```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docke
r-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list al
l files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/wm811k-wafer-map/LSWMD.pkl
```

In [2]:

```
import os
from os.path import join

import numpy as np
import pandas as pd

import tensorflow as tf
import keras
from keras import layers, Input, models
from keras.utils import to_categorical
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt

datapath = join('data', 'wafer')

print(os.listdir("../input"))
import warnings
warnings.filterwarnings("ignore")
```

```
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
16: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
17: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
18: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
```

```
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
19: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
20: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
25: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
['wm811k-wafer-map']
```

In [3]:

```
df=pd.read_pickle("../input/wm811k-wafer-map/LSWMD.pkl")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 811457 entries, 0 to 811456
Data columns (total 6 columns):
waferMap          811457 non-null object
dieSize           811457 non-null float64
```

```
lotName          811457 non-null object
waferIndex       811457 non-null float64
trianTestLabel   811457 non-null object
failureType      811457 non-null object
dtypes: float64(2), object(4)
memory usage: 37.1+ MB
```

In [4]:

```python
df.tail()
```

Out[4]:

	waferMap	dieSize	lotName	waferIndex	trianTestLabel	failureType
811452	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1,...	600.0	lot47542	23.0	[[Test]]	[[Edge-Ring]]
811453	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 1,...	600.0	lot47542	24.0	[[Test]]	[[Edge-Loc]]
811454	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1,...	600.0	lot47542	25.0	[[Test]]	[[Edge-Ring]]
811455	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,...	600.0	lot47543	1.0	[]	[]
811456	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1,...	600.0	lot47543	2.0	[]	[]

In [5]:

```python
df = df.drop(['waferIndex'], axis = 1)
```

In [6]:

```python
def find_dim(x):
    dim0=np.size(x,axis=0)
    dim1=np.size(x,axis=1)
    return dim0,dim1
df['waferMapDim']=df.waferMap.apply(find_dim)
df.sample(5)
```

Out[6]:

	waferMap	dieSize	lotName	trianTestLabel	failureType	waferMapDim
584337	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	846.0	lot36456	[]	[]	(33, 33)
617623	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	846.0	lot38766	[]	[]	(33, 33)
245636	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1893.0	lot15374	[[Training]]	[[none]]	(50, 49)
236952	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,...	1073.0	lot14795	[[Training]]	[[Edge-Ring]]	(36, 38)
152945	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	3532.0	lot9921	[]	[]	(64, 71)

In [7]:

```
df['failureNum']=df.failureType
df['trainTestNum']=df.trianTestLabel
mapping_type={'Center':0,'Donut':1,'Edge-Loc':2,'Edge-Ring':3,'Loc':4,'Random':5,'Scratch':6,'Near-full':7,'none':8}
mapping_traintest={'Training':0,'Test':1}
df=df.replace({'failureNum':mapping_type, 'trainTestNum':mapping_traintest})
```

In [8]:

```
tol_wafers = df.shape[0]
tol_wafers
```

Out[8]:

```
811457
```

In [9]:

```
df_withlabel = df[(df['failureNum']>=0) & (df['failureNum']<=8)]
df_withlabel =df_withlabel.reset_index()
df_withpattern = df[(df['failureNum']>=0) & (df['failureNum']<=7)]
df_withpattern = df_withpattern.reset_index()
df_nonpattern = df[(df['failureNum']==8)]
df_withlabel.shape[0], df_withpattern.shape[0], df_nonpattern.shape[0]
```

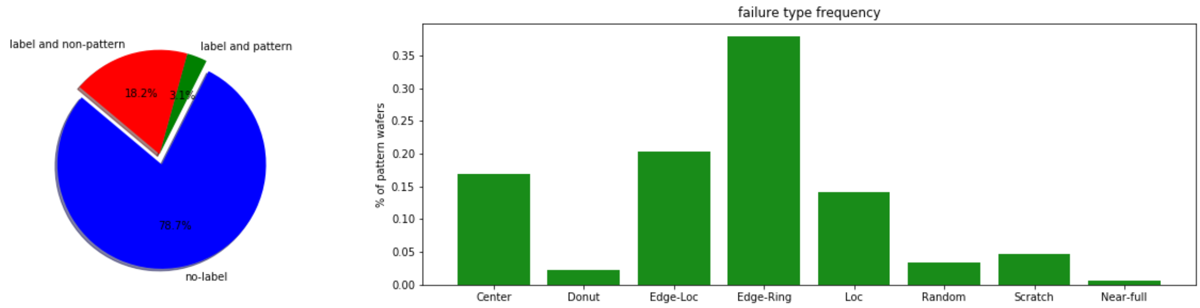
Out[9]:

```
(172950, 25519, 147431)
```

In [10]:

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

from matplotlib import gridspec
fig = plt.figure(figsize=(20, 4.5))
```

```python
gs = gridspec.GridSpec(1, 2, width_ratios=[1, 2.5])
ax1 = plt.subplot(gs[0])
ax2 = plt.subplot(gs[1])

no_wafers=[tol_wafers-df_withlabel.shape[0], df_withpattern.shape[0], df_nonpatter
n.shape[0]]

colors = ['blue', 'green', 'red']
explode = (0.1, 0, 0)  # explode 1st slice
labels = ['no-label','label and pattern','label and non-pattern']
ax1.pie(no_wafers, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%
', shadow=True, startangle=140)

uni_pattern=np.unique(df_withpattern.failureNum, return_counts=True)
labels2 = ['','Center','Donut','Edge-Loc','Edge-Ring','Loc','Random','Scratch','Ne
ar-full']
ax2.bar(uni_pattern[0],uni_pattern[1]/df_withpattern.shape[0], color='green', alig
n='center', alpha=0.9)
ax2.set_title("failure type frequency")
ax2.set_ylabel("% of pattern wafers")
ax2.set_xticklabels(labels2)

plt.show()
```



In [11]:

```python
sub_df = df.loc[df['waferMapDim'] == (26, 26)]
sub_wafer = sub_df['waferMap'].values

sw = np.ones((1, 26, 26))
label = list()

for i in range(len(sub_df)):
    # skip null label
    if len(sub_df.iloc[i,:]['failureType']) == 0:
        continue
    sw = np.concatenate((sw, sub_df.iloc[i,:]['waferMap'].reshape(1, 26, 26)))
    label.append(sub_df.iloc[i,:]['failureType'][0][0])
```

In [12]:

```python
x = sw[1:]
y = np.array(label).reshape((-1,1))
```

In [13]:

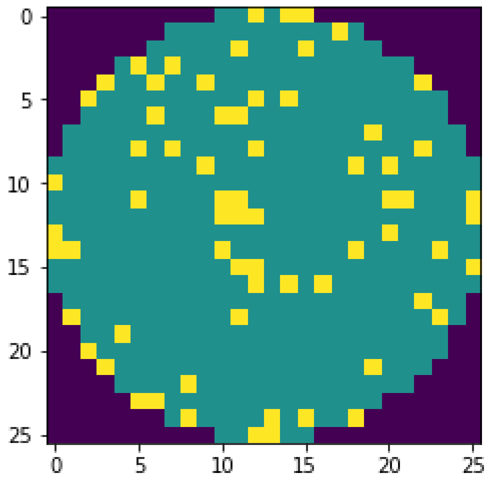
```python
print('x shape : {}, y shape : {}'.format(x.shape, y.shape))
```

```
x shape : (14366, 26, 26), y shape : (14366, 1)
```

In [14]:

```python
# plot 1st data
plt.imshow(x[2040])
plt.show()
```

```python
# check faulty case
print('Faulty case : {} '.format(y[2040]))
```




```
Faulty case : ['none']
```

In [15]:

```python
x = x.reshape((-1, 26, 26, 1))
```

In [16]:

```python
faulty_case = np.unique(y)
print('Faulty case list : {}'.format(faulty_case))
```

```
Faulty case list : ['Center' 'Donut' 'Edge-Loc' 'Edge-Ring' 'Loc' 'Near-full'
'Random'
 'Scratch' 'none']
```

In [17]:

```python
for f in faulty_case :
    print('{} : {}'.format(f, len(y[y==f])))
```

```
Center : 90
Donut : 1
Edge-Loc : 296
Edge-Ring : 31
Loc : 297
Near-full : 16
Random : 74
Scratch : 72
none : 13489
```

In [18]:

```python
new_x = np.zeros((len(x), 26, 26, 3))

for w in range(len(x)):
    for i in range(26):
        for j in range(26):
            new_x[w, i, j, int(x[w, i, j])] = 1
```

In [19]:

```python
new_x.shape
```

Out[19]:

```
(14366, 26, 26, 3)
```

In [20]:

```python
# Encoder
input_shape = (26, 26, 3)
```

```
input_tensor = Input(input_shape)
encode = layers.Conv2D(64, (3,3), padding='same', activation='relu')(input_tensor)

latent_vector = layers.MaxPool2D()(encode)

# Decoder
decode_layer_1 = layers.Conv2DTranspose(64, (3,3), padding='same', activation='rel
u')
decode_layer_2 = layers.UpSampling2D()
output_tensor = layers.Conv2DTranspose(3, (3,3), padding='same', activation='sigmo
id')

# connect decoder layers
decode = decode_layer_1(latent_vector)
decode = decode_layer_2(decode)

ae = models.Model(input_tensor, output_tensor(decode))
ae.compile(optimizer = 'Adam',
              loss = 'mse',
             )
```

In [21]:

```
epoch=30
batch_size=1024
```

In [22]:

```
# start train
ae.fit(new_x, new_x,
       batch_size=batch_size,
       epochs=epoch,
       verbose=2)
```

```
Epoch 1/30
 - 7s - loss: 0.1606
Epoch 2/30
 - 1s - loss: 0.1015
Epoch 3/30
 - 1s - loss: 0.0837
Epoch 4/30
 - 1s - loss: 0.0730
Epoch 5/30
 - 1s - loss: 0.0634
Epoch 6/30
 - 1s - loss: 0.0554
Epoch 7/30
 - 1s - loss: 0.0487
Epoch 8/30
 - 1s - loss: 0.0431
Epoch 9/30
 - 1s - loss: 0.0380
Epoch 10/30
 - 1s - loss: 0.0333
Epoch 11/30
 - 1s - loss: 0.0293
Epoch 12/30
 - 1s - loss: 0.0262
Epoch 13/30
 - 1s - loss: 0.0236
Epoch 14/30
```

```
 - 1s - loss: 0.0215
Epoch 15/30
 - 1s - loss: 0.0198
Epoch 16/30
 - 1s - loss: 0.0183
Epoch 17/30
 - 1s - loss: 0.0170
Epoch 18/30
 - 1s - loss: 0.0159
Epoch 19/30
 - 1s - loss: 0.0149
Epoch 20/30
 - 1s - loss: 0.0139
Epoch 21/30
 - 1s - loss: 0.0131
Epoch 22/30
 - 1s - loss: 0.0123
Epoch 23/30
 - 1s - loss: 0.0116
Epoch 24/30
 - 1s - loss: 0.0109
Epoch 25/30
 - 1s - loss: 0.0103
Epoch 26/30
 - 1s - loss: 0.0098
Epoch 27/30
 - 1s - loss: 0.0093
Epoch 28/30
 - 1s - loss: 0.0088
Epoch 29/30
 - 1s - loss: 0.0084
Epoch 30/30
 - 1s - loss: 0.0080
```

Out[22]:

```
<keras.callbacks.History at 0x7fc1cd5e3588>
```

In [23]:

```
encoder = models.Model(input_tensor, latent_vector)
```

In [24]:

```
decoder_input = Input((13, 13, 64))
decode = decode_layer_1(decoder_input)
decode = decode_layer_2(decode)

decoder = models.Model(decoder_input, output_tensor(decode))
```

In [25]:

```
# Encode original faulty wafer
encoded_x = encoder.predict(new_x)
```

In [26]:

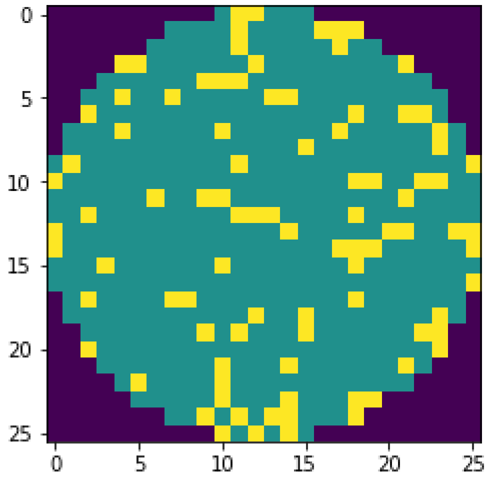
```
# Add noise to encoded latent faulty wafers vector.
noised_encoded_x = encoded_x + np.random.normal(loc=0, scale=0.1, size = (len(enco
ded_x), 13, 13, 64))
```

In [27]:

```
# check original faulty wafer data
plt.imshow(np.argmax(new_x[3], axis=2))
```

Out[27]:

```
<matplotlib.image.AxesImage at 0x7fc1e01fe4a8>
```

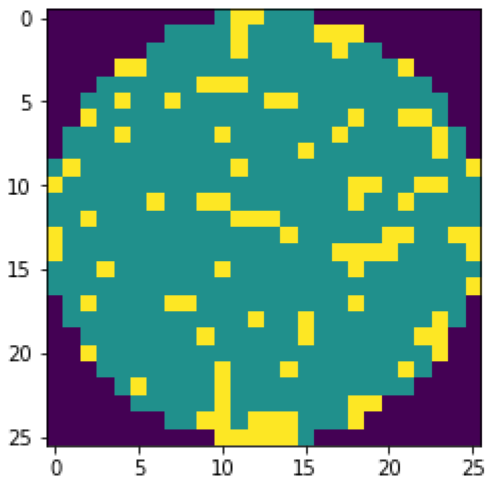


In [28]:

```
# check new noised faulty wafer data
noised_gen_x = np.argmax(decoder.predict(noised_encoded_x), axis=3)
plt.imshow(noised_gen_x[3])
```

Out[28]:

```
<matplotlib.image.AxesImage at 0x7fc1e0394f98>
```



In [29]:

```
# augment function define
def gen_data(wafer, label):
    # Encode input wafer
    encoded_x = encoder.predict(wafer)

    # dummy array for collecting noised wafer
    gen_x = np.zeros((1, 26, 26, 3))

    # Make wafer until total # of wafer to 2000
    for i in range((2000//len(wafer)) + 1):
        noised_encoded_x = encoded_x + np.random.normal(loc=0, scale=0.1, size = (
len(encoded_x), 13, 13, 64))
        noised_gen_x = decoder.predict(noised_encoded_x)
        gen_x = np.concatenate((gen_x, noised_gen_x), axis=0)
    # also make label vector with same length
    gen_y = np.full((len(gen_x), 1), label)
```

```
    # return date without 1st dummy data.
    return gen_x[1:], gen_y[1:]
```

In [30]:

```
# Augmentation for all faulty case.
for f in faulty_case :
    # skip none case
    if f == 'none' :
        continue

    gen_x, gen_y = gen_data(new_x[np.where(y==f)[0]], f)
    new_x = np.concatenate((new_x, gen_x), axis=0)
    y = np.concatenate((y, gen_y))
```

In [31]:

```
print('After Generate new_x shape : {}, new_y shape : {}'.format(new_x.shape, y.sh
ape))
```

```
After Generate new_x shape : (30707, 26, 26, 3), new_y shape : (30707, 1)
```

In [32]:

```
for f in faulty_case :
    print('{} : {}'.format(f, len(y[y==f])))
```

```
Center : 2160
Donut : 2002
Edge-Loc : 2368
Edge-Ring : 2046
Loc : 2376
Near-full : 2032
Random : 2146
Scratch : 2088
none : 13489
```

In [33]:

```
none_idx = np.where(y=='none')[0][np.random.choice(len(np.where(y=='none')[0]), si
ze=11000, replace=False)]
```

In [34]:

```
new_x = np.delete(new_x, none_idx, axis=0)
new_y = np.delete(y, none_idx, axis=0)
```

In [35]:

```
print('After Delete "none" class new_x shape : {}, new_y shape : {}'.format(new_x.
shape, new_y.shape))
```

```
After Delete "none" class new_x shape : (19707, 26, 26, 3), new_y shape : (197
07, 1)
```

In [36]:

```
for f in faulty_case :
    print('{} : {}'.format(f, len(new_y[new_y==f])))
```

```
Center : 2160
Donut : 2002
Edge-Loc : 2368
Edge-Ring : 2046
Loc : 2376
Near-full : 2032
Random : 2146
Scratch : 2088
none : 2489
```

In [37]:

```
for i, l in enumerate(faulty_case):
    new_y[new_y==l] = i

# one-hot-encoding
new_y = to_categorical(new_y)
```

In [38]:

```
new_X=new_x[0:19000]
new_Y=new_y[0:19000]
test_x=new_x[19001:19706]
test_y=new_y[19001:19706]
test_x.shape
```

Out[38]:

```
(705, 26, 26, 3)
```

In [39]:

```
x_train, x_test, y_train, y_test = train_test_split(new_X, new_Y,
                                                    test_size=0.33,
                                                    random_state=2019)
```

In [40]:

```
print('Train x : {}, y : {}'.format(x_train.shape, y_train.shape))
print('Test x: {}, y : {}'.format(x_test.shape, y_test.shape))
```

```
Train x : (12730, 26, 26, 3), y : (12730, 9)
Test x: (6270, 26, 26, 3), y : (6270, 9)
```

In [41]:

```
def create_model():
    input_shape = (26, 26, 3)
    input_tensor = Input(input_shape)

    conv_1 = layers.Conv2D(16, (3,3), activation='relu', padding='same')(input_ten
sor)
    conv_2 = layers.Conv2D(64, (3,3), activation='relu', padding='same')(conv_1)
    conv_3 = layers.Conv2D(128, (3,3), activation='relu', padding='same')(conv_2)

    flat = layers.Flatten()(conv_3)

    dense_1 = layers.Dense(512, activation='relu')(flat)
    dense_2 = layers.Dense(128, activation='relu')(dense_1)
    output_tensor = layers.Dense(9, activation='softmax')(dense_2)

    model = models.Model(input_tensor, output_tensor)
    model.compile(optimizer='Adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

In [42]:

```
model = KerasClassifier(build_fn=create_model, epochs=30, batch_size=1024, verbose
=2)
# 3-Fold Crossvalidation
kfold = KFold(n_splits=3, shuffle=True, random_state=2019)
results = cross_val_score(model, x_train, y_train, cv=kfold)
# Check 3-fold model's mean accuracy
print('Simple CNN Cross validation score : {:.4f}'.format(np.mean(results)))
```

```
Epoch 1/30
```

```
 - 4s - loss: 2.2454 - acc: 0.3998
Epoch 2/30
 - 1s - loss: 0.8462 - acc: 0.6914
Epoch 3/30
 - 1s - loss: 0.4931 - acc: 0.8223
Epoch 4/30
 - 1s - loss: 0.5228 - acc: 0.8505
Epoch 5/30
 - 1s - loss: 0.2987 - acc: 0.9187
Epoch 6/30
 - 1s - loss: 0.1622 - acc: 0.9566
Epoch 7/30
 - 1s - loss: 0.0928 - acc: 0.9760
Epoch 8/30
 - 1s - loss: 0.0763 - acc: 0.9802
Epoch 9/30
 - 1s - loss: 0.0419 - acc: 0.9903
Epoch 10/30
 - 1s - loss: 0.0273 - acc: 0.9948
Epoch 11/30
 - 1s - loss: 0.0166 - acc: 0.9976
Epoch 12/30
 - 1s - loss: 0.0121 - acc: 0.9980
Epoch 13/30
 - 1s - loss: 0.0111 - acc: 0.9972
Epoch 14/30
 - 1s - loss: 0.0082 - acc: 0.9986
Epoch 15/30
 - 1s - loss: 0.0066 - acc: 0.9987
Epoch 16/30
 - 1s - loss: 0.0091 - acc: 0.9981
Epoch 17/30
 - 1s - loss: 0.0103 - acc: 0.9979
Epoch 18/30
 - 1s - loss: 0.0096 - acc: 0.9987
Epoch 19/30
 - 1s - loss: 0.0066 - acc: 0.9989
Epoch 20/30
 - 1s - loss: 0.0058 - acc: 0.9988
Epoch 21/30
 - 1s - loss: 0.0043 - acc: 0.9989
Epoch 22/30
 - 1s - loss: 0.0044 - acc: 0.9989
Epoch 23/30
 - 1s - loss: 0.0053 - acc: 0.9984
Epoch 24/30
 - 1s - loss: 0.0039 - acc: 0.9985
Epoch 25/30
 - 1s - loss: 0.0046 - acc: 0.9988
Epoch 26/30
 - 1s - loss: 0.0059 - acc: 0.9985
Epoch 27/30
 - 1s - loss: 0.0059 - acc: 0.9989
Epoch 28/30
 - 1s - loss: 0.0058 - acc: 0.9981
Epoch 29/30
```

```
 - 1s - loss: 0.0039 - acc: 0.9988
Epoch 30/30
 - 1s - loss: 0.0043 - acc: 0.9982
Epoch 1/30
 - 3s - loss: 2.9482 - acc: 0.2222
Epoch 2/30
 - 1s - loss: 1.1951 - acc: 0.6080
Epoch 3/30
 - 1s - loss: 0.8000 - acc: 0.7104
Epoch 4/30
 - 1s - loss: 0.5120 - acc: 0.8187
Epoch 5/30
 - 1s - loss: 0.3506 - acc: 0.8742
Epoch 6/30
 - 1s - loss: 0.2443 - acc: 0.9269
Epoch 7/30
 - 1s - loss: 0.1692 - acc: 0.9511
Epoch 8/30
 - 1s - loss: 0.0986 - acc: 0.9770
Epoch 9/30
 - 1s - loss: 0.0612 - acc: 0.9879
Epoch 10/30
 - 1s - loss: 0.0349 - acc: 0.9945
Epoch 11/30
 - 1s - loss: 0.0213 - acc: 0.9979
Epoch 12/30
 - 1s - loss: 0.0170 - acc: 0.9975
Epoch 13/30
 - 1s - loss: 0.0116 - acc: 0.9981
Epoch 14/30
 - 1s - loss: 0.0084 - acc: 0.9987
Epoch 15/30
 - 1s - loss: 0.0084 - acc: 0.9986
Epoch 16/30
 - 1s - loss: 0.0093 - acc: 0.9984
Epoch 17/30
 - 1s - loss: 0.0093 - acc: 0.9987
Epoch 18/30
 - 1s - loss: 0.0073 - acc: 0.9982
Epoch 19/30
 - 1s - loss: 0.0060 - acc: 0.9989
Epoch 20/30
 - 1s - loss: 0.0061 - acc: 0.9985
Epoch 21/30
 - 1s - loss: 0.0059 - acc: 0.9986
Epoch 22/30
 - 1s - loss: 0.0096 - acc: 0.9979
Epoch 23/30
 - 1s - loss: 0.0347 - acc: 0.9934
Epoch 24/30
 - 1s - loss: 5.0369 - acc: 0.5391
Epoch 25/30
 - 1s - loss: 3.0918 - acc: 0.5187
Epoch 26/30
 - 1s - loss: 1.0979 - acc: 0.6730
Epoch 27/30
```

```
 - 1s - loss: 0.4224 - acc: 0.8465
Epoch 28/30
 - 1s - loss: 0.1743 - acc: 0.9532
Epoch 29/30
 - 1s - loss: 0.0794 - acc: 0.9775
Epoch 30/30
 - 1s - loss: 0.0457 - acc: 0.9897
Epoch 1/30
 - 3s - loss: 2.2367 - acc: 0.3223
Epoch 2/30
 - 1s - loss: 0.9060 - acc: 0.6612
Epoch 3/30
 - 1s - loss: 0.7930 - acc: 0.7651
Epoch 4/30
 - 1s - loss: 0.5420 - acc: 0.8343
Epoch 5/30
 - 1s - loss: 0.3333 - acc: 0.8924
Epoch 6/30
 - 1s - loss: 0.1972 - acc: 0.9441
Epoch 7/30
 - 1s - loss: 0.1181 - acc: 0.9684
Epoch 8/30
 - 1s - loss: 0.0641 - acc: 0.9846
Epoch 9/30
 - 1s - loss: 0.0339 - acc: 0.9932
Epoch 10/30
 - 1s - loss: 0.0207 - acc: 0.9961
Epoch 11/30
 - 1s - loss: 0.0151 - acc: 0.9975
Epoch 12/30
 - 1s - loss: 0.0131 - acc: 0.9973
Epoch 13/30
 - 1s - loss: 0.0124 - acc: 0.9975
Epoch 14/30
 - 1s - loss: 0.0140 - acc: 0.9968
Epoch 15/30
 - 1s - loss: 0.0118 - acc: 0.9973
Epoch 16/30
 - 1s - loss: 0.0101 - acc: 0.9979
Epoch 17/30
 - 1s - loss: 0.0121 - acc: 0.9982
Epoch 18/30
 - 1s - loss: 0.0095 - acc: 0.9981
Epoch 19/30
 - 1s - loss: 0.0083 - acc: 0.9982
Epoch 20/30
 - 1s - loss: 0.0104 - acc: 0.9976
Epoch 21/30
 - 1s - loss: 0.0102 - acc: 0.9975
Epoch 22/30
 - 1s - loss: 0.0107 - acc: 0.9980
Epoch 23/30
 - 1s - loss: 0.0092 - acc: 0.9980
Epoch 24/30
 - 1s - loss: 0.0109 - acc: 0.9972
Epoch 25/30
```

```
 - 1s - loss: 0.0098 - acc: 0.9978
Epoch 26/30
 - 1s - loss: 0.0094 - acc: 0.9978
Epoch 27/30
 - 1s - loss: 0.0075 - acc: 0.9981
Epoch 28/30
 - 1s - loss: 0.0080 - acc: 0.9982
Epoch 29/30
 - 1s - loss: 0.0061 - acc: 0.9986
Epoch 30/30
 - 1s - loss: 0.0073 - acc: 0.9986
Simple CNN Cross validation score : 0.9833
```

In [43]:

```
history = model.fit(x_train, y_train,
          validation_data=[x_test, y_test],
          epochs=epoch,
          batch_size=batch_size,
          )
```

```
Train on 12730 samples, validate on 6270 samples
Epoch 1/30
 - 4s - loss: 1.9970 - acc: 0.3897 - val_loss: 0.9990 - val_acc: 0.6754
Epoch 2/30
 - 2s - loss: 0.8095 - acc: 0.7253 - val_loss: 0.6484 - val_acc: 0.8051
Epoch 3/30
 - 2s - loss: 0.4719 - acc: 0.8554 - val_loss: 0.3474 - val_acc: 0.8684
Epoch 4/30
 - 2s - loss: 0.2440 - acc: 0.9266 - val_loss: 0.1800 - val_acc: 0.9475
Epoch 5/30
 - 2s - loss: 0.1248 - acc: 0.9649 - val_loss: 0.1054 - val_acc: 0.9662
Epoch 6/30
 - 2s - loss: 0.0691 - acc: 0.9827 - val_loss: 0.0764 - val_acc: 0.9839
Epoch 7/30
 - 2s - loss: 0.0371 - acc: 0.9926 - val_loss: 0.0351 - val_acc: 0.9914
Epoch 8/30
 - 2s - loss: 0.0183 - acc: 0.9967 - val_loss: 0.0316 - val_acc: 0.9920
Epoch 9/30
 - 2s - loss: 0.0156 - acc: 0.9966 - val_loss: 0.0298 - val_acc: 0.9922
Epoch 10/30
 - 2s - loss: 0.0111 - acc: 0.9977 - val_loss: 0.0324 - val_acc: 0.9907
Epoch 11/30
 - 2s - loss: 0.0093 - acc: 0.9980 - val_loss: 0.0242 - val_acc: 0.9935
Epoch 12/30
 - 2s - loss: 0.0104 - acc: 0.9977 - val_loss: 0.0316 - val_acc: 0.9896
Epoch 13/30
 - 2s - loss: 0.0085 - acc: 0.9984 - val_loss: 0.0288 - val_acc: 0.9925
Epoch 14/30
 - 2s - loss: 0.0093 - acc: 0.9977 - val_loss: 0.0266 - val_acc: 0.9927
Epoch 15/30
 - 2s - loss: 0.0089 - acc: 0.9980 - val_loss: 0.0231 - val_acc: 0.9943
Epoch 16/30
 - 2s - loss: 0.0076 - acc: 0.9984 - val_loss: 0.0265 - val_acc: 0.9936
Epoch 17/30
 - 2s - loss: 0.0066 - acc: 0.9987 - val_loss: 0.0338 - val_acc: 0.9923
Epoch 18/30
 - 2s - loss: 0.0077 - acc: 0.9986 - val_loss: 0.0223 - val_acc: 0.9946
Epoch 19/30
```

```
 - 2s - loss: 0.0068 - acc: 0.9984 - val_loss: 0.0237 - val_acc: 0.9941
Epoch 20/30
 - 2s - loss: 0.0065 - acc: 0.9985 - val_loss: 0.0308 - val_acc: 0.9917
Epoch 21/30
 - 2s - loss: 0.0065 - acc: 0.9980 - val_loss: 0.0337 - val_acc: 0.9917
Epoch 22/30
 - 2s - loss: 0.0093 - acc: 0.9979 - val_loss: 0.0375 - val_acc: 0.9917
Epoch 23/30
 - 2s - loss: 0.0092 - acc: 0.9977 - val_loss: 0.0271 - val_acc: 0.9923
Epoch 24/30
 - 2s - loss: 0.0057 - acc: 0.9988 - val_loss: 0.0225 - val_acc: 0.9947
Epoch 25/30
 - 2s - loss: 0.0054 - acc: 0.9987 - val_loss: 0.0239 - val_acc: 0.9938
Epoch 26/30
 - 2s - loss: 0.0046 - acc: 0.9987 - val_loss: 0.0255 - val_acc: 0.9939
Epoch 27/30
 - 2s - loss: 0.0063 - acc: 0.9982 - val_loss: 0.0252 - val_acc: 0.9944
Epoch 28/30
 - 2s - loss: 0.0067 - acc: 0.9984 - val_loss: 0.0463 - val_acc: 0.9895
Epoch 29/30
 - 2s - loss: 0.0090 - acc: 0.9980 - val_loss: 0.0271 - val_acc: 0.9938
Epoch 30/30
 - 2s - loss: 0.0084 - acc: 0.9981 - val_loss: 0.0303 - val_acc: 0.9930
```

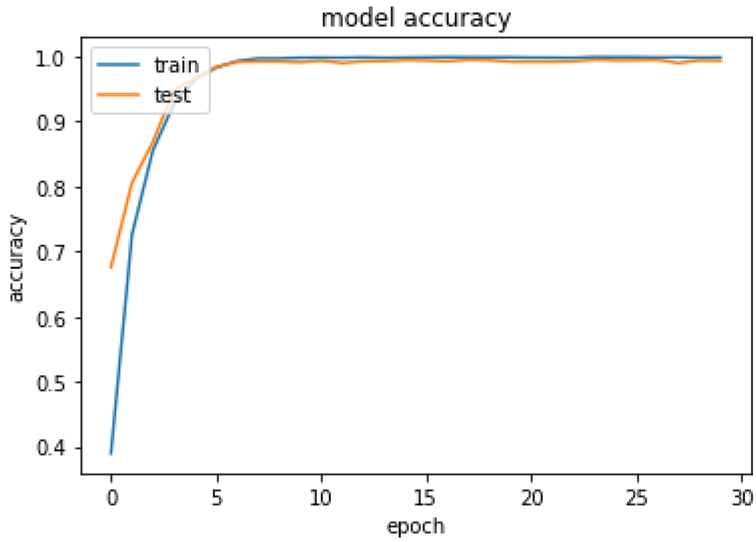
In [44]:

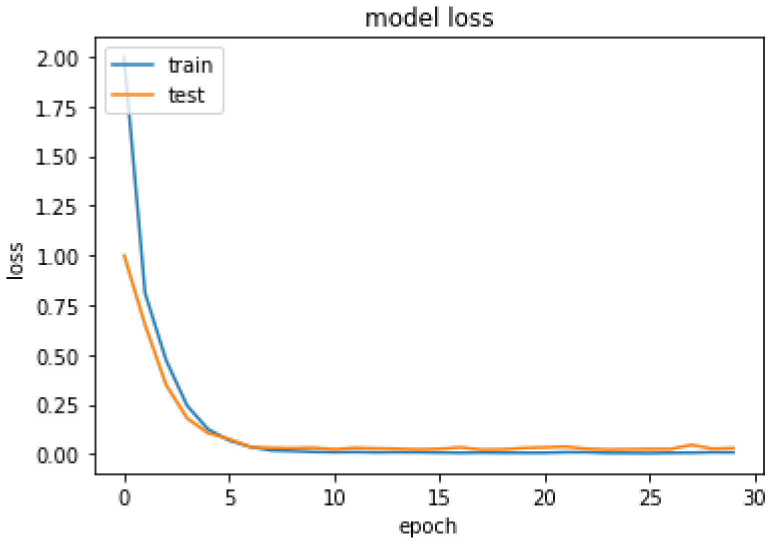
```
score = model.score(x_test, y_test)
#print('Test Loss:', score[0])
#print('Test accuracy:', score[1])
print('Testing Accuracy:',score)
```

```
Testing Accuracy: 0.9929824555509589
```

In [45]:

```
# accuracy plot
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# loss plot
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```






```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docke
r-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list al
l files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/wm811k-wafer-map/LSWMD.pkl
```

In [2]:

```
import os
from os.path import join
```

```
import numpy as np
import pandas as pd

import tensorflow as tf
import keras
from keras import layers, Input, models
from keras.utils import to_categorical
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt

datapath = join('data', 'wafer')

print(os.listdir("../input"))
import warnings
warnings.filterwarnings("ignore")
```

```
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
16: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
17: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
18: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
19: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
20: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/conda/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:5
25: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprec
ated; in a future version of numpy, it will be understood as (type, (1,)) / '(
1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
```

```
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/opt/conda/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtyp
es.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is
deprecated; in a future version of numpy, it will be understood as (type, (1,)
) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Using TensorFlow backend.
['wm811k-wafer-map']
```

In [3]:

```
df=pd.read_pickle("../input/wm811k-wafer-map/LSWMD.pkl")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 811457 entries, 0 to 811456
Data columns (total 6 columns):
waferMap          811457 non-null object
dieSize           811457 non-null float64
lotName           811457 non-null object
waferIndex        811457 non-null float64
trianTestLabel    811457 non-null object
failureType       811457 non-null object
dtypes: float64(2), object(4)
memory usage: 37.1+ MB
```

In [4]:

```
df.tail()
```

Out[4]:

	waferMap	dieSize	lotName	waferIndex	trianTestLabel	failureType
811452	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1,...	600.0	lot47542	23.0	[[Test]]	[[Edge-Ring]]
811453	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 1,...	600.0	lot47542	24.0	[[Test]]	[[Edge-Loc]]

	waferMap	dieSize	lotName	waferIndex	trianTestLabel	failureType
811454	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1,...	600.0	lot47542	25.0	[[Test]]	[[Edge-Ring]]
811455	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,...	600.0	lot47543	1.0	[]	[]
811456	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1,...	600.0	lot47543	2.0	[]	[]

In [5]:

```
df = df.drop(['waferIndex'], axis = 1)
```

In [6]:

```
def find_dim(x):
    dim0=np.size(x,axis=0)
    dim1=np.size(x,axis=1)
    return dim0,dim1
df['waferMapDim']=df.waferMap.apply(find_dim)
df.sample(5)
```

Out[6]:

	waferMap	dieSize	lotName	trianTestLabel	failureType	waferMapDim
584337	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	846.0	lot36456	[]	[]	(33, 33)
617623	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	846.0	lot38766	[]	[]	(33, 33)
245636	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1893.0	lot15374	[[Training]]	[[none]]	(50, 49)
236952	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,...	1073.0	lot14795	[[Training]]	[[Edge-Ring]]	(36, 38)
152945	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	3532.0	lot9921	[]	[]	(64, 71)

In [7]:

```
df['failureNum']=df.failureType
df['trainTestNum']=df.trianTestLabel
mapping_type={'Center':0,'Donut':1,'Edge-Loc':2,'Edge-Ring':3,'Loc':4,'Random':5,'
Scratch':6,'Near-full':7,'none':8}
mapping_traintest={'Training':0,'Test':1}
df=df.replace({'failureNum':mapping_type, 'trainTestNum':mapping_traintest})
```

In [8]:

```
tol_wafers = df.shape[0]
tol_wafers
```

Out[8]:

```
811457
```

In [9]:

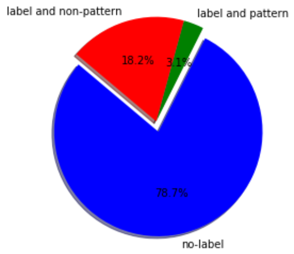
```
df_withlabel = df[(df['failureNum']>=0) & (df['failureNum']<=8)]
df_withlabel =df_withlabel.reset_index()
df_withpattern = df[(df['failureNum']>=0) & (df['failureNum']<=7)]
df_withpattern = df_withpattern.reset_index()
df_nonpattern = df[(df['failureNum']==8)]
df_withlabel.shape[0], df_withpattern.shape[0], df_nonpattern.shape[0]
```

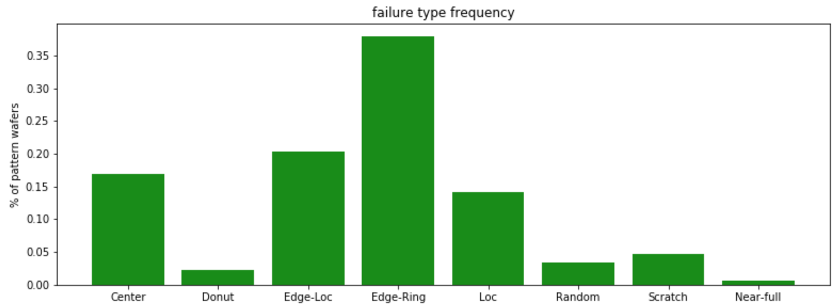
Out[9]:

```
(172950, 25519, 147431)
```

In [10]:

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

from matplotlib import gridspec
fig = plt.figure(figsize=(20, 4.5))
gs = gridspec.GridSpec(1, 2, width_ratios=[1, 2.5])
ax1 = plt.subplot(gs[0])
ax2 = plt.subplot(gs[1])

no_wafers=[tol_wafers-df_withlabel.shape[0], df_withpattern.shape[0], df_nonpatter
n.shape[0]]

colors = ['blue', 'green', 'red']
explode = (0.1, 0, 0)  # explode 1st slice
labels = ['no-label','label and pattern','label and non-pattern']
ax1.pie(no_wafers, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%
', shadow=True, startangle=140)

uni_pattern=np.unique(df_withpattern.failureNum, return_counts=True)
labels2 = ['','Center','Donut','Edge-Loc','Edge-Ring','Loc','Random','Scratch','Ne
ar-full']
ax2.bar(uni_pattern[0],uni_pattern[1]/df_withpattern.shape[0], color='green', alig
n='center', alpha=0.9)
ax2.set_title("failure type frequency")
ax2.set_ylabel("% of pattern wafers")
ax2.set_xticklabels(labels2)

plt.show()
```





In [11]:

```
sub_df = df.loc[df['waferMapDim'] == (26, 26)]
sub_wafer = sub_df['waferMap'].values

sw = np.ones((1, 26, 26))
label = list()

for i in range(len(sub_df)):
    # skip null label
    if len(sub_df.iloc[i,:]['failureType']) == 0:
        continue
    sw = np.concatenate((sw, sub_df.iloc[i,:]['waferMap'].reshape(1, 26, 26)))
    label.append(sub_df.iloc[i,:]['failureType'][0][0])
```

In [12]:

```
x = sw[1:]
y = np.array(label).reshape((-1,1))
```

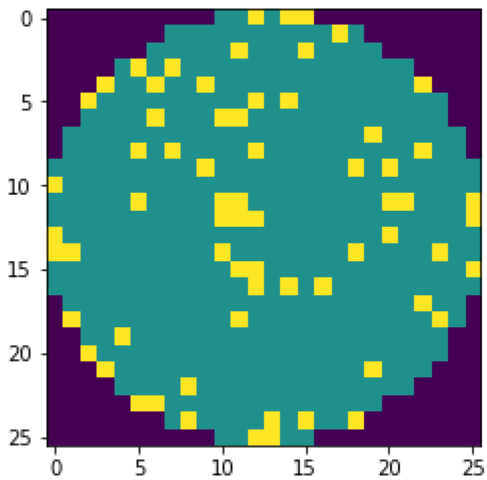
In [13]:

```
print('x shape : {}, y shape : {}'.format(x.shape, y.shape))
```

```
x shape : (14366, 26, 26), y shape : (14366, 1)
```

In [14]:

```
# plot 1st data
plt.imshow(x[2040])
plt.show()

# check faulty case
print('Faulty case : {} '.format(y[2040]))
```




```
Faulty case : ['none']
```

In [15]:

```
x = x.reshape((-1, 26, 26, 1))
```

In [16]:

```
faulty_case = np.unique(y)
print('Faulty case list : {}'.format(faulty_case))
```

```
Faulty case list : ['Center' 'Donut' 'Edge-Loc' 'Edge-Ring' 'Loc' 'Near-full'
'Random'
 'Scratch' 'none']
```

In [17]:

```
for f in faulty_case :
    print('{} : {}'.format(f, len(y[y==f])))
```

```
Center : 90
Donut : 1
Edge-Loc : 296
Edge-Ring : 31
Loc : 297
Near-full : 16
Random : 74
Scratch : 72
none : 13489
```

In [18]:

```
new_x = np.zeros((len(x), 26, 26, 3))

for w in range(len(x)):
    for i in range(26):
        for j in range(26):
            new_x[w, i, j, int(x[w, i, j])] = 1
```

In [19]:

```
new_x.shape
```

Out[19]:

```
(14366, 26, 26, 3)
```

In [20]:

```
# Encoder
input_shape = (26, 26, 3)
input_tensor = Input(input_shape)
encode = layers.Conv2D(64, (3,3), padding='same', activation='relu')(input_tensor)

latent_vector = layers.MaxPool2D()(encode)

# Decoder
decode_layer_1 = layers.Conv2DTranspose(64, (3,3), padding='same', activation='rel
u')
decode_layer_2 = layers.UpSampling2D()
output_tensor = layers.Conv2DTranspose(3, (3,3), padding='same', activation='sigmo
id')

# connect decoder layers
decode = decode_layer_1(latent_vector)
decode = decode_layer_2(decode)

ae = models.Model(input_tensor, output_tensor(decode))
ae.compile(optimizer = 'Adam',
              loss = 'mse',
             )
```

In [21]:

```
epoch=30
```

```python
batch_size=1024
```

In [22]:

```python
# start train
ae.fit(new_x, new_x,
       batch_size=batch_size,
       epochs=epoch,
       verbose=2)
```

```
Epoch 1/30
 - 7s - loss: 0.1606
Epoch 2/30
 - 1s - loss: 0.1015
Epoch 3/30
 - 1s - loss: 0.0837
Epoch 4/30
 - 1s - loss: 0.0730
Epoch 5/30
 - 1s - loss: 0.0634
Epoch 6/30
 - 1s - loss: 0.0554
Epoch 7/30
 - 1s - loss: 0.0487
Epoch 8/30
 - 1s - loss: 0.0431
Epoch 9/30
 - 1s - loss: 0.0380
Epoch 10/30
 - 1s - loss: 0.0333
Epoch 11/30
 - 1s - loss: 0.0293
Epoch 12/30
 - 1s - loss: 0.0262
Epoch 13/30
 - 1s - loss: 0.0236
Epoch 14/30
 - 1s - loss: 0.0215
Epoch 15/30
 - 1s - loss: 0.0198
Epoch 16/30
 - 1s - loss: 0.0183
Epoch 17/30
 - 1s - loss: 0.0170
Epoch 18/30
 - 1s - loss: 0.0159
Epoch 19/30
 - 1s - loss: 0.0149
Epoch 20/30
 - 1s - loss: 0.0139
Epoch 21/30
 - 1s - loss: 0.0131
Epoch 22/30
 - 1s - loss: 0.0123
Epoch 23/30
 - 1s - loss: 0.0116
Epoch 24/30
 - 1s - loss: 0.0109
Epoch 25/30
```

```
 - 1s - loss: 0.0103
Epoch 26/30
 - 1s - loss: 0.0098
Epoch 27/30
 - 1s - loss: 0.0093
Epoch 28/30
 - 1s - loss: 0.0088
Epoch 29/30
 - 1s - loss: 0.0084
Epoch 30/30
 - 1s - loss: 0.0080
```

Out[22]:

```
<keras.callbacks.History at 0x7fc1cd5e3588>
```

In [23]:

```
encoder = models.Model(input_tensor, latent_vector)
```

In [24]:

```
decoder_input = Input((13, 13, 64))
decode = decode_layer_1(decoder_input)
decode = decode_layer_2(decode)

decoder = models.Model(decoder_input, output_tensor(decode))
```

In [25]:

```
# Encode original faulty wafer
encoded_x = encoder.predict(new_x)
```

In [26]:

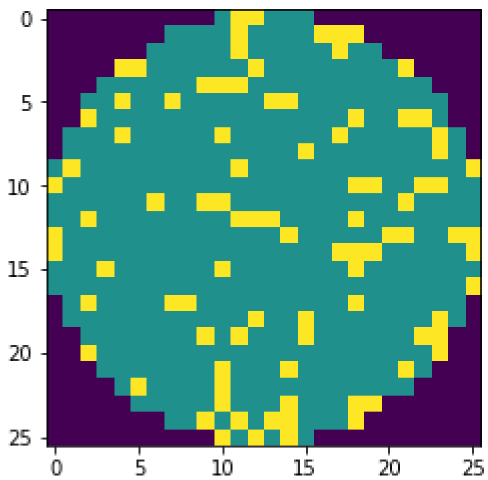
```
# Add noise to encoded latent faulty wafers vector.
noised_encoded_x = encoded_x + np.random.normal(loc=0, scale=0.1, size = (len(enco
ded_x), 13, 13, 64))
```

In [27]:

```
# check original faulty wafer data
plt.imshow(np.argmax(new_x[3], axis=2))
```

Out[27]:

```
<matplotlib.image.AxesImage at 0x7fc1e01fe4a8>
```

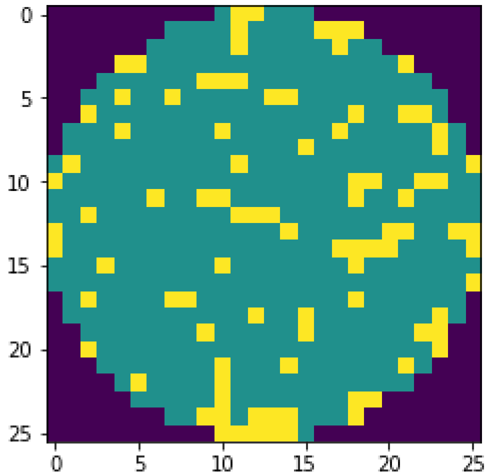


In [28]:

```
# check new noised faulty wafer data
noised_gen_x = np.argmax(decoder.predict(noised_encoded_x), axis=3)
plt.imshow(noised_gen_x[3])
```

Out[28]:

```
<matplotlib.image.AxesImage at 0x7fc1e0394f98>
```



In [29]:

```
# augment function define
def gen_data(wafer, label):
    # Encode input wafer
    encoded_x = encoder.predict(wafer)

    # dummy array for collecting noised wafer
    gen_x = np.zeros((1, 26, 26, 3))

    # Make wafer until total # of wafer to 2000
    for i in range((2000//len(wafer)) + 1):
        noised_encoded_x = encoded_x + np.random.normal(loc=0, scale=0.1, size = (
len(encoded_x), 13, 13, 64))
        noised_gen_x = decoder.predict(noised_encoded_x)
        gen_x = np.concatenate((gen_x, noised_gen_x), axis=0)
    # also make label vector with same length
    gen_y = np.full((len(gen_x), 1), label)

    # return date without 1st dummy data.
    return gen_x[1:], gen_y[1:]
```

In [30]:

```
# Augmentation for all faulty case.
for f in faulty_case :
    # skip none case
    if f == 'none' :
        continue

    gen_x, gen_y = gen_data(new_x[np.where(y==f)[0]], f)
    new_x = np.concatenate((new_x, gen_x), axis=0)
    y = np.concatenate((y, gen_y))
```

In [31]:

```
print('After Generate new_x shape : {}, new_y shape : {}'.format(new_x.shape, y.sh
ape))
```

```
After Generate new_x shape : (30707, 26, 26, 3), new_y shape : (30707, 1)
```

In [32]:

```
for f in faulty_case :
    print('{} : {}'.format(f, len(y[y==f])))
```

```
Center : 2160
```

```
Donut : 2002
Edge-Loc : 2368
Edge-Ring : 2046
Loc : 2376
Near-full : 2032
Random : 2146
Scratch : 2088
none : 13489
```

In [33]:

```
none_idx = np.where(y=='none')[0][np.random.choice(len(np.where(y=='none')[0]), si
ze=11000, replace=False)]
```

In [34]:

```
new_x = np.delete(new_x, none_idx, axis=0)
new_y = np.delete(y, none_idx, axis=0)
```

In [35]:

```
print('After Delete "none" class new_x shape : {}, new_y shape : {}'.format(new_x.
shape, new_y.shape))
```

```
After Delete "none" class new_x shape : (19707, 26, 26, 3), new_y shape : (197
07, 1)
```

In [36]:

```
for f in faulty_case :
    print('{} : {}'.format(f, len(new_y[new_y==f])))
```

```
Center : 2160
Donut : 2002
Edge-Loc : 2368
Edge-Ring : 2046
Loc : 2376
Near-full : 2032
Random : 2146
Scratch : 2088
none : 2489
```

In [37]:

```
for i, l in enumerate(faulty_case):
    new_y[new_y==l] = i

# one-hot-encoding
new_y = to_categorical(new_y)
```

In [38]:

```
new_X=new_x[0:19000]
new_Y=new_y[0:19000]
test_x=new_x[19001:19706]
test_y=new_y[19001:19706]
test_x.shape
```

Out[38]:

```
(705, 26, 26, 3)
```

In [39]:

```
x_train, x_test, y_train, y_test = train_test_split(new_X, new_Y,
                                                    test_size=0.33,
                                                    random_state=2019)
```

In [40]:

```
print('Train x : {}, y : {}'.format(x_train.shape, y_train.shape))
print('Test x: {}, y : {}'.format(x_test.shape, y_test.shape))
```

```
Train x : (12730, 26, 26, 3), y : (12730, 9)
Test x: (6270, 26, 26, 3), y : (6270, 9)
```

In [41]:

```python
def create_model():
    input_shape = (26, 26, 3)
    input_tensor = Input(input_shape)

    conv_1 = layers.Conv2D(16, (3,3), activation='relu', padding='same')(input_ten
sor)
    conv_2 = layers.Conv2D(64, (3,3), activation='relu', padding='same')(conv_1)
    conv_3 = layers.Conv2D(128, (3,3), activation='relu', padding='same')(conv_2)

    flat = layers.Flatten()(conv_3)

    dense_1 = layers.Dense(512, activation='relu')(flat)
    dense_2 = layers.Dense(128, activation='relu')(dense_1)
    output_tensor = layers.Dense(9, activation='softmax')(dense_2)

    model = models.Model(input_tensor, output_tensor)
    model.compile(optimizer='Adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

In [42]:

```python
model = KerasClassifier(build_fn=create_model, epochs=30, batch_size=1024, verbose
=2)
# 3-Fold Crossvalidation
kfold = KFold(n_splits=3, shuffle=True, random_state=2019)
results = cross_val_score(model, x_train, y_train, cv=kfold)
# Check 3-fold model's mean accuracy
print('Simple CNN Cross validation score : {:.4f}'.format(np.mean(results)))
```

```
Epoch 1/30
 - 4s - loss: 2.2454 - acc: 0.3998
Epoch 2/30
 - 1s - loss: 0.8462 - acc: 0.6914
Epoch 3/30
 - 1s - loss: 0.4931 - acc: 0.8223
Epoch 4/30
 - 1s - loss: 0.5228 - acc: 0.8505
Epoch 5/30
 - 1s - loss: 0.2987 - acc: 0.9187
Epoch 6/30
 - 1s - loss: 0.1622 - acc: 0.9566
Epoch 7/30
 - 1s - loss: 0.0928 - acc: 0.9760
Epoch 8/30
 - 1s - loss: 0.0763 - acc: 0.9802
Epoch 9/30
 - 1s - loss: 0.0419 - acc: 0.9903
Epoch 10/30
 - 1s - loss: 0.0273 - acc: 0.9948
Epoch 11/30
 - 1s - loss: 0.0166 - acc: 0.9976
Epoch 12/30
 - 1s - loss: 0.0121 - acc: 0.9980
```

```
Epoch 13/30
 - 1s - loss: 0.0111 - acc: 0.9972
Epoch 14/30
 - 1s - loss: 0.0082 - acc: 0.9986
Epoch 15/30
 - 1s - loss: 0.0066 - acc: 0.9987
Epoch 16/30
 - 1s - loss: 0.0091 - acc: 0.9981
Epoch 17/30
 - 1s - loss: 0.0103 - acc: 0.9979
Epoch 18/30
 - 1s - loss: 0.0096 - acc: 0.9987
Epoch 19/30
 - 1s - loss: 0.0066 - acc: 0.9989
Epoch 20/30
 - 1s - loss: 0.0058 - acc: 0.9988
Epoch 21/30
 - 1s - loss: 0.0043 - acc: 0.9989
Epoch 22/30
 - 1s - loss: 0.0044 - acc: 0.9989
Epoch 23/30
 - 1s - loss: 0.0053 - acc: 0.9984
Epoch 24/30
 - 1s - loss: 0.0039 - acc: 0.9985
Epoch 25/30
 - 1s - loss: 0.0046 - acc: 0.9988
Epoch 26/30
 - 1s - loss: 0.0059 - acc: 0.9985
Epoch 27/30
 - 1s - loss: 0.0059 - acc: 0.9989
Epoch 28/30
 - 1s - loss: 0.0058 - acc: 0.9981
Epoch 29/30
 - 1s - loss: 0.0039 - acc: 0.9988
Epoch 30/30
 - 1s - loss: 0.0043 - acc: 0.9982
Epoch 1/30
 - 3s - loss: 2.9482 - acc: 0.2222
Epoch 2/30
 - 1s - loss: 1.1951 - acc: 0.6080
Epoch 3/30
 - 1s - loss: 0.8000 - acc: 0.7104
Epoch 4/30
 - 1s - loss: 0.5120 - acc: 0.8187
Epoch 5/30
 - 1s - loss: 0.3506 - acc: 0.8742
Epoch 6/30
 - 1s - loss: 0.2443 - acc: 0.9269
Epoch 7/30
 - 1s - loss: 0.1692 - acc: 0.9511
Epoch 8/30
 - 1s - loss: 0.0986 - acc: 0.9770
Epoch 9/30
 - 1s - loss: 0.0612 - acc: 0.9879
Epoch 10/30
 - 1s - loss: 0.0349 - acc: 0.9945
```

```
Epoch 11/30
 - 1s - loss: 0.0213 - acc: 0.9979
Epoch 12/30
 - 1s - loss: 0.0170 - acc: 0.9975
Epoch 13/30
 - 1s - loss: 0.0116 - acc: 0.9981
Epoch 14/30
 - 1s - loss: 0.0084 - acc: 0.9987
Epoch 15/30
 - 1s - loss: 0.0084 - acc: 0.9986
Epoch 16/30
 - 1s - loss: 0.0093 - acc: 0.9984
Epoch 17/30
 - 1s - loss: 0.0093 - acc: 0.9987
Epoch 18/30
 - 1s - loss: 0.0073 - acc: 0.9982
Epoch 19/30
 - 1s - loss: 0.0060 - acc: 0.9989
Epoch 20/30
 - 1s - loss: 0.0061 - acc: 0.9985
Epoch 21/30
 - 1s - loss: 0.0059 - acc: 0.9986
Epoch 22/30
 - 1s - loss: 0.0096 - acc: 0.9979
Epoch 23/30
 - 1s - loss: 0.0347 - acc: 0.9934
Epoch 24/30
 - 1s - loss: 5.0369 - acc: 0.5391
Epoch 25/30
 - 1s - loss: 3.0918 - acc: 0.5187
Epoch 26/30
 - 1s - loss: 1.0979 - acc: 0.6730
Epoch 27/30
 - 1s - loss: 0.4224 - acc: 0.8465
Epoch 28/30
 - 1s - loss: 0.1743 - acc: 0.9532
Epoch 29/30
 - 1s - loss: 0.0794 - acc: 0.9775
Epoch 30/30
 - 1s - loss: 0.0457 - acc: 0.9897
Epoch 1/30
 - 3s - loss: 2.2367 - acc: 0.3223
Epoch 2/30
 - 1s - loss: 0.9060 - acc: 0.6612
Epoch 3/30
 - 1s - loss: 0.7930 - acc: 0.7651
Epoch 4/30
 - 1s - loss: 0.5420 - acc: 0.8343
Epoch 5/30
 - 1s - loss: 0.3333 - acc: 0.8924
Epoch 6/30
 - 1s - loss: 0.1972 - acc: 0.9441
Epoch 7/30
 - 1s - loss: 0.1181 - acc: 0.9684
Epoch 8/30
 - 1s - loss: 0.0641 - acc: 0.9846
```

```
Epoch 9/30
 - 1s - loss: 0.0339 - acc: 0.9932
Epoch 10/30
 - 1s - loss: 0.0207 - acc: 0.9961
Epoch 11/30
 - 1s - loss: 0.0151 - acc: 0.9975
Epoch 12/30
 - 1s - loss: 0.0131 - acc: 0.9973
Epoch 13/30
 - 1s - loss: 0.0124 - acc: 0.9975
Epoch 14/30
 - 1s - loss: 0.0140 - acc: 0.9968
Epoch 15/30
 - 1s - loss: 0.0118 - acc: 0.9973
Epoch 16/30
 - 1s - loss: 0.0101 - acc: 0.9979
Epoch 17/30
 - 1s - loss: 0.0121 - acc: 0.9982
Epoch 18/30
 - 1s - loss: 0.0095 - acc: 0.9981
Epoch 19/30
 - 1s - loss: 0.0083 - acc: 0.9982
Epoch 20/30
 - 1s - loss: 0.0104 - acc: 0.9976
Epoch 21/30
 - 1s - loss: 0.0102 - acc: 0.9975
Epoch 22/30
 - 1s - loss: 0.0107 - acc: 0.9980
Epoch 23/30
 - 1s - loss: 0.0092 - acc: 0.9980
Epoch 24/30
 - 1s - loss: 0.0109 - acc: 0.9972
Epoch 25/30
 - 1s - loss: 0.0098 - acc: 0.9978
Epoch 26/30
 - 1s - loss: 0.0094 - acc: 0.9978
Epoch 27/30
 - 1s - loss: 0.0075 - acc: 0.9981
Epoch 28/30
 - 1s - loss: 0.0080 - acc: 0.9982
Epoch 29/30
 - 1s - loss: 0.0061 - acc: 0.9986
Epoch 30/30
 - 1s - loss: 0.0073 - acc: 0.9986
Simple CNN Cross validation score : 0.9833
```

In [43]:

```
history = model.fit(x_train, y_train,
         validation_data=[x_test, y_test],
         epochs=epoch,
         batch_size=batch_size,
         )
```

```
Train on 12730 samples, validate on 6270 samples
Epoch 1/30
 - 4s - loss: 1.9970 - acc: 0.3897 - val_loss: 0.9990 - val_acc: 0.6754
Epoch 2/30
 - 2s - loss: 0.8095 - acc: 0.7253 - val_loss: 0.6484 - val_acc: 0.8051
```

```
Epoch 3/30
 - 2s - loss: 0.4719 - acc: 0.8554 - val_loss: 0.3474 - val_acc: 0.8684
Epoch 4/30
 - 2s - loss: 0.2440 - acc: 0.9266 - val_loss: 0.1800 - val_acc: 0.9475
Epoch 5/30
 - 2s - loss: 0.1248 - acc: 0.9649 - val_loss: 0.1054 - val_acc: 0.9662
Epoch 6/30
 - 2s - loss: 0.0691 - acc: 0.9827 - val_loss: 0.0764 - val_acc: 0.9839
Epoch 7/30
 - 2s - loss: 0.0371 - acc: 0.9926 - val_loss: 0.0351 - val_acc: 0.9914
Epoch 8/30
 - 2s - loss: 0.0183 - acc: 0.9967 - val_loss: 0.0316 - val_acc: 0.9920
Epoch 9/30
 - 2s - loss: 0.0156 - acc: 0.9966 - val_loss: 0.0298 - val_acc: 0.9922
Epoch 10/30
 - 2s - loss: 0.0111 - acc: 0.9977 - val_loss: 0.0324 - val_acc: 0.9907
Epoch 11/30
 - 2s - loss: 0.0093 - acc: 0.9980 - val_loss: 0.0242 - val_acc: 0.9935
Epoch 12/30
 - 2s - loss: 0.0104 - acc: 0.9977 - val_loss: 0.0316 - val_acc: 0.9896
Epoch 13/30
 - 2s - loss: 0.0085 - acc: 0.9984 - val_loss: 0.0288 - val_acc: 0.9925
Epoch 14/30
 - 2s - loss: 0.0093 - acc: 0.9977 - val_loss: 0.0266 - val_acc: 0.9927
Epoch 15/30
 - 2s - loss: 0.0089 - acc: 0.9980 - val_loss: 0.0231 - val_acc: 0.9943
Epoch 16/30
 - 2s - loss: 0.0076 - acc: 0.9984 - val_loss: 0.0265 - val_acc: 0.9936
Epoch 17/30
 - 2s - loss: 0.0066 - acc: 0.9987 - val_loss: 0.0338 - val_acc: 0.9923
Epoch 18/30
 - 2s - loss: 0.0077 - acc: 0.9986 - val_loss: 0.0223 - val_acc: 0.9946
Epoch 19/30
 - 2s - loss: 0.0068 - acc: 0.9984 - val_loss: 0.0237 - val_acc: 0.9941
Epoch 20/30
 - 2s - loss: 0.0065 - acc: 0.9985 - val_loss: 0.0308 - val_acc: 0.9917
Epoch 21/30
 - 2s - loss: 0.0065 - acc: 0.9980 - val_loss: 0.0337 - val_acc: 0.9917
Epoch 22/30
 - 2s - loss: 0.0093 - acc: 0.9979 - val_loss: 0.0375 - val_acc: 0.9917
Epoch 23/30
 - 2s - loss: 0.0092 - acc: 0.9977 - val_loss: 0.0271 - val_acc: 0.9923
Epoch 24/30
 - 2s - loss: 0.0057 - acc: 0.9988 - val_loss: 0.0225 - val_acc: 0.9947
Epoch 25/30
 - 2s - loss: 0.0054 - acc: 0.9987 - val_loss: 0.0239 - val_acc: 0.9938
Epoch 26/30
 - 2s - loss: 0.0046 - acc: 0.9987 - val_loss: 0.0255 - val_acc: 0.9939
Epoch 27/30
 - 2s - loss: 0.0063 - acc: 0.9982 - val_loss: 0.0252 - val_acc: 0.9944
Epoch 28/30
 - 2s - loss: 0.0067 - acc: 0.9984 - val_loss: 0.0463 - val_acc: 0.9895
Epoch 29/30
 - 2s - loss: 0.0090 - acc: 0.9980 - val_loss: 0.0271 - val_acc: 0.9938
Epoch 30/30
 - 2s - loss: 0.0084 - acc: 0.9981 - val_loss: 0.0303 - val_acc: 0.9930
```

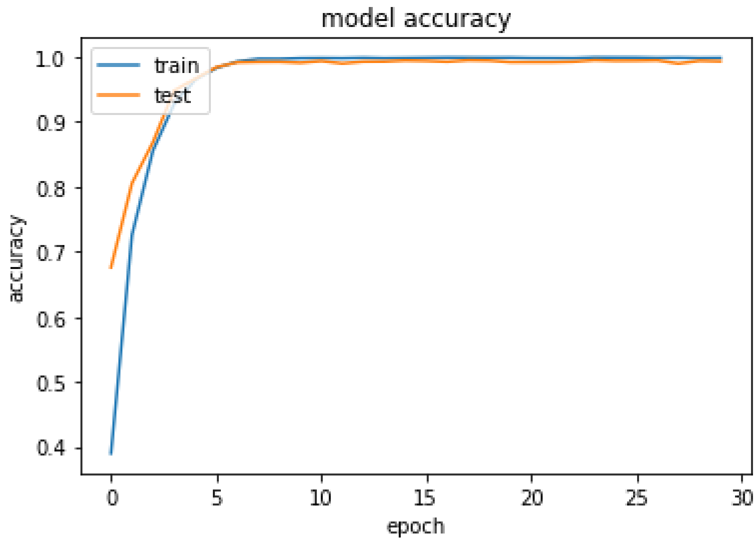
In [44]:

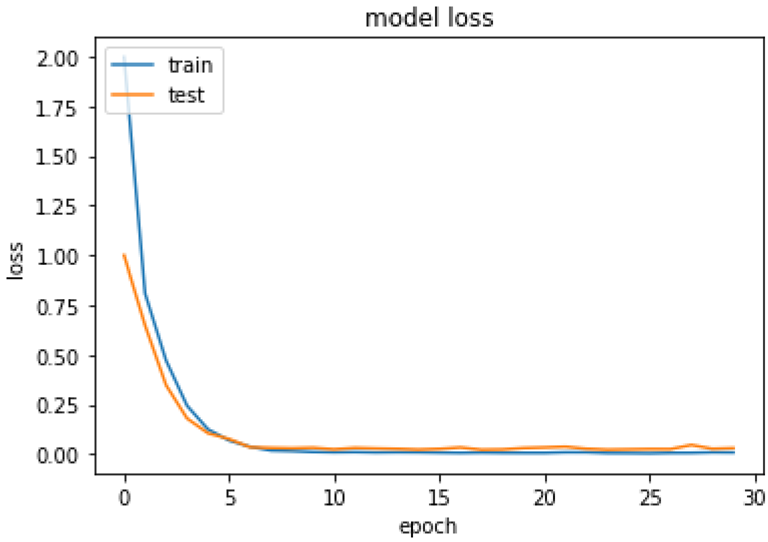
```
score = model.score(x_test, y_test)
#print('Test Loss:', score[0])
#print('Test accuracy:', score[1])
print('Testing Accuracy:',score)
```

```
Testing Accuracy: 0.9929824555509589
```

In [45]:

```
# accuracy plot
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# loss plot
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




model loss
train
test
loss
epoch
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
25
30