In [167]:

```
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
```

In [168]:

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

Out[168]:

| | 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 [169]:

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

Out[169]:

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

In [170]:

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

In [171]:

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

Out[171]:

```
0
```

In [172]:

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

Out[172]:

| | 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 [173]:

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

Out[173]:

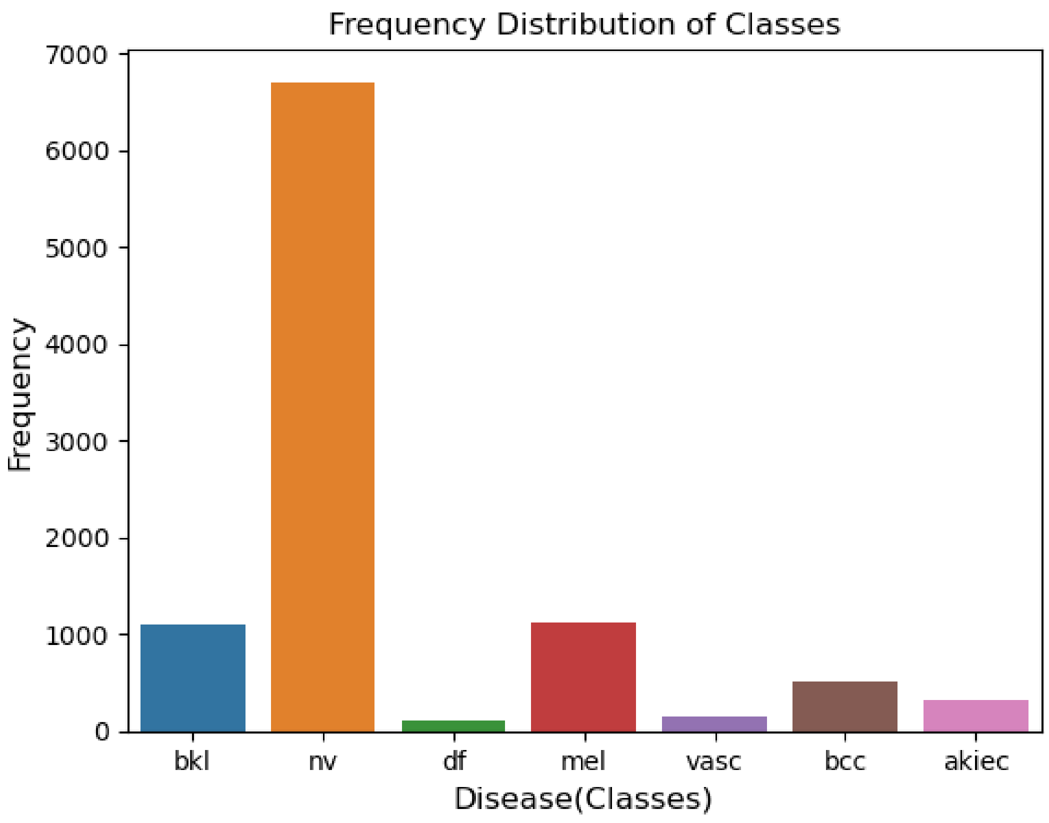
```
array(['bkl', 'nv', 'df', 'mel', 'vasc', 'bcc', 'akiec'], dtype=object)
```

In [174]:

```
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[174]:

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



In [175]:

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

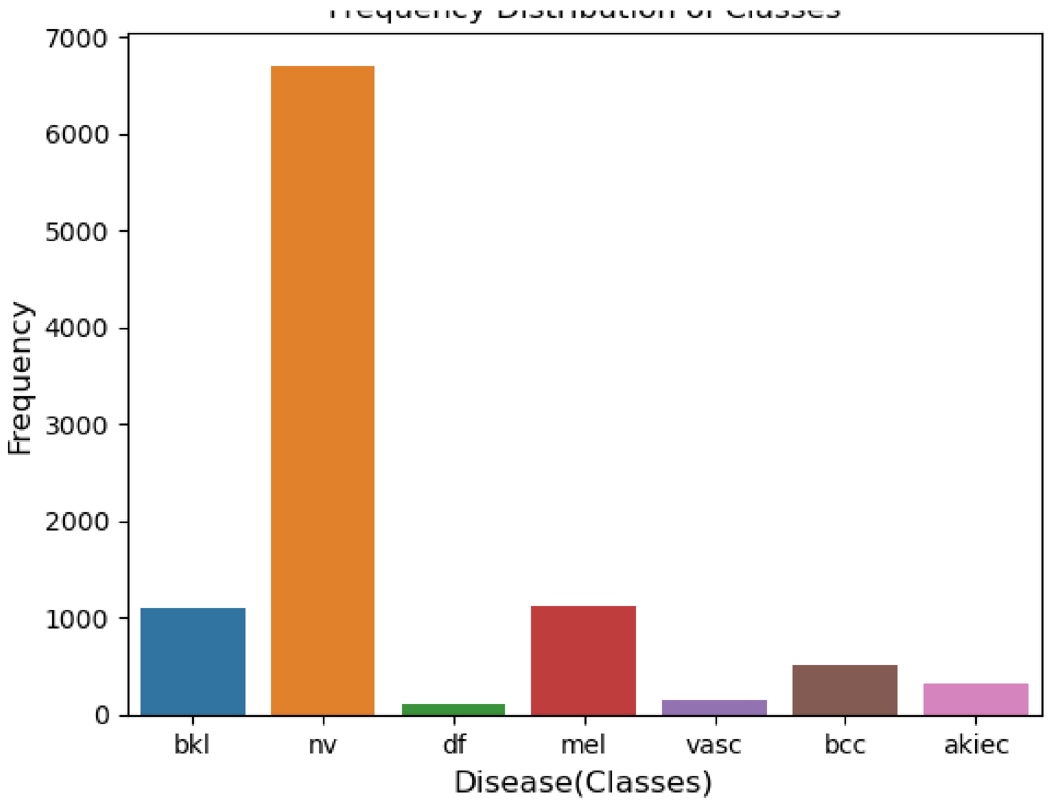
```
(10015, 2352) (10015,)
(46935, 2352) (46935,)
```

In [176]:

```
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[176]:

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



In [ ]:

In [177]:

```
# 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 [178]:

