Projet Fédérateur

IMRANE OU EL FAQUIR

Diagnostic du cancer de la peau

Importing Libraries

In [1]:

```
import os
from pathlib import Path
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from keras.models import Sequential
from keras.layers import Activation, BatchNormalization, Dense, Dropout, Flat
from keras.applications.resnet import preprocess_input
from keras_preprocessing.image import ImageDataGenerator
```

Function for loading dataset

In [2]:

```
def load_dataset(path: str):
    dir = Path(path)
    filepaths = list(dir.glob(r'**/*.jpg'))
    labels = list(map(lambda l: os.path.split(os.path.split(l)[0])[1], filepa
    filepaths = pd.Series(filepaths, name='FilePaths').astype(str)
    labels = pd.Series(labels, name='Labels').astype(str)
    df = pd.merge(filepaths, labels, right_index=True, left_index=True)
    return df.sample(frac=1).reset_index(drop=True)
```

Unzip dataset.zip file and loading images

In [ ]:

```
!unzip /content/drive/MyDrive/S5/dataset.zip
```

In [5]:

```
df = load_dataset('/content/dataset/Train/')
```

In [6]:

```
df.head(2)
```

Out[6]:

	FilePaths	Labels
0	/content/dataset/Train/melanoma/ISIC_0010648.jpg	melanoma
1	/content/dataset/Train/melanoma/ISIC_0011140.jpg	melanoma

Viewing Dataset Labels

In [7]:

```
labels_count = df['Labels'].value_counts(ascending=True)
```

In [8]:

```
labels_count
```

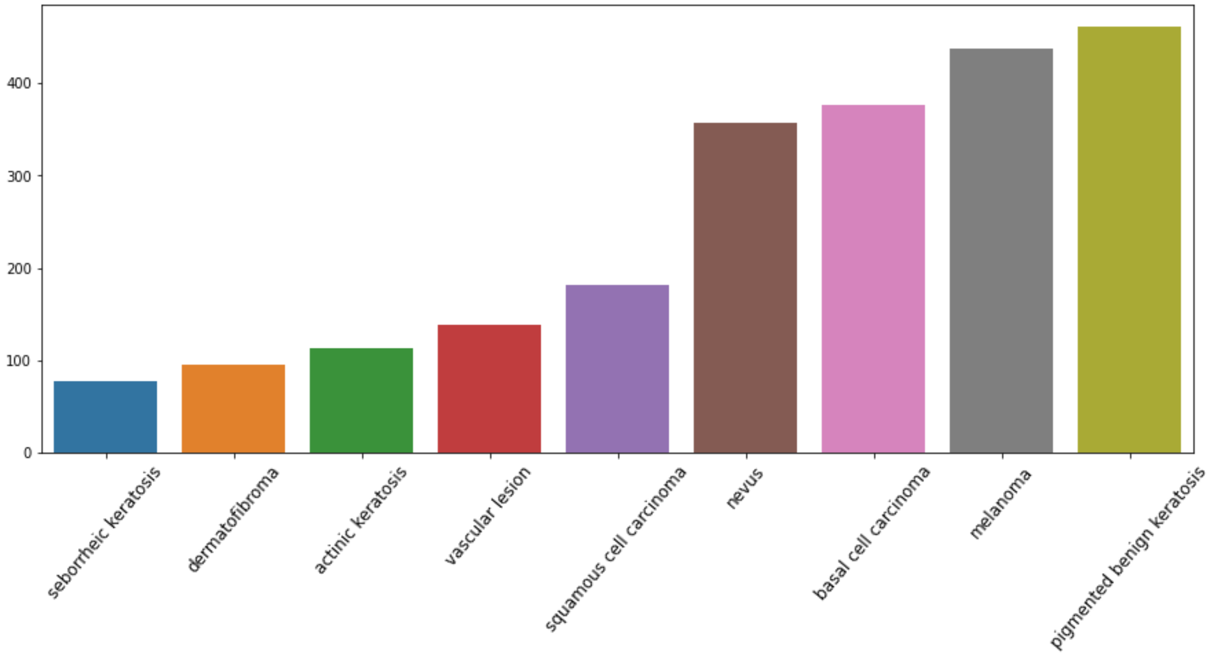
Out[8]:

```
seborrheic keratosis          77
dermatofibroma                95
actinic keratosis            114
vascular lesion              139
squamous cell carcinoma      181
nevus                        357
basal cell carcinoma         376
melanoma                     438
pigmented benign keratosis   462
Name: Labels, dtype: int64
```

