Problem statement:

To build a CNN based model which can accurately detect melanoma. Melanoma is a type of cancer that can be deadly if not detected early. It accounts for 75% of skin cancer deaths. A solution that can evaluate images and alert dermatologists about the presence of melanoma has the potential to reduce a lot of manual effort needed in diagnosis.

Dataset:

The dataset consists of 2357 images of malignant and benign oncological diseases, which were formed from the International Skin Imaging Collaboration (ISIC). All images were sorted according to the classification taken with ISIC, and all subsets were divided into the same number of images, with the exception of melanomas and moles, whose images are slightly dominant.

The data set contains the following diseases:

- Actinic keratosis
- Basal cell carcinoma
- Dermatofibroma
- Melanoma
- Nevus
- Pigmented benign keratosis
- Seborrheic keratosis
- Squamous cell carcinoma
- Vascular lesion

In [1]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remoun
t, call drive.mount("/content/drive", force_remount=True).
```

In [2]:

```
import os
root='/content/drive/MyDrive/Upgrad/cnn_assignment'
print(os.getcwd())
os.chdir(root)
print(os.getcwd())
```

```
/content
/content/drive/MyDrive/Upgrad/cnn_assignment
```

In [3]:

```
!pip install Augmentor
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simpl
e,) https://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-
python.pkg.dev/colab-wheels/public/simple/)
Requirement already satisfied: Augmentor in /usr/local/lib/python3.10/
dist-packages (0.2.12)
Requirement already satisfied: tqdm>=4.9.0 in /usr/local/lib/python3.1
0/dist-packages (from Augmentor) (4.65.0)
Requirement already satisfied: numpy>=1.11.0 in /usr/local/lib/python
3.10/dist-packages (from Augmentor) (1.22.4)
Requirement already satisfied: Pillow>=5.2.0 in /usr/local/lib/python
3.10/dist-packages (from Augmentor) (8.4.0)
```

In [4]:

```
# import
import pathlib
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

import PIL
import Augmentor
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
```

In [5]:

```
# set data path
data_dir_train = pathlib.Path("/content/drive/MyDrive/Upgrad/cnn_assignment/Skin car
data_dir_test = pathlib.Path('/content/drive/MyDrive/Upgrad/cnn_assignment/Skin canc
```

In [6]:

```
# Get the count of images
image_count_train = len(list(data_dir_train.glob('*/*.jpg')))
print(f"Total train images : {image_count_train}")
image_count_test = len(list(data_dir_test.glob('*/*.jpg')))
print(f"Total test images  : {image_count_test}")
```

```
Total train images : 2239
Total test images  : 118
```

In [7]:

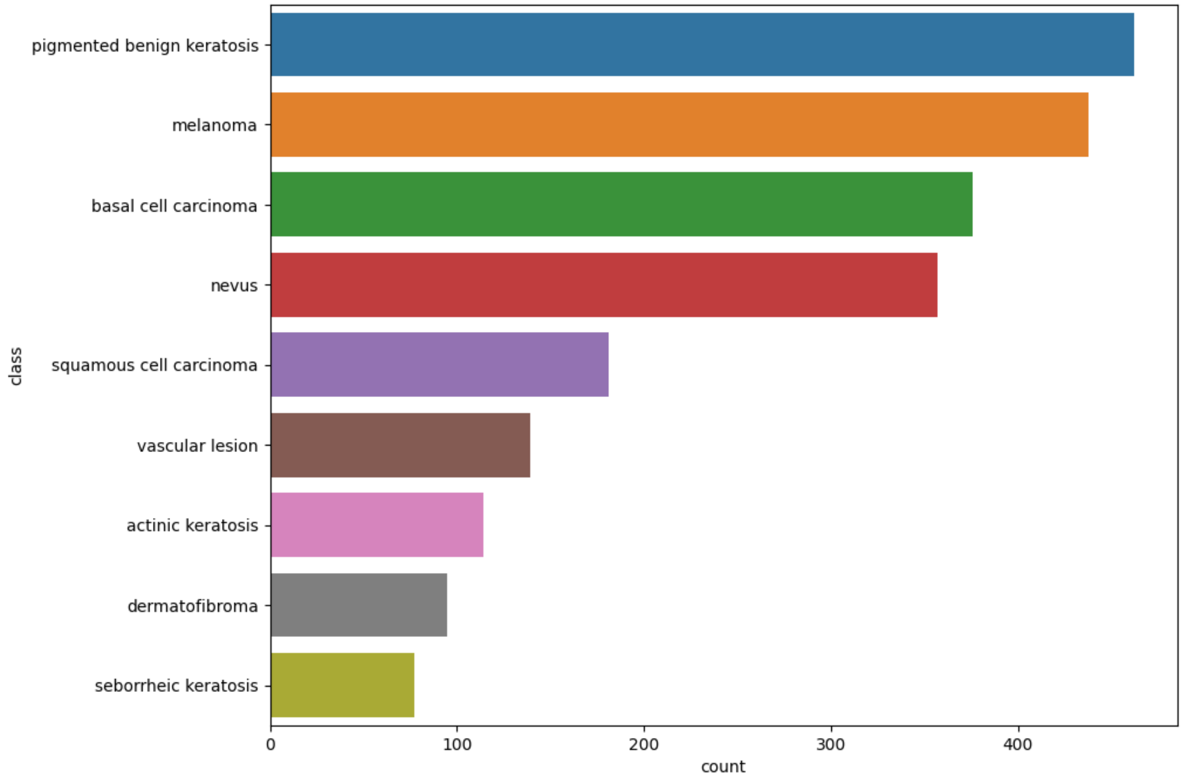
```
# check for class balance
classes=[]
count=[]
pct=[]

for d in os.listdir(data_dir_train):
    pth=pathlib.Path(os.path.join(data_dir_train,d))
    img_count=len(list(pth.glob('*.jpg')))
    pctg=round((img_count/image_count_train)*100,2)
    # print(f"{d}  : {img_count} | {pctg}%")
    classes.append(d)
    count.append(img_count)
    pct.append(pctg)

stats_df=pd.DataFrame(columns=['class','count','percentage'])
stats_df['class']=pd.Series(classes)
stats_df['count']=pd.Series(count)
stats_df['percentage']=pd.Series(pct)
classes=list(stats_df['class'])
stats_df
```

Out[7]:

	class	count	percentage
0	actinic keratosis	114	5.09
1	basal cell carcinoma	376	16.79
2	dermatofibroma	95	4.24
3	melanoma	438	19.56
4	nevus	357	15.94
5	pigmented benign keratosis	462	20.63
6	seborrheic keratosis	77	3.44
7	squamous cell carcinoma	181	8.08
8	vascular lesion	139	6.21

In [8]:

```
# plot the class balance
stats_df=stats_df.sort_values(by='count', ascending=False)
plt.figure(figsize=(10, 8))
sns.barplot(x="count", y="class", data=stats_df, label="class")
plt.show()
```



From the above plot we can understand that there is a class imabalance in the dataset.
seborrheic has only 77 samples (class with least number of samples) where as pigmented benign keratosis has 462 samples (class with highest number of samples)

In [9]:

```
# Get the count of images
image_count_train = len(list(data_dir_train.glob('*/*.jpg')))
print(f"Total train images : {image_count_train}")
image_count_test = len(list(data_dir_test.glob('*/*.jpg')))
print(f"Total test images  : {image_count_test}")
```

```
Total train images : 2239
Total test images  : 118
```

Load the data

In [10]:

```
# creating training and validation data sets

batch_size = 32
img_height = 180
img_width = 180

# Train dataset
train_ds = tf.keras.preprocessing.image_dataset_from_directory(data_dir_train,
                                                               batch_size=batch_size
                                                               image_size=(img_heigh
                                                               label_mode='categoric
                                                               seed=123,
                                                               subset="training",
                                                               validation_split=0.2)


# validation dataset
val_ds =tf.keras.preprocessing.image_dataset_from_directory(data_dir_train,
                                                            batch_size=batch_size,
                                                            image_size=(img_height,i
                                                            label_mode='categorical'
                                                            seed=123,
                                                            subset="validation",
                                                            validation_split=0.2)
```

```
Found 2239 files belonging to 9 classes.
Using 1792 files for training.
Found 2239 files belonging to 9 classes.
Using 447 files for validation.
```

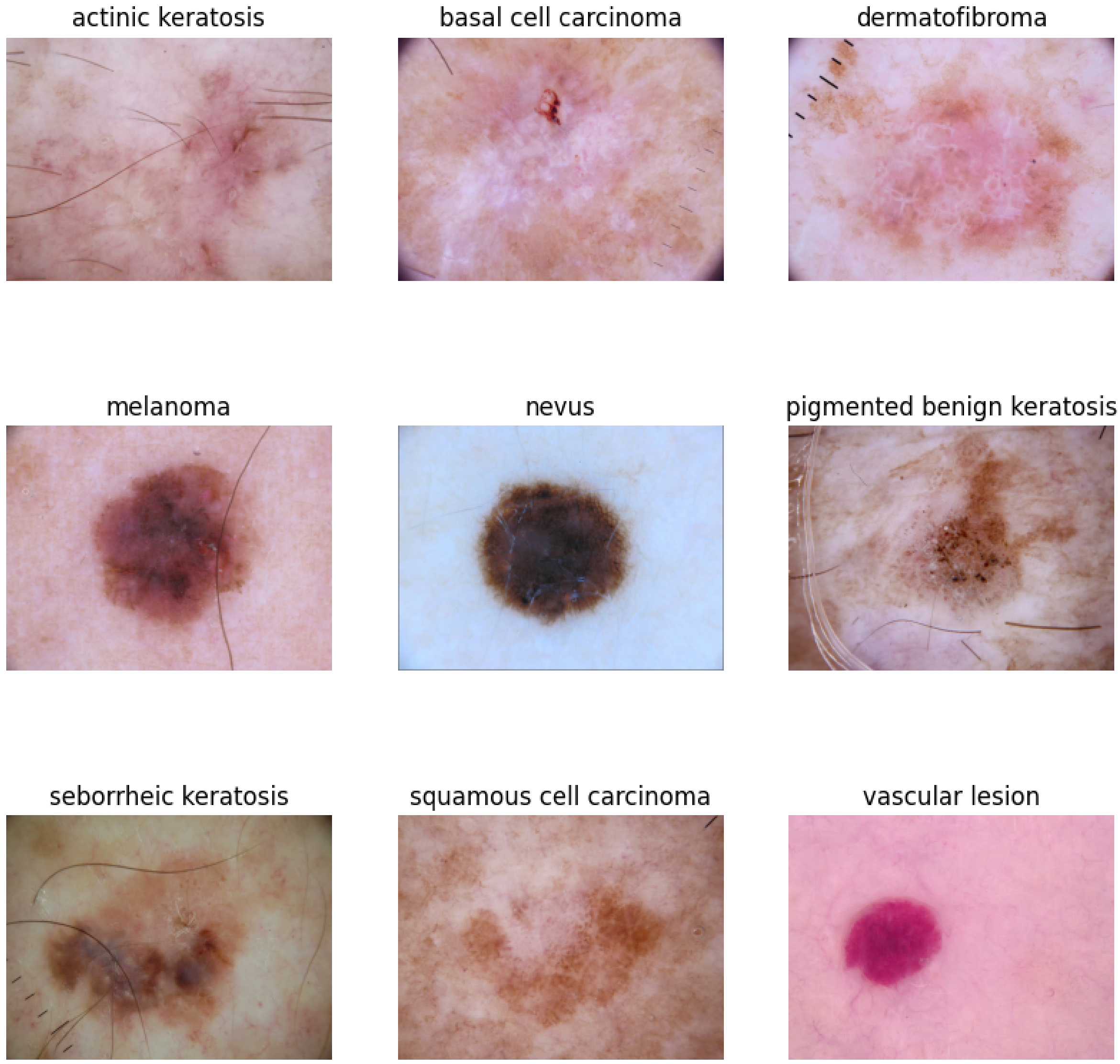
In [11]:

```
class_names = train_ds.class_names
print(class_names)
```

```
['actinic keratosis', 'basal cell carcinoma', 'dermatofibroma', 'melan
oma', 'nevus', 'pigmented benign keratosis', 'seborrheic keratosis',
'squamous cell carcinoma', 'vascular lesion']
```

In [13]:

```
# visualize the data
plt.figure(figsize=(10,10))
for i in range(9):
  plt.subplot(3, 3, i + 1)
  image = plt.imread(str(list(data_dir_train.glob(class_names[i]+'/*.jpg'))[1]))
  plt.title(class_names[i])
  plt.imshow(image)
  plt.axis("off")
```



Create train and test datasets

In [14]:

```
# defines appropriate number of processes that are free for working.
AUTOTUNE = tf.data.experimental.AUTOTUNE
# keeps the images in memory after they're loaded off disk during the first epoch.
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
# overlaps data preprocessing and model execution while training.
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Build the Model

In [15]:

```
model = Sequential()
#Rescaling Layer
model.add(layers.experimental.preprocessing.Rescaling(1./255,input_shape=(180,180,3)

# Conv 1
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
# Maxpool1
model.add(MaxPool2D(pool_size=(2,2)))
# Conv 2
model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
# Maxpool2
model.add(MaxPool2D(pool_size=(2,2)))
# Conv 3
model.add(Conv2D(128,kernel_size=(3,3),activation='relu'))
# Maxpool3
model.add(MaxPool2D(pool_size=(2,2)))
# Dropout layer 50%
# model.add(Dropout(0.5))
# Flatten Layer
model.add(Flatten())
# Dense Layer
model.add(Dense(128,activation='relu'))
# Dropout layer 25%
# model.add(Dropout(0.25))
model.add(layers.Dense(len(class_names),activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling (Rescaling)       (None, 180, 180, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 178, 178, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 89, 89, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 87, 87, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 43, 43, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 41, 41, 128)       73856     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 20, 20, 128)      0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 51200)             0         
                                                                 
 dense (Dense)               (None, 128)               6553728   
                                                                 
 dense_1 (Dense)             (None, 9)                 1161      
                                                                 
=================================================================
Total params: 6,648,137
Trainable params: 6,648,137
Non-trainable params: 0
_________________________________________________________________
```

Train the model

In [16]:

```
#Compile the Model
model.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=["accuracy"])
# checkpoint = ModelCheckpoint("model.h5",monitor="val_accuracy",save_best_only=True

# train the model
epochs = 30
history = model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

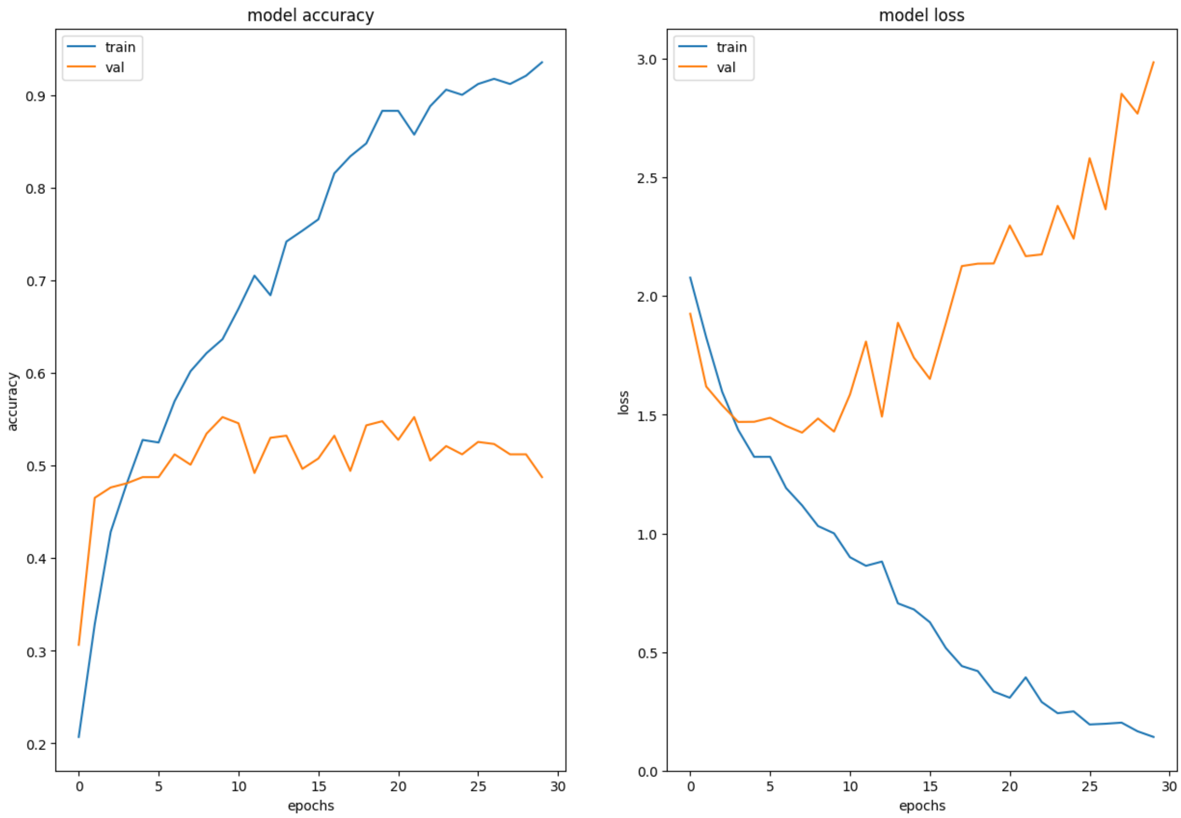
```
Epoch 1/30
56/56 [==============================] - 11s 37ms/step - loss: 2.0779
- accuracy: 0.2070 - val_loss: 1.9257 - val_accuracy: 0.3065
Epoch 2/30
56/56 [==============================] - 1s 11ms/step - loss: 1.8260 -
accuracy: 0.3287 - val_loss: 1.6189 - val_accuracy: 0.4653
Epoch 3/30
56/56 [==============================] - 1s 11ms/step - loss: 1.5963 -
accuracy: 0.4286 - val_loss: 1.5395 - val_accuracy: 0.4765
Epoch 4/30
56/56 [==============================] - 1s 11ms/step - loss: 1.4362 -
accuracy: 0.4805 - val_loss: 1.4697 - val_accuracy: 0.4810
Epoch 5/30
56/56 [==============================] - 1s 11ms/step - loss: 1.3225 -
accuracy: 0.5279 - val_loss: 1.4702 - val_accuracy: 0.4877
Epoch 6/30
56/56 [==============================] - 1s 12ms/step - loss: 1.3227 -
accuracy: 0.5251 - val_loss: 1.4871 - val_accuracy: 0.4877
Epoch 7/30
56/56 [==============================] - 1s 11ms/step - loss: 1.1909 -
accuracy: 0.5698 - val_loss: 1.4528 - val_accuracy: 0.5123
Epoch 8/30
56/56 [==============================] - 1s 11ms/step - loss: 1.1180 -
accuracy: 0.6021 - val_loss: 1.4247 - val_accuracy: 0.5011
Epoch 9/30
56/56 [==============================] - 1s 11ms/step - loss: 1.0309 -
accuracy: 0.6217 - val_loss: 1.4843 - val_accuracy: 0.5347
Epoch 10/30
56/56 [==============================] - 1s 11ms/step - loss: 1.0001 -
accuracy: 0.6367 - val_loss: 1.4290 - val_accuracy: 0.5526
Epoch 11/30
56/56 [==============================] - 1s 11ms/step - loss: 0.8994 -
accuracy: 0.6696 - val_loss: 1.5856 - val_accuracy: 0.5459
Epoch 12/30
56/56 [==============================] - 1s 11ms/step - loss: 0.8632 -
accuracy: 0.7054 - val_loss: 1.8086 - val_accuracy: 0.4922
Epoch 13/30
56/56 [==============================] - 1s 11ms/step - loss: 0.8812 -
accuracy: 0.6842 - val_loss: 1.4921 - val_accuracy: 0.5302
Epoch 14/30
56/56 [==============================] - 1s 11ms/step - loss: 0.7050 -
accuracy: 0.7422 - val_loss: 1.8875 - val_accuracy: 0.5324
Epoch 15/30
56/56 [==============================] - 1s 11ms/step - loss: 0.6793 -
accuracy: 0.7539 - val_loss: 1.7408 - val_accuracy: 0.4966
Epoch 16/30
56/56 [==============================] - 1s 11ms/step - loss: 0.6254 -
accuracy: 0.7662 - val_loss: 1.6510 - val_accuracy: 0.5078
Epoch 17/30
56/56 [==============================] - 1s 11ms/step - loss: 0.5166 -
accuracy: 0.8158 - val_loss: 1.8854 - val_accuracy: 0.5324
Epoch 18/30
56/56 [==============================] - 1s 11ms/step - loss: 0.4407 -
accuracy: 0.8343 - val_loss: 2.1265 - val_accuracy: 0.4944
Epoch 19/30
56/56 [==============================] - 1s 11ms/step - loss: 0.4194 -
accuracy: 0.8482 - val_loss: 2.1368 - val_accuracy: 0.5436
Epoch 20/30
56/56 [==============================] - 1s 11ms/step - loss: 0.3330 -
accuracy: 0.8834 - val_loss: 2.1377 - val_accuracy: 0.5481
Epoch 21/30
```

```
56/56 [==============================] - 1s 11ms/step - loss: 0.3072 -
accuracy: 0.8834 - val_loss: 2.2974 - val_accuracy: 0.5280
Epoch 22/30
56/56 [==============================] - 1s 11ms/step - loss: 0.3932 -
accuracy: 0.8577 - val_loss: 2.1684 - val_accuracy: 0.5526
Epoch 23/30
56/56 [==============================] - 1s 11ms/step - loss: 0.2892 -
accuracy: 0.8884 - val_loss: 2.1758 - val_accuracy: 0.5056
Epoch 24/30
56/56 [==============================] - 1s 11ms/step - loss: 0.2420 -
accuracy: 0.9062 - val_loss: 2.3806 - val_accuracy: 0.5213
Epoch 25/30
56/56 [==============================] - 1s 11ms/step - loss: 0.2497 -
accuracy: 0.9007 - val_loss: 2.2423 - val_accuracy: 0.5123
Epoch 26/30
56/56 [==============================] - 1s 11ms/step - loss: 0.1944 -
accuracy: 0.9124 - val_loss: 2.5813 - val_accuracy: 0.5257
Epoch 27/30
56/56 [==============================] - 1s 11ms/step - loss: 0.1976 -
accuracy: 0.9180 - val_loss: 2.3659 - val_accuracy: 0.5235
Epoch 28/30
56/56 [==============================] - 1s 11ms/step - loss: 0.2021 -
accuracy: 0.9124 - val_loss: 2.8529 - val_accuracy: 0.5123
Epoch 29/30
56/56 [==============================] - 1s 11ms/step - loss: 0.1657 -
accuracy: 0.9213 - val_loss: 2.7693 - val_accuracy: 0.5123
Epoch 30/30
56/56 [==============================] - 1s 11ms/step - loss: 0.1420 -
accuracy: 0.9358 - val_loss: 2.9856 - val_accuracy: 0.4877
```

Since it is a multi class classification problem we are using categorical cross entropy as the loss function. Also, using Adam as an optimizer.

In [17]:

```
# Plot the training curves
plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)

#Plot Model Accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'val'], loc='upper left')

#Plot Model Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



The train accuracy and test accuracy started having huge difference after 5 epochs. This could be a possible case of overfitting. Similarly the train loss and val loss shoe huge differnces after 10 epochs

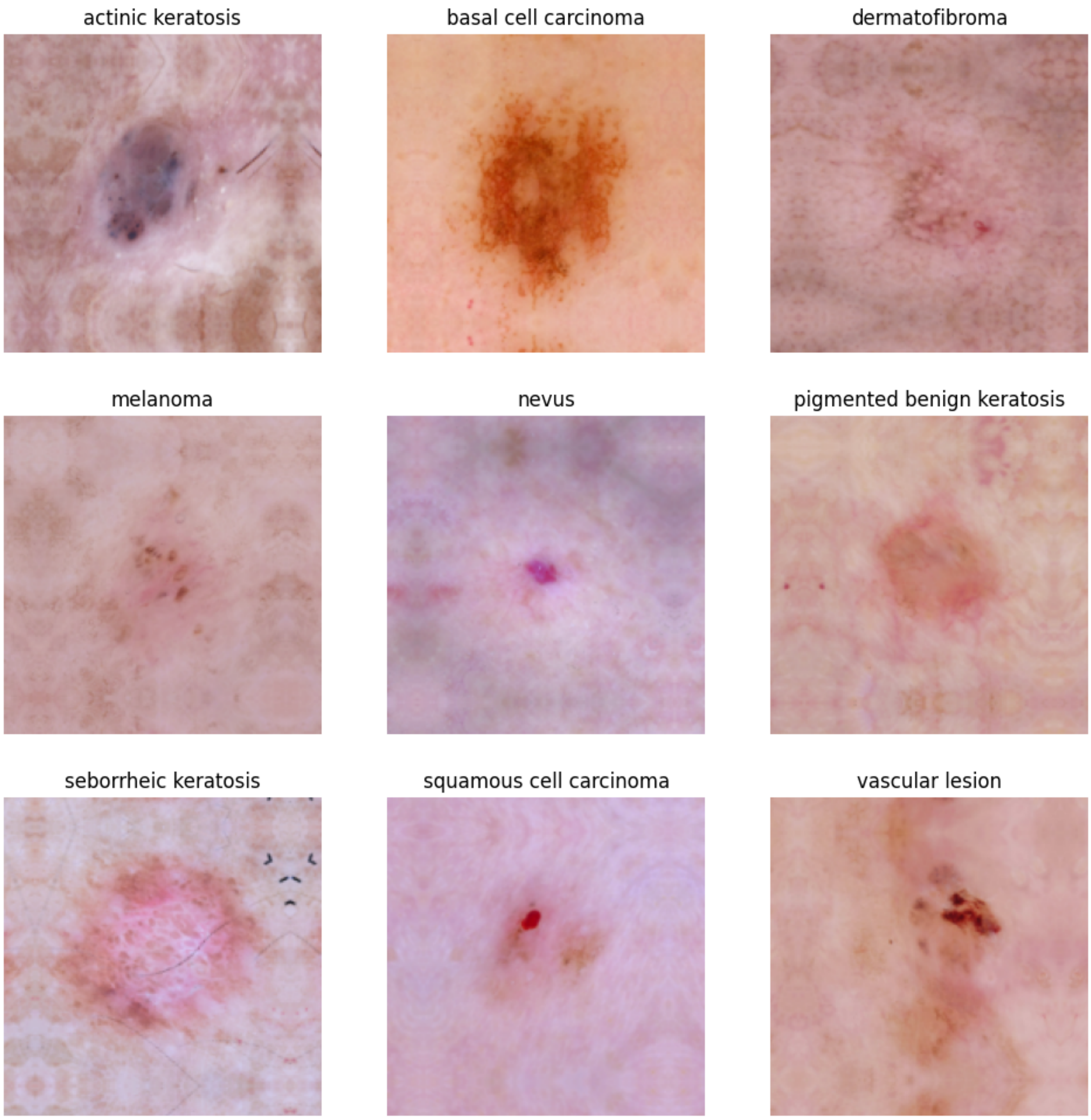
Let's try to fit a better model by adding data augmentation in data preprocessing

In [18]:

```
# using data augmentation
data_augument = keras.Sequential([
                             layers.experimental.preprocessing.RandomFlip(mode="hori
                             layers.experimental.preprocessing.RandomRotation(0.2, f
                             layers.experimental.preprocessing.RandomZoom(height_fac
])
```

In [19]:

```
# visualize the augmented data
plt.figure(figsize=(12, 12))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(data_augument(images)[i].numpy().astype("uint8"))
        plt.title(class_names[i])
        plt.axis("off")
```



In [20]:

```
# build the cnn architecture
model = Sequential([ data_augument,
                        layers.experimental.preprocessing.Rescaling(1./255, input_sh
                    ])

# Conv 1
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
# Maxpool1
model.add(MaxPool2D(pool_size=(2,2)))
# Conv 2
model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
# Maxpool2
model.add(MaxPool2D(pool_size=(2,2)))
# Conv 3
model.add(Conv2D(128,kernel_size=(3,3),activation='relu'))
# Maxpool3
model.add(MaxPool2D(pool_size=(2,2)))
# Dropout layer 50%
# model.add(Dropout(0.5))
# Flatten Layer
model.add(Flatten())
# Dense Layer
model.add(Dense(128,activation='relu'))
# Dropout layer 25%
# model.add(Dropout(0.25))
model.add(layers.Dense(len(class_names),activation='softmax'))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d_3 (Conv2D)           (None, 178, 178, 32)      896       
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 89, 89, 32)       0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 87, 87, 64)        18496     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 43, 43, 64)       0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 41, 41, 128)       73856     
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 20, 20, 128)      0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 51200)             0         
                                                                 
 dense_2 (Dense)             (None, 128)               6553728   
                                                                 
 dense_3 (Dense)             (None, 9)                 1161      
                                                                 
=================================================================
Total params: 6,648,137
Trainable params: 6,648,137
Non-trainable params: 0
_________________________________________________________________
```

In [21]:

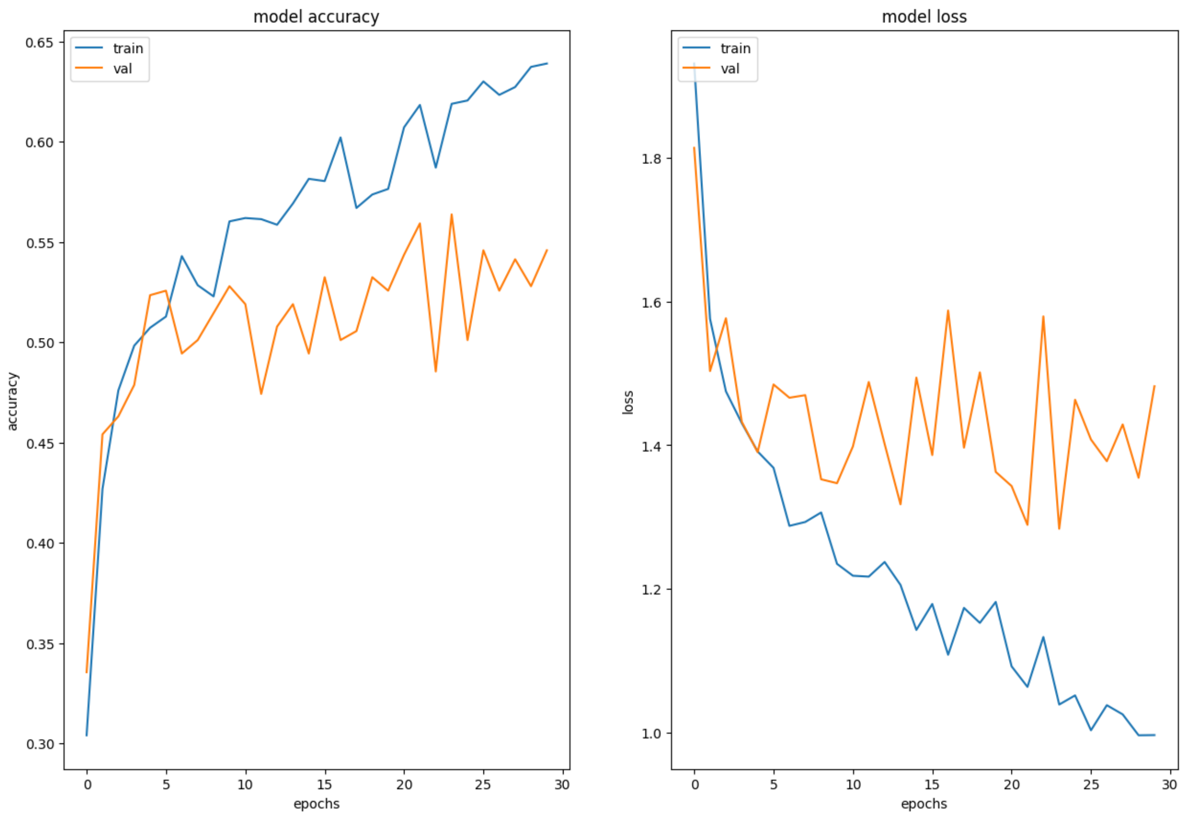
```
# compile the model
model.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=["accuracy"])
checkpoint = ModelCheckpoint("model.h5",monitor="val_accuracy",save_best_only=True,m

# train the model
epochs = 30
history = model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

```
Epoch 1/30
56/56 [==============================] - 3s 15ms/step - loss: 1.9314 -
accuracy: 0.3041 - val_loss: 1.8140 - val_accuracy: 0.3356
Epoch 2/30
56/56 [==============================] - 1s 13ms/step - loss: 1.5759 -
accuracy: 0.4269 - val_loss: 1.5032 - val_accuracy: 0.4541
Epoch 3/30
56/56 [==============================] - 1s 12ms/step - loss: 1.4750 -
accuracy: 0.4760 - val_loss: 1.5768 - val_accuracy: 0.4631
Epoch 4/30
56/56 [==============================] - 1s 12ms/step - loss: 1.4307 -
accuracy: 0.4983 - val_loss: 1.4326 - val_accuracy: 0.4787
Epoch 5/30
56/56 [==============================] - 1s 12ms/step - loss: 1.3912 -
accuracy: 0.5073 - val_loss: 1.3903 - val_accuracy: 0.5235
Epoch 6/30
56/56 [==============================] - 1s 12ms/step - loss: 1.3684 -
accuracy: 0.5128 - val_loss: 1.4846 - val_accuracy: 0.5257
Epoch 7/30
56/56 [==============================] - 1s 12ms/step - loss: 1.2877 -
accuracy: 0.5430 - val_loss: 1.4661 - val_accuracy: 0.4944
Epoch 8/30
56/56 [==============================] - 1s 12ms/step - loss: 1.2931 -
accuracy: 0.5285 - val_loss: 1.4697 - val_accuracy: 0.5011
Epoch 9/30
56/56 [==============================] - 1s 12ms/step - loss: 1.3063 -
accuracy: 0.5229 - val_loss: 1.3525 - val_accuracy: 0.5145
Epoch 10/30
56/56 [==============================] - 1s 12ms/step - loss: 1.2348 -
accuracy: 0.5603 - val_loss: 1.3471 - val_accuracy: 0.5280
Epoch 11/30
56/56 [==============================] - 1s 12ms/step - loss: 1.2183 -
accuracy: 0.5619 - val_loss: 1.3982 - val_accuracy: 0.5190
Epoch 12/30
56/56 [==============================] - 1s 13ms/step - loss: 1.2170 -
accuracy: 0.5614 - val_loss: 1.4880 - val_accuracy: 0.4743
Epoch 13/30
56/56 [==============================] - 1s 13ms/step - loss: 1.2374 -
accuracy: 0.5586 - val_loss: 1.4016 - val_accuracy: 0.5078
Epoch 14/30
56/56 [==============================] - 1s 13ms/step - loss: 1.2054 -
accuracy: 0.5692 - val_loss: 1.3177 - val_accuracy: 0.5190
Epoch 15/30
56/56 [==============================] - 1s 12ms/step - loss: 1.1428 -
accuracy: 0.5815 - val_loss: 1.4941 - val_accuracy: 0.4944
Epoch 16/30
56/56 [==============================] - 1s 12ms/step - loss: 1.1789 -
accuracy: 0.5804 - val_loss: 1.3863 - val_accuracy: 0.5324
Epoch 17/30
56/56 [==============================] - 1s 12ms/step - loss: 1.1082 -
accuracy: 0.6021 - val_loss: 1.5875 - val_accuracy: 0.5011
Epoch 18/30
56/56 [==============================] - 1s 12ms/step - loss: 1.1734 -
accuracy: 0.5670 - val_loss: 1.3966 - val_accuracy: 0.5056
Epoch 19/30
56/56 [==============================] - 1s 13ms/step - loss: 1.1526 -
accuracy: 0.5737 - val_loss: 1.5015 - val_accuracy: 0.5324
Epoch 20/30
56/56 [==============================] - 1s 12ms/step - loss: 1.1818 -
accuracy: 0.5765 - val_loss: 1.3628 - val_accuracy: 0.5257
Epoch 21/30
```

```
56/56 [==============================] - 1s 12ms/step - loss: 1.0921 -
accuracy: 0.6071 - val_loss: 1.3431 - val_accuracy: 0.5436
Epoch 22/30
56/56 [==============================] - 1s 12ms/step - loss: 1.0635 -
accuracy: 0.6183 - val_loss: 1.2891 - val_accuracy: 0.5593
Epoch 23/30
56/56 [==============================] - 1s 12ms/step - loss: 1.1330 -
accuracy: 0.5871 - val_loss: 1.5794 - val_accuracy: 0.4855
Epoch 24/30
56/56 [==============================] - 1s 12ms/step - loss: 1.0389 -
accuracy: 0.6189 - val_loss: 1.2837 - val_accuracy: 0.5638
Epoch 25/30
56/56 [==============================] - 1s 12ms/step - loss: 1.0516 -
accuracy: 0.6205 - val_loss: 1.4633 - val_accuracy: 0.5011
Epoch 26/30
56/56 [==============================] - 1s 12ms/step - loss: 1.0031 -
accuracy: 0.6300 - val_loss: 1.4079 - val_accuracy: 0.5459
Epoch 27/30
56/56 [==============================] - 1s 12ms/step - loss: 1.0380 -
accuracy: 0.6233 - val_loss: 1.3777 - val_accuracy: 0.5257
Epoch 28/30
56/56 [==============================] - 1s 12ms/step - loss: 1.0252 -
accuracy: 0.6272 - val_loss: 1.4289 - val_accuracy: 0.5414
Epoch 29/30
56/56 [==============================] - 1s 12ms/step - loss: 0.9961 -
accuracy: 0.6373 - val_loss: 1.3546 - val_accuracy: 0.5280
Epoch 30/30
56/56 [==============================] - 1s 12ms/step - loss: 0.9964 -
accuracy: 0.6390 - val_loss: 1.4821 - val_accuracy: 0.5459
```

In [22]:

```
# Plot the training curves
plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)

#Plot Model Accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'val'], loc='upper left')

#Plot Model Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



Still there is huge difference in the train accuracy, val accuracy. Similarly in train loss and val loss. Again there seems to be a overfitting condition

Let's try

1. Augmentor to increase the total samples from all the classes
2. drop out layers to reduce overfitting

Augmentor to increase the sample size and handle class imbalance

In [23]:

```python
# Augment more images for each class
for i in class_names:
    pth=str(data_dir_train)+"/"+i
    p = Augmentor.Pipeline(pth)
    p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
    p.sample(500)
```

```
Initialised with 114 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/S
kin cancer ISIC The International Skin Imaging Collaboration/Train/act
inic keratosis/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FA1DA99A
6B0>: 100%|██████████| 500/500 [00:04<00:00, 123.62 Samples/s]

Initialised with 376 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/S
kin cancer ISIC The International Skin Imaging Collaboration/Train/bas
al cell carcinoma/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FA1DAA40
820>: 100%|██████████| 500/500 [00:04<00:00, 116.95 Samples/s]

Initialised with 95 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/S
kin cancer ISIC The International Skin Imaging Collaboration/Train/der
matofibroma/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FA1901AD
720>: 100%|██████████| 500/500 [00:04<00:00, 122.75 Samples/s]

Initialised with 438 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/S
kin cancer ISIC The International Skin Imaging Collaboration/Train/mel
anoma/output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=3072
x2304 at 0x7FA11A5E5C90>: 100%|██████████| 500/500 [00:17<00:00, 28.43
Samples/s]

Initialised with 357 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/S
kin cancer ISIC The International Skin Imaging Collaboration/Train/nev
us/output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1008
x711 at 0x7FA1DA96E7A0>: 100%|██████████| 500/500 [00:15<00:00, 33.15
Samples/s]

Initialised with 462 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/S
kin cancer ISIC The International Skin Imaging Collaboration/Train/pig
mented benign keratosis/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FA190150
6D0>: 100%|██████████| 500/500 [00:03<00:00, 125.56 Samples/s]
```

```
Initialised with 77 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/Skin cancer ISIC The International Skin Imaging Collaboration/Train/seborrheic keratosis/output.

Processing <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=1024x768 at 0x7FA11A5E7D60>: 100%|██████████| 500/500 [00:07<00:00, 65.26 Samples/s]

Initialised with 181 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/Skin cancer ISIC The International Skin Imaging Collaboration/Train/squamous cell carcinoma/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FA190150D90>: 100%|██████████| 500/500 [00:04<00:00, 120.47 Samples/s]

Initialised with 139 image(s) found.
Output directory set to /content/drive/MyDrive/Upgrad/cnn_assignment/Skin cancer ISIC The International Skin Imaging Collaboration/Train/vascular lesion/output.

Processing <PIL.Image.Image image mode=RGB size=600x450 at 0x7FA1C805A620>: 100%|██████████| 500/500 [00:04<00:00, 122.30 Samples/s]
```

In [24]:

```
# data_dir_train1 = pathlib.Path("/content/drive/MyDrive/Upgrad/cnn_assignment/Skin
image_count_train = len(list(data_dir_train.glob('*/output/*.jpg')))
print(image_count_train)
```

```
4500
```

In [25]:

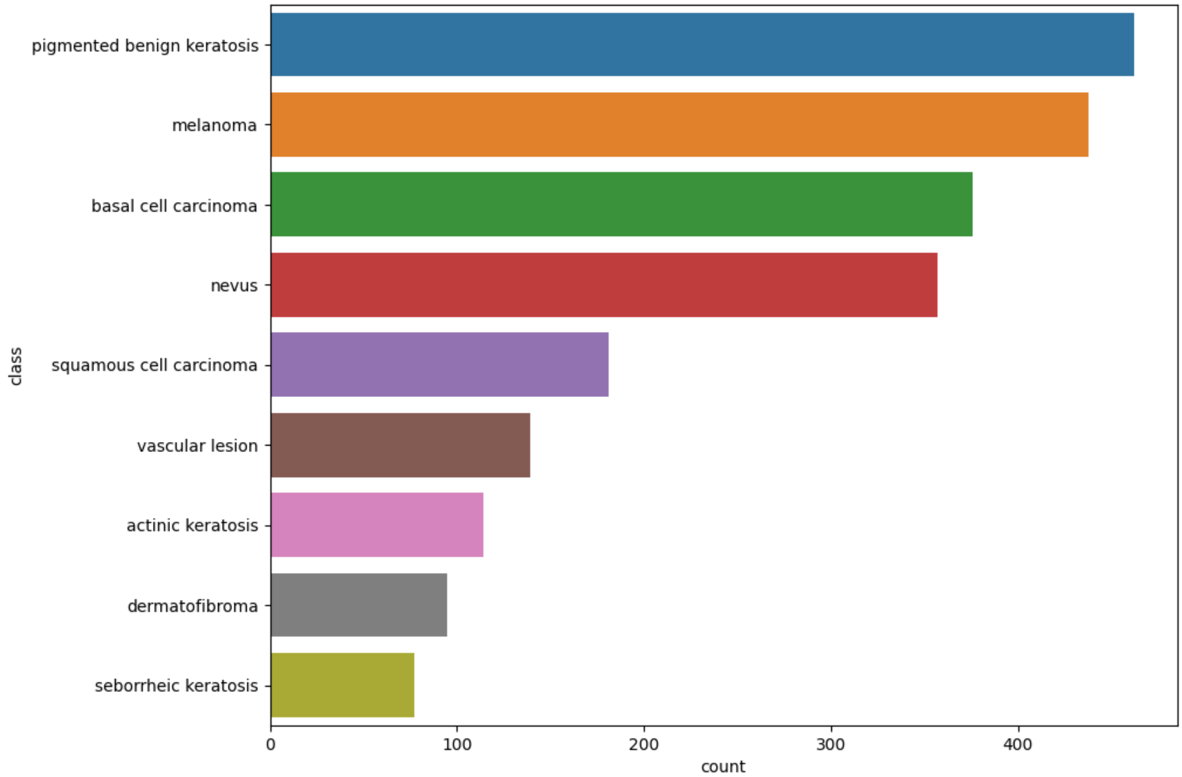
```
# check for class balance
classes=[]
count=[]
pct=[]

for d in os.listdir(data_dir_train):
    pth=pathlib.Path(os.path.join(data_dir_train,d))
    img_count=len(list(pth.glob('*.jpg')))
    pctg=round((img_count/image_count_train)*100,2)
    # print(f"{d}  : {img_count} | {pctg}%")
    classes.append(d)
    count.append(img_count)
    pct.append(pctg)

stats_df=pd.DataFrame(columns=['class','count','percentage'])
stats_df['class']=pd.Series(classes)
stats_df['count']=pd.Series(count)
stats_df['percentage']=pd.Series(pct)
classes=list(stats_df['class'])
stats_df
```

Out[25]:

	class	count	percentage
0	actinic keratosis	114	2.53
1	basal cell carcinoma	376	8.36
2	dermatofibroma	95	2.11
3	melanoma	438	9.73
4	nevus	357	7.93
5	pigmented benign keratosis	462	10.27
6	seborrheic keratosis	77	1.71
7	squamous cell carcinoma	181	4.02
8	vascular lesion	139	3.09

In [26]:

```
# plot the class balance
stats_df=stats_df.sort_values(by='count', ascending=False)
plt.figure(figsize=(10, 8))
sns.barplot(x="count", y="class", data=stats_df, label="class")
plt.show()
```



In [27]:

```
# train data set
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123,
  validation_split = 0.2,
  subset = "training",
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

In [28]:

```
# val dataset
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir_train,
  seed=123,
  validation_split = 0.2,
  subset = 'validation',
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

In [32]:

```
# build the cnn architecture
model = Sequential([ data_augument,
                        layers.experimental.preprocessing.Rescaling(1./255, input_sh
                    ])

# Conv 1
model.add(Conv2D(32,kernel_size=(3,3),activation='relu'))
# Maxpool1
model.add(MaxPool2D(pool_size=(2,2)))
# Conv 2
model.add(Conv2D(64,kernel_size=(3,3),activation='relu'))
# Maxpool2
model.add(MaxPool2D(pool_size=(2,2)))
# Conv 3
model.add(Conv2D(128,kernel_size=(3,3),activation='relu'))
# Maxpool3
model.add(MaxPool2D(pool_size=(2,2)))
# Dropout layer 50%
model.add(Dropout(0.5))
# Flatten Layer
model.add(Flatten())
# Dense Layer
model.add(Dense(128,activation='relu'))
# Dropout layer 25%
model.add(Dropout(0.25))
model.add(layers.Dense(len(class_names),activation='softmax'))
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential_1 (Sequential)   (None, 180, 180, 3)       0         
                                                                 
 rescaling_3 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d_12 (Conv2D)          (None, 178, 178, 32)      896       
                                                                 
 max_pooling2d_9 (MaxPooling  (None, 89, 89, 32)       0         
 2D)                                                             
                                                                 
 conv2d_13 (Conv2D)          (None, 87, 87, 64)        18496     
                                                                 
 max_pooling2d_10 (MaxPoolin  (None, 43, 43, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_14 (Conv2D)          (None, 41, 41, 128)       73856     
                                                                 
 max_pooling2d_11 (MaxPoolin  (None, 20, 20, 128)      0         
 g2D)                                                            
                                                                 
 dropout_2 (Dropout)         (None, 20, 20, 128)       0         
                                                                 
 flatten_3 (Flatten)         (None, 51200)             0         
                                                                 
 dense_6 (Dense)             (None, 128)               6553728   
                                                                 
 dropout_3 (Dropout)         (None, 128)               0         
                                                                 
 dense_7 (Dense)             (None, 9)                 1161      
                                                                 
=================================================================
Total params: 6,648,137
Trainable params: 6,648,137
Non-trainable params: 0
_________________________________________________________________
```

In [33]:

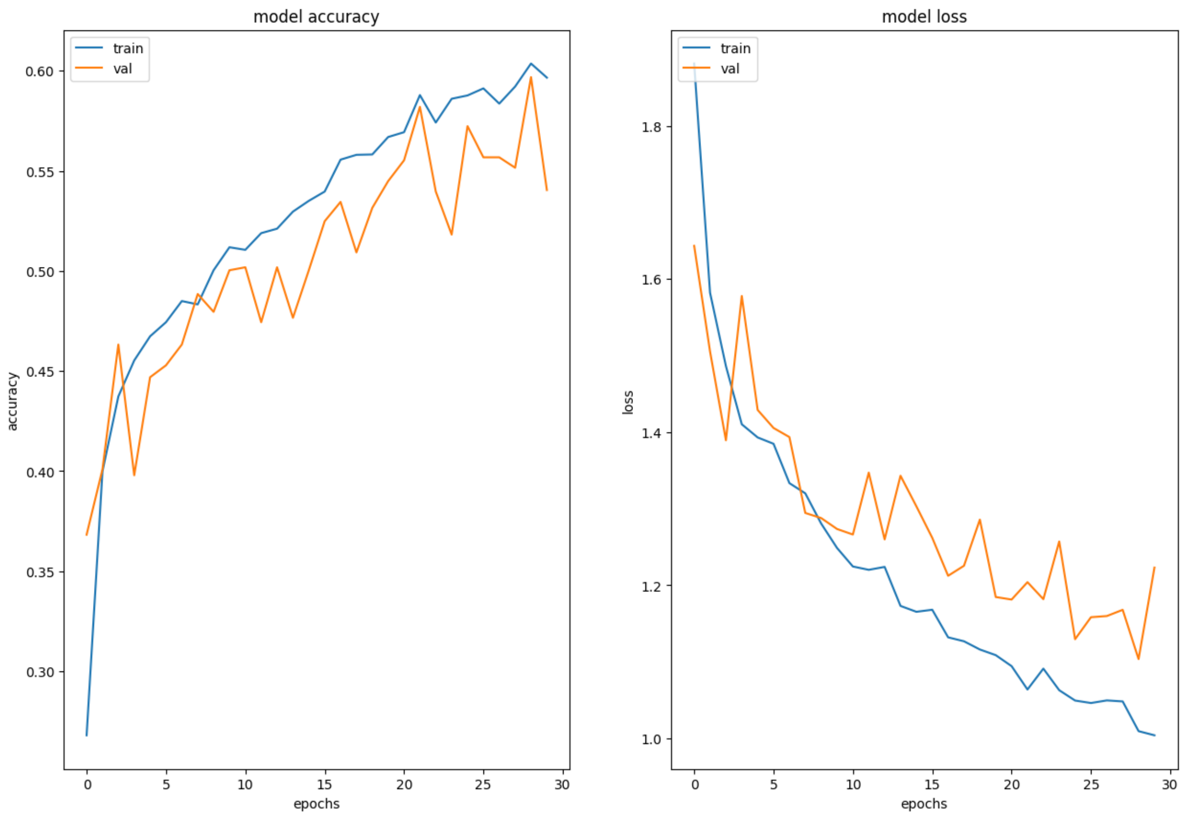
```
# Compile the Model
model.compile(optimizer="Adam",loss="sparse_categorical_crossentropy",metrics=["accu
checkpoint = ModelCheckpoint("model.h5",monitor="val_accuracy",save_best_only=True,m

# train the model
epochs = 30
history = model.fit(train_ds,validation_data=val_ds,epochs=epochs)
```

```
Epoch 1/30
169/169 [==============================] - 14s 64ms/step - loss: 1.881
2 - accuracy: 0.2680 - val_loss: 1.6432 - val_accuracy: 0.3682
Epoch 2/30
169/169 [==============================] - 11s 63ms/step - loss: 1.582
0 - accuracy: 0.3999 - val_loss: 1.5052 - val_accuracy: 0.4009
Epoch 3/30
169/169 [==============================] - 11s 62ms/step - loss: 1.486
2 - accuracy: 0.4373 - val_loss: 1.3893 - val_accuracy: 0.4633
Epoch 4/30
169/169 [==============================] - 11s 62ms/step - loss: 1.410
2 - accuracy: 0.4553 - val_loss: 1.5777 - val_accuracy: 0.3979
Epoch 5/30
169/169 [==============================] - 11s 62ms/step - loss: 1.393
0 - accuracy: 0.4674 - val_loss: 1.4289 - val_accuracy: 0.4469
Epoch 6/30
169/169 [==============================] - 11s 63ms/step - loss: 1.384
8 - accuracy: 0.4744 - val_loss: 1.4054 - val_accuracy: 0.4529
Epoch 7/30
169/169 [==============================] - 11s 62ms/step - loss: 1.333
4 - accuracy: 0.4850 - val_loss: 1.3935 - val_accuracy: 0.4633
Epoch 8/30
169/169 [==============================] - 11s 63ms/step - loss: 1.320
1 - accuracy: 0.4833 - val_loss: 1.2945 - val_accuracy: 0.4885
Epoch 9/30
169/169 [==============================] - 11s 61ms/step - loss: 1.280
7 - accuracy: 0.5004 - val_loss: 1.2878 - val_accuracy: 0.4796
Epoch 10/30
169/169 [==============================] - 11s 62ms/step - loss: 1.248
8 - accuracy: 0.5119 - val_loss: 1.2735 - val_accuracy: 0.5004
Epoch 11/30
169/169 [==============================] - 11s 62ms/step - loss: 1.224
6 - accuracy: 0.5106 - val_loss: 1.2663 - val_accuracy: 0.5019
Epoch 12/30
169/169 [==============================] - 11s 62ms/step - loss: 1.220
2 - accuracy: 0.5189 - val_loss: 1.3471 - val_accuracy: 0.4744
Epoch 13/30
169/169 [==============================] - 11s 62ms/step - loss: 1.224
0 - accuracy: 0.5211 - val_loss: 1.2599 - val_accuracy: 0.5019
Epoch 14/30
169/169 [==============================] - 11s 63ms/step - loss: 1.173
1 - accuracy: 0.5297 - val_loss: 1.3430 - val_accuracy: 0.4766
Epoch 15/30
169/169 [==============================] - 11s 62ms/step - loss: 1.165
4 - accuracy: 0.5351 - val_loss: 1.3031 - val_accuracy: 0.5004
Epoch 16/30
169/169 [==============================] - 11s 61ms/step - loss: 1.168
1 - accuracy: 0.5397 - val_loss: 1.2620 - val_accuracy: 0.5249
Epoch 17/30
169/169 [==============================] - 11s 63ms/step - loss: 1.132
1 - accuracy: 0.5556 - val_loss: 1.2124 - val_accuracy: 0.5345
Epoch 18/30
169/169 [==============================] - 11s 62ms/step - loss: 1.126
9 - accuracy: 0.5580 - val_loss: 1.2255 - val_accuracy: 0.5093
Epoch 19/30
169/169 [==============================] - 11s 63ms/step - loss: 1.116
2 - accuracy: 0.5582 - val_loss: 1.2858 - val_accuracy: 0.5316
Epoch 20/30
169/169 [==============================] - 11s 62ms/step - loss: 1.108
8 - accuracy: 0.5670 - val_loss: 1.1847 - val_accuracy: 0.5449
Epoch 21/30
```

```
169/169 [==============================] - 11s 63ms/step - loss: 1.094
5 - accuracy: 0.5694 - val_loss: 1.1814 - val_accuracy: 0.5553
Epoch 22/30
169/169 [==============================] - 11s 62ms