```
# 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 [179]:

```
model = Sequential()

model.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation = 'relu'
))
model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu'))
model.add(Conv2D(64, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2)))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(7, activation='softmax'))
model.summary()
```

```
Model: "sequential_3"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_12 (Conv2D)          (None, 26, 26, 16)        448

 conv2d_13 (Conv2D)          (None, 24, 24, 32)        4640

 max_pooling2d_8 (MaxPooling  (None, 12, 12, 32)       0
 2D)

 conv2d_14 (Conv2D)          (None, 10, 10, 32)        9248

 conv2d_15 (Conv2D)          (None, 8, 8, 64)          18496

 max_pooling2d_9 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 flatten_3 (Flatten)         (None, 1024)              0

 dense_13 (Dense)            (None, 64)                65600

 dense_14 (Dense)            (None, 7)                 455

=================================================================
Total params: 98,887
Trainable params: 98,887
Non-trainable params: 0
_________________________________________________________________
```

In [180]:

```
model.compile(loss = 'sparse_categorical_crossentropy',
             optimizer = 'adam',
              metrics = ['accuracy'])
history = model.fit(X_train,
                    Y_train,
                    validation_split=0.2,
                    batch_size = 128,
                    epochs = 50)
```

```
Epoch 1/50

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipykernel_27/1092788544.py in <module>
      6                     validation_split=0.2,
      7                     batch_size = 128,
----> 8                     epochs = 50)

/opt/conda/lib/python3.7/site-packages/keras/utils/traceback_utils.py in error_handler(*a
rgs, **kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

/opt/conda/lib/python3.7/site-packages/keras/engine/training.py in fit(self, x, y, batch_
size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weigh
t, sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size
, validation_freq, max_queue_size, workers, use_multiprocessing)
   1648                         ):
   1649                             callbacks.on_train_batch_begin(step)
-> 1650                             tmp_logs = self.train_function(iterator)
   1651                             if data_handler.should_sync:
   1652                                 context.async_wait()

/opt/conda/lib/python3.7/site-packages/tensorflow/python/util/traceback_utils.py in error
_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
```

```
    152         filtered_tb = _process_traceback_frames(e.__traceback__)

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/polym
orphic_function.py in __call__(self, *args, **kwds)
    878 
    879         with OptionalXlaContext(self._jit_compile):
--> 880           result = self._call(*args, **kwds)
    881 
    882         new_tracing_count = self.experimental_get_tracing_count()

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/polym
orphic_function.py in _call(self, *args, **kwds)
    926         # This is the first call of __call__, so we have to initialize.
    927         initializers = []
--> 928         self._initialize(args, kwds, add_initializers_to=initializers)
    929       finally:
    930         # At this point we know that the initialization is complete (or less

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/polym
orphic_function.py in _initialize(self, args, kwds, add_initializers_to)
    749           self._variable_creation_fn    # pylint: disable=protected-access
    750           ._get_concrete_function_internal_garbage_collected(
--> 751               *args, **kwds))
    752 
    753     def invalid_creator_scope(*unused_args, **unused_kwds):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/traci
ng_compiler.py in _get_concrete_function_internal_garbage_collected(self, *args, **kwargs
)
    160       """Returns a concrete function which cleans up its graph function."""
    161       with self._lock:
--> 162         concrete_function, _ = self._maybe_define_concrete_function(args, kwargs)
    163       return concrete_function
    164 

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/traci
ng_compiler.py in _maybe_define_concrete_function(self, args, kwargs)
    155         kwargs = {}
    156 
--> 157       return self._maybe_define_function(args, kwargs)
    158 
    159     def _get_concrete_function_internal_garbage_collected(self, *args, **kwargs):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/traci
ng_compiler.py in _maybe_define_function(self, args, kwargs)
    358             args, kwargs = generalized_func_key._placeholder_value()  # pylint:
disable=protected-access
    359 
--> 360           concrete_function = self._create_concrete_function(args, kwargs)
    361 
    362           graph_capture_container = concrete_function.graph._capture_func_lib  #
pylint: disable=protected-access

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/traci
ng_compiler.py in _create_concrete_function(self, args, kwargs)
    291             autograph_options=self._autograph_options,
    292             arg_names=arg_names,
--> 293             capture_by_value=self._capture_by_value),
    294           self._function_attributes,
    295           spec=self.function_spec,

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/func_graph.py in func_
graph_from_py_func(name, python_func, args, kwargs, signature, func_graph, autograph, aut
ograph_options, add_control_dependencies, arg_names, op_return_value, collections, captur
e_by_value, acd_record_initial_resource_uses)
   1281           _, original_func = tf_decorator.unwrap(python_func)
   1282 
-> 1283         func_outputs = python_func(*func_args, **func_kwargs)
   1284 
   1285         # invariant: `func_outputs` contains only Tensors, CompositeTensors,

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/polymorphic_function/polym
```

```
orphic_function.py in wrapped_fn(*args, **kwds)
    643         # the function a weak reference to itself to avoid a reference cycle.
    644         with OptionalXlaContext(compile_with_xla):
--> 645           out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    646         return out
    647 

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/func_graph.py in autog
raph_handler(*args, **kwargs)
   1263                   recursive=True,
   1264                   optional_features=autograph_options,
-> 1265                   user_requested=True,
   1266               ))
   1267           except Exception as e:  # pylint:disable=broad-except

/opt/conda/lib/python3.7/site-packages/tensorflow/python/autograph/impl/api.py in convert
ed_call(f, args, kwargs, caller_fn_scope, options)
    437     try:
    438       if kwargs is not None:
--> 439         result = converted_f(*effective_args, **kwargs)
    440       else:
    441         result = converted_f(*effective_args)

/opt/conda/lib/python3.7/site-packages/keras/engine/training.py in tf__train_function(ite
rator)
     13                 try:
     14                     do_return = True
---> 15                     retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.
ld(self), ag__.ld(iterator)), None, fscope)
     16                 except:
     17                     do_return = False

/opt/conda/lib/python3.7/site-packages/tensorflow/python/autograph/impl/api.py in convert
ed_call(f, args, kwargs, caller_fn_scope, options)
    375 
    376   if not options.user_requested and conversion.is_allowlisted(f):
--> 377     return _call_unconverted(f, args, kwargs, options)
    378 
    379   # internal_convert_user_code is for example turned off when issuing a dynamic

/opt/conda/lib/python3.7/site-packages/tensorflow/python/autograph/impl/api.py in _call_u
nconverted(f, args, kwargs, options, update_cache)
    457   if kwargs is not None:
    458     return f(*args, **kwargs)
--> 459   return f(*args)
    460 
    461 

/opt/conda/lib/python3.7/site-packages/keras/engine/training.py in step_function(model, i
terator)
   1231               )
   1232           data = next(iterator)
-> 1233           outputs = model.distribute_strategy.run(run_step, args=(data,))
   1234           outputs = reduce_per_replica(
   1235               outputs,

/opt/conda/lib/python3.7/site-packages/tensorflow/python/distribute/distribute_lib.py in
run(***failed resolving arguments***)
   1314       fn = autograph.tf_convert(
   1315           fn, autograph_ctx.control_status_ctx(), convert_by_default=False)
-> 1316       return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
   1317 
   1318   def reduce(self, reduce_op, value, axis):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/distribute/distribute_lib.py in
call_for_each_replica(self, fn, args, kwargs)
   2893       kwargs = {}
   2894     with self._container_strategy().scope():
-> 2895       return self._call_for_each_replica(fn, args, kwargs)
   2896 
   2897   def _call_for_each_replica(self, fn, args, kwargs):
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/distribute/distribute_lib.py in
_call_for_each_replica(self, fn, args, kwargs)
   3694   def _call_for_each_replica(self, fn, args, kwargs):
   3695     with ReplicaContext(self._container_strategy(), replica_id_in_sync_group=0):
-> 3696       return fn(*args, **kwargs)
   3697
   3698   def _reduce_to(self, reduce_op, value, destinations, options):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/autograph/impl/api.py in wrapper
(*args, **kwargs)
    687       try:
    688         with conversion_ctx:
--> 689           return converted_call(f, args, kwargs, options=options)
    690       except Exception as e:  # pylint:disable=broad-except
    691         if hasattr(e, 'ag_error_metadata'):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/autograph/impl/api.py in convert
ed_call(f, args, kwargs, caller_fn_scope, options)
    375
    376   if not options.user_requested and conversion.is_allowlisted(f):
--> 377     return _call_unconverted(f, args, kwargs, options)
    378
    379   # internal_convert_user_code is for example turned off when issuing a dynamic

/opt/conda/lib/python3.7/site-packages/tensorflow/python/autograph/impl/api.py in _call_u
nconverted(f, args, kwargs, options, update_cache)
    456
    457   if kwargs is not None:
--> 458     return f(*args, **kwargs)
    459   return f(*args)
    460

/opt/conda/lib/python3.7/site-packages/keras/engine/training.py in run_step(data)
   1220
   1221             def run_step(data):
-> 1222                 outputs = model.train_step(data)
   1223                 # Ensure counter is updated only if `train_step` succeeds.
   1224                 with tf.control_dependencies(_minimum_control_deps(outputs)):

/opt/conda/lib/python3.7/site-packages/keras/engine/training.py in train_step(self, data)
   1021         # Run forward pass.
   1022         with tf.GradientTape() as tape:
-> 1023             y_pred = self(x, training=True)
   1024             loss = self.compute_loss(x, y, y_pred, sample_weight)
   1025         self._validate_target_and_loss(y, loss)

/opt/conda/lib/python3.7/site-packages/keras/utils/traceback_utils.py in error_handler(*a
rgs, **kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

/opt/conda/lib/python3.7/site-packages/keras/engine/training.py in __call__(self, *args,
**kwargs)
    559             layout_map_lib._map_subclass_model_variable(self, self._layout_map)
    560
--> 561         return super().__call__(*args, **kwargs)
    562
    563     @doc_controls.doc_in_current_and_subclasses

/opt/conda/lib/python3.7/site-packages/keras/utils/traceback_utils.py in error_handler(*a
rgs, **kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

/opt/conda/lib/python3.7/site-packages/keras/engine/base_layer.py in __call__(self, *args
, **kwargs)
```

```
   1130                         self._compute_dtype_object
   1131                     ):
-> 1132                         outputs = call_fn(inputs, *args, **kwargs)
   1133
   1134                     if self._activity_regularizer:

/opt/conda/lib/python3.7/site-packages/keras/utils/traceback_utils.py in error_handler(*a
rgs, **kwargs)
     94         bound_signature = None
     95         try:
---> 96             return fn(*args, **kwargs)
     97         except Exception as e:
     98             if hasattr(e, "_keras_call_info_injected"):

/opt/conda/lib/python3.7/site-packages/keras/engine/sequential.py in call(self, inputs, t
raining, mask)
    411             if not self.built:
    412                 self._init_graph_network(self.inputs, self.outputs)
--> 413             return super().call(inputs, training=training, mask=mask)
    414
    415         outputs = inputs  # handle the corner case where self.layers is empty

/opt/conda/lib/python3.7/site-packages/keras/engine/functional.py in call(self, inputs, t
raining, mask)
    509             a list of tensors if there are more than one outputs.
    510         """
--> 511         return self._run_internal_graph(inputs, training=training, mask=mask)
    512
    513     def compute_output_shape(self, input_shape):

/opt/conda/lib/python3.7/site-packages/keras/engine/functional.py in _run_internal_graph(
self, inputs, training, mask)
    666
    667                 args, kwargs = node.map_arguments(tensor_dict)
--> 668                 outputs = node.layer(*args, **kwargs)
    669
    670                 # Update tensor_dict.

/opt/conda/lib/python3.7/site-packages/keras/utils/traceback_utils.py in error_handler(*a
rgs, **kwargs)
     63         filtered_tb = None
     64         try:
---> 65             return fn(*args, **kwargs)
     66         except Exception as e:
     67             filtered_tb = _process_traceback_frames(e.__traceback__)

/opt/conda/lib/python3.7/site-packages/keras/engine/base_layer.py in __call__(self, *args
, **kwargs)
   1130                         self._compute_dtype_object
   1131                     ):
-> 1132                         outputs = call_fn(inputs, *args, **kwargs)
   1133
   1134                     if self._activity_regularizer:

/opt/conda/lib/python3.7/site-packages/keras/utils/traceback_utils.py in error_handler(*a
rgs, **kwargs)
     94         bound_signature = None
     95         try:
---> 96             return fn(*args, **kwargs)
     97         except Exception as e:
     98             if hasattr(e, "_keras_call_info_injected"):

/opt/conda/lib/python3.7/site-packages/keras/layers/core/dense.py in call(self, inputs)
    239                 )
    240             else:
--> 241                 outputs = tf.matmul(a=inputs, b=self.kernel)
    242         # Broadcast kernel to inputs.
    243         else:

/opt/conda/lib/python3.7/site-packages/tensorflow/python/util/traceback_utils.py in error
_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

/opt/conda/lib/python3.7/site-packages/tensorflow/python/util/dispatch.py in op_dispatch_
handler(*args, **kwargs)
   1174       # Fallback dispatch system (dispatch v1):
   1175       try:
-> 1176         return dispatch_target(*args, **kwargs)
   1177       except (TypeError, ValueError):
   1178         # Note: convert_to_eager_tensor currently raises a ValueError, not a

/opt/conda/lib/python3.7/site-packages/tensorflow/python/ops/math_ops.py in matmul(a, b,
transpose_a, transpose_b, adjoint_a, adjoint_b, a_is_sparse, b_is_sparse, output_type, na
me)
   3634     else:
   3635       a = ops.convert_to_tensor(a, name="a")
-> 3636       b = ops.convert_to_tensor(b, dtype_hint=a.dtype.base_dtype, name="b")
   3637 
   3638     # TODO(apassos) remove _shape_tuple here when it is not needed.

/opt/conda/lib/python3.7/site-packages/tensorflow/python/profiler/trace.py in wrapped(*ar
gs, **kwargs)
    181         with Trace(trace_name, **trace_kwargs):
    182           return func(*args, **kwargs)
--> 183       return func(*args, **kwargs)
    184 
    185     return wrapped

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/ops.py in convert_to_t
ensor(value, dtype, name, as_ref, preferred_dtype, dtype_hint, ctx, accepted_result_types
)
   1618       try:
   1619         ret = conversion_func(
-> 1620             value, dtype=preferred_dtype, name=name, as_ref=as_ref)
   1621       except (TypeError, ValueError):
   1622         # Could not coerce the conversion to use the preferred dtype.

/opt/conda/lib/python3.7/site-packages/tensorflow/python/ops/resource_variable_ops.py in
_dense_var_to_tensor(var, dtype, name, as_ref)
   2225 
   2226 def _dense_var_to_tensor(var, dtype=None, name=None, as_ref=False):
-> 2227   return var._dense_var_to_tensor(dtype=dtype, name=name, as_ref=as_ref)  # pylin
t: disable=protected-access
   2228 
   2229 

/opt/conda/lib/python3.7/site-packages/tensorflow/python/ops/resource_variable_ops.py in
_dense_var_to_tensor(***failed resolving arguments***)
   1467       return self.read_value().op.inputs[0]
   1468     else:
-> 1469       return self.value()
   1470 
   1471   def __iadd__(self, unused_other):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/ops/resource_variable_ops.py in
value(self)
    582       return self._cached_value
    583     with ops.colocate_with(None, ignore_existing=True):
--> 584       return self._read_variable_op()
    585 
    586   def _as_graph_element(self):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/ops/resource_variable_ops.py in
_read_variable_op(self, no_copy)
    704         result = read_and_set_handle(no_copy)
    705     else:
--> 706       result = read_and_set_handle(no_copy)
    707 
    708     if not context.executing_eagerly():
```

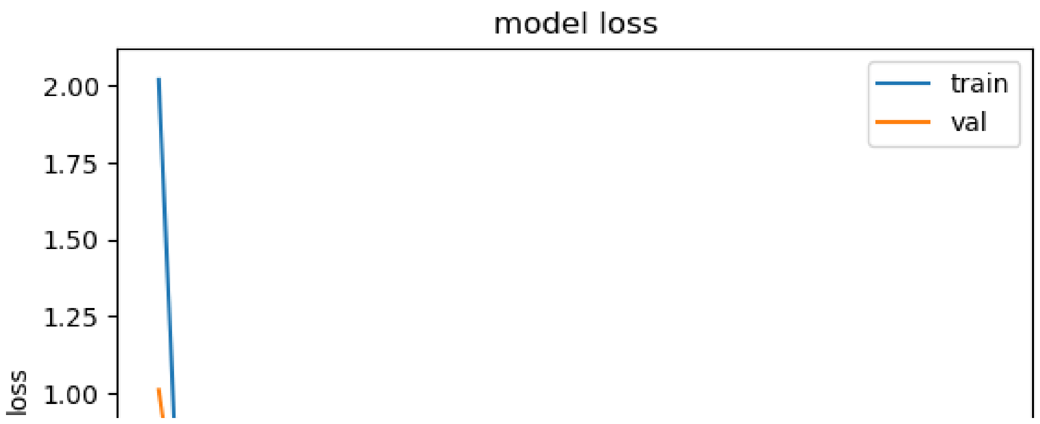
```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/ops/resource_variable_ops.py in
read_and_set_handle(no_copy)
    695         gen_resource_variable_ops.disable_copy_on_read(self.handle)
    696       result = gen_resource_variable_ops.read_variable_op(
--> 697           self.handle, self._dtype)
    698       _maybe_set_handle_data(self._dtype, self.handle, result)
    699       return result

/opt/conda/lib/python3.7/site-packages/tensorflow/python/ops/gen_resource_variable_ops.py
in read_variable_op(resource, dtype, name)
    538   dtype = _execute.make_type(dtype, "dtype")
    539   _, _, _op, _outputs = _op_def_library._apply_op_helper(
--> 540         "ReadVariableOp", resource=resource, dtype=dtype, name=name)
    541   _result = _outputs[:]
    542   if _execute.must_record_gradient():

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/op_def_library.py in _
apply_op_helper(op_type_name, name, **keywords)
    777       _ExtractInputsAndAttrs(op_type_name, op_def, allowed_list_attr_map,
    778                              keywords, default_type_attr_map, attrs, inputs,
--> 779                              input_types)
    780       _ExtractRemainingAttrs(op_type_name, op_def, keywords,
    781                              default_type_attr_map, attrs)

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/op_def_library.py in _
ExtractInputsAndAttrs(op_type_name, op_def, allowed_list_attr_map, keywords, default_type
_attr_map, attrs, inputs, input_types)
    553               dtype=dtype,
    554               as_ref=input_arg.is_ref,
--> 555               preferred_dtype=default_dtype)
    556       except TypeError as err:
    557         if dtype is None:

/opt/conda/lib/python3.7/site-packages/tensorflow/python/profiler/trace.py in wrapped(*ar
gs, **kwargs)
    181         with Trace(trace_name, **trace_kwargs):
    182           return func(*args, **kwargs)
--> 183       return func(*args, **kwargs)
    184 
    185     return wrapped

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/ops.py in convert_to_t
ensor(value, dtype, name, as_ref, preferred_dtype, dtype_hint, ctx, accepted_result_types
)
   1589                 "building a function.",
   1590                 name=name))
-> 1591       return graph.capture(value, name=name)
   1592 
   1593   if dtype is not None:

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/func_graph.py in captu
re(self, tensor, name, shape)
    783 
    784       # Large EagerTensors and resources are captured with Placeholder ops
--> 785       return self._capture_helper(tensor, name, shape)
    786     if tensor.graph is not self:
    787       if name is None:

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/func_graph.py in _capt
ure_helper(self, tensor, name, shape)
    819     if capture is None:
    820       placeholder = _create_substitute_placeholder(
--> 821           tensor, name=name, dtype=tensor.dtype, shape=shape)
    822       # Record the composite device as an attribute to the placeholder.
    823       # This attribute would be propogated into the arg_attr of the FunctionDef.

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/func_graph.py in _crea
te_substitute_placeholder(value, name, dtype, shape)
   1456   with ops.control_dependencies(None):
   1457     placeholder = graph_placeholder(
-> 1458         dtype=dtype or value.dtype, shape=shape, name=name)
   1459   handle_data_util.copy_handle_data(value, placeholder)
   1460     return placeholder

/opt/conda/lib/python3.7/site-packages/tensorflow/python/eager/graph_only_ops.py in graph
_placeholder(dtype, shape, name)
     34   op = g._create_op_internal(  # pylint: disable=protected-access
     35       "Placeholder", [], [dtype], input_types=[],
---> 36       attrs=attrs, name=name)
     37   result, = op.outputs
     38   if op_callbacks.should_invoke_op_callbacks():

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/func_graph.py in _crea
te_op_internal(self, op_type, inputs, dtypes, input_types, name, attrs, op_def, compute_d
evice)
    749     return super(FuncGraph, self)._create_op_internal(  # pylint: disable=protect
ed-access
    750         op_type, captured_inputs, dtypes, input_types, name, attrs, op_def,
--> 751         compute_device)
    752
    753   def capture(self, tensor, name=None, shape=None):

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/ops.py in _create_op_i
nternal(self, op_type, inputs, dtypes, input_types, name, attrs, op_def, compute_device)
   3792
   3793     input_ops = set(t.op for t in inputs)
-> 3794     control_inputs = self._control_dependencies_for_inputs(input_ops)
   3795     # _create_op_helper mutates the new Operation. `_mutation_lock` ensures a
   3796     # Session.run call cannot occur between creating and mutating the op.

/opt/conda/lib/python3.7/site-packages/tensorflow/python/framework/ops.py in _control_dep
endencies_for_inputs(self, input_ops)
   4966       # do not need to add control dependencies for this controller's inputs.
   4967       dominated = False
-> 4968       for op in input_ops:
   4969         if controller.op_in_group(op):
   4970           dominated = True

KeyboardInterrupt:
```

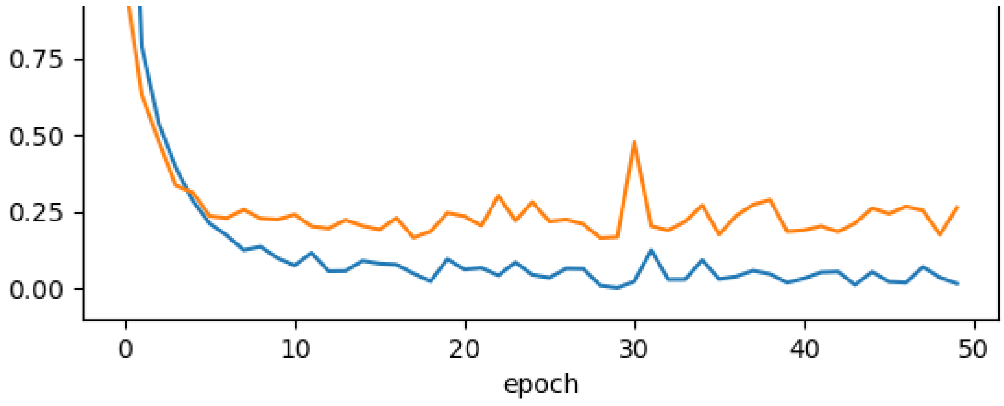
In [ ]:

```
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 [83]:

```
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 [96]:

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

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

```
    Test Loss: 0.24558
Test Accuracy: 96.27%
```

In [181]:

```
from sklearn.metrics import confusion_matrix , classification_report

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

```
294/294 [==============================] - 3s 10ms/step
Y Actual Values : [5, 1, 4, 0, 5, 0, 2, 0, 3, 2]
Y Predicted Values : [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

In [182]:

```
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 [183]:

```
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')}
```

In [184]:

```
cm = confusion_matrix(y_true,y_pred,labels=classes_labels)
print(confusion_matrix(y_true,y_pred,labels=classes_labels))
```

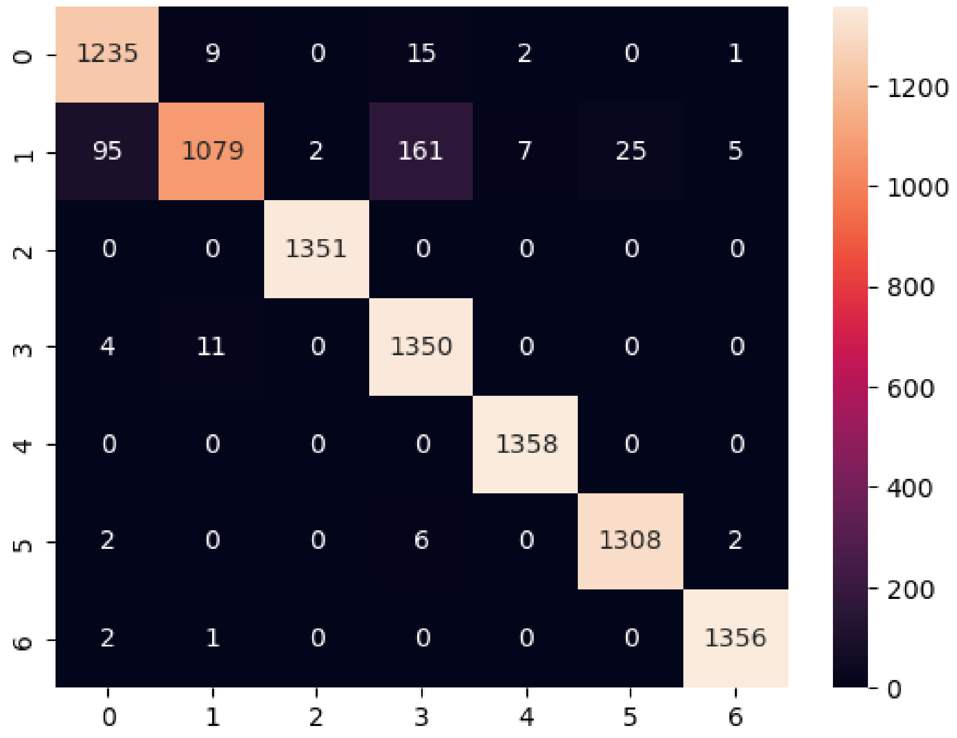
```
[[   0    0 1262    0    0    0    0]
 [   0    0 1374    0    0    0    0]
 [   0    0 1351    0    0    0    0]
 [   0    0 1365    0    0    0    0]
 [   0    0 1358    0    0    0    0]
 [   0    0 1318    0    0    0    0]
 [   0    0 1359    0    0    0    0]]
```

In [101]:

```
sns.heatmap(cm, annot = True, fmt='')
```

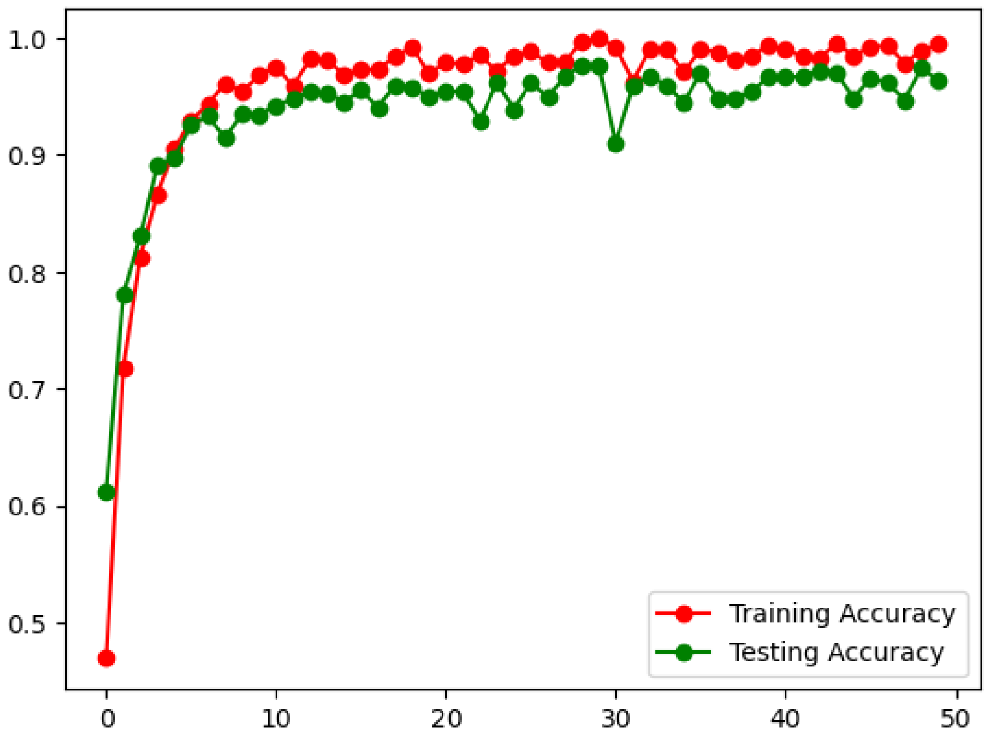
Out[101]:

```
<AxesSubplot:>
```



In [102]:

```
#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 [137]:

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

```
294/294 [==============================] - 3s 10ms/step
```

In [138]:

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

```
9387    9387
                                                               precision    recall  f1-
score   support

('akiec', 'Actinic keratoses and intraepithelial carcinomae')       0.99      1.00      0
.99      1359
                                  ('bcc', ' basal cell carcinoma')       0.98      0.99
0.99      1318
                          ('bkl', 'benign keratosis-like lesions')       0.92      0.98
0.95      1262
                                        ('df', 'dermatofibroma')       1.00      1.00
1.00      1351
                                     ('nv', ' melanocytic nevi')       0.98      0.79
0.87      1374
                   ('vasc', ' pyogenic granulomas and hemorrhage')       0.99      1.00
1.00      1358
                                              ('mel', 'melanoma')       0.88      0.99
0.93      1365

                                                          accuracy
0.96      9387
                                                         macro avg       0.96      0.96
0.96      9387
                                                      weighted avg       0.96      0.96
0.96      9387
```

In [ ]:

```

```

## NEW CNN Model 4-Layers

In [164]:

```
model_CNN = Sequential()
    model_CNN.add(Conv2D(16, kernel_size = (3,3), input_shape = (28, 28, 3), activation
= 'relu', 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, activation='relu'))
    model_CNN.add(Dense(7, 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_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_8 (Conv2D)           (None, 28, 28, 16)        448       
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 14, 14, 16)       0         
 2D)                                                             
                                                                 
 conv2d_9 (Conv2D)           (None, 14, 14, 32)        4640      
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 7, 7, 32)         0         
 2D)                                                             
                                                                 
 conv2d_10 (Conv2D)          (None, 7, 7, 64)          18496     
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 4, 4, 64)         0         
 2D)                                                             
                                                                 
 conv2d_11 (Conv2D)          (None, 4, 4, 128)         73856     
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 2, 2, 128)        0         
 2D)                                                             
                                                                 
 flatten_2 (Flatten)         (None, 512)               0         
                                                                 
 dense_10 (Dense)            (None, 64)                32832     
                                                                 
 dense_11 (Dense)            (None, 32)                2080      
                                                                 
 dense_12 (Dense)            (None, 7)                 231       
                                                                 
=================================================================
Total params: 132,583
Trainable params: 132,583
Non-trainable params: 0
_________________________________________________________________
None
```

In [165]:

```
history = model_CNN.fit(X_train,
                    Y_train,
                    validation_split=0.2,
                    batch_size = 128,
                    epochs = 50)
```

```
Epoch 1/50
235/235 [==============================] - 20s 77ms/step - loss: 2.5885 - accuracy: 0.358
7 - val_loss: 1.2554 - val_accuracy: 0.5305
Epoch 2/50
235/235 [==============================] - 17s 74ms/step - loss: 1.1541 - accuracy: 0.557
0 - val_loss: 1.1120 - val_accuracy: 0.5659
Epoch 3/50
235/235 [==============================] - 18s 76ms/step - loss: 0.9099 - accuracy: 0.660
7 - val_loss: 0.8082 - val_accuracy: 0.6985
Epoch 4/50
235/235 [==============================] - 17s 74ms/step - loss: 0.7223 - accuracy: 0.729
5 - val_loss: 0.6220 - val_accuracy: 0.7659
Epoch 5/50
235/235 [==============================] - 18s 76ms/step - loss: 0.5812 - accuracy: 0.786
1 - val_loss: 0.5774 - val_accuracy: 0.7812
Epoch 6/50
235/235 [==============================] - 18s 76ms/step - loss: 0.4820 - accuracy: 0.822
1 - val_loss: 0.4526 - val_accuracy: 0.8395
Epoch 7/50
235/235 [==============================] - 18s 75ms/step - loss: 0.3894 - accuracy: 0.860
3 - val_loss: 0.4067 - val_accuracy: 0.8519
```

```
Epoch 8/50
235/235 [==============================] - 18s 76ms/step - loss: 0.3418 - accuracy: 0.878
1 - val_loss: 0.3304 - val_accuracy: 0.8892
Epoch 9/50
235/235 [==============================] - 17s 74ms/step - loss: 0.2954 - accuracy: 0.894
7 - val_loss: 0.2797 - val_accuracy: 0.9016
Epoch 10/50
235/235 [==============================] - 18s 76ms/step - loss: 0.2300 - accuracy: 0.922
7 - val_loss: 0.3263 - val_accuracy: 0.8794
Epoch 11/50
235/235 [==============================] - 17s 74ms/step - loss: 0.2043 - accuracy: 0.929
3 - val_loss: 0.3116 - val_accuracy: 0.8868
Epoch 12/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1833 - accuracy: 0.934
9 - val_loss: 0.2169 - val_accuracy: 0.9298
Epoch 13/50
235/235 [==============================] - 18s 75ms/step - loss: 0.1614 - accuracy: 0.943
4 - val_loss: 0.1812 - val_accuracy: 0.9414
Epoch 14/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1248 - accuracy: 0.957
9 - val_loss: 0.2776 - val_accuracy: 0.9116
Epoch 15/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1389 - accuracy: 0.951
1 - val_loss: 0.1496 - val_accuracy: 0.9519
Epoch 16/50
235/235 [==============================] - 18s 75ms/step - loss: 0.1079 - accuracy: 0.960
9 - val_loss: 0.1818 - val_accuracy: 0.9463
Epoch 17/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1246 - accuracy: 0.956
3 - val_loss: 0.1977 - val_accuracy: 0.9329
Epoch 18/50
235/235 [==============================] - 17s 74ms/step - loss: 0.1242 - accuracy: 0.956
6 - val_loss: 0.2103 - val_accuracy: 0.9258
Epoch 19/50
235/235 [==============================] - 18s 76ms/step - loss: 0.1123 - accuracy: 0.960
0 - val_loss: 0.1526 - val_accuracy: 0.9545
Epoch 20/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0595 - accuracy: 0.980
2 - val_loss: 0.1912 - val_accuracy: 0.9458
Epoch 21/50
235/235 [==============================] - 18s 77ms/step - loss: 0.1614 - accuracy: 0.944
5 - val_loss: 0.1343 - val_accuracy: 0.9569
Epoch 22/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0461 - accuracy: 0.985
0 - val_loss: 0.1567 - val_accuracy: 0.9567
Epoch 23/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0785 - accuracy: 0.972
4 - val_loss: 0.1857 - val_accuracy: 0.9401
Epoch 24/50
235/235 [==============================] - 17s 74ms/step - loss: 0.1033 - accuracy: 0.964
4 - val_loss: 0.1391 - val_accuracy: 0.9626
Epoch 25/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0570 - accuracy: 0.980
6 - val_loss: 0.2287 - val_accuracy: 0.9398
Epoch 26/50
235/235 [==============================] - 18s 77ms/step - loss: 0.1085 - accuracy: 0.963
7 - val_loss: 0.2011 - val_accuracy: 0.9433
Epoch 27/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0944 - accuracy: 0.967
2 - val_loss: 0.1748 - val_accuracy: 0.9610
Epoch 28/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0219 - accuracy: 0.992
9 - val_loss: 0.0906 - val_accuracy: 0.9760
Epoch 29/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0151 - accuracy: 0.995
5 - val_loss: 0.1514 - val_accuracy: 0.9636
Epoch 30/50
235/235 [==============================] - 19s 80ms/step - loss: 0.1210 - accuracy: 0.960
4 - val_loss: 0.1581 - val_accuracy: 0.9511
Epoch 31/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0422 - accuracy: 0.985
5 - val_loss: 0.1387 - val_accuracy: 0.9634
```

```
Epoch 32/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0629 - accuracy: 0.978
2 - val_loss: 0.1702 - val_accuracy: 0.9534
Epoch 33/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0402 - accuracy: 0.986
2 - val_loss: 0.2046 - val_accuracy: 0.9582
Epoch 34/50
235/235 [==============================] - 18s 77ms/step - loss: 0.1231 - accuracy: 0.959
8 - val_loss: 0.1331 - val_accuracy: 0.9676
Epoch 35/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0392 - accuracy: 0.986
6 - val_loss: 0.1718 - val_accuracy: 0.9603
Epoch 36/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0354 - accuracy: 0.988
3 - val_loss: 0.1165 - val_accuracy: 0.9688
Epoch 37/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0188 - accuracy: 0.993
9 - val_loss: 0.1273 - val_accuracy: 0.9679
Epoch 38/50
235/235 [==============================] - 17s 74ms/step - loss: 0.1026 - accuracy: 0.968
0 - val_loss: 0.2632 - val_accuracy: 0.9322
Epoch 39/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0430 - accuracy: 0.985
4 - val_loss: 0.1801 - val_accuracy: 0.9628
Epoch 40/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0325 - accuracy: 0.989
1 - val_loss: 0.2478 - val_accuracy: 0.9406
Epoch 41/50
235/235 [==============================] - 18s 77ms/step - loss: 0.0386 - accuracy: 0.986
2 - val_loss: 0.3104 - val_accuracy: 0.9280
Epoch 42/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0592 - accuracy: 0.980
9 - val_loss: 0.1586 - val_accuracy: 0.9654
Epoch 43/50
235/235 [==============================] - 18s 78ms/step - loss: 0.0350 - accuracy: 0.988
2 - val_loss: 0.1852 - val_accuracy: 0.9582
Epoch 44/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0712 - accuracy: 0.976
3 - val_loss: 0.1430 - val_accuracy: 0.9648
Epoch 45/50
235/235 [==============================] - 17s 74ms/step - loss: 0.0280 - accuracy: 0.990
8 - val_loss: 0.2069 - val_accuracy: 0.9593
Epoch 46/50
235/235 [==============================] - 18s 77ms/step - loss: 0.0474 - accuracy: 0.984
8 - val_loss: 0.1993 - val_accuracy: 0.9559
Epoch 47/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0275 - accuracy: 0.990
5 - val_loss: 0.1815 - val_accuracy: 0.9622
Epoch 48/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0636 - accuracy: 0.979
6 - val_loss: 0.2474 - val_accuracy: 0.9486
Epoch 49/50
235/235 [==============================] - 18s 75ms/step - loss: 0.0473 - accuracy: 0.984
0 - val_loss: 0.1976 - val_accuracy: 0.9570
Epoch 50/50
235/235 [==============================] - 18s 76ms/step - loss: 0.0170 - accuracy: 0.994
1 - val_loss: 0.1731 - val_accuracy: 0.9656
```

In [185]:

```
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: 0.17276
Test Accuracy: 96.59%
```

## NEW CNN Model 4-Layers with Early Stopping & Reduce Learning Rate

In [187]:

```
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 [==============================] - 19s 41ms/step - loss: 0.2260 - accuracy: 0.930
8 - val_loss: 0.2124 - val_accuracy: 0.9465 - lr: 0.0010
Epoch 2/50
470/470 [==============================] - 20s 42ms/step - loss: 0.0955 - accuracy: 0.968
2 - val_loss: 0.1509 - val_accuracy: 0.9577 - lr: 0.0010
Epoch 3/50
470/470 [==============================] - 19s 41ms/step - loss: 0.0529 - accuracy: 0.982
0 - val_loss: 0.1256 - val_accuracy: 0.9635 - lr: 0.0010
Epoch 4/50
470/470 [==============================] - 20s 42ms/step - loss: 0.0847 - accuracy: 0.971
4 - val_loss: 0.1910 - val_accuracy: 0.9485 - lr: 0.0010
Epoch 5/50
470/470 [==============================] - 20s 42ms/step - loss: 0.0492 - accuracy: 0.983
7 - val_loss: 0.2246 - val_accuracy: 0.9537 - lr: 0.0010
Epoch 6/50
470/470 [==============================] - 19s 41ms/step - loss: 0.0567 - accuracy: 0.981
9 - val_loss: 0.1146 - val_accuracy: 0.9714 - lr: 0.0010
Epoch 7/50
470/470 [==============================] - 20s 42ms/step - loss: 0.0861 - accuracy: 0.971
5 - val_loss: 0.2055 - val_accuracy: 0.9474 - lr: 0.0010
Epoch 8/50
470/470 [==============================] - 19s 41ms/step - loss: 0.0383 - accuracy: 0.987
4 - val_loss: 0.1679 - val_accuracy: 0.9646 - lr: 0.0010
Epoch 9/50
470/470 [==============================] - ETA: 0s - loss: 0.0515 - accuracy: 0.9822
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
470/470 [==============================] - 19s 41ms/step - loss: 0.0515 - accuracy: 0.982
2 - val_loss: 0.1842 - val_accuracy: 0.9559 - lr: 0.0010
Epoch 10/50
470/470 [==============================] - 20s 42ms/step - loss: 0.0083 - accuracy: 0.998
0 - val_loss: 0.0953 - val_accuracy: 0.9796 - lr: 1.0000e-04
Epoch 11/50
470/470 [==============================] - 19s 41ms/step - loss: 0.0023 - accuracy: 0.999
8 - val_loss: 0.0915 - val_accuracy: 0.9812 - lr: 1.0000e-04
Epoch 12/50
470/470 [==============================] - 21s 44ms/step - loss: 0.0014 - accuracy: 0.999
9 - val_loss: 0.0917 - val_accuracy: 0.9816 - lr: 1.0000e-04
Epoch 13/50
470/470 [==============================] - 20s 42ms/step - loss: 9.7418e-04 - accuracy: 1
.0000 - val_loss: 0.0947 - val_accuracy: 0.9811 - lr: 1.0000e-04
Epoch 14/50
470/470 [==============================] - ETA: 0s - loss: 7.1054e-04 - accuracy: 1.0000
Epoch 14: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
470/470 [==============================] - 19s 41ms/step - loss: 7.1054e-04 - accuracy: 1
.0000 - val_loss: 0.0997 - val_accuracy: 0.9806 - lr: 1.0000e-04
Epoch 15/50
470/470 [==============================] - 20s 42ms/step - loss: 5.5160e-04 - accuracy: 1
.0000 - val_loss: 0.0991 - val_accuracy: 0.9807 - lr: 1.0000e-05
Epoch 16/50
470/470 [==============================] - 19s 41ms/step - loss: 5.3296e-04 - accuracy: 1
.0000 - val_loss: 0.0987 - val_accuracy: 0.9807 - lr: 1.0000e-05
Epoch 17/50
469/470 [============================>.] - ETA: 0s - loss: 5.1187e-04 - accuracy: 1.0000
Epoch 17: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
470/470 [==============================] - 20s 42ms/step - loss: 5.1157e-04 - accuracy: 1
.0000 - val_loss: 0.0991 - val_accuracy: 0.9807 - lr: 1.0000e-05
Epoch 18/50
```

```
470/470 [==============================] - 21s 44ms/step - loss: 4.9137e-04 - accuracy: 1
.0000 - val_loss: 0.0992 - val_accuracy: 0.9807 - lr: 1.0000e-06
Epoch 19/50
470/470 [==============================] - 19s 41ms/step - loss: 4.8836e-04 - accuracy: 1
.0000 - val_loss: 0.0992 - val_accuracy: 0.9807 - lr: 1.0000e-06
Epoch 20/50
469/470 [============================>.] - ETA: 0s - loss: 4.8349e-04 - accuracy: 1.0000
Epoch 20: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
470/470 [==============================] - 20s 42ms/step - loss: 4.8474e-04 - accuracy: 1
.0000 - val_loss: 0.0993 - val_accuracy: 0.9807 - lr: 1.0000e-06
Epoch 21/50
470/470 [==============================] - 20s 42ms/step - loss: 4.8106e-04 - accuracy: 1
.0000 - val_loss: 0.0993 - val_accuracy: 0.9807 - lr: 1.0000e-07
Epoch 21: early stopping
```

In [188]:

```
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: 0.10330
Test Accuracy: 98.09%
```

In [189]:

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

```
294/294 [==============================] - 3s 9ms/step
```

In [190]:

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

```
9387    9387
                                                               precision    recall  f1-
score   support

('akiec', 'Actinic keratoses and intraepithelial carcinomae')       0.99      1.00      1
.00      1359
                              ('bcc', ' basal cell carcinoma')       0.99      1.00
0.99      1318
                      ('bkl', 'benign keratosis-like lesions')       0.95      1.00
0.97      1262
                                      ('df', 'dermatofibroma')       1.00      1.00
1.00      1351
                                   ('nv', ' melanocytic nevi')       1.00      0.87
0.93      1374
               ('vasc', ' pyogenic granulomas and hemorrhage')       1.00      1.00
1.00      1358
                                          ('mel', 'melanoma')       0.94      1.00
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
```

In [ ]: