In [1]:
```
from tensorflow.keras import layers
```

```
C:\Users\krishnendu\Desktop\sample_project\env\lib\site-packages\scipy\__init__.p
y:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this vers
ion of SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
import pathlib
```

In [3]:
```
import tensorflow as tf
```

In [4]:
```
from tensorflow import keras
from tensorflow.keras.models import Sequential
```

In [5]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import PIL
```

In [6]:
```
''' train_ds have totle 2239
    test_ds have totle 118 files
    totle =2357 files , and train and test have 9 classe
'''
```

Out[6]:
```
' train_ds have totle 2239\n    test_ds have totle 118 files \n    totle =2357 fil
es , and train and test have 9 classe \n'
```

Create a dataset

Some parameter of data loader

In [7]:
```
batch_size = 32 # the number of images to be processed in each batch during traini
#that the model will process 32 images at a time before updating the weights.

channel = 3 # indicates the number of color channels in the images. In this case, 
# representing the Red, Green, and Blue (RGB) channels.

img_height = 180
img_width = 180  #images will be resized or cropped to these dimensions
```

In [8]:
```
# Writting the train dataset
# Resize the images with the mentioned image dimensions

train_data = keras.preprocessing.image_dataset_from_directory(r"Skin cancer ISIC Th
                                                              shuffle=True,
                                                              image_size=(img_heigh
                                                              batch_size= batch_siz
                                                              )
```

```
Found 2239 files belonging to 9 classes.
```

In [9]:
```
len(train_data)
```

Out[9]:
```
70
```

The totle files is 2239 and btach_size is 32 totole_batch = (2239/32) ~ 70

its mean 70 iteration

In [10]:
```
## Numbar of classes in the training data set
train_data.class_names
```

Out[10]:
```
['act_keratosis',
 'basal_carc',
 'dermatofibroma',
 'melanoma',
 'nevus',
 'pig_kerato',
 'sebo_kerato',
 'squamous_car',
 'vas_lesion']
```

- 'act_keratosis' : 'actinic keratosis',
- 'basal_carc' : 'basal cell carcinoma',
- 'dermatofibroma' : 'dermatofibroma',
- 'melanoma': 'melanoma',
- 'nevus' : 'nevus',
- 'pig_kerato' : 'pigmented benign keratosis',
- 'sebo_kerato' : 'seborrheic keratosis',
- 'squamous_car' : 'squamous cell carcinoma',
- 'vas_lesion' : 'vascular lesion'

In [11]:
```
# Writting the validation dataset
# Resize the images with the mentioned image dimensions

val_data = keras.preprocessing.image_dataset_from_directory(r"Skin cancer ISIC The
                                                             shuffle=True,
                                                             image_size=(img_heigh
                                                             batch_size= batch_siz
                                                             )
```

```
Found 118 files belonging to 9 classes.
```

The totle files is 118 and btach_size is 32

totole_batch = (118/32) ~ 4

its mean 4 iteration

In [12]:
```
len(val_data)
```

Out[12]:
```
4
```

In [13]:
```
## Numbar of classes in the validation data set
val_data.class_names
```

Out[13]:
```
['act_keratosis',
 'basal_carc',
 'dermatofibroma',
 'melanoma',
 'nevus',
 'pig_kerato',
 'sebo_kerato',
 'squamous_car',
 'vas_lesion']
```

In [14]:
```
val_data
```

Out[14]:
```
<BatchDataset element_spec=(TensorSpec(shape=(None, 180, 180, 3), dtype=tf.float3
2, name=None), TensorSpec(shape=(None,), dtype=tf.int32, name=None))>
```

In [15]:
```
train_data
```

Out[15]:
```
<BatchDataset element_spec=(TensorSpec(shape=(None, 180, 180, 3), dtype=tf.float3
2, name=None), TensorSpec(shape=(None,), dtype=tf.int32, name=None))>
```

The data type is tensor

In [16]:
```
# convert simple form
# fetch 1 batch , 1 batch have 32 files
for img_batch, label_batch in val_data.take(1):
    print("This is one batch shape : ", img_batch.shape)
    print("The label shape is :", label_batch.shape)
```

```
This is one batch shape :  (32, 180, 180, 3)
The label shape is : (32,)
```

In [17]:
```
## Normal form
for img_batch , label_batch in train_data.take(1):
    print(img_batch.numpy())
    print("\n")
    print("Convergt the label in numeric",label_batch.numpy())
```

```
[[[[170.41667  150.45833  149.41666 ]
   [173.875    156.625    160.      ]
   [176.875    157.91667  163.16667 ]
   ...
   [161.7084   135.37503  141.62503 ]
   [160.375    133.5      143.125   ]
   [159.62502  134.37502  143.37505 ]]

  [[172.83334  153.08333  152.08333 ]
   [174.875    158.75     162.125   ]
   [176.       157.375    159.91666 ]
   ...
   [161.91669  137.0416   143.74994 ]
   [161.75     136.875    146.75    ]
   [161.04167  136.66669  149.2083  ]]

  [[174.625    154.20833  154.29166 ]
   [176.625    157.125    163.25    ]
   [175.91667  158.79167  161.75    ]
   ...
   [161.58331  139.83331  150.66663 ]
   [162.875    139.875    152.375   ]
   [163.7917   139.75     150.45839 ]]

  ...

  [[157.33334  131.875    132.91667 ]
   [157.125    132.125    131.625   ]
   [150.54166  125.16667  125.41667 ]
   ...
   [148.0416   121.416565 125.666626]
   [144.25     117.       121.875   ]
   [142.95833  113.95833  117.29164 ]]

  [[154.91667  130.45833  131.20833 ]
   [158.125    132.125    134.625   ]
   [154.20833  125.875    128.79167 ]
   ...
   [146.2916   117.99994  121.83325 ]
   [145.125    116.5      119.75    ]
   [141.99997  115.874954 117.16666 ]]

  [[159.58334  130.70834  133.41667 ]
   [157.875    131.25     134.375   ]
   [156.66666  128.79166  132.33334 ]
   ...
   [142.99994  116.50006  119.66669 ]
   [142.375    113.375    117.875   ]
   [145.99994  117.99994  116.99994 ]]]


 [[[195.79167  134.91667  130.375   ]
   [195.75     134.25     130.375   ]
   [193.5      130.91666  128.33333 ]
   ...
   [187.00006  121.83331  126.00006 ]
   [187.25     125.625    122.5     ]
   [185.83331  125.12508  123.625046]]

  [[196.29167  133.75     130.91667 ]
   [198.375    133.5      139.25    ]
   [194.58333  128.16667  127.333336]
   ...
   [186.29147  127.458405 130.12479 ]
```

```
[[[216.73334  152.73334  106.73333 ]
  [217.       154.       110.      ]
  [217.01295  155.01295  108.01296 ]
  ...
  [219.72217  157.72217  108.72217 ]
  [215.63333  153.63333  103.9     ]
  [220.42929  162.42929  115.69596 ]]

 [[213.65445  150.65445  106.65444 ]
  [217.00333  154.00333  111.003334]
  [216.1      155.1      110.1     ]
  ...
  [217.62216  157.62216  107.62217 ]
  [219.83669  159.63669  109.73669 ]
  [217.80998  160.60999  115.70998 ]]

 [[216.39075  151.89075  104.39074 ]
  [217.20001  152.70001  108.86667 ]
  [217.9352   153.4352   107.60185 ]
  ...
  [216.72217  154.48145  106.71281 ]
  [215.03333  155.03333  103.033325]
  [216.53523  155.53523  110.535225]]

 ...

 [[224.59816  155.59816  116.59816 ]
  [224.39987  155.39987  114.23325 ]
  [225.5556   156.5556   115.555595]
  ...
  [227.4166   163.69443  119.97226 ]
  [228.       166.49988  127.33325 ]
  [233.28703  173.28703  136.28703 ]]

 [[227.44449  158.44449  116.54452 ]
  [224.13004  153.13004  110.92996 ]
  [228.50002  156.20007  112.70555 ]
  ...
  [230.14449  166.35008  127.37209 ]
  [234.96335  172.06339  129.26346 ]
  [233.83438  172.03445  135.13449 ]]

 [[225.41516  152.78186  117.948364]
  [224.       154.3667   118.5332  ]
  [226.92595  157.45563  121.19078 ]
  ...
  [229.53142  165.80925  120.364914]
  [232.6544   174.0211   134.0211  ]
  [228.24432  168.61102  129.77753 ]]]]

Convergt the label in numeric [1 5 4 4 1 3 7 7 5 1 1 8 5 5 7 5 7 5 8 8 5 5 4 1 5 2
 7 1 4 4 3 4]
```

In [18]:

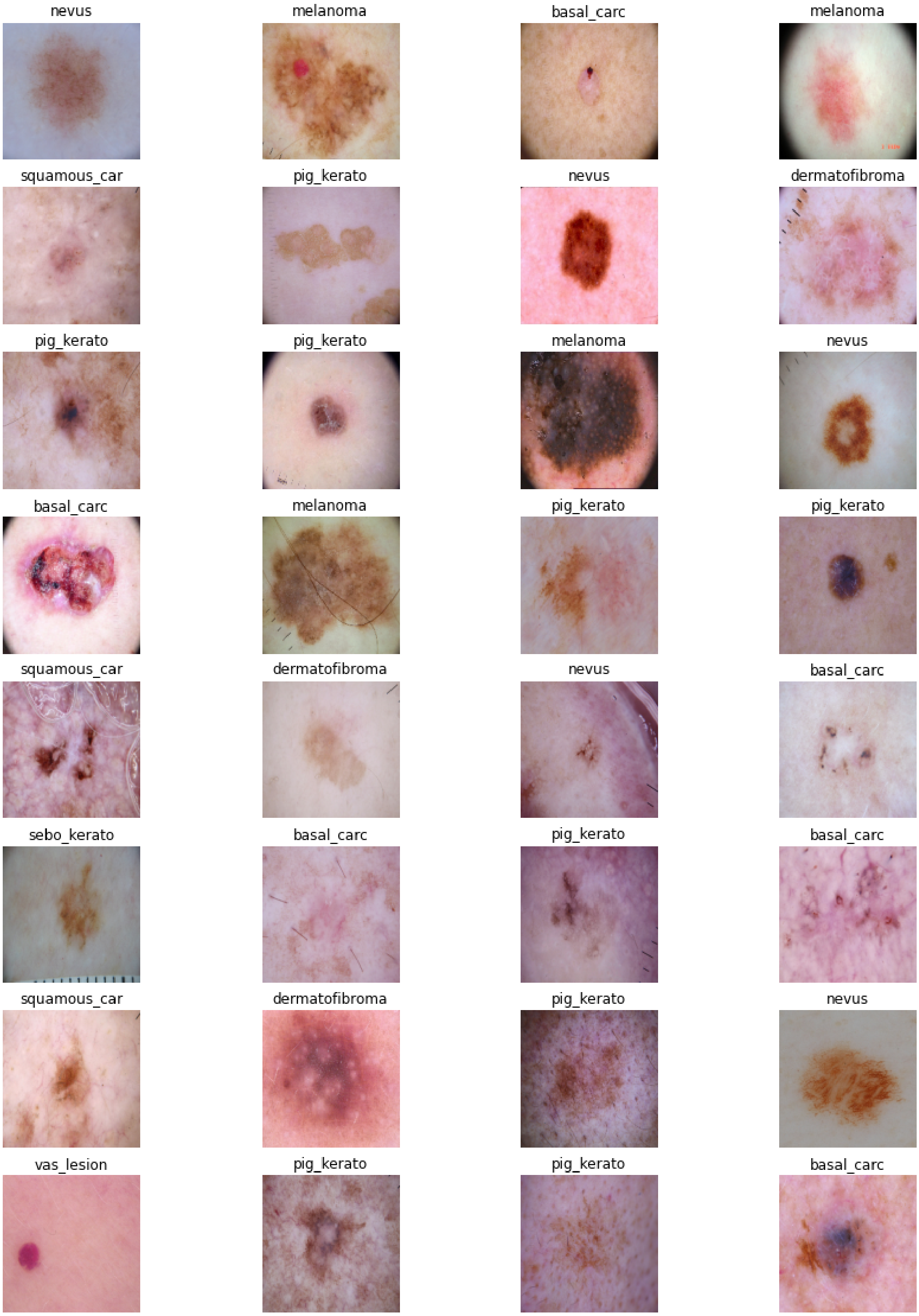
```
class_names = train_data.class_names
```

Visualize the data

Todo, create a code to visualize one instance of all the nine classes present in the dataset

In [19]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,20))

# The visualisation of training data has been done here
for image_batch , labels_batch in train_data.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
    for i in range(32):
        plt.subplot(8,4,i+1)
        plt.imshow(image_batch[i].numpy().astype('uint8'))
        plt.title(class_names[labels_batch[i]])
        plt.axis('off')
```

```
(32, 180, 180, 3)
[4 3 1 3 7 5 4 2 5 5 3 4 1 3 5 5 7 2 4 1 6 1 5 1 7 2 5 4 8 5 5 1]
```



The *img_batch* is a tensor of the shape $(32, 180, 180, 3)$. This is a batch of 32 images of shape $180x180x3$ (the last dimension refers to color channels RGB). The label_batch is a tensor of the shape (32,), these are corresponding labels to the 32 images.

$Dataset.cache()$ keeps the images in memory after they're loaded off disk during the first epoch.

Dataset.prefetch() overlaps data preprocessing and model execution while training.

In [20]:
```
# incress the perfomance
autotune = tf.data.experimental.AUTOTUNE
train_data = train_data.shuffle(1000).prefetch(buffer_size= autotune)
val_data = val_data.shuffle(1000).prefetch(buffer_size = autotune)
```

In [21]:
```
# Resize and Rescale the values
resize_and_resacle = keras.Sequential([
    keras.layers.experimental.preprocessing.Resizing(img_height,img_width),
    keras.layers.experimental.preprocessing.Rescaling(1.0/255)
])
```

In [22]:
```
## Data augentation
data_augmentation = keras.Sequential([
    keras.layers.experimental.preprocessing.RandomFlip('horizontal_and_vertical'),
    keras.layers.experimental.preprocessing.RandomRotation(0.3)
])
```

Create the model

Todo: Create a CNN model, which can accurately detect 9 classes present in the dataset. Use layers.experimental.preprocessing.Rescaling to normalize pixel values between (0,1). The RGB channel values are in the [0, 255] range. This is not ideal for a neural network. Here, it is good to standardize values to be in the [0, 1]

In [23]:
```
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, M
```

In [24]:
```
input_shape = (batch_size , img_height , img_width , channel)
n_class = len(class_names)
```

In [25]:
```
## CREAT CNN MODEL
model = Sequential()

model.add(resize_and_resacle)
model.add(data_augmentation)

## Covlutional layes
model.add(Conv2D(32,kernel_size=(3,3),padding='same', activation='relu',input_shap

model.add(Conv2D(64,kernel_size=(3,3),padding='same', activation='relu'))
model.add(MaxPooling2D(2,2))

model.add(Conv2D(64,kernel_size = (3,3), padding= 'same', activation= 'relu'))
model.add(MaxPooling2D(2,2))
model.add(Flatten())

## Add dense layers to connect the layers
model.add(Dense(32, activation= 'relu'))
model.add(Dense(64, activation= 'relu'))

## Add dropuout as regularisation
model.add(Dropout(0.10))
model.add(Dense(n_class))
```

Compile the model

Choose an appropirate optimiser and loss function for model training

In CNN models, various optimizers and loss functions are used to train the network effectively and optimize the learning process. Here are some commonly used optimizers and loss functions in CNN models, along with their functionalities and typical usage:

Optimizers:

Stochastic Gradient Descent (SGD):

Functionality: SGD is a classic optimization algorithm that updates the model's weights based on the gradients computed on small batches of training data. Usage: SGD is a simple optimizer often used as a baseline. It can be effective for training shallow CNN models or when dealing with limited computational resources.

Adam:

Functionality: Adam is an adaptive optimization algorithm that combines the benefits of both AdaGrad and RMSProp. It adjusts the learning rate adaptively based on the gradients' magnitudes. Usage: Adam is a popular optimizer choice in CNN models. It performs well across a wide range of problems and provides faster convergence compared to SGD.

Adagrad:

Functionality: Adagrad adapts the learning rate for each parameter based on the historical gradients. It reduces the learning rate more for frequently updated parameters. Usage: Adagrad is suitable when dealing with sparse data or when different features have significantly different frequencies. It can handle learning rate scheduling automatically.

RMSProp:

Functionality: RMSProp also adapts the learning rate based on the moving average of squared gradients. It divides the learning rate by a running average of the recent gradient magnitudes. Usage: RMSProp is effective in dealing with non-stationary or noisy gradients. It is commonly used in CNN models and performs well in various scenarios.

Loss Functions:

Binary Crossentropy:

Functionality: Binary Crossentropy is used for binary classification tasks, where the model predicts probabilities for two classes. Usage: Binary Crossentropy is typically used as the loss function in the output layer of CNN models when performing binary classification tasks.

Categorical Crossentropy:

Functionality: Categorical Crossentropy is suitable for multi-class classification tasks, where the model predicts probabilities across multiple mutually exclusive classes. Usage: Categorical Crossentropy is commonly used as the loss function in the output layer of CNN models for multi-class classification tasks.

Mean Squared Error (MSE):

Functionality: MSE calculates the average squared difference between the predicted and target values. It is often used for regression tasks. Usage: MSE is commonly used as the loss function in CNN models when performing regression tasks, such as predicting continuous values or coordinates.

Sparse Categorical Crossentropy:

Functionality: Sparse Categorical Crossentropy is similar to Categorical Crossentropy but accepts integer labels instead of one-hot encoded vectors. Usage: Sparse Categorical Crossentropy is used as the loss function in the output layer of CNN models for multi-class classification tasks with integer labels.

In [26]:
```
### Here we have to choose an appropriate optimiser and also a loss function
model.compile(optimizer= 'adam',
              loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              #expected to be a logits tensor. By  #default, we assume that `y_pre
              #CNN models for multi-class classification tasks with integer labels
              metrics=['accuracy'])
```

In [27]:
```
model.build(input_shape)
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
```

In [28]:
```
# View the summary of all layers
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential (Sequential)     (32, 180, 180, 3)         0         
                                                                 
 sequential_1 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 180, 180, 32)      896       
                                                                 
 conv2d_1 (Conv2D)           (None, 180, 180, 64)      18496     
                                                                 
 max_pooling2d (MaxPooling2D  (None, 90, 90, 64)       0         
 )                                                               
                                                                 
 conv2d_2 (Conv2D)           (None, 90, 90, 64)        36928     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 45, 45, 64)       0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 129600)            0         
                                                                 
 dense (Dense)               (None, 32)                4147232   
                                                                 
 dense_1 (Dense)             (None, 64)                2112      
                                                                 
 dropout (Dropout)           (None, 64)                0         
                                                                 
 dense_2 (Dense)             (None, 9)                 585       
                                                                 
=================================================================
Total params: 4,206,249
Trainable params: 4,206,249
Non-trainable params: 0
_________________________________________________________________
```

In [29]:
```
epochs = 10

history = model.fit(train_data,validation_data= val_data, epochs=epochs)
```

```
Epoch 1/10
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
70/70 [==============================] - 220s 3s/step - loss: 2.0347 - accuracy:
0.2144 - val_loss: 2.1485 - val_accuracy: 0.2288
Epoch 2/10
70/70 [==============================] - 226s 3s/step - loss: 1.8645 - accuracy:
0.3198 - val_loss: 2.3261 - val_accuracy: 0.3220
Epoch 3/10
70/70 [==============================] - 206s 3s/step - loss: 1.6526 - accuracy:
0.4288 - val_loss: 2.0924 - val_accuracy: 0.3475
Epoch 4/10
70/70 [==============================] - 197s 3s/step - loss: 1.5123 - accuracy:
0.4730 - val_loss: 2.1662 - val_accuracy: 0.3390
Epoch 5/10
70/70 [==============================] - 202s 3s/step - loss: 1.4644 - accuracy:
0.5060 - val_loss: 2.1836 - val_accuracy: 0.3305
Epoch 6/10
70/70 [==============================] - 206s 3s/step - loss: 1.3763 - accuracy:
0.5252 - val_loss: 2.0576 - val_accuracy: 0.3729
Epoch 7/10
70/70 [==============================] - 203s 3s/step - loss: 1.3472 - accuracy:
0.5284 - val_loss: 2.1824 - val_accuracy: 0.3814
Epoch 8/10
70/70 [==============================] - 209s 3s/step - loss: 1.3252 - accuracy:
0.5360 - val_loss: 2.1387 - val_accuracy: 0.3814
```

```
Epoch 9/10
70/70 [==============================] - 221s 3s/step - loss: 1.3099 - accuracy:
0.5333 - val_loss: 2.1448 - val_accuracy: 0.3814
Epoch 10/10
70/70 [==============================] - 209s 3s/step - loss: 1.2863 - accuracy:
0.5395 - val_loss: 1.9936 - val_accuracy: 0.3559
```

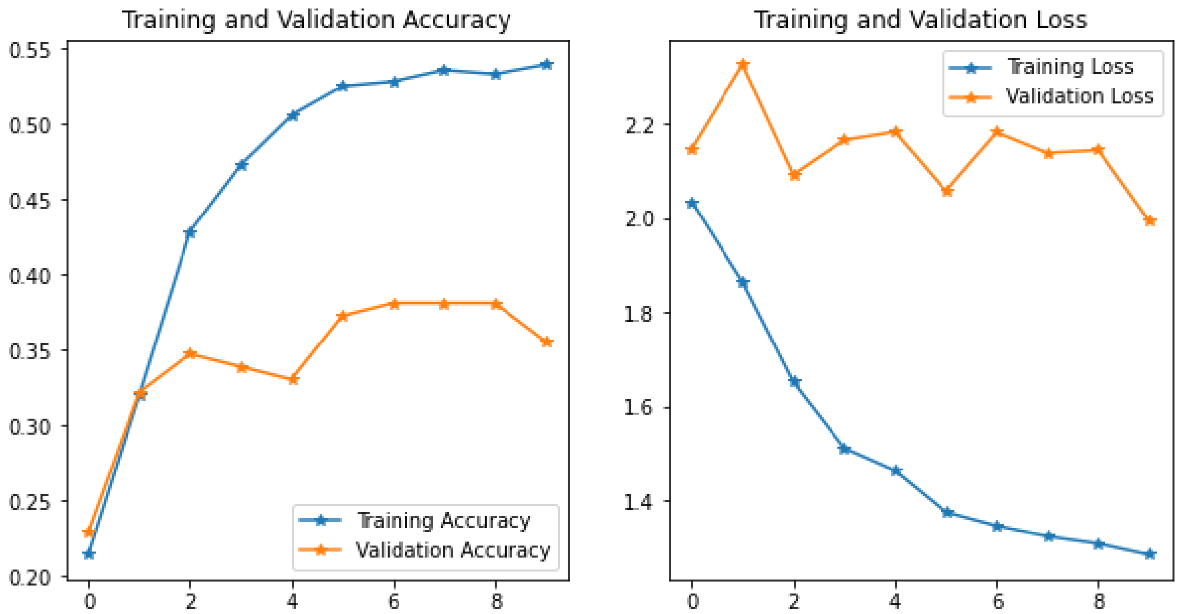
In [30]:

```
# accuracy of model each epochs
print("Accuraccy propagation :",history.history['accuracy'])
print('\n')
# validation accuracy of model each epochs
print("Validation accuracy propagation :",history.history['val_accuracy'])
print('\n')

#Loss of model each epochs
print("Loss propagation through epochs",history.history["loss"])
```

```
Accuraccy propagation : [0.21438142657279968, 0.31978562474250793, 0.4287628531455
9937, 0.47297900915145874, 0.506029486656189, 0.5252344608306885, 0.52836090326309
2, 0.5359535217285156, 0.5332737565040588, 0.5395265817642212]


Validation accuracy propagation : [0.2288135588169098, 0.32203391194343567, 0.3474
57617521286, 0.33898305892944336, 0.3305084705352783, 0.37288135290145874, 0.38135
59412956238, 0.3813559412956238, 0.3813559412956238, 0.35593220591545105]


Loss propagation through epochs [2.034680128097534, 1.8644813299179077, 1.65256571
76971436, 1.5122623443603516, 1.464416742324829, 1.3762773275375366, 1.34715235233
30688, 1.3252204656600952, 1.3099390268325806, 1.2862755060195923]
```

In [31]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy',marker = '*')
plt.plot(epochs_range, val_acc, label='Validation Accuracy',marker = '*')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss',marker = '*')
plt.plot(epochs_range, val_loss, label='Validation Loss',marker = '*')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Findings of the model (Underfit or Overfit)

The **Training_accuracy** = 0.50 and the **Validation accuracy** = 0.35 Which both are very less, So the model is clearly **underfit**

Sol = We will different augmentation to create more datas.

In [32]:
```
import warnings

# Ignore all warnings
warnings.filterwarnings("ignore")

# Your code here

# Reset warning behavior (optional)
warnings.resetwarnings()
```

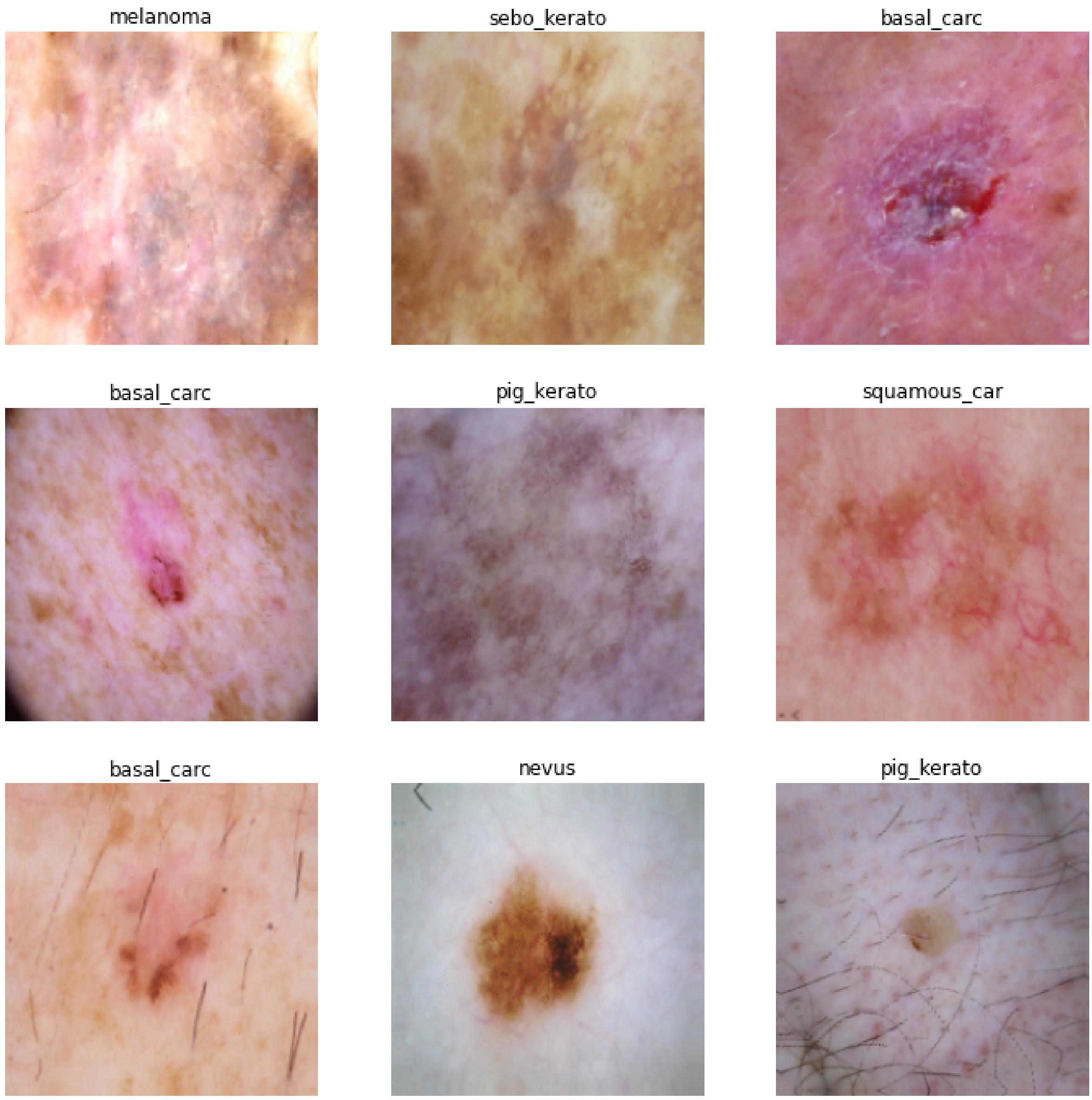
In [33]:
```
data_augmentation = keras.Sequential([
    keras.layers.experimental.preprocessing.RandomFlip('horizontal',input_shape=(i
                                                          img_width,3)),
    keras.layers.experimental.preprocessing.RandomRotation(0.1),
    keras.layers.experimental.preprocessing.RandomZoom(0.1)
])
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
```

In [34]:

```
# Todo, visualize how your augmentation strategy works for one instance of training
# Your code goes here
plt.figure(figsize=(12, 12))
for images, labels in train_data.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

```
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
```



Create the model, compile and train the model

In [35]:

```
## You can use Dropout layer if there is an evidence of overfitting in your finding

## Your code goes here

model = Sequential()
model.add(data_augmentation)
model.add(resize_and_resacle)

model.add(Conv2D(16, kernel_size= (3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(2,2))

model.add(Conv2D(32,kernel_size=(3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(2,2))
```

```
model.add(Conv2D(64, kernel_size=(3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(2,2))

model.add(Dropout(0.2))
model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dense(n_class))
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
```

Compiling the model

In [36]:

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

Training the model

In [37]:

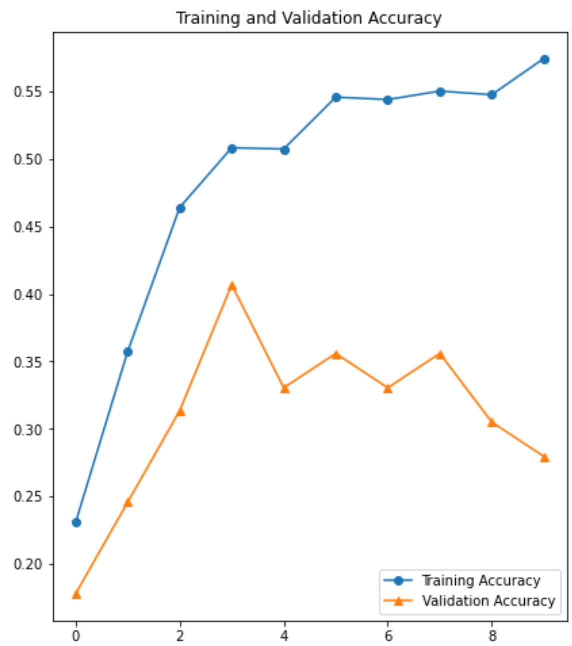
```
history = model.fit(train_data , validation_data= val_data, epochs=10)
```

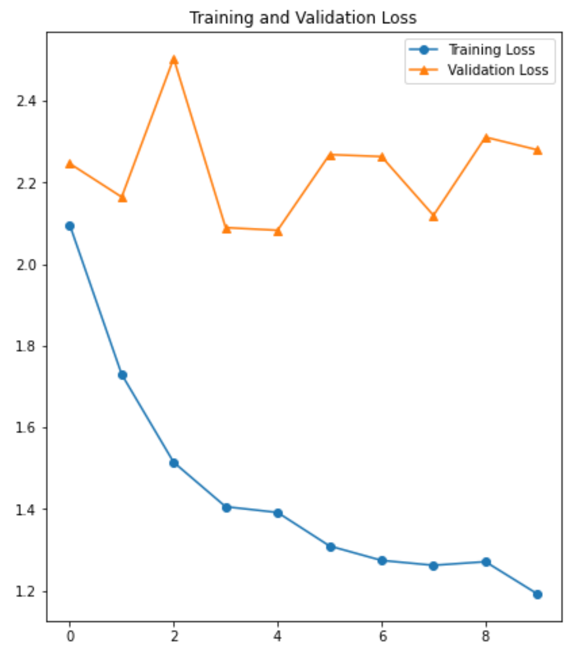
```
Epoch 1/10
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is
no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no reg
istered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 caus
e there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 ca
use there is no registered converter for this op.
70/70 [==============================] - 62s 708ms/step - loss: 2.0961 - accuracy:
0.2309 - val_loss: 2.2464 - val_accuracy: 0.1780
Epoch 2/10
70/70 [==============================] - 61s 765ms/step - loss: 1.7298 - accuracy:
0.3573 - val_loss: 2.1644 - val_accuracy: 0.2458
Epoch 3/10
70/70 [==============================] - 62s 728ms/step - loss: 1.5145 - accuracy:
0.4640 - val_loss: 2.5041 - val_accuracy: 0.3136
Epoch 4/10
70/70 [==============================] - 62s 758ms/step - loss: 1.4057 - accuracy:
0.5083 - val_loss: 2.0894 - val_accuracy: 0.4068
Epoch 5/10
70/70 [==============================] - 61s 756ms/step - loss: 1.3915 - accuracy:
0.5074 - val_loss: 2.0829 - val_accuracy: 0.3305
Epoch 6/10
70/70 [==============================] - 58s 730ms/step - loss: 1.3095 - accuracy:
0.5458 - val_loss: 2.2683 - val_accuracy: 0.3559
Epoch 7/10
70/70 [==============================] - 63s 769ms/step - loss: 1.2741 - accuracy:
0.5440 - val_loss: 2.2635 - val_accuracy: 0.3305
Epoch 8/10
70/70 [==============================] - 60s 732ms/step - loss: 1.2622 - accuracy:
0.5502 - val_loss: 2.1194 - val_accuracy: 0.3559
```

```
Epoch 9/10
70/70 [==============================] - 60s 743ms/step - loss: 1.2709 - accuracy: 0.5476 - val_loss: 2.3111 - val_accuracy: 0.3051
Epoch 10/10
70/70 [==============================] - 62s 740ms/step - loss: 1.1920 - accuracy: 0.5744 - val_loss: 2.2799 - val_accuracy: 0.2797
```

Visualizing the results

In [38]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(15, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy',marker = 'o')
plt.plot(epochs_range, val_acc, label='Validation Accuracy',marker = '^')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss',marker = 'o')
plt.plot(epochs_range, val_loss, label='Validation Loss',marker = '^')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```





Todo: Write your findings after the model fit, see if there is an evidence of model overfit or underfit. Do you think there is some improvement now as compared to the previous model run? This model like underfiting beacuse

Traning accuracy = 57 and

Validation accuracy = 44

Todo: Find the distribution of classes in the training dataset. **Context:** Many times real life datasets can have class imbalance, one class can have proportionately higher number of samples compared to the others. Class imbalance can have a detrimental effect on the final model quality. Hence as a sanity check it becomes important to check what is the distribution of classes in the data.

In [40]:
```
data_dir = pathlib.Path(r"Skin cancer ISIC The International Skin Imaging Collabora
```

In [41]:
```
image_count_train = len(list(data_dir.glob('*/*.jpg'))) ## How many images are ther
print(image_count_train)
```

```
2239
```

In [42]:
```
class_names
```

Out[42]:
```
['act_keratosis',
 'basal_carc',
 'dermatofibroma',
 'melanoma',
 'nevus',
 'pig_kerato',
 'sebo_kerato',
 'squamous_car',
 'vas_lesion']
```

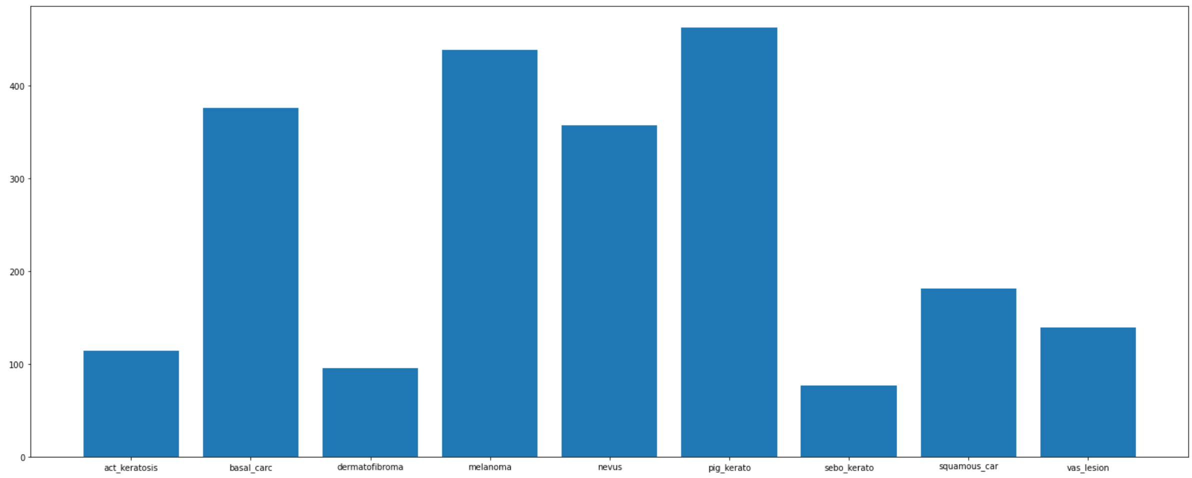
In [43]:
```
#plot number of images in each Class
count=[]
for name in class_names:
    count.append(len(list(data_dir.glob(name+'/*.jpg'))))
```

In [44]:
```
count
```

Out[44]:
```
[114, 376, 95, 438, 357, 462, 77, 181, 139]
```

In [45]:
```
plt.figure(figsize=(25,10))
plt.bar(class_names,count)
```

Out[45]:
```
<BarContainer object of 9 artists>
```



So clearly there is a class imbalancein the data set.

Todo: Write your findings here:

- Which class has the least number of samples?

- Which classes dominate the data in terms proportionate number of samples? Todo: Rectify the class imbalance Context: You can use a python package known as Augmentor (https://augmentor.readthedocs.io/en/master/) to add more samples across all classes so that none of the classes have very few samples.

In [ ]:

```

```

In [46]:

```python
import Augmentor as aug
```

To use Augmentor, the following general procedure is followed:

Instantiate a **Pipeline** object pointing to a directory containing your initial image data set. Define a number of operations to perform on this data set using your **Pipeline** object. Execute these operations by calling the **Pipeline's sample()** method.

In [47]:

```python
data_dir
```

Out[47]:

```
WindowsPath('Skin cancer ISIC The International Skin Imaging Collaboration/Train')
```

In [48]:

```python
path_to_training_dataset=str(data_dir)+'/'
for i in class_names:
    p = aug.Pipeline(path_to_training_dataset + str(i))
    p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
    p.sample(500) ## We are adding 500 samples per class to make sure that none of
```

```
Initialised with 114 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/act_keratosis\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x14F0E6E7580>: 100%|█|
500/500 [00:07<00:00, 71.30 Samples/
Initialised with 376 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/basal_carc\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x14F0E5A0DC0>: 100%|█|
500/500 [00:06<00:00, 72.11 Samples/
Initialised with 95 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/dermatofibroma\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x14F0CA203A0>: 100%|█|
500/500 [00:07<00:00, 63.22 Samples/
Initialised with 438 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/melanoma\output.
Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x14F0D0134C0>: 100%|█
| 500/500 [00:30<00:00, 16.26 Samples
Initialised with 357 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/nevus\output.
Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x14F1120D8D0>: 100%|█
| 500/500 [00:43<00:00, 11.54 Samples
Initialised with 462 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/pig_kerato\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x14F07842C80>: 100%|█|
500/500 [00:12<00:00, 39.21 Samples/
Initialised with 77 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/sebo_kerato\output.
```

```
Processing <PIL.Image.Image image mode=RGB size=1024x768 at 0x14F11178DF0>: 100%|█
| 500/500 [00:25<00:00, 19.32 Samples
Initialised with 181 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/squamous_car\output.
Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=600x450 at 0x14F
0C8EEE60>: 100%|█| 500/500 [00:12<00:
Initialised with 139 image(s) found.
Output directory set to Skin cancer ISIC The International Skin Imaging Collaborat
ion\Train/vas_lesion\output.
Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x14F0E6E7A90>: 100%|█|
500/500 [00:13<00:00, 35.84 Samples/
```

In [49]:

```
#Augmentor has stored the augmented images in the output sub-directory of each of t
#Lets take a look at total count of augmented images.

image_count_train = len(list(data_dir.glob('*/output/*.jpg')))
print(image_count_train)
```

```
4500
```

In [50]:

```
class_names
```

Out[50]:

```
['act_keratosis',
 'basal_carc',
 'dermatofibroma',
 'melanoma',
 'nevus',
 'pig_kerato',
 'sebo_kerato',
 'squamous_car',
 'vas_lesion']
```

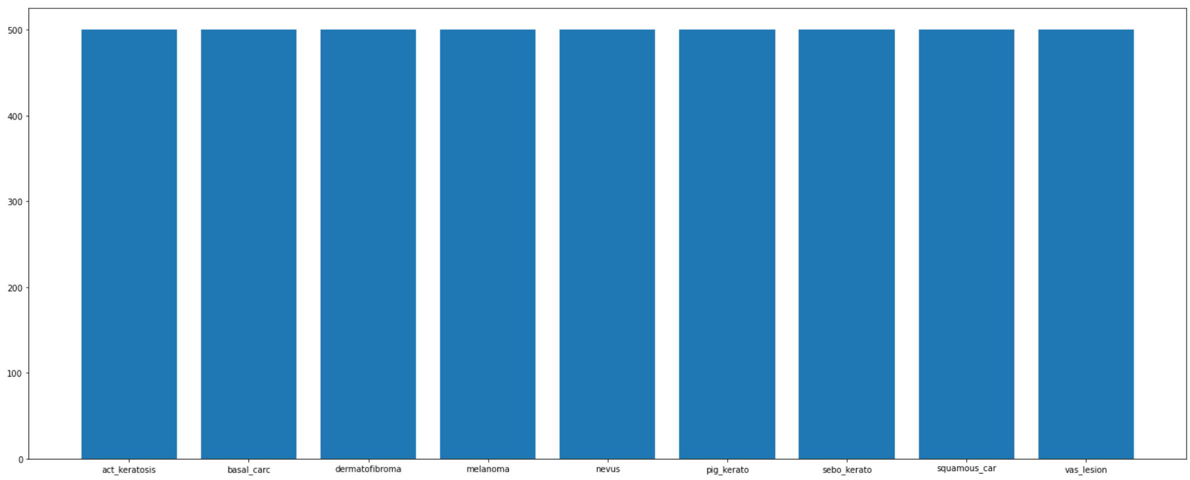
Lets see the distribution of augmented data after adding new images to the original training data.

In [52]:

```
# Check the distribution of data again.
count=[]
for name in class_names:
    count.append(len(list(data_dir.glob(name+'*/output/*.jpg'))))
plt.figure(figsize=(25,10))
plt.bar(class_names,count)
```

Out[52]:

```
<BarContainer object of 9 artists>
```



Lets see the distribution of augmented data after adding new images to the original training data.

In [53]:
```
import os
from glob import glob
```

In [55]:
```
path_list_new = [x for x in glob(os.path.join(data_dir, '*','output', '*.jpg'))]
path_list_new
```

```
\\output\\basal_carc_original_ISIC_0031442.jpg_8a07da38-4adf-4185-b9d8-05ad5da1b62
b.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031450.jpg_29e8dbfc-164b-472b-87b1-ea8d8448b2c
5.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031450.jpg_3e83bbad-6c75-4871-8197-00a9ffe9693
b.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031450.jpg_a4b97b8a-02b1-4fde-bd6d-aebd9e58c35
d.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031470.jpg_5009c4e1-71cb-40dc-83e4-ac53dbcd283
2.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031470.jpg_695091f7-a053-480c-a323-72bd1fa1821
8.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031489.jpg_74b9a0b6-357b-4459-b012-61678929d69
d.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031513.jpg_027af825-64bc-409b-a01a-3709ae7ed1d
5.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031520.jpg_b581c67c-b6f6-4fe9-88b8-2030605113a
6.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031526.jpg_2cd439b7-4683-425b-86a1-3bf466b1c2b
a.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031526.jpg_43c2374d-a532-4b95-aa7e-87228a6bb0b
0.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031531.jpg_5e300e81-75db-47b6-a273-342331f0d5b
5.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031552.jpg_ec99455b-71ab-43a3-981d-e6f01363c1b
5.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031597.jpg_09bb4e39-131e-4705-8630-856ba751d2c
f.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031597.jpg_a5bd89a7-5b82-4524-83f7-1d2374d49d5
2.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031614.jpg_641b24e9-1dea-4dd3-80f2-56041b248c1
6.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031614.jpg_a4eee3c5-761a-4a9b-a019-18e90e437dd
3.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031614.jpg_cfd93c14-dd95-4013-b263-c0bef4d3d27
b.jpg',
 'Skin cancer ISIC The International Skin Imaging Collaboration\\Train\\basal_carc
\\output\\basal_carc_original_ISIC_0031614.jpg_e843140b-e0b2-4e3c-90cd-a89c668bd1d
3.jpg',
 ...]
```

In [58]:
```
lesion_list_new = [os.path.basename(os.path.dirname(os.path.dirname(y))) for y in g
len(lesion_list_new)
```

Out[58]:
```
4500
```

In [60]:
```
dataframe_dict_new = dict(zip(path_list_new,lesion_list_new))
path_list = []
class_name = []
for name in class_names:
    for file in data_dir.glob(name + '/*.jpg'):
        path_list.append(file)
        class_name.append(name)
```

In [61]:
```
dataframe_dict_original=dict(zip(path_list,class_name))
original_df=pd.DataFrame(list(dataframe_dict_original.items()),columns=['Path','Lal
```

In [62]:
```
original_df
```

Out[62]:

	Path	Label
0	Skin cancer ISIC The International Skin Imagin...	act_keratosis
1	Skin cancer ISIC The International Skin Imagin...	act_keratosis
2	Skin cancer ISIC The International Skin Imagin...	act_keratosis
3	Skin cancer ISIC The International Skin Imagin...	act_keratosis
4	Skin cancer ISIC The International Skin Imagin...	act_keratosis
...	...	...
2234	Skin cancer ISIC The International Skin Imagin...	vas_lesion
2235	Skin cancer ISIC The International Skin Imagin...	vas_lesion
2236	Skin cancer ISIC The International Skin Imagin...	vas_lesion
2237	Skin cancer ISIC The International Skin Imagin...	vas_lesion
2238	Skin cancer ISIC The International Skin Imagin...	vas_lesion

2239 rows × 2 columns

In [63]:
```
df2 = pd.DataFrame(list(dataframe_dict_new.items()),columns=['Path','Label'])
```

In [64]:
```
new_df = original_df.append(df2)
```

```
C:\Users\krishnendu\AppData\Local\Temp\ipykernel_9100\1731109560.py:1: FutureWarni
ng: The frame.append method is deprecated and will be removed from pandas in a fut
ure version. Use pandas.concat instead.
  new_df = original_df.append(df2)
```

In [65]:
```
new_df
```

Out[65]:

	Path	Label
0	Skin cancer ISIC The International Skin Imagin...	act_keratosis
1	Skin cancer ISIC The International Skin Imagin...	act_keratosis
2	Skin cancer ISIC The International Skin Imagin...	act_keratosis
3	Skin cancer ISIC The International Skin Imagin...	act_keratosis
4	Skin cancer ISIC The International Skin Imagin...	act_keratosis
...	...	...
4495	Skin cancer ISIC The International Skin Imagin...	vas_lesion
4496	Skin cancer ISIC The International Skin Imagin...	vas_lesion
4497	Skin cancer ISIC The International Skin Imagin...	vas_lesion
4498	Skin cancer ISIC The International Skin Imagin...	vas_lesion
4499	Skin cancer ISIC The International Skin Imagin...	vas_lesion

6739 rows × 2 columns

In [66]:
```
new_df['Label'].value_counts()
```

Out[66]:
```
pig_kerato        962
melanoma          938
basal_carc        876
nevus             857
squamous_car      681
vas_lesion        639
act_keratosis     614
dermatofibroma    595
sebo_kerato       577
Name: Label, dtype: int64
```

So, now we have added 500 images to all the classes to maintain some class balance. We can add more images as we want to improve training process.

Create training set

In [67]:
```
traing_ds = keras.preprocessing.image_dataset_from_directory(data_dir,image_size=(
                                                              validation_split= 0.2
```

```
Found 6739 files belonging to 9 classes.
Using 5392 files for training.
```

Create validation set

In [68]:
```
validation_ds = keras.preprocessing.image_dataset_from_directory(data_dir,image_si
                                                              validation_split= 0.2
```

```
Found 6739 files belonging to 9 classes.
Using 1347 files for validation.
```

Create your model again (make sure to include normalization)

In [73]:
```
## code
AUTOTUNE = tf.data.experimental.AUTOTUNE

traing_ds = traing_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
validation_ds = validation_ds.cache().prefetch(buffer_size = AUTOTUNE)

model = Sequential([
    layers.experimental.preprocessing.Rescaling(1.0/255),
    layers.Conv2D(16,3,padding='same',activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32,3,padding='same',activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64,3,padding='same', activation = 'relu'),
    layers.MaxPool2D(),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(n_class)
])
```

In [74]:
```
### Compile the model
model.compile(optimizer='adam',
             loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True),
             metrics=['accuracy'])
```

In [75]:
```
## fitting the model
epochs = 15

history = model.fit(
  traing_ds,
  validation_data= validation_ds,
  epochs=epochs
)
```

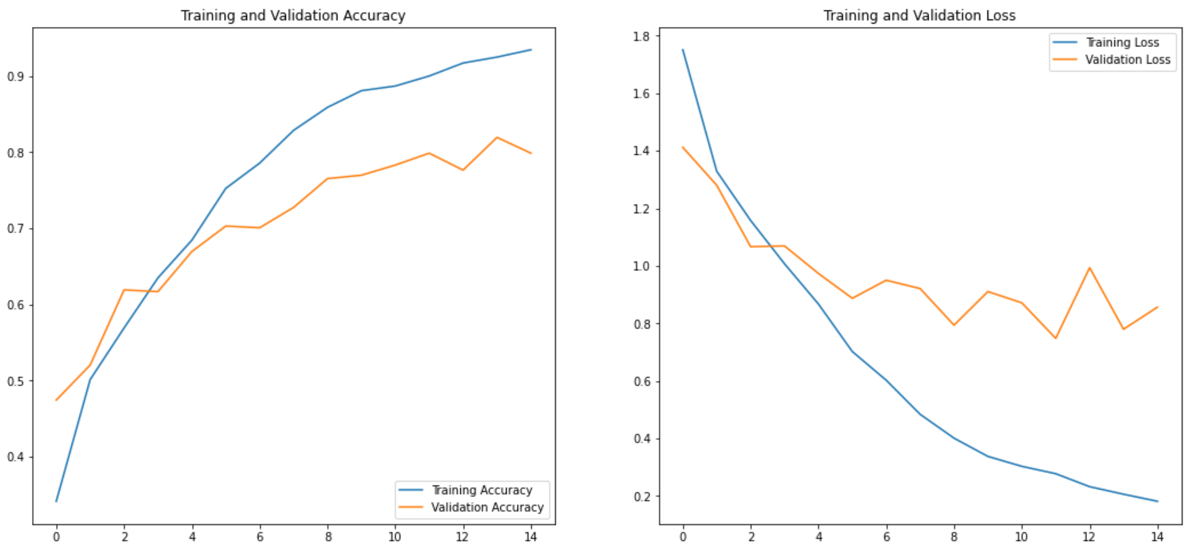
```
Epoch 1/15
169/169 [==============================] - 166s 862ms/step - loss: 1.7510 - accura
cy: 0.3414 - val_loss: 1.4119 - val_accuracy: 0.4744
Epoch 2/15
169/169 [==============================] - 128s 760ms/step - loss: 1.3289 - accura
cy: 0.5013 - val_loss: 1.2801 - val_accuracy: 0.5204
Epoch 3/15
169/169 [==============================] - 119s 704ms/step - loss: 1.1579 - accura
cy: 0.5692 - val_loss: 1.0667 - val_accuracy: 0.6192
Epoch 4/15
169/169 [==============================] - 130s 768ms/step - loss: 1.0073 - accura
cy: 0.6350 - val_loss: 1.0693 - val_accuracy: 0.6169
Epoch 5/15
169/169 [==============================] - 118s 697ms/step - loss: 0.8675 - accura
cy: 0.6845 - val_loss: 0.9737 - val_accuracy: 0.6696
Epoch 6/15
169/169 [==============================] - 130s 770ms/step - loss: 0.7022 - accura
cy: 0.7526 - val_loss: 0.8874 - val_accuracy: 0.7030
Epoch 7/15
169/169 [==============================] - 130s 766ms/step - loss: 0.6026 - accura
cy: 0.7858 - val_loss: 0.9500 - val_accuracy: 0.7008
Epoch 8/15
169/169 [==============================] - 135s 799ms/step - loss: 0.4842 - accura
cy: 0.8288 - val_loss: 0.9211 - val_accuracy: 0.7275
Epoch 9/15
169/169 [==============================] - 128s 757ms/step - loss: 0.4013 - accura
cy: 0.8591 - val_loss: 0.7939 - val_accuracy: 0.7654
Epoch 10/15
169/169 [==============================] - 119s 702ms/step - loss: 0.3378 - accura
cy: 0.8809 - val_loss: 0.9107 - val_accuracy: 0.7699
Epoch 11/15
169/169 [==============================] - 125s 740ms/step - loss: 0.3033 - accura
cy: 0.8871 - val_loss: 0.8715 - val_accuracy: 0.7832
Epoch 12/15
169/169 [==============================] - 123s 726ms/step - loss: 0.2778 - accura
cy: 0.9002 - val_loss: 0.7481 - val_accuracy: 0.7988
Epoch 13/15
169/169 [==============================] - 126s 745ms/step - loss: 0.2326 - accura
cy: 0.9173 - val_loss: 0.9936 - val_accuracy: 0.7765
Epoch 14/15
169/169 [==============================] - 126s 744ms/step - loss: 0.2065 - accura
cy: 0.9251 - val_loss: 0.7799 - val_accuracy: 0.8196
Epoch 15/15
169/169 [==============================] - 132s 779ms/step - loss: 0.1819 - accura
cy: 0.9347 - val_loss: 0.8565 - val_accuracy: 0.7988
```

Visualize the model results

In [77]:

```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(18, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
```

```
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



This is good

Traning Accuracy = 93

Validation Accuracy = 80

thats good

```
In [ ]:
```