In [9]:

```
fig = plt.figure(figsize=(15, 6))
s = sns.barplot(labels_count.index,labels_count.values)
f = s.set_xticklabels(s.get_xticklabels(), fontsize = 12, rotation = 50)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



In [10]:

```
# This function is for plotting images per label
def plot_images_per_label(df, label, cols: int, size: tuple):
    fig, axs = plt.subplots(nrows=1, ncols=cols, figsize=size)
    cntMax = cols
    cntCur = 0
    for index, row in df.iterrows():
        if(row['Labels'] == label and cntCur < cntMax):
            axs[cntCur].imshow(plt.imread(df.FilePaths[index]))
            axs[cntCur].set_title(df.Labels[index])

            cntCur += 1
        else:
            if(cntCur >= cntMax):
                break
```

```
    plt.tight_layout()
    plt.show()
```

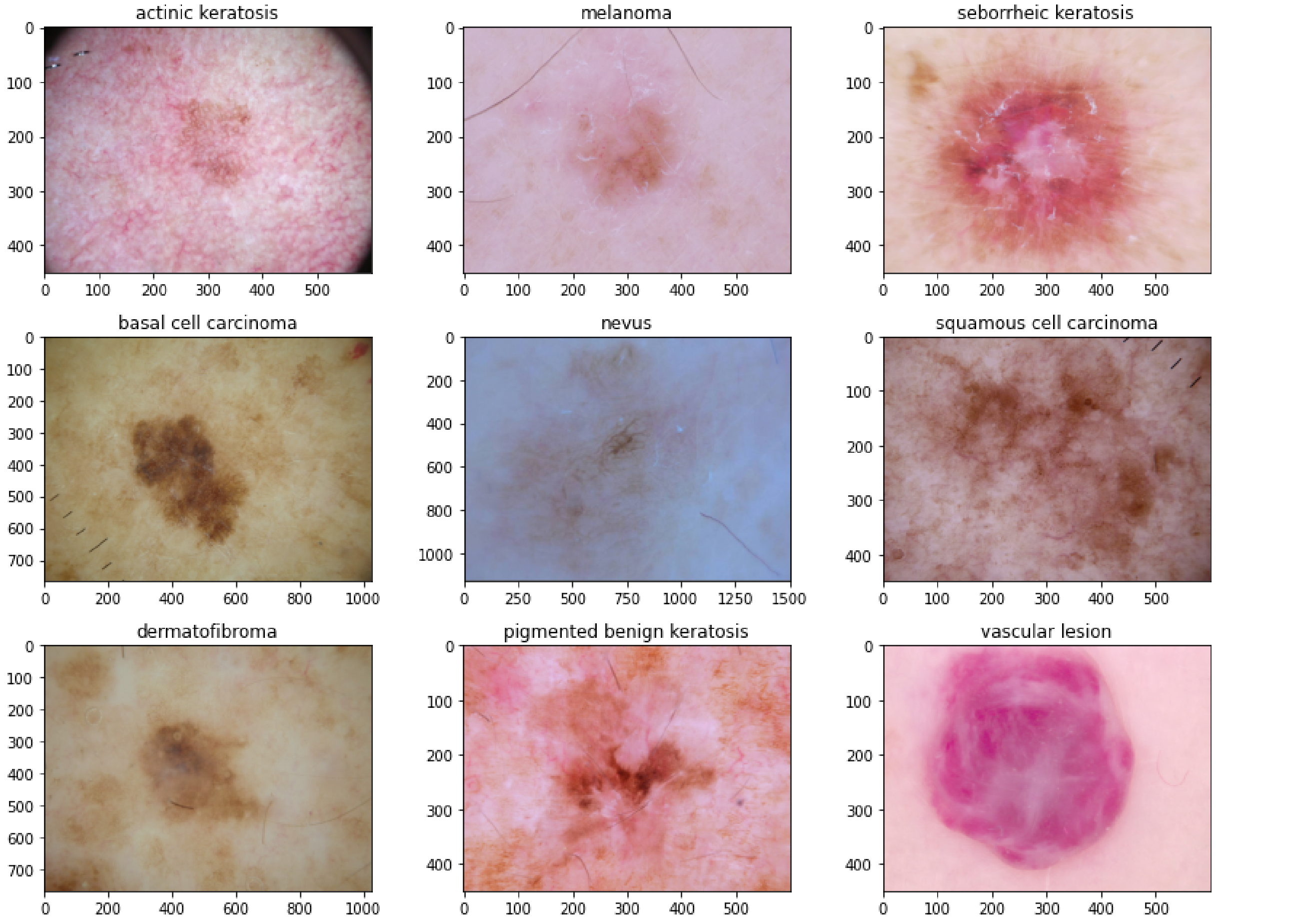
In [11]:

```
labels = sorted(df['Labels'].unique())
dictionnary = dict()
for label in labels:
    dictionnary.update({label:df[df.Labels == label].iloc[0][0]})
```

Plotting images of the differents skin cancer types

In [13]:

```
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(12,9))
keys = list(dictionnary.keys())
for i in range(3):
    for j in range(3):
        axs[i][j].imshow(plt.imread(dictionnary[keys[j + 3*(i%3)]]))
        axs[j][i].set_title(keys[j + 3*(i%3)])
plt.tight_layout()
plt.show()
```



Split the dataset into Train- and Val-datasets

stratified train and val (20%) datasets

In [37]:

```
X_train, X_val = train_test_split(df, test_size=0.2, stratify=df['Labels'], r
```

In [38]:

```
print('Train Data: ', X_train.shape)
print('Val Data: ', X_val.shape)
```

```
Train Data:  (1791, 2)
Val Data:  (448, 2)
```

Imrage preprocessing

We specify the batch size which (BATCH_SIZE) is the number of images per iteration, the image size (IMG_SIZE) and the number of epochs (EPOCHS)

In [39]:

```
BATCH_SIZE = 32
IMG_SIZE = (224, 224)
EPOCHS = 50
```

Image preprocessing

In [40]:

```
img_data_gen = ImageDataGenerator(shear_range=0.2, zoom_range=0.2, horizontal
```

In [41]:

```
X_train = img_data_gen.flow_from_dataframe(
    dataframe=X_train,
    x_col='FilePaths',
    y_col='Labels',
    target_size=IMG_SIZE,
    color_mode='rgb',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    seed=1
)

X_val = img_data_gen.flow_from_dataframe(
    dataframe=X_val,
    x_col='FilePaths',
    y_col='Labels',
    target_size=IMG_SIZE,
    color_mode='rgb',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    seed=1
)
```

```
Found 1791 validated image filenames belonging to 9 classes.
Found 448 validated image filenames belonging to 9 classes.
```

Plotting images after preprocessing

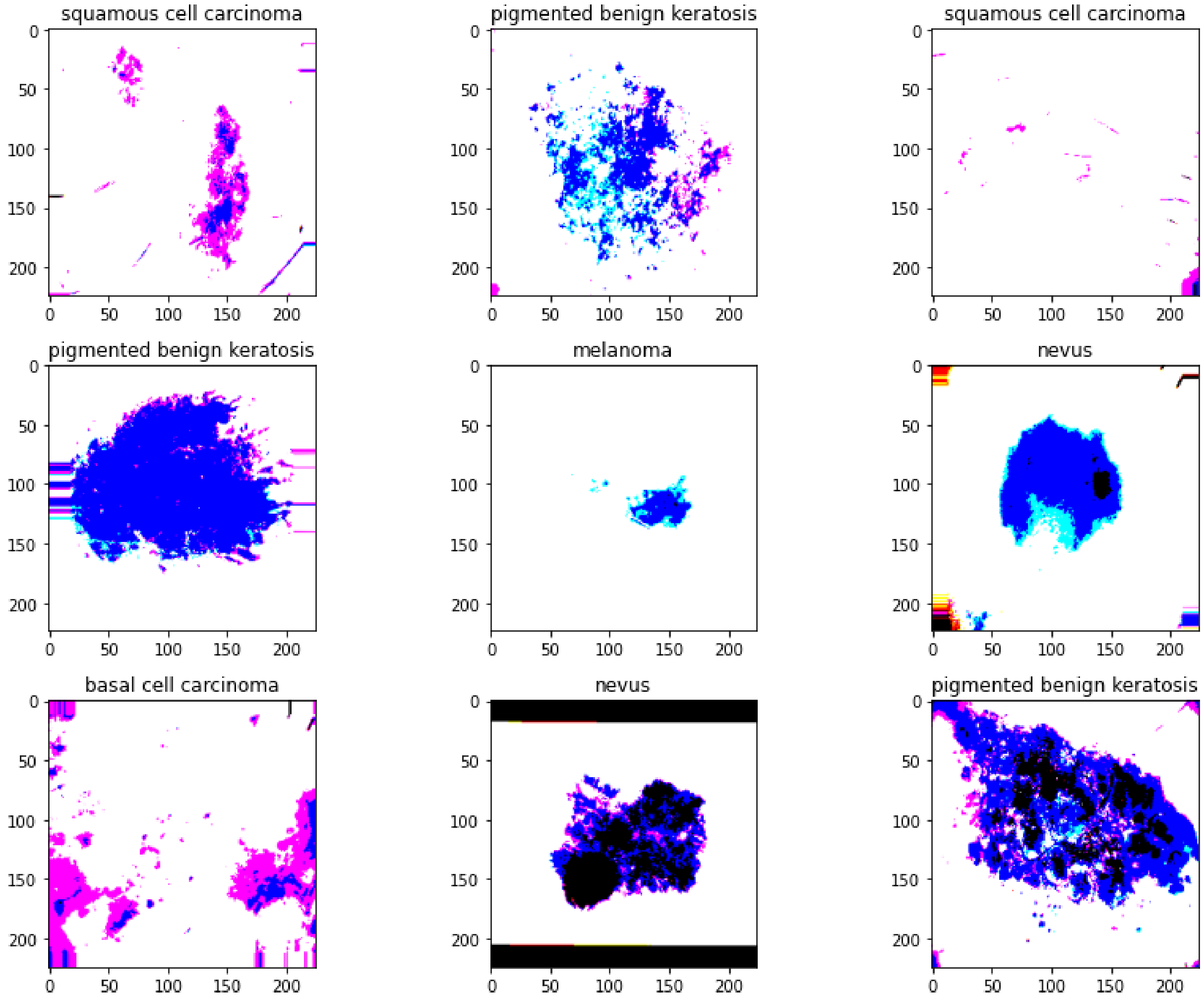
In [42]:

```
fit, axs = plt.subplots(nrows=3, ncols=3, figsize=(12,9))

for i, a in enumerate(axs.flat):
    img, label = X_train.next()
    a.imshow(img[0],)
    a.set_title(labels[np.where(label[0] == 1)[0][0]])
plt.tight_layout()
plt.show()
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
```

```
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
```



Creating CNN Model

In [44]:
```
model = Sequential()
```

Scale image size to 0.1

In [ ]:
```
model.add(tf.keras.layers.experimental.preprocessing.Rescaling(1./255))
```

1. Conv2D layer

In [45]:
```
model.add(Conv2D(filters=32, kernel_size=(3,3), padding='same', input_shape=(
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))
```

2. Conv2D layer

In [46]:
```
model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))
```

3. Conv2D layer

In [47]:
```
model.add(Conv2D(filters=128, kernel_size=(3,3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=2, padding='valid'))
```

Scale to 1 dimensional input for NN

In [48]:
```
model.add(Flatten())
```

Hidden fully connected layer

In [49]:
```
model.add(Dense(256))
model.add(Activation('relu'))
```

Inhibit overfitting

In [50]:
```
model.add(Dropout(0.2))
```

Output fully connected layer

In [51]:
```
model.add(Dense(9))
model.add(Activation('softmax'))
```

Compile model

In [52]:
```
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["ac
```

In [53]:
```
history = model.fit(X_train, validation_data=X_val, epochs=EPOCHS)
```

```
Epoch 1/50
56/56 [==============================] - 55s 971ms/step - loss: 15.8016 - acc
uracy: 0.2831 - val_loss: 46.9220 - val_accuracy: 0.0424
Epoch 2/50
56/56 [==============================] - 51s 917ms/step - loss: 2.2410 - accu
racy: 0.2915 - val_loss: 41.6504 - val_accuracy: 0.0424
Epoch 3/50
56/56 [==============================] - 51s 917ms/step - loss: 1.8882 - accu
racy: 0.3540 - val_loss: 22.4105 - val_accuracy: 0.1518
Epoch 4/50
56/56 [==============================] - 51s 913ms/step - loss: 1.8121 - accu
```

```
racy: 0.3830 - val_loss: 19.8426 - val_accuracy: 0.1875
Epoch 5/50
56/56 [==============================] - 51s 911ms/step - loss: 1.8004 - accu
racy: 0.3724 - val_loss: 24.7421 - val_accuracy: 0.2098
Epoch 6/50
56/56 [==============================] - 51s 905ms/step - loss: 1.7676 - accu
racy: 0.3735 - val_loss: 18.2236 - val_accuracy: 0.2567
Epoch 7/50
56/56 [==============================] - 51s 915ms/step - loss: 1.7670 - accu
racy: 0.3964 - val_loss: 13.0326 - val_accuracy: 0.2522
Epoch 8/50
56/56 [==============================] - 51s 915ms/step - loss: 1.6927 - accu
racy: 0.4143 - val_loss: 11.1575 - val_accuracy: 0.2835
Epoch 9/50
56/56 [==============================] - 51s 915ms/step - loss: 1.6384 - accu
racy: 0.4042 - val_loss: 4.9046 - val_accuracy: 0.3817
Epoch 10/50
56/56 [==============================] - 51s 912ms/step - loss: 1.6002 - accu
racy: 0.4165 - val_loss: 3.5385 - val_accuracy: 0.4196
Epoch 11/50
56/56 [==============================] - 51s 916ms/step - loss: 1.5602 - accu
racy: 0.4260 - val_loss: 1.7731 - val_accuracy: 0.4397
Epoch 12/50
56/56 [==============================] - 51s 911ms/step - loss: 1.6237 - accu
racy: 0.4098 - val_loss: 1.6083 - val_accuracy: 0.4754
Epoch 13/50
56/56 [==============================] - 51s 920ms/step - loss: 1.5507 - accu
racy: 0.4590 - val_loss: 1.7268 - val_accuracy: 0.4933
Epoch 14/50
56/56 [==============================] - 51s 914ms/step - loss: 1.5518 - accu
racy: 0.4618 - val_loss: 1.5181 - val_accuracy: 0.4576
Epoch 15/50
56/56 [==============================] - 51s 911ms/step - loss: 1.5464 - accu
racy: 0.4327 - val_loss: 1.6770 - val_accuracy: 0.3929
Epoch 16/50
56/56 [==============================] - 51s 910ms/step - loss: 1.4998 - accu
racy: 0.4394 - val_loss: 1.5379 - val_accuracy: 0.4643
Epoch 17/50
56/56 [==============================] - 51s 909ms/step - loss: 1.4684 - accu
racy: 0.4763 - val_loss: 1.5338 - val_accuracy: 0.5112
Epoch 18/50
56/56 [==============================] - 51s 911ms/step - loss: 1.4622 - accu
racy: 0.4796 - val_loss: 1.4207 - val_accuracy: 0.4821
Epoch 19/50
56/56 [==============================] - 51s 911ms/step - loss: 1.4762 - accu
racy: 0.4584 - val_loss: 1.5735 - val_accuracy: 0.4732
Epoch 20/50
56/56 [==============================] - 51s 916ms/step - loss: 1.4724 - accu
racy: 0.4791 - val_loss: 1.5169 - val_accuracy: 0.4933
Epoch 21/50
56/56 [==============================] - 51s 914ms/step - loss: 1.4507 - accu
racy: 0.4668 - val_loss: 1.3910 - val_accuracy: 0.4844
Epoch 22/50
56/56 [==============================] - 51s 914ms/step - loss: 1.4312 - accu
racy: 0.4690 - val_loss: 1.4613 - val_accuracy: 0.5268
Epoch 23/50
56/56 [==============================] - 51s 908ms/step - loss: 1.3938 - accu
racy: 0.5075 - val_loss: 1.3870 - val_accuracy: 0.4955
Epoch 24/50
56/56 [==============================] - 51s 912ms/step - loss: 1.3731 - accu
racy: 0.4980 - val_loss: 1.4087 - val_accuracy: 0.5134
Epoch 25/50
56/56 [==============================] - 51s 906ms/step - loss: 1.4039 - accu
racy: 0.4975 - val_loss: 1.5133 - val_accuracy: 0.4955
```

```
Epoch 26/50
56/56 [==============================] - 51s 909ms/step - loss: 1.3865 - accu
racy: 0.4964 - val_loss: 1.4155 - val_accuracy: 0.5000
Epoch 27/50
56/56 [==============================] - 51s 911ms/step - loss: 1.3286 - accu
racy: 0.5087 - val_loss: 1.3194 - val_accuracy: 0.5290
Epoch 28/50
56/56 [==============================] - 51s 908ms/step - loss: 1.3629 - accu
racy: 0.5137 - val_loss: 1.5433 - val_accuracy: 0.4375
Epoch 29/50
56/56 [==============================] - 51s 908ms/step - loss: 1.3896 - accu
racy: 0.4980 - val_loss: 1.5736 - val_accuracy: 0.4554
Epoch 30/50
56/56 [==============================] - 51s 911ms/step - loss: 1.3734 - accu
racy: 0.4891 - val_loss: 1.3613 - val_accuracy: 0.4933
Epoch 31/50
56/56 [==============================] - 51s 909ms/step - loss: 1.3392 - accu
racy: 0.5204 - val_loss: 1.4302 - val_accuracy: 0.4844
Epoch 32/50
56/56 [==============================] - 51s 910ms/step - loss: 1.3868 - accu
racy: 0.4969 - val_loss: 1.4773 - val_accuracy: 0.5246
Epoch 33/50
56/56 [==============================] - 51s 906ms/step - loss: 1.4412 - accu
racy: 0.4796 - val_loss: 1.4045 - val_accuracy: 0.4777
Epoch 34/50
56/56 [==============================] - 51s 910ms/step - loss: 1.3576 - accu
racy: 0.4997 - val_loss: 1.4224 - val_accuracy: 0.4844
Epoch 35/50
56/56 [==============================] - 51s 911ms/step - loss: 1.3229 - accu
racy: 0.5126 - val_loss: 1.4315 - val_accuracy: 0.4821
Epoch 36/50
56/56 [==============================] - 51s 905ms/step - loss: 1.3785 - accu
racy: 0.4947 - val_loss: 1.3367 - val_accuracy: 0.5045
Epoch 37/50
56/56 [==============================] - 51s 909ms/step - loss: 1.3204 - accu
racy: 0.5221 - val_loss: 1.7139 - val_accuracy: 0.5112
Epoch 38/50
56/56 [==============================] - 51s 909ms/step - loss: 1.2774 - accu
racy: 0.5360 - val_loss: 1.4030 - val_accuracy: 0.4844
Epoch 39/50
56/56 [==============================] - 51s 909ms/step - loss: 1.2757 - accu
racy: 0.5226 - val_loss: 1.3431 - val_accuracy: 0.5156
Epoch 40/50
56/56 [==============================] - 51s 907ms/step - loss: 1.2801 - accu
racy: 0.5377 - val_loss: 1.4148 - val_accuracy: 0.5402
Epoch 41/50
56/56 [==============================] - 51s 912ms/step - loss: 1.3303 - accu
racy: 0.5310 - val_loss: 1.5031 - val_accuracy: 0.5089
Epoch 42/50
56/56 [==============================] - 51s 911ms/step - loss: 1.2833 - accu
racy: 0.5366 - val_loss: 1.5746 - val_accuracy: 0.4888
Epoch 43/50
56/56 [==============================] - 51s 915ms/step - loss: 1.2374 - accu
racy: 0.5343 - val_loss: 1.2918 - val_accuracy: 0.5469
Epoch 44/50
56/56 [==============================] - 51s 914ms/step - loss: 1.2735 - accu
racy: 0.5293 - val_loss: 1.3787 - val_accuracy: 0.4911
Epoch 45/50
56/56 [==============================] - 51s 912ms/step - loss: 1.2708 - accu
racy: 0.5165 - val_loss: 1.4066 - val_accuracy: 0.5469
Epoch 46/50
56/56 [==============================] - 51s 913ms/step - loss: 1.3429 - accu
racy: 0.5165 - val_loss: 1.6572 - val_accuracy: 0.4754
Epoch 47/50
```

```
56/56 [==============================] - 51s 917ms/step - loss: 1.3415 - accu
racy: 0.5148 - val_loss: 1.3488 - val_accuracy: 0.4978
Epoch 48/50
56/56 [==============================] - 51s 915ms/step - loss: 1.2729 - accu
racy: 0.5288 - val_loss: 1.2861 - val_accuracy: 0.5312
Epoch 49/50
56/56 [==============================] - 51s 911ms/step - loss: 1.2152 - accu
racy: 0.5438 - val_loss: 1.5197 - val_accuracy: 0.5424
Epoch 50/50
56/56 [==============================] - 51s 912ms/step - loss: 1.2264 - accu
racy: 0.5388 - val_loss: 1.2538 - val_accuracy: 0.5536
```

Saving the model as a tflite file in order to deploy it in a mobile application.

Convert the model.

In [54]:

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

```
INFO:tensorflow:Assets written to: /tmp/tmpyd653x7p/assets
INFO:tensorflow:Assets written to: /tmp/tmpyd653x7p/assets
WARNING:absl:Buffer deduplication procedure will be skipped when flatbuffer l
ibrary is not properly loaded
```

Plotting the accuraccy and loss of Training vs Validation

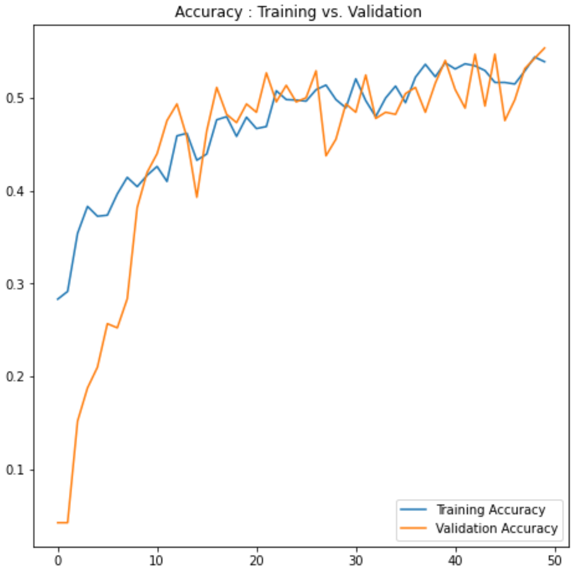
In [55]:

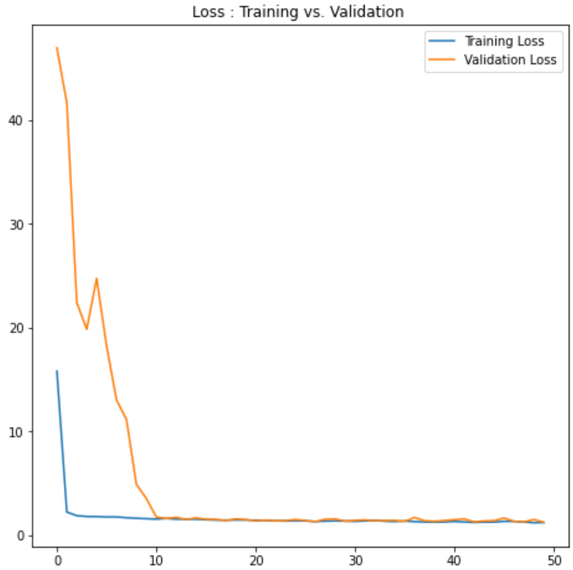
```
accuracy = history.history['accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
val_accuracy = history.history['val_accuracy']
```

In [56]:

```
plt.figure(figsize=(17, 17))
plt.subplot(2, 2, 1)
plt.plot(range(EPOCHS), accuracy, label='Training Accuracy')
plt.plot(range(EPOCHS), val_accuracy, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Accuracy : Training vs. Validation ')

plt.subplot(2, 2, 2)
plt.plot(range(EPOCHS), loss, label='Training Loss')
plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
plt.title('Loss : Training vs. Validation ')
plt.legend(loc='upper right')
plt.show()
```





Loading testing dataset

In [57]:

```
X_test = load_dataset('/content/dataset/Test/')
```

Shape of Test Data and ordered count of rows per unique label

In [58]:

```
print('Test Data: ', X_test.shape)

# ordered count of rows per unique label
X_test['Labels'].value_counts(ascending=True)
```

Out[58]:

```
Test Data:  (118, 2)
seborrheic keratosis          3
vascular lesion               3
actinic keratosis            16
melanoma                     16
dermatofibroma               16
pigmented benign keratosis   16
squamous cell carcinoma      16
basal cell carcinoma         16
nevus                        16
Name: Labels, dtype: int64
```

In [59]:

```
# image preprocessing
X_test = img_data_gen.flow_from_dataframe(
    dataframe=X_test,
    x_col='FilePaths',
    y_col='Labels',
    target_size=IMG_SIZE,
    color_mode='rgb',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=False, # necessary fpr confusion matrix
    seed=1
)
```

```
Found 118 validated image filenames belonging to 9 classes.
```

In [60]:

```
res = model.evaluate(X_test)
```

```
4/4 [==============================] - 9s 2s/step - loss: 3.0716 - accuracy:
0.3305
```

In [62]:

```
# accuracy
print(f'Train Accuracy: {history.history["accuracy"][-1:][0] * 100:.2f}')
print(f'Val Accuracy: {history.history["val_accuracy"][-1:][0] * 100:.2f}')
print(f'Test Accuracy: {res[1] * 100:.2f}')
# loss
print(f'Train Loss: {history.history["loss"][-1:][0] * 100:.2f}')
print(f'Val Loss: {history.history["val_loss"][-1:][0] * 100:.2f}')
print(f'Test Loss: {res[0] * 100:.2f}')
```

```
Train Accuracy: 53.88
Val Accuracy: 55.36
Test Accuracy: 33.05
Train Loss: 122.64
Val Loss: 125.38
Test Loss: 307.16
```

In [63]:

```
# predicted labels
Y_pred = model.predict(X_test)
print("Y_pred", Y_pred.shape)
# rounded labels
y_pred = np.argmax(Y_pred, axis=1)
print("y_pred", y_pred.size)
```

```
Y_pred (118, 9)
y_pred 118
```

In [64]:

```
# true labels
y_true = X_test.classes
print("y_pred", len(y_pred))
# label classes
class_labels = list(X_test.class_indices.keys())
print("labels", len(class_labels))
```

```
y_pred 118
labels 9
```

In [64]: