

In [1]:

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

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 read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all 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))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserve
d as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of
the current session
```

```
/kaggle/input/skin-cancer-mnist-ham10000/hmnist_8_8_RGB.csv
/kaggle/input/skin-cancer-mnist-ham10000/hmnist_28_28_RGB.csv
/kaggle/input/skin-cancer-mnist-ham10000/hmnist_8_8_L.csv
/kaggle/input/skin-cancer-mnist-ham10000/hmnist_28_28_L.csv
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028933.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028394.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027799.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028100.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027960.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028872.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026412.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0024872.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026232.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027031.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026692.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025729.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028582.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0024754.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027742.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025408.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026474.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027164.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027228.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028059.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028588.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0029299.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027221.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0024912.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028259.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025844.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028090.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025454.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0028842.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026980.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026662.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025037.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025876.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0024901.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025611.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027201.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0025536.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0027941.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026620.jpg
/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_images_part_1/ISIC_0026352.jpg
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



















[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]















[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



















[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



















[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]







[illegible]



[illegible]

[illegible]





































[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]







[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0034010.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0031255.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0031832.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030545.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030994.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0031911.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032993.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032613.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032376.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033485.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0029948.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033802.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032466.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0029393.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0029423.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032058.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032155.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0031859.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0034076.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033220.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030845.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033255.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033703.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0034090.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032771.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030265.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0029330.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032485.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033781.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0031869.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030474.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030459.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033906.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032422.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0031726.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0031279.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033165.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032972.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030091.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030780.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030674.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0029733.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0033470.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0032153.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030216.jpg  
/kaggle/input/skin-cancer-mnist-ham10000/ham10000\_images\_part\_2/ISIC\_0030344.jpg

In [2]:

```
import tensorflow as tf
from keras.callbacks import ReduceLROnPlateau
from imblearn.over_sampling import RandomOverSampler
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import to_categorical
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix , classification_report
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

## Loading the Dataset

In [3]:

```
FilePath = "../input/skin-cancer-mnist-ham10000/hmnist_28_28_RGB.csv"
dataSet = pd.read_csv(FilePath)
```

```
In [4]:
```

```
Label = dataSet["label"]  
Data = dataSet.drop(columns=["label"])
```

### Count labels using countlabel() method

```
In [5]:
```

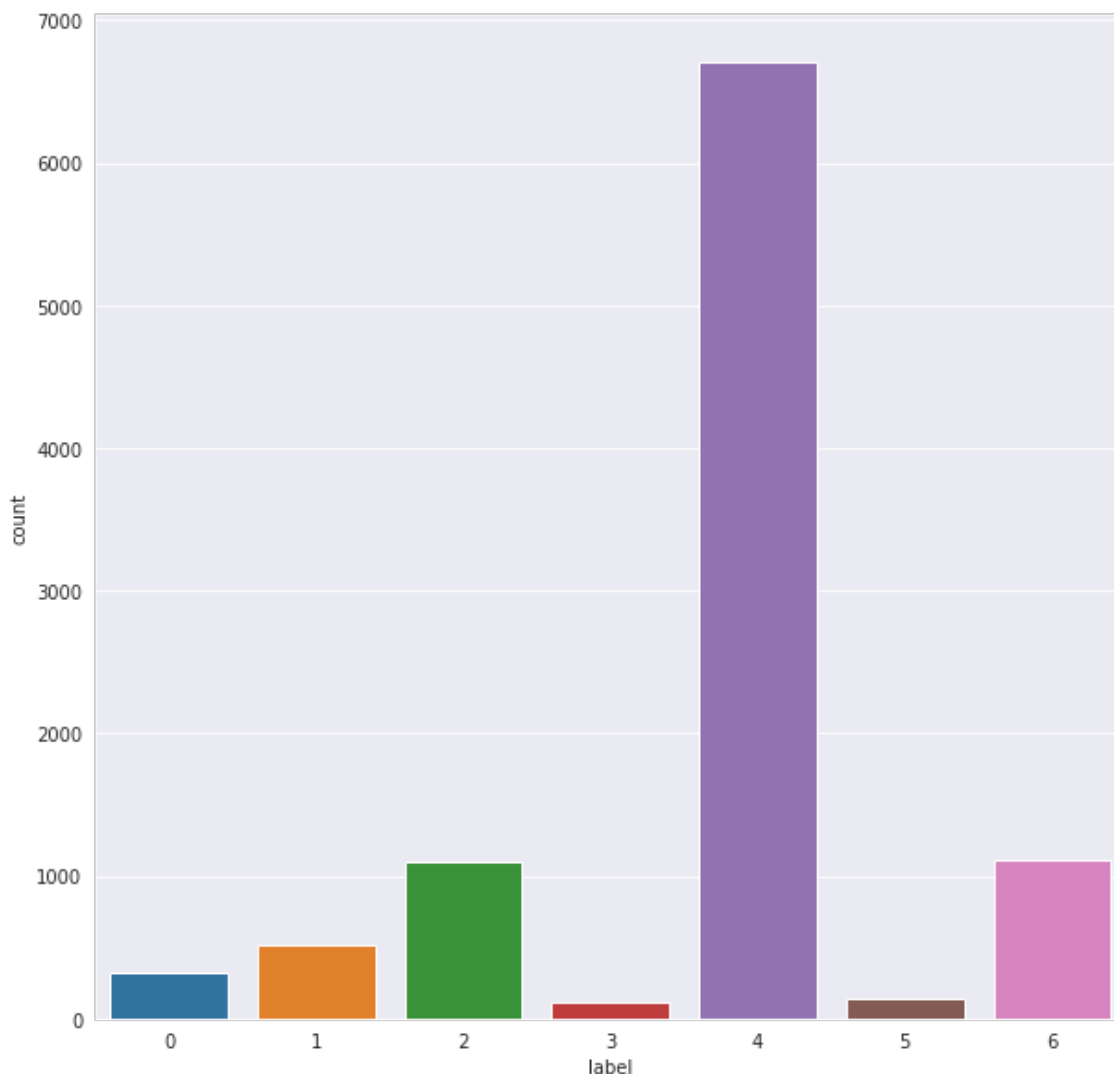
```
plt.figure(figsize = (10,10))  
sns.set_style("darkgrid")  
sns.countplot(Label)
```

/opt/conda/lib/python3.7/site-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

FutureWarning

```
Out[5]:
```

```
<AxesSubplot:xlabel='label', ylabel='count'>
```



**As you see the data is imbalanced , so let's make it balanced**

```
In [6]:
```

```
oversample = RandomOverSampler()  
Data,Label = oversample.fit_resample(Data,Label)  
Data = np.array(Data).reshape(-1,28,28,3)  
print('Shape of Data :',Data.shape)
```

```
Shape of Data : (46935, 28, 28, 3)
```

```
In [7]:
```

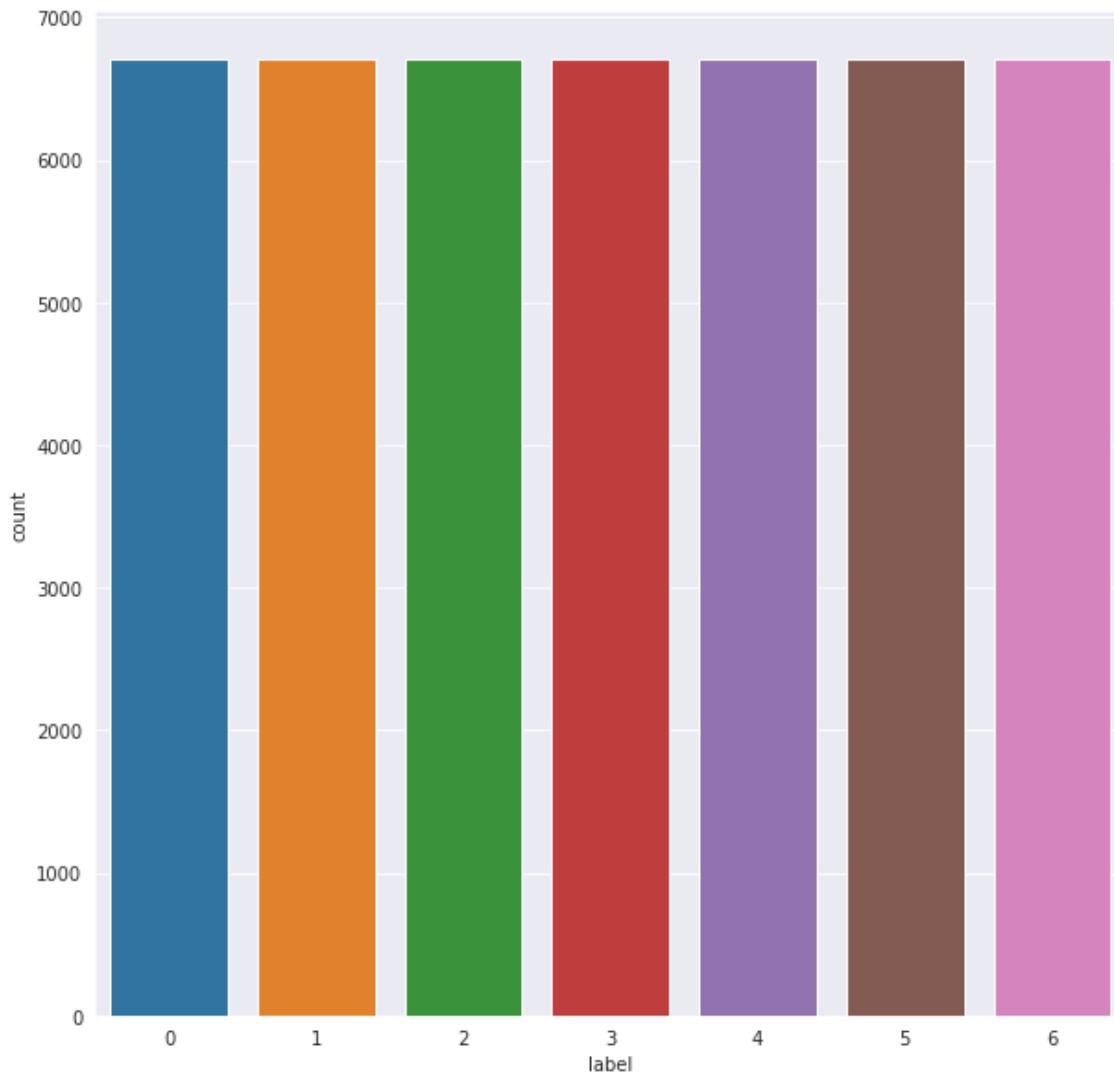
```
plt.figure(figsize = (10,10))
sns.set_style("darkgrid")
sns.countplot(Label)
```

/opt/conda/lib/python3.7/site-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

FutureWarning

Out[7]:

<AxesSubplot:xlabel='label', ylabel='count'>



In [8]:

```
Label = np.array(Label)
Label
```

Out[8]:

array([2, 2, 2, ..., 6, 6, 6])

In [9]:

```
classes = {4: ('nv', ' melanocytic nevi'),
            6: ('mel', 'melanoma'),
            2: ('bkl', 'benign keratosis-like lesions'),
            1: ('bcc', ' basal cell carcinoma'),
            5: ('vasc', ' pyogenic granulomas and hemorrhage'),
            0: ('akiec', 'Actinic keratoses and intraepithelial carcinomae'),
            3: ('df', 'dermatofibroma')}
```

## SPLIT DATA INTO TRAIN AND TEST DATA



In [10]:

```
X_train , X_test , y_train , y_test = train_test_split(Data , Label , test_size = 0.25 ,
random_state = 49)
```

In [11]:

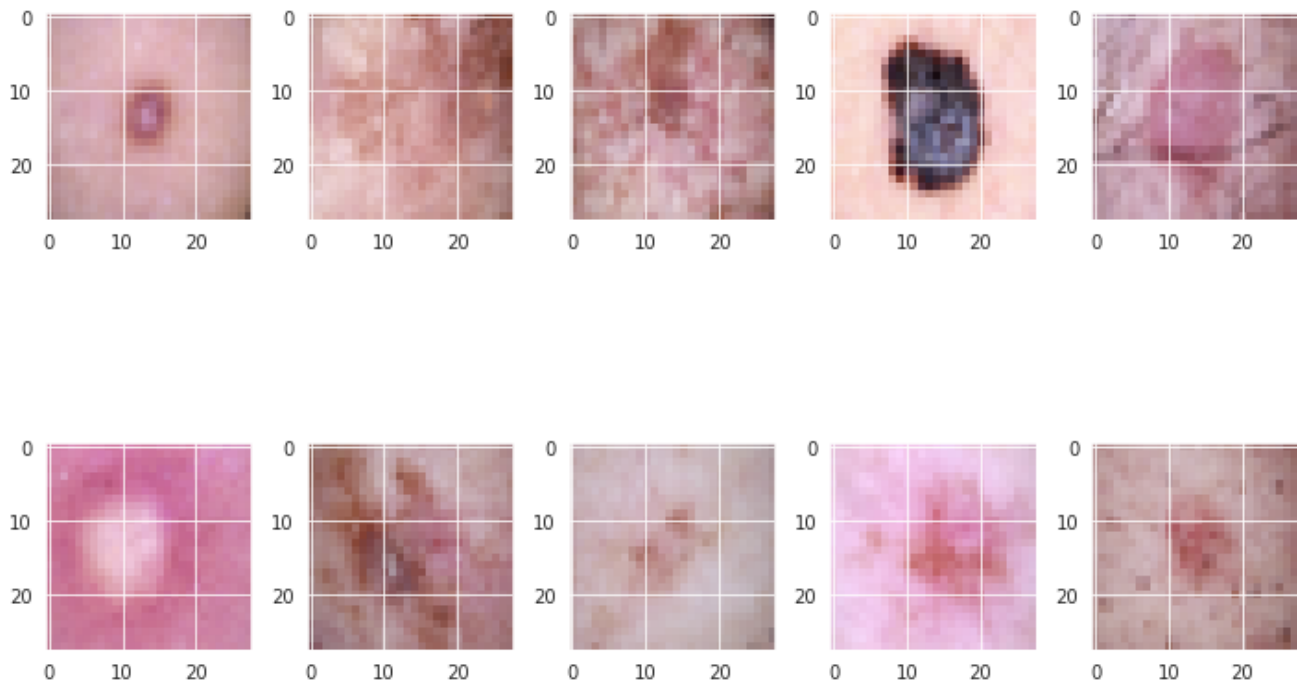
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(35201, 28, 28, 3)
(11734, 28, 28, 3)
(35201,)
(11734,)
```

## Plot images

In [12]:

```
f , ax = plt.subplots(2,5)
f.set_size_inches(10, 10)
k = 0
for i in range(2):
    for j in range(5):
        ax[i,j].imshow(X_train[k].reshape(28,28,3))
        k = k + 1
plt.tight_layout()
```



## Convert lables into One-hot encoding

In [13]:

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [14]:

```
print(y_train)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
```

```
...
[1. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 0. 0.]
[0. 0. 1. ... 0. 0. 0.]]
```

## use Data Augmentation to avoid Overfitting , using by ImageDataGenerator method

In [15]:

```
datagen = ImageDataGenerator(rescale=(1./255)
                             ,rotation_range=10
                             ,zoom_range = 0.1
                             ,width_shift_range=0.1
                             ,height_shift_range=0.1)
testgen = ImageDataGenerator(rescale=(1./255))
```

## Learning Rate Decay

In [16]:

```
learning_rate_reduction = ReduceLROnPlateau (monitor='val_accuracy'
                                              , patience = 2
                                              , verbose=1
                                              , factor=0.5
                                              , min_lr=0.00001)
```

# Build Our Model

In [17]:

```
def My_Model():
    input_ = keras.layers.Input(shape = [28,28,3])
    x = keras.layers.Conv2D(32 , (3,3) , activation='relu',padding='same' , kernel_initializer='he_normal')(input_)
    x = keras.layers.MaxPooling2D()(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(64 , (3,3) , activation='relu',padding='same' , kernel_initializer='he_normal')(x)
    x = keras.layers.Conv2D(64 , (3,3) , activation='relu',padding='same' , kernel_initializer='he_normal')(x)
    x = keras.layers.MaxPooling2D()(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(128 , (3,3) , activation='relu',padding='same' , kernel_initializer='he_normal')(x)
    x = keras.layers.Conv2D(128 , (3,3) , activation='relu',padding='same' , kernel_initializer='he_normal')(x)
    x = keras.layers.MaxPooling2D()(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(256 , (3,3) , activation='relu' ,padding='same', kernel_initializer='he_normal')(x)
    x = keras.layers.Conv2D(256 , (3,3) , activation='relu' ,padding='same', kernel_initializer='he_normal')(x)
    x = keras.layers.MaxPooling2D()(x)
    flatten = keras.layers.Flatten()(x)
    classifier = keras.layers.Dropout(rate = 0.2)(flatten)
    classifier = keras.layers.Dense(units = 256 , activation = 'relu' , kernel_initializer = 'he_normal')(classifier)
    classifier = keras.layers.BatchNormalization()(classifier)
    classifier = keras.layers.Dense(units = 128 , activation = 'relu' , kernel_initializer = 'he_normal')(classifier)
    classifier = keras.layers.BatchNormalization()(classifier)
    classifier = keras.layers.Dense(units = 64 , activation = 'relu' , kernel_initializer = 'he_normal')(classifier)
    classifier = keras.layers.BatchNormalization()(classifier)
    classifier = keras.layers.Dense(units = 32 , activation = 'relu' , kernel_initializer = 'he_normal' , kernel_regularizer=keras.regularizers.L1L2())(classifier)
    classifier = keras.layers.BatchNormalization()(classifier)
    classifier = keras.layers.Dense(units = 7 , activation='softmax' , kernel_initializer="
```

```
glorot_uniform" , name = 'classifier')(classifier)

return keras.models.Model(inputs = input_ ,outputs = classifier )
```

In [18]:

```
model = My_Model()
```

In [19]:

```
model.summary()
```

Model: "model"

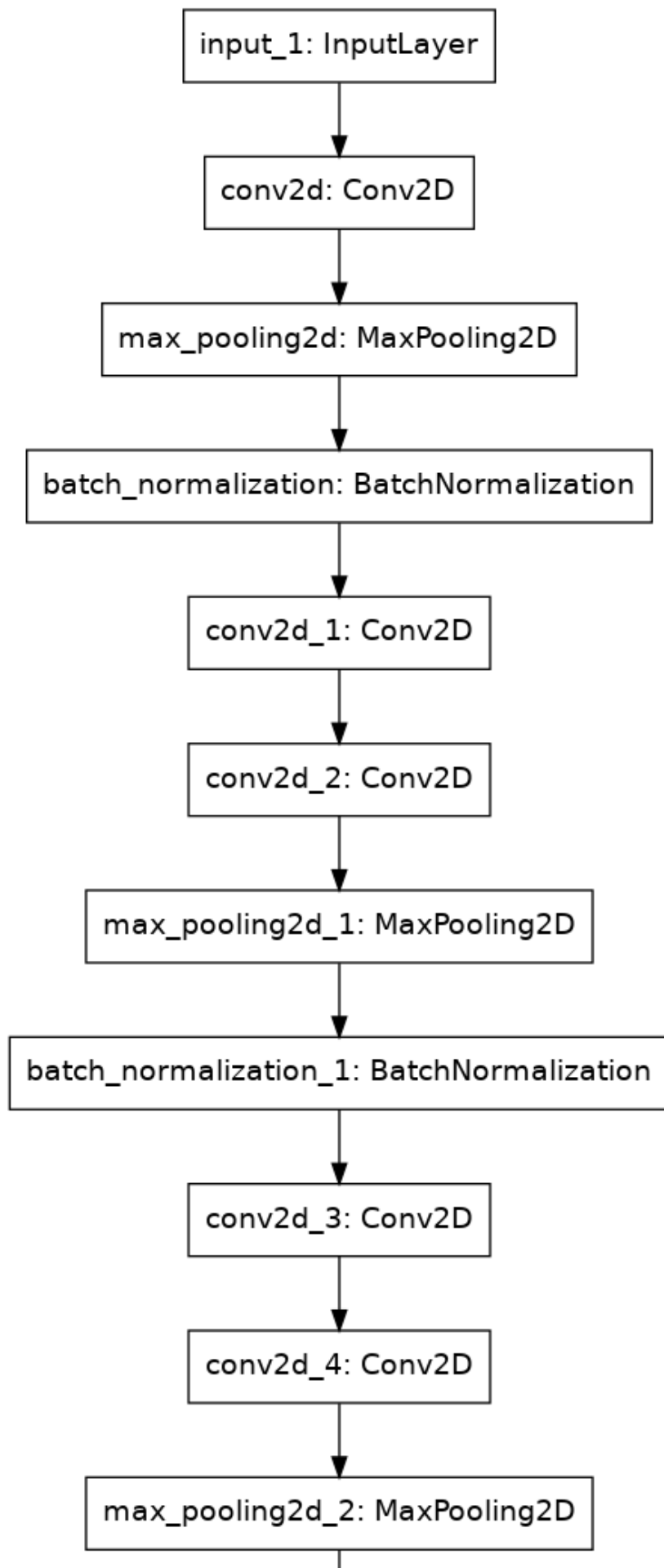
Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 28, 28, 3)]	0
conv2d (Conv2D)	(None, 28, 28, 32)	896
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
batch_normalization (BatchNo	(None, 14, 14, 32)	128
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18496
conv2d_2 (Conv2D)	(None, 14, 14, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 7, 7, 64)	0
batch_normalization_1 (Batch	(None, 7, 7, 64)	256
conv2d_3 (Conv2D)	(None, 7, 7, 128)	73856
conv2d_4 (Conv2D)	(None, 7, 7, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 3, 3, 128)	0
batch_normalization_2 (Batch	(None, 3, 3, 128)	512
conv2d_5 (Conv2D)	(None, 3, 3, 256)	295168
conv2d_6 (Conv2D)	(None, 3, 3, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 1, 1, 256)	0
flatten (Flatten)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
dense (Dense)	(None, 256)	65792
batch_normalization_3 (Batch	(None, 256)	1024
dense_1 (Dense)	(None, 128)	32896
batch_normalization_4 (Batch	(None, 128)	512
dense_2 (Dense)	(None, 64)	8256
batch_normalization_5 (Batch	(None, 64)	256
dense_3 (Dense)	(None, 32)	2080
batch_normalization_6 (Batch	(None, 32)	128
classifier (Dense)	(None, 7)	231
=====		
Total params: 1,275,079		
Trainable params: 1,273,671		
Non-trainable params: 1,408		

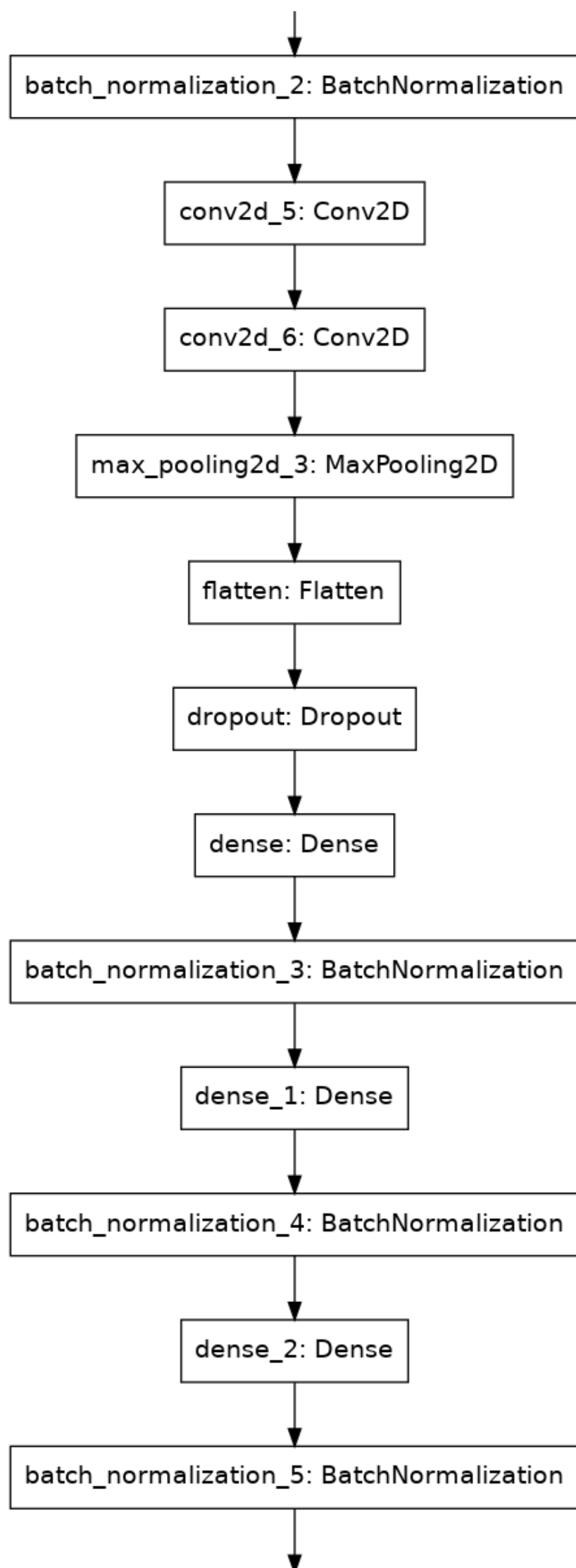
plot model to see connected layers

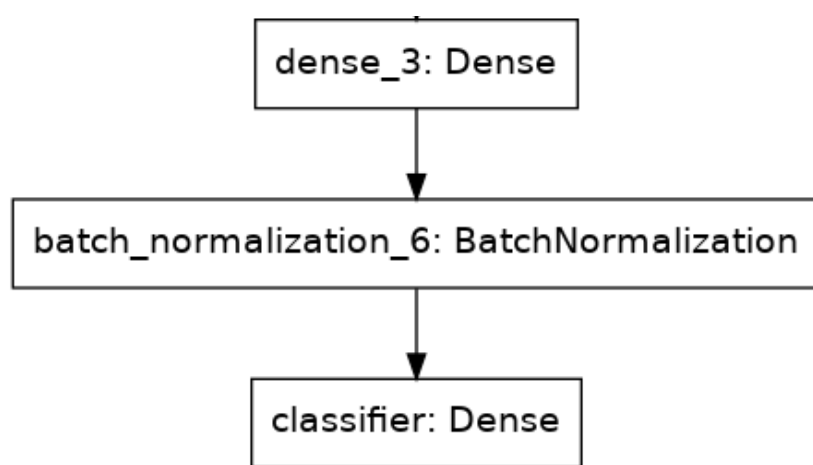
In [20]:

```
keras.utils.plot_model(model)
```

Out[20]:







In [21]:

```
model.compile(optimizer='adam' , loss = keras.losses.CategoricalCrossentropy() ,metrics
= ['accuracy'])
```

### Training step

In [22]:

```
history = model.fit(X_train ,
                    y_train ,
                    epochs=25 ,
                    batch_size=128,
                    validation_data=(X_test , y_test) ,
                    callbacks=[learning_rate_reduction])
```

```
Epoch 1/25
276/276 [=====] - 8s 15ms/step - loss: 1.4456 - accuracy: 0.4618
- val_loss: 1.0054 - val_accuracy: 0.6242
Epoch 2/25
276/276 [=====] - 3s 11ms/step - loss: 0.5232 - accuracy: 0.8120
- val_loss: 0.6355 - val_accuracy: 0.7733
Epoch 3/25
276/276 [=====] - 3s 11ms/step - loss: 0.3069 - accuracy: 0.8873
- val_loss: 0.4428 - val_accuracy: 0.8424
Epoch 4/25
276/276 [=====] - 3s 11ms/step - loss: 0.2065 - accuracy: 0.9275
- val_loss: 0.4280 - val_accuracy: 0.8610
Epoch 5/25
276/276 [=====] - 3s 11ms/step - loss: 0.1452 - accuracy: 0.9488
- val_loss: 0.4402 - val_accuracy: 0.8659
Epoch 6/25
276/276 [=====] - 3s 11ms/step - loss: 0.1322 - accuracy: 0.9533
- val_loss: 0.4819 - val_accuracy: 0.8365
Epoch 7/25
276/276 [=====] - 3s 11ms/step - loss: 0.0952 - accuracy: 0.9670
- val_loss: 0.6498 - val_accuracy: 0.8238

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 8/25
276/276 [=====] - 3s 12ms/step - loss: 0.0440 - accuracy: 0.9853
- val_loss: 0.0586 - val_accuracy: 0.9830
Epoch 9/25
276/276 [=====] - 3s 12ms/step - loss: 0.0183 - accuracy: 0.9947
- val_loss: 0.0738 - val_accuracy: 0.9807
Epoch 10/25
276/276 [=====] - 3s 11ms/step - loss: 0.0139 - accuracy: 0.9958
- val_loss: 0.1272 - val_accuracy: 0.9579

Epoch 00010: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 11/25
276/276 [=====] - 3s 12ms/step - loss: 0.0128 - accuracy: 0.9962
- val_loss: 0.0515 - val_accuracy: 0.9875
Epoch 12/25
276/276 [=====] - 3s 11ms/step - loss: 0.0037 - accuracy: 0.9991
```

```

- val_loss: 0.0527 - val_accuracy: 0.9872
Epoch 13/25
276/276 [=====] - 3s 11ms/step - loss: 0.0038 - accuracy: 0.9991
- val_loss: 0.0697 - val_accuracy: 0.9845

Epoch 00013: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 14/25
276/276 [=====] - 3s 11ms/step - loss: 0.0021 - accuracy: 0.9997
- val_loss: 0.0523 - val_accuracy: 0.9883
Epoch 15/25
276/276 [=====] - 3s 11ms/step - loss: 0.0020 - accuracy: 0.9998
- val_loss: 0.0504 - val_accuracy: 0.9892
Epoch 16/25
276/276 [=====] - 3s 11ms/step - loss: 0.0011 - accuracy: 0.9998
- val_loss: 0.0618 - val_accuracy: 0.9866
Epoch 17/25
276/276 [=====] - 3s 11ms/step - loss: 0.0013 - accuracy: 0.9998
- val_loss: 0.0521 - val_accuracy: 0.9889

Epoch 00017: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 18/25
276/276 [=====] - 3s 12ms/step - loss: 6.9502e-04 - accuracy: 1.
0000 - val_loss: 0.0519 - val_accuracy: 0.9891
Epoch 19/25
276/276 [=====] - 3s 12ms/step - loss: 6.8005e-04 - accuracy: 0.
9999 - val_loss: 0.0567 - val_accuracy: 0.9880

Epoch 00019: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 20/25
276/276 [=====] - 3s 11ms/step - loss: 6.3631e-04 - accuracy: 0.
9999 - val_loss: 0.0535 - val_accuracy: 0.9889
Epoch 21/25
276/276 [=====] - 3s 12ms/step - loss: 6.1578e-04 - accuracy: 0.
9999 - val_loss: 0.0572 - val_accuracy: 0.9884

Epoch 00021: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
Epoch 22/25
276/276 [=====] - 3s 11ms/step - loss: 6.8307e-04 - accuracy: 0.
9999 - val_loss: 0.0575 - val_accuracy: 0.9884
Epoch 23/25
276/276 [=====] - 3s 11ms/step - loss: 4.8331e-04 - accuracy: 1.
0000 - val_loss: 0.0590 - val_accuracy: 0.9882

Epoch 00023: ReduceLROnPlateau reducing learning rate to 1e-05.
Epoch 24/25
276/276 [=====] - 3s 11ms/step - loss: 4.7633e-04 - accuracy: 1.
0000 - val_loss: 0.0557 - val_accuracy: 0.9886
Epoch 25/25
276/276 [=====] - 3s 12ms/step - loss: 6.2609e-04 - accuracy: 0.
9999 - val_loss: 0.0569 - val_accuracy: 0.9885

```

In [23]:

```
model.evaluate(X_test , y_test)
```

```
367/367 [=====] - 1s 3ms/step - loss: 0.0569 - accuracy: 0.9885
```

Out[23]:

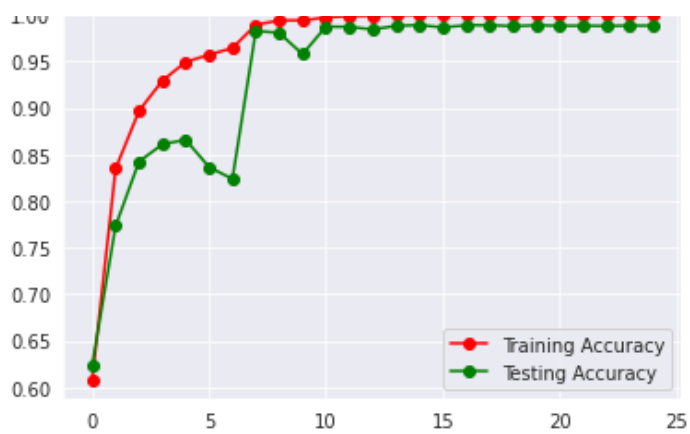
```
[0.05687971040606499, 0.9884949922561646]
```

## Analysis after Training

In [24]:

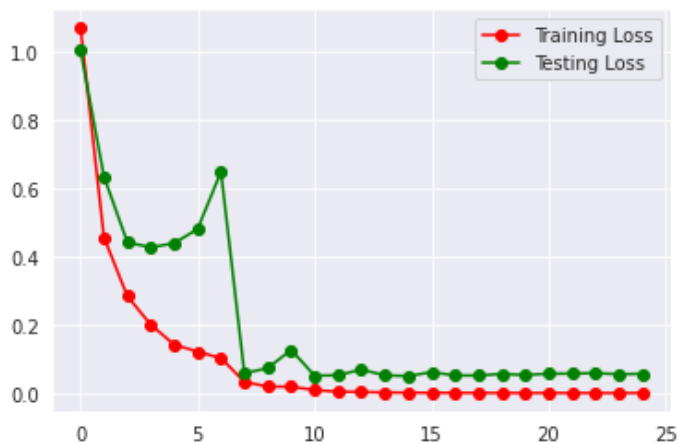
```
plt.plot(history.history["accuracy"] , 'ro-' , label = "Training Accuracy")
plt.plot(history.history["val_accuracy"] , 'go-' , label = "Testing Accuracy")
plt.legend()
plt.show()
```





In [25]:

```
plt.plot(history.history["loss"] , 'ro-' , label = "Training Loss")
plt.plot(history.history["val_loss"] , 'go-' , label = "Testing Loss")
plt.legend()
plt.show()
```



In [26]:

```
y_pred = model.predict(X_test).round()
```

In [27]:

```
target_names = [f"{classes[i]}" for i in range(7)]
print(classification_report(y_test , y_pred , target_names =target_names ))
```

score	support		precision	recall	f1-
1.00	1667	('akiec', 'Actinic keratoses and intraepithelial carcinomae')	1.00	1.00	1
1.00	1689	('bcc', ' basal cell carcinoma')	0.99	1.00	
0.98	1651	('bkl', 'benign keratosis-like lesions')	0.97	1.00	
1.00	1629	('df', 'dermatofibroma')	1.00	1.00	
0.96	1663	('nv', ' melanocytic nevi')	1.00	0.93	
1.00	1680	('vasc', ' pyogenic granulomas and hemorrhage')	1.00	1.00	
0.98	1755	('mel', 'melanoma')	0.97	1.00	
0.99	11734	micro avg	0.99	0.99	
0.99	11734	macro avg	0.99	0.99	
0.99	11734	weighted avg	0.99	0.99	
0.99	11734	samples avg	0.99	0.99	

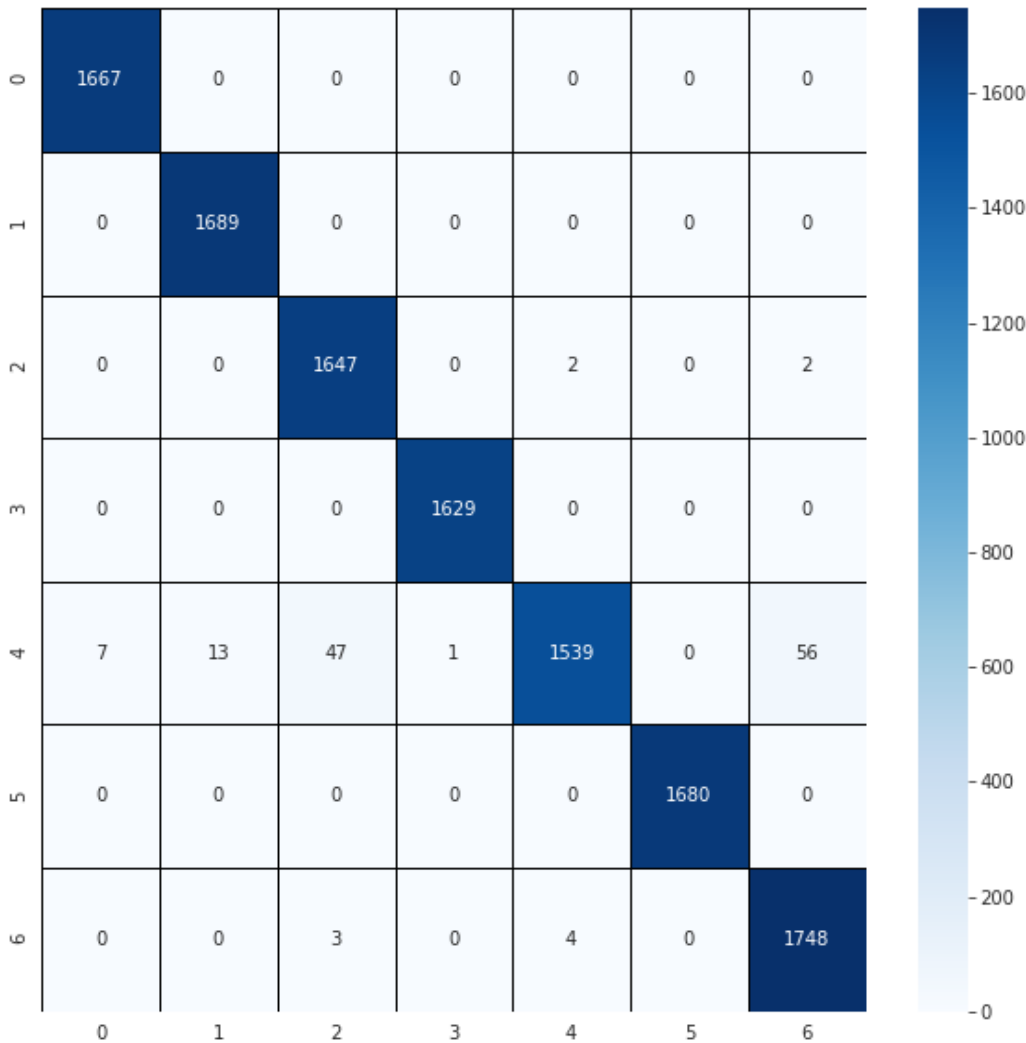
```
/opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1245: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in samples with
no predicted labels. Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
```

In [28]:

```
cm = confusion_matrix(y_test.argmax(axis = 1) , y_pred.argmax(axis = 1))
cm = pd.DataFrame(cm , index = [i for i in range(7)] , columns = [i for i in range(7)])
plt.figure(figsize = (10,10))
sns.heatmap(cm,cmap= "Blues", linecolor = 'black' , linewidth = 1 , annot = True, fmt='')
)
```

Out[28]:

<AxesSubplot:>

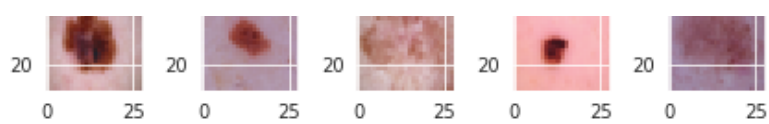


## Plot Test Images

In [29]:

```
correct = np.nonzero(y_pred == y_test)[0]
k = 0
for c in correct[:10]:
    plt.subplot(2,5,k+1)
    plt.imshow(X_test[k].reshape(28,28,3) , interpolation='none')
    plt.title(f"pred : {y_pred[k].argmax(axis = 0)},Actual :{y_test[k].argmax(ax
is = 0)} ")
    plt.tight_layout()
    k += 1
```

pred : 6,Actual:6 pred : 6,Actual:6 pred : 0,Actual:0 pred : 6,Actual:6 pred : 6,Actual:6



In [ ]: