In [2]:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from imblearn.over_sampling import RandomOverSampler
import numpy as np
from sklearn.model_selection import train_test_split
import os, cv2
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D, Activation
```

In [3]:

```
data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/hmnist_28_28_RGB.csv')
data.head()
```

Out[3]:

	pixel0000	pixel0001	pixel0002	pixel0003	pixel0004	pixel0005	pixel0006	pixel0007	pixel0008	pixel0009	...	pixel2343	pix
0	192	153	193	195	155	192	197	154	185	202	...	173	
1	25	14	30	68	48	75	123	93	126	158	...	60	
2	192	138	153	200	145	163	201	142	160	206	...	167	
3	38	19	30	95	59	72	143	103	119	171	...	44	
4	158	113	139	194	144	174	215	162	191	225	...	209	

5 rows × 2353 columns

In [4]:

```
data['label'].unique()
```

Out[4]:

```
array([2, 4, 3, 6, 5, 1, 0])
```

In [5]:

```
y = data['label']
x = data.drop(columns = ['label'])
```

In [6]:

```
data.isnull().sum().sum() #no null values present
```

Out[6]:

```
0
```

In [7]:

```
meta_data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')
meta_data.head()
```

Out[7]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp

	lesion_id	image_id	dx	dx_type	age	sex	localization
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
~~4~~	~~HAM_0001466~~	~~ISIC_0031633~~	~~bkl~~	~~histo~~	~~75.0~~	~~male~~	~~ear~~

In [8]:

```
meta_data['dx'].unique()
```

Out[8]:

```
array(['bkl', 'nv', 'df', 'mel', 'vasc', 'bcc', 'akiec'], dtype=object)
```

In [9]:

```
y = data['label']
x = data.drop(columns = ['label'])
```

In [10]:

```
data.isnull().sum().sum() #no null values present
```

Out[10]:

```
0
```

In [11]:

```
meta_data = pd.read_csv('/kaggle/input/skin-cancer-mnist-ham10000/HAM10000_metadata.csv')
meta_data.head()
```

Out[11]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear

In [12]:

```
meta_data['dx'].unique()
```

Out[12]:

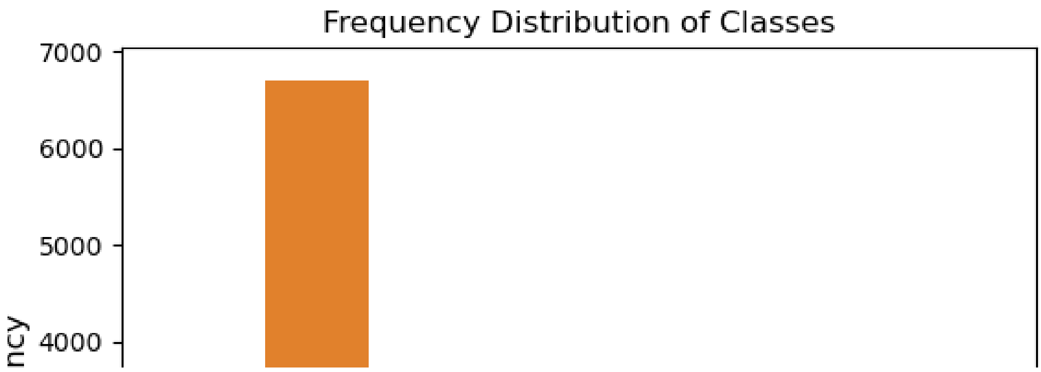
```
array(['bkl', 'nv', 'df', 'mel', 'vasc', 'bcc', 'akiec'], dtype=object)
```

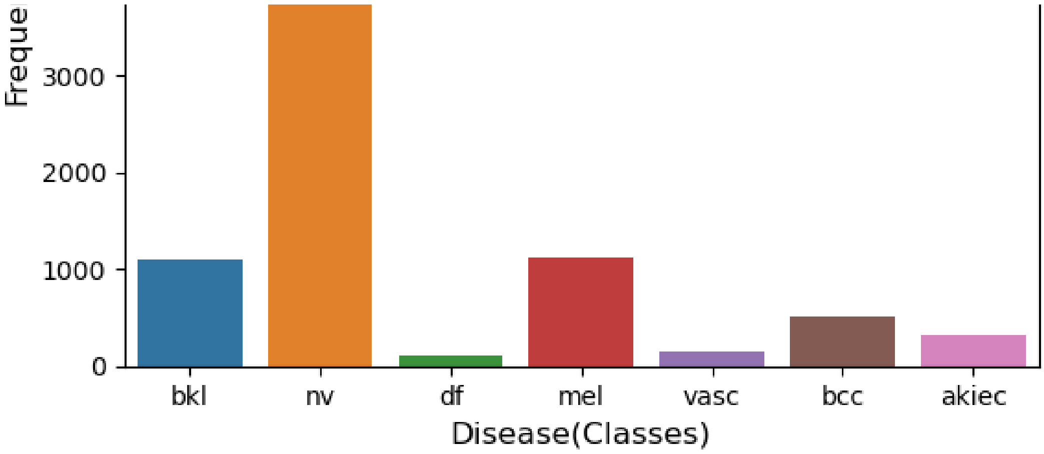
In [13]:

```
sns.countplot(x = 'dx', data = meta_data)
plt.xlabel('Disease(Classes)', size=12)
plt.ylabel('Frequency', size=12)
plt.title('Frequency Distribution of Classes')
```

Out[13]:

```
Text(0.5, 1.0, 'Frequency Distribution of Classes')
```





In [14]:

```
classes = {2:'bkl', 4:'nv', 3:'df', 6:'mel', 5:'vasc', 1:'bcc', 0:'akiec'}

classes_labels=[]
for key in classes.keys():
    classes_labels.append(key)
print(classes_labels)
```

```
[2, 4, 3, 6, 5, 1, 0]
```

In [15]:

```
print(x.shape,y.shape)
# To overcome class imbalace
oversample = RandomOverSampler()
x,y  = oversample.fit_resample(x,y)
print(x.shape,y.shape)
```

```
(10015, 2352) (10015,)
(46935, 2352) (46935,)
```

In [16]:

```
# reshaping the data so that it can be taken by convolution neural network(without distur
bing the no. of samples)
x = np.array(x).reshape(-1,28,28,3)
print('Shape of X :',x.shape)
print('Shape of y :',y.shape)
```

```
Shape of X : (46935, 28, 28, 3)
Shape of y : (46935,)
```

In [17]:

```
# Splitting Data
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=0.2, random_state=1)
print(X_train.shape,Y_train.shape)
print(X_test.shape , Y_test.shape)
```

```
(37548, 28, 28, 3) (37548,)
(9387, 28, 28, 3) (9387,)
```

In [18]:

```
model_CNN = Sequential()
model_CNN.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation = 'r
elu', padding = 'same'))
model_CNN.add(MaxPool2D(pool_size = (2,2)))

model_CNN.add(Conv2D(32, kernel_size = (3,3), activation = 'relu', padding = 'same'))
model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))

model_CNN.add(Conv2D(64, kernel_size = (3,3), activation = 'relu', padding = 'same'))
model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))
model_CNN.add(Conv2D(128, kernel_size = (3,3), activation = 'relu', padding = 'same'))
model_CNN.add(MaxPool2D(pool_size = (2,2), padding = 'same'))

model_CNN.add(Flatten())
model_CNN.add(Dense(64, activation = 'relu'))
model_CNN.add(Dense(32))
model_CNN.add(Activation(activation='relu'))
model_CNN.add(Dense(7))
model_CNN.add(Activation(activation='softmax'))

optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)

model_CNN.compile(loss = 'sparse_categorical_crossentropy',
                 optimizer = optimizer,
                  metrics = ['accuracy'])
print(model_CNN.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 16)        448       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 14, 14, 16)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 14, 14, 32)        4640      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 7, 7, 32)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 7, 7, 64)          18496     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 4, 4, 64)         0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 4, 4, 128)         73856     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 2, 2, 128)        0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 512)               0         
                                                                 
 dense (Dense)               (None, 64)                32832     
                                                                 
 dense_1 (Dense)             (None, 32)                2080      
                                                                 
 activation (Activation)     (None, 32)                0         
                                                                 
 dense_2 (Dense)             (None, 7)                 231       
                                                                 
 activation_1 (Activation)   (None, 7)                 0         
                                                                 
=================================================================
Total params: 132,583
Trainable params: 132,583
Non-trainable params: 0
_________________________________________________________________
None
```

In [19]:

```
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=10, verbose=1, mode='auto')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=3, verbose=1, mod
e='auto')
history = model_CNN.fit(X_train,
                    Y_train,
                    validation_split=0.2,
                    batch_size = 64,
                    epochs = 50,
                    callbacks = [reduce_lr, early_stop])
```

```
Epoch 1/50
470/470 [==============================] - 11s 7ms/step - loss: 1.5439 - accuracy: 0.4495
- val_loss: 0.9909 - val_accuracy: 0.6160 - lr: 0.0010
Epoch 2/50
470/470 [==============================] - 3s 7ms/step - loss: 0.8075 - accuracy: 0.6954
- val_loss: 0.6681 - val_accuracy: 0.7455 - lr: 0.0010
Epoch 3/50
470/470 [==============================] - 3s 6ms/step - loss: 0.5480 - accuracy: 0.7950
- val_loss: 0.4580 - val_accuracy: 0.8229 - lr: 0.0010
Epoch 4/50
470/470 [==============================] - 3s 7ms/step - loss: 0.3927 - accuracy: 0.8528
- val_loss: 0.4805 - val_accuracy: 0.8229 - lr: 0.0010
Epoch 5/50
470/470 [==============================] - 3s 6ms/step - loss: 0.3179 - accuracy: 0.8828
- val_loss: 0.3661 - val_accuracy: 0.8638 - lr: 0.0010
Epoch 6/50
470/470 [==============================] - 3s 6ms/step - loss: 0.2621 - accuracy: 0.9038
- val_loss: 0.2403 - val_accuracy: 0.9129 - lr: 0.0010
Epoch 7/50
470/470 [==============================] - 3s 6ms/step - loss: 0.2084 - accuracy: 0.9229
- val_loss: 0.2247 - val_accuracy: 0.9185 - lr: 0.0010
Epoch 8/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1949 - accuracy: 0.9288
- val_loss: 0.2296 - val_accuracy: 0.9190 - lr: 0.0010
Epoch 9/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1832 - accuracy: 0.9337
- val_loss: 0.2104 - val_accuracy: 0.9274 - lr: 0.0010
Epoch 10/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1653 - accuracy: 0.9412
- val_loss: 0.2971 - val_accuracy: 0.9113 - lr: 0.0010
Epoch 11/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1587 - accuracy: 0.9438
- val_loss: 0.1348 - val_accuracy: 0.9547 - lr: 0.0010
Epoch 12/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1409 - accuracy: 0.9498
- val_loss: 0.1849 - val_accuracy: 0.9337 - lr: 0.0010
Epoch 13/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1100 - accuracy: 0.9607
- val_loss: 0.1329 - val_accuracy: 0.9594 - lr: 0.0010
Epoch 14/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1066 - accuracy: 0.9621
- val_loss: 0.1350 - val_accuracy: 0.9553 - lr: 0.0010
Epoch 15/50
470/470 [==============================] - 3s 6ms/step - loss: 0.1117 - accuracy: 0.9614
- val_loss: 0.1987 - val_accuracy: 0.9325 - lr: 0.0010
Epoch 16/50
466/470 [============================>.] - ETA: 0s - loss: 0.1239 - accuracy: 0.9570
Epoch 16: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
470/470 [==============================] - 3s 6ms/step - loss: 0.1241 - accuracy: 0.9570
- val_loss: 0.1929 - val_accuracy: 0.9379 - lr: 0.0010
Epoch 17/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0295 - accuracy: 0.9912
- val_loss: 0.0916 - val_accuracy: 0.9752 - lr: 1.0000e-04
Epoch 18/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0144 - accuracy: 0.9967
- val_loss: 0.0867 - val_accuracy: 0.9778 - lr: 1.0000e-04
Epoch 19/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0095 - accuracy: 0.9983
- val_loss: 0.0876 - val_accuracy: 0.9784 - lr: 1.0000e-04
Epoch 20/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0066 - accuracy: 0.9991
- val_loss: 0.0889 - val_accuracy: 0.9784 - lr: 1.0000e-04
Epoch 21/50
466/470 [============================>.] - ETA: 0s - loss: 0.0046 - accuracy: 0.9997
Epoch 21: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
470/470 [==============================] - 3s 6ms/step - loss: 0.0046 - accuracy: 0.9997
- val_loss: 0.0931 - val_accuracy: 0.9790 - lr: 1.0000e-04
Epoch 22/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0032 - accuracy: 0.9999
- val_loss: 0.0920 - val_accuracy: 0.9794 - lr: 1.0000e-05
Epoch 23/50
```

```
470/470 [==============================] - 3s 6ms/step - loss: 0.0030 - accuracy: 0.9999
- val_loss: 0.0917 - val_accuracy: 0.9795 - lr: 1.0000e-05
Epoch 24/50
468/470 [============================>.] - ETA: 0s - loss: 0.0029 - accuracy: 0.9999
Epoch 24: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
470/470 [==============================] - 3s 7ms/step - loss: 0.0029 - accuracy: 0.9999
- val_loss: 0.0916 - val_accuracy: 0.9796 - lr: 1.0000e-05
Epoch 25/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0027 - accuracy: 0.9999
- val_loss: 0.0920 - val_accuracy: 0.9796 - lr: 1.0000e-06
Epoch 26/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0027 - accuracy: 0.9999
- val_loss: 0.0925 - val_accuracy: 0.9794 - lr: 1.0000e-06
Epoch 27/50
463/470 [============================>.] - ETA: 0s - loss: 0.0027 - accuracy: 0.9999
Epoch 27: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
470/470 [==============================] - 3s 6ms/step - loss: 0.0027 - accuracy: 0.9999
- val_loss: 0.0926 - val_accuracy: 0.9795 - lr: 1.0000e-06
Epoch 28/50
470/470 [==============================] - 3s 6ms/step - loss: 0.0026 - accuracy: 0.9999
- val_loss: 0.0926 - val_accuracy: 0.9794 - lr: 1.0000e-07
Epoch 28: early stopping
```

In [20]:

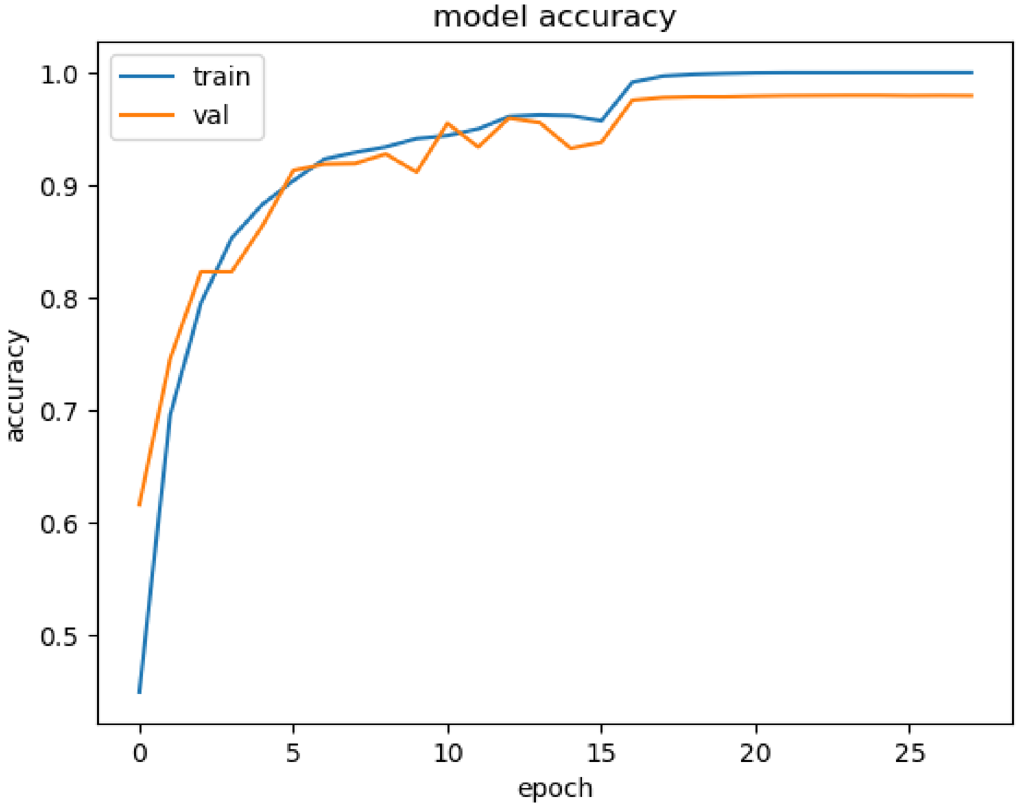
```
results = model_CNN.evaluate(X_test , Y_test, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 0.09621
Test Accuracy: 97.82%
```

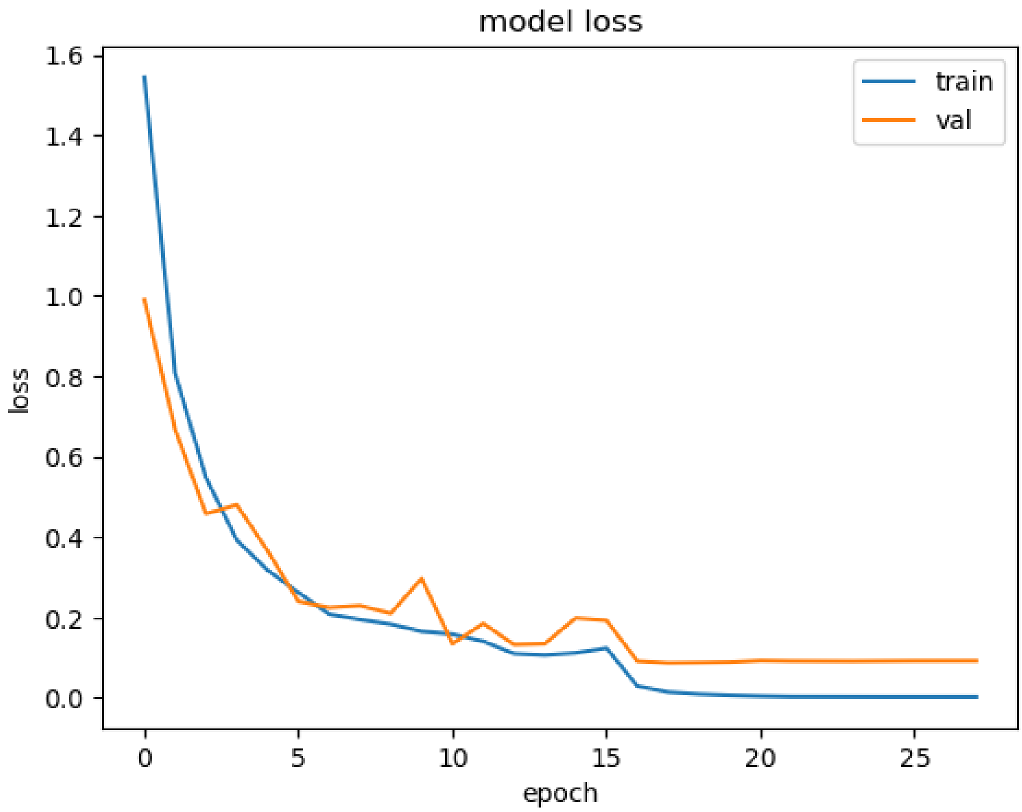
In [21]:

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



In [22]:

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



In [23]:

```
from sklearn.metrics import confusion_matrix , classification_report

y_true_CNN = list(Y_test)
y_pred_CNN = model_CNN.predict(X_test)
y_pred_CNN = list(map(lambda x: np.argmax(x), y_pred_CNN))
print('Y Actual Values :' , y_true_CNN[0:10])
print('Y Predicted Values :' , y_pred_CNN[0:10])
```

```
294/294 [==============================] - 1s 2ms/step
Y Actual Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
Y Predicted Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
```

In [24]:

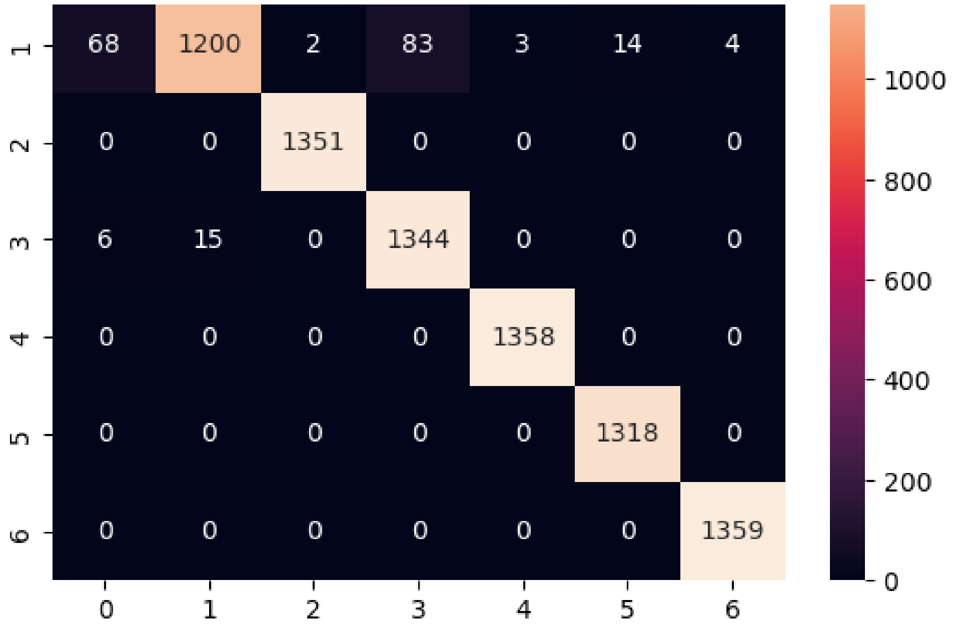
```
cm_CNN = confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels)
print(confusion_matrix(y_true_CNN,y_pred_CNN,labels=classes_labels))
sns.heatmap(cm_CNN, annot = True, fmt='')
```

```
[[1252    1    0    1    1    4    3]
 [  68 1200    2   83    3   14    4]
 [   0    0 1351    0    0    0    0]
 [   6   15    0 1344    0    0    0]
 [   0    0    0    0 1358    0    0]
 [   0    0    0    0    0 1318    0]
 [   0    0    0    0    0    0 1359]]
```

Out[24]:

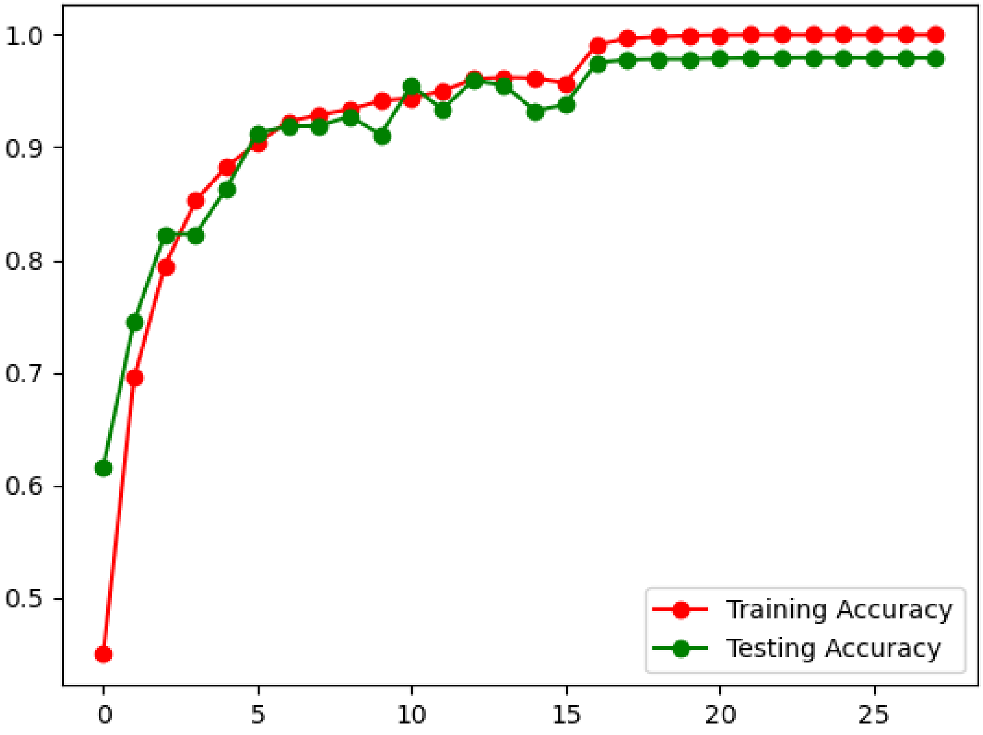
```
<AxesSubplot:>
```

0
1252
1
0
1
1
4
3
1200



In [25]:

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



In [26]:

```
#predicting
y_pred_CNN  = model_CNN.predict(X_test)
target_names = [f"{classes[i]}" for i in range(7)]
print(len(Y_test) ,"  ",len(y_pred_CNN))
y_pred_CNN = list(map(lambda x: np.argmax(x), y_pred_CNN))
print(classification_report(Y_test , y_pred_CNN,target_names=target_names))
```

```
294/294 [==============================] - 1s 2ms/step
9387    9387
              precision    recall  f1-score   support

       akiec       0.99      1.00      1.00      1359
         bcc       0.99      1.00      0.99      1318
         bkl       0.94      0.99      0.97      1262
          df       1.00      1.00      1.00      1351
          nv       0.99      0.87      0.93      1374
        vasc       1.00      1.00      1.00      1358
         mel       0.94      0.98      0.96      1365

    accuracy                           0.98      9387
   macro avg       0.98      0.98      0.98      9387
weighted avg       0.98      0.98      0.98      9387
```

In [27]:

```
# Layers definitions
from keras import backend as K
for l in range(len(model_CNN.layers)):
    print(l, model_CNN.layers[l])
```

```
0 <keras.layers.convolutional.conv2d.Conv2D object at 0x7fd095ac7a90>
1 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7fd0ffd3d5d0>
2 <keras.layers.convolutional.conv2d.Conv2D object at 0x7fd0ffd3d4d0>
3 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7fd091448210>
4 <keras.layers.convolutional.conv2d.Conv2D object at 0x7fd0914608d0>
5 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7fd0ffbd2190>
6 <keras.layers.convolutional.conv2d.Conv2D object at 0x7fd091457790>
7 <keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x7fd09144dd10>
8 <keras.layers.reshaping.flatten.Flatten object at 0x7fd09142c210>
9 <keras.layers.core.dense.Dense object at 0x7fd0ffab7850>
10 <keras.layers.core.dense.Dense object at 0x7fd0ffafae10>
11 <keras.layers.core.activation.Activation object at 0x7fd09144d790>
12 <keras.layers.core.dense.Dense object at 0x7fd091448890>
13 <keras.layers.core.activation.Activation object at 0x7fd0ffb3e590>
```

In [28]:

```
import os
os.environ["KERAS_BACKEND"] = "tensorflow"
kerasBKED = os.environ["KERAS_BACKEND"]
print(kerasBKED)
```

```
tensorflow
```

In [29]:

```
import tensorflow as tf
feature_extractor = tf.keras.Model(inputs=model_CNN.input,
                                    outputs=model_CNN.get_layer('activation').output)
output_layers_model =tf.keras.Model(inputs=model_CNN.input, outputs=model_CNN.output)
```

In [30]:

```
# Extract features from input data using the CNN model
X_train_cnn = model_CNN.predict(X_train)
X_test_cnn = model_CNN.predict(X_test)
```

```
1174/1174 [==============================] - 3s 2ms/step
294/294 [==============================] - 1s 2ms/step
```

In [38]:

```
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

parameters = {'kernel':['rbf'],
              'C':[1, 10, 100, 1000],
              'gamma':[1e-3, 1e-4]}
clf = GridSearchCV(SVC(), parameters)
clf.fit(X_train_cnn, Y_train)
# Evaluate the combined CNN-SVM model on a test dataset
svm_accuracy = clf.score(X_test_cnn, Y_test)
print('SVM Accuracy:', svm_accuracy*100)
```

```
SVM Accuracy: 98.13571961222968
SVM Accuracy: 98.13571961222968
```

In [66]:

```
svm_accuracy = clf.score(X_test_cnn, Y_test)
print('SVM Accuracy:', svm_accuracy*100)
svmclf = clf.best_estimator_
print(svmclf)
svmclf.fit(X_train_cnn, Y_train)
```

```
SVM Accuracy: 98.13571961222968
SVC(C=1000, gamma=0.001)
```

Out[66]:

```
SVC(C=1000, gamma=0.001)
```

In [68]:

```
y_testSVM = svmclf.predict(X_test_cnn)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

print(classification_report(Y_test, y_testSVM))
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testSVM)*100))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1359
           1       0.99      1.00      0.99      1318
           2       0.95      0.99      0.97      1262
           3       1.00      1.00      1.00      1351
           4       0.99      0.90      0.94      1374
           5       1.00      1.00      1.00      1358
           6       0.95      0.98      0.97      1365

    accuracy                           0.98      9387
   macro avg       0.98      0.98      0.98      9387
weighted avg       0.98      0.98      0.98      9387

Accuracy: 98.13571961222968
```

In [64]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

parameters = {"max_depth": [3],
              "max_features": [1],
              "min_samples_split": [3],
              "min_samples_leaf": [3],
              "bootstrap": [True],
              "criterion": ["entropy"],
              "n_estimators": [10]}
rclf = RandomForestClassifier()
rgclf = GridSearchCV(rclf, param_grid=parameters)
rgclf.fit(X_test_cnn, Y_test)
RFC_accuracy = clf.score(X_test_cnn, Y_test)
print('Random Forest Classifier Accuracy:', RFC_accuracy*100)
```

```
Random Forest Classifier Accuracy: 98.13571961222968
```

In [71]:

```
RFclf = rgclf.best_estimator_
RFclf.fit(X_test_cnn, Y_test)
print(RFclf)
y_testRFC = RFclf.predict(X_test_cnn)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

print(classification_report(Y_test, y_testRFC))
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testRFC)*100))
```

```
RandomForestClassifier(criterion='entropy', max_depth=3, max_features=1,
                     min_samples_leaf=3, min_samples_split=3,
                     n_estimators=10)
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1359
           1       0.99      1.00      1.00      1318
           2       0.97      0.99      0.98      1262
           3       1.00      1.00      1.00      1351
           4       0.97      0.93      0.95      1374
           5       1.00      1.00      1.00      1358
           6       0.97      0.98      0.97      1365

    accuracy                           0.99      9387
   macro avg       0.99      0.99      0.99      9387
weighted avg       0.99      0.99      0.99      9387

Accuracy: 98.51922872057101
```

In [72]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

parameters = {"n_neighbors": [1, 5, 10, 30],
              "weights": ['uniform', 'distance'],
              "metric": ['minkowski','euclidean','manhattan'],
              "algorithm": ['auto', 'ball_tree', 'kd_tree', 'brute']}
kclf = KNeighborsClassifier()
kgclf = GridSearchCV(kclf, param_grid=parameters)
kgclf.fit(X_test_cnn, Y_test)
KNN_accuracy = kgclf.score(X_test_cnn, Y_test)
print('KNN Classifier Accuracy:', KNN_accuracy*100)
```

```
KNN Classifier Accuracy: 100.0
```

In [73]:

```
y_testKNN = kgclf.predict(X_test_cnn)
KNNclf = kgclf.best_estimator_
print(KNNclf)
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

print(classification_report(Y_test, y_testKNN))
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testKNN)*100))
```

```
KNeighborsClassifier(n_neighbors=30, weights='distance')
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1359
           1       1.00      1.00      1.00      1318
           2       1.00      1.00      1.00      1262
           3       1.00      1.00      1.00      1351
           4       1.00      1.00      1.00      1374
           5       1.00      1.00      1.00      1358
           6       1.00      1.00      1.00      1365

    accuracy                           1.00      9387
   macro avg       1.00      1.00      1.00      9387
weighted avg       1.00      1.00      1.00      9387

Accuracy: 100.0
```

In [77]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```python
# Create a logistic regression object
lr = LogisticRegression()

# Define the hyperparameter grid to search over
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}

# Perform grid search with 5-fold cross-validation
grid_search_LR = GridSearchCV(lr, param_grid, cv=5)
grid_search_LR.fit(X_test_cnn, Y_test)

# Print the best hyperparameters and the corresponding accuracy score
print("Best hyperparameters: ", grid_search_LR.best_params_)
y_testKNN = grid_search_LR.predict(X_test_cnn)

print(classification_report(Y_test, y_testKNN))
print("Accuracy: {0}".format(accuracy_score(Y_test, y_testKNN)*100))
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
```

```
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_validation.py:372: FitFai
ledWarning:
30 fits failed out of a total of 60.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='rai
se'.

Below are more details about the failures:
--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_validation.py", l
ine 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py", line 1
461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py", line 4
49, in _check_solver
    % (solver, penalty)
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/opt/conda/lib/python3.7/site-packages/sklearn/model_selection/_search.py:972: UserWarnin
g: One or more of the test scores are non-finite: [       nan 0.97837456        nan 0.978
80066        nan 0.98007913
        nan 0.98114426        nan 0.98189002        nan 0.98220956]
  category=UserWarning,
```

```
Best hyperparameters:  {'C': 100, 'penalty': 'l2'}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1359
           1       0.99      1.00      0.99      1318
           2       0.96      0.99      0.97      1262
           3       1.00      1.00      1.00      1351
           4       0.98      0.91      0.94      1374
           5       1.00      1.00      1.00      1358
           5       1.00      1.00      1.00      1358
           6       0.95      0.98      0.97      1365

    accuracy                           0.98      9387
   macro avg       0.98      0.98      0.98      9387
weighted avg       0.98      0.98      0.98      9387

Accuracy: 98.25290295088953
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:818: Convergence
Warning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

In [ ]: