range(1,(epochs)+1))

plt.plot(x,model_5.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model_5.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 995s 17ms/step - loss: 14.4
375 - acc: 0.0992 - val_loss: 14.4902 - val_acc: 0.1010
Epoch 2/5
60000/60000 [==============================] - 990s 16ms/step - loss: 14.5
031 - acc: 0.1002 - val_loss: 14.4902 - val_acc: 0.1010
Epoch 3/5
60000/60000 [==============================] - 989s 16ms/step - loss: 14.4
990 - acc: 0.1004 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 4/5
60000/60000 [==============================] - 984s 16ms/step - loss: 14.5
173 - acc: 0.0993 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 5/5
60000/60000 [==============================] - 983s 16ms/step - loss: 14.5
291 - acc: 0.0986 - val_loss: 14.5385 - val_acc: 0.0980
Test loss: 14.538521841430665
Test accuracy: 0.098
```



In [0]:

```
```

7 LAYER ARCHITECTURE

In [0]:

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

batch_size = 128
num_classes = 10
epochs = 5

# 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)

model_6= Sequential()
model_6.add(Conv2D(32, kernel_size=(2,2),padding='same',
                 activation='relu',
                 input_shape=input_shape))
model_6.add(Conv2D(32, (2,2),padding='same', activation='relu'))
model_6.add(MaxPooling2D(pool_size=(2, 2)))
model_6.add(Dropout(0.5))
model_6.add(Conv2D(64, (2,2),padding='same', activation='relu'))
model_6.add(MaxPooling2D(pool_size=(1,1)))
model_6.add(Dropout(0.5))
model_6.add(Conv2D(64, (2,2),padding='same', activation='relu'))
model_6.add(MaxPooling2D(pool_size=(1,1)))
model_6.add(Dropout(0.5))
model_6.add(Conv2D(128, (2,2), padding='same',activation='relu'))
model_6.add(MaxPooling2D(pool_size=(2, 2)))
model_6.add(Dropout(0.5))
model_6.add(Conv2D(128, (2,2), padding='same',activation='relu'))
model_6.add(MaxPooling2D(pool_size=(2, 2)))
model_6.add(Dropout(0.5))
model_6.add(Conv2D(128, (2,2),padding='same', activation='relu'))
```

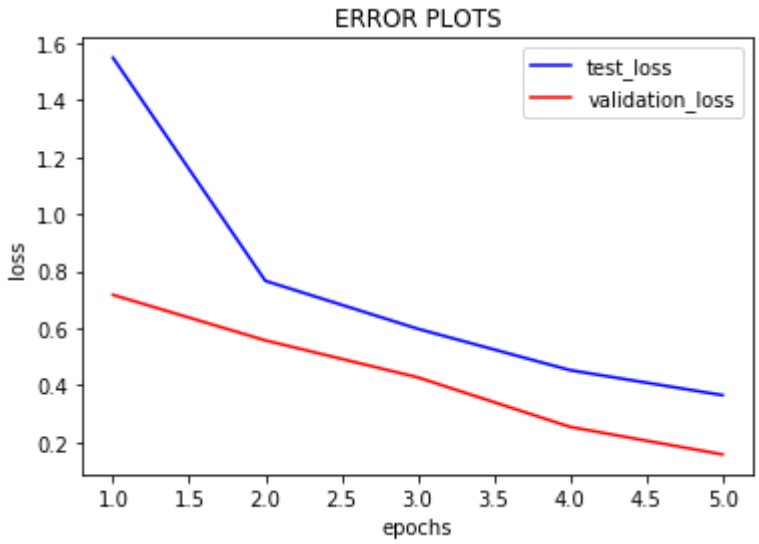
```
model_6.add(MaxPooling2D(pool_size=(2, 2)))
model_6.add(Dropout(0.5))
model_6.add(Flatten())
model_6.add(Dense(128, activation='relu'))
model_6.add(Dropout(0.5))
model_6.add(Dense(num_classes, activation='softmax'))
model_6.summary()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_33 (Conv2D)           (None, 28, 28, 32)        160
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 28, 28, 32)        4128
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
dropout_33 (Dropout)         (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 14, 14, 64)        8256
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_34 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 14, 14, 64)        16448
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_35 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 14, 14, 128)       32896
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_36 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 7, 7, 128)         65664
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
dropout_37 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 3, 3, 128)         65664
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
dropout_38 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
flatten_10 (Flatten)         (None, 128)               0
_________________________________________________________________
dense_19 (Dense)             (None, 128)               16512
_________________________________________________________________
dropout_39 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_20 (Dense)             (None, 10)                1290
=================================================================
Total params: 211,018
Trainable params: 211,018
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
model_6.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model_6.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model_6.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

x=list(range(1,(epochs)+1))

plt.plot(x,model_6.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model_6.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 240s 4ms/step - loss: 1.549
1 - acc: 0.4491 - val_loss: 0.7166 - val_acc: 0.7862
Epoch 2/5
60000/60000 [==============================] - 238s 4ms/step - loss: 0.765
8 - acc: 0.7435 - val_loss: 0.5568 - val_acc: 0.8259
Epoch 3/5
60000/60000 [==============================] - 238s 4ms/step - loss: 0.597
4 - acc: 0.8100 - val_loss: 0.4269 - val_acc: 0.8660
Epoch 4/5
60000/60000 [==============================] - 240s 4ms/step - loss: 0.452
2 - acc: 0.8660 - val_loss: 0.2530 - val_acc: 0.9299
Epoch 5/5
60000/60000 [==============================] - 240s 4ms/step - loss: 0.364
5 - acc: 0.8950 - val_loss: 0.1563 - val_acc: 0.9569
Test loss: 0.15629133698493242
Test accuracy: 0.9569
```



In [0]:

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

batch_size = 128
num_classes = 10
epochs = 5

# 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)

model_7= Sequential()
model_7.add(Conv2D(32, kernel_size=(3,3),padding='same',
                 activation='relu',
                 input_shape=input_shape))
model_7.add(Conv2D(32, (3,3),padding='same', activation='relu'))
model_7.add(MaxPooling2D(pool_size=(2,2)))
model_7.add(Dropout(0.5))
model_7.add(Conv2D(64, (3,3),padding='same', activation='relu'))
model_7.add(MaxPooling2D(pool_size=(1,1)))
model_7.add(Dropout(0.5))
model_7.add(Conv2D(64, (3,3),padding='same', activation='relu'))
model_7.add(MaxPooling2D(pool_size=(1,1)))
model_7.add(Dropout(0.5))
model_7.add(Conv2D(128, (3,3), padding='same',activation='relu'))
model_7.add(MaxPooling2D(pool_size=(2, 2)))
model_7.add(Dropout(0.5))
model_7.add(Conv2D(128, (3,3), padding='same',activation='relu'))
model_7.add(MaxPooling2D(pool_size=(2,2)))
model_7.add(Dropout(0.5))
model_7.add(Conv2D(128, (3,3),padding='same', activation='relu'))
```

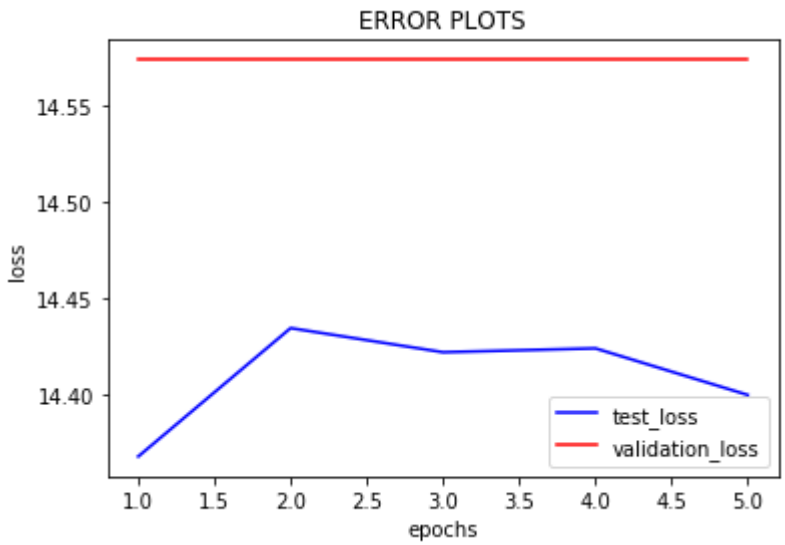
```
model_7.add(MaxPooling2D(pool_size=(2,2)))
model_7.add(Dropout(0.5))
model_7.add(Flatten())
model_7.add(Dense(128, activation='relu'))
model_7.add(Dropout(0.5))
model_7.add(Dense(num_classes, activation='softmax'))
model_7.summary()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_40 (Conv2D)           (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
dropout_40 (Dropout)         (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_41 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 14, 14, 64)        0
_________________________________________________________________
dropout_42 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 14, 14, 128)       73856
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_43 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 7, 7, 128)         147584
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
dropout_44 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 3, 3, 128)         147584
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
dropout_45 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
flatten_11 (Flatten)         (None, 128)               0
_________________________________________________________________
dense_21 (Dense)             (None, 128)               16512
_________________________________________________________________
dropout_46 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_22 (Dense)             (None, 10)                1290
=================================================================
Total params: 451,818
Trainable params: 451,818
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
model_7.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model_7.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model_7.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
x=list(range(1,(epochs)+1))

plt.plot(x,model_7.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model_7.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 492s 8ms/step - loss: 14.36
83 - acc: 0.1011 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 2/5
60000/60000 [==============================] - 488s 8ms/step - loss: 14.43
48 - acc: 0.1044 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 3/5
60000/60000 [==============================] - 485s 8ms/step - loss: 14.42
23 - acc: 0.1052 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 4/5
60000/60000 [==============================] - 486s 8ms/step - loss: 14.42
43 - acc: 0.1051 - val_loss: 14.5740 - val_acc: 0.0958
Epoch 5/5
60000/60000 [==============================] - 484s 8ms/step - loss: 14.40
02 - acc: 0.1066 - val_loss: 14.5740 - val_acc: 0.0958
Test loss: 14.573981651306152
Test accuracy: 0.0958
```



In [7]:

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

batch_size = 128
num_classes = 10
epochs = 3

# 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)

model_8= Sequential()
model_8.add(Conv2D(32, kernel_size=(5,5),padding='same',
                 activation='relu',
                 input_shape=input_shape))
model_8.add(Conv2D(32, (5,5),padding='same', activation='relu'))
model_8.add(MaxPooling2D(pool_size=(2,2)))
model_8.add(Dropout(0.5))
model_8.add(Conv2D(64, (5,5),padding='same', activation='relu'))
model_8.add(MaxPooling2D(pool_size=(1,1)))
model_8.add(Dropout(0.5))
model_8.add(Conv2D(64, (5,5),padding='same', activation='relu'))
model_8.add(MaxPooling2D(pool_size=(1,1)))
model_8.add(Dropout(0.5))
model_8.add(Conv2D(128, (5,5), padding='same',activation='relu'))
model_8.add(MaxPooling2D(pool_size=(2,2)))
model_8.add(Dropout(0.5))
model_8.add(Conv2D(128, (5,5), padding='same',activation='relu'))
model_8.add(MaxPooling2D(pool_size=(2,2)))
model_8.add(Dropout(0.5))
model_8.add(Conv2D(128, (5,5),padding='same', activation='relu'))
```

```
model_8.add(MaxPooling2D(pool_size=(2,2)))
model_8.add(Dropout(0.5))
model_8.add(Flatten())
model_8.add(Dense(128, activation='relu'))
model_8.add(Dropout(0.5))
model_8.add(Dense(num_classes, activation='softmax'))
model_8.summary()
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 32)        25632
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 14, 14, 64)        51264
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_9 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 64)        102464
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_10 (Dropout)         (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 128)       204928
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
dropout_11 (Dropout)         (None, 7, 7, 128)         0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 7, 7, 128)         409728
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
dropout_12 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 3, 3, 128)         409728
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 1, 1, 128)         0
_________________________________________________________________
dropout_13 (Dropout)         (None, 1, 1, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_14 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,222,378
Trainable params: 1,222,378
Non-trainable params: 0
_________________________________________________________________
```

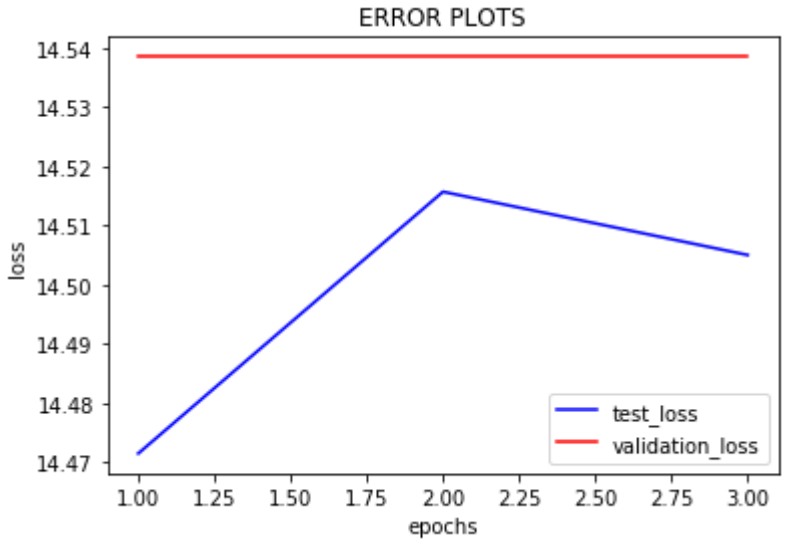
In [8]:

```
model_8.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model_8.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model_8.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 1/3
60000/60000 [==============================] - 1181s 20ms/step - loss: 14.
4714 - acc: 0.0987 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 2/3
60000/60000 [==============================] - 1178s 20ms/step - loss: 14.
5157 - acc: 0.0994 - val_loss: 14.5385 - val_acc: 0.0980
Epoch 3/3
60000/60000 [==============================] - 1182s 20ms/step - loss: 14.
5050 - acc: 0.1001 - val_loss: 14.5385 - val_acc: 0.0980
Test loss: 14.538521841430665
Test accuracy: 0.098
```

In [9]:

```
import matplotlib.pyplot as plt

x=list(range(1,(epochs)+1))

plt.plot(x,model_8.history.history['loss'],color='blue', label="test_loss")
plt.plot(x,model_8.history.history['val_loss'],color='red', label="validation_loss")
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('ERROR PLOTS')
plt.legend()
plt.show()
plt.close()
```



CONCLUSION

In [10]:

```
from prettytable import PrettyTable
x=PrettyTable(['NO OF HIDDEN LAYERS','KERNEL SIZE','TEST ACCURACY','TEST LOSS'])
x.add_row(['3','3*3','0.9841','0.0588'])
x.add_row(['3','5*5','0.9011','0.3789'])
x.add_row(['5','2*2','0.0958','14.5739'])
x.add_row(['5','3*3','0.098' ,'14.538'])
x.add_row(['5','5*5','0.098','14.5385'])
x.add_row(['7','2*2','0.9569','0.1562'])
x.add_row(['7','3*3','0.0958','14.5739'])
x.add_row(['7','5*5','0.098','14.5385'])


print(x.get_string(start=0,end=9))
```

```
+---------------------+-------------+---------------+-----------+
| NO OF HIDDEN LAYERS | KERNEL SIZE | TEST ACCURACY | TEST LOSS |
+---------------------+-------------+---------------+-----------+
|          3          |     3*3     |     0.9841    |   0.0588  |
|          3          |     5*5     |     0.9011    |   0.3789  |
|          5          |     2*2     |     0.0958    |  14.5739  |
|          5          |     3*3     |     0.098     |   14.538  |
|          5          |     5*5     |     0.098     |  14.5385  |
|          7          |     2*2     |     0.9569    |   0.1562  |
|          7          |     3*3     |     0.0958    |  14.5739  |
|          7          |     5*5     |     0.098     |  14.5385  |
+---------------------+-------------+---------------+-----------+
```

In [0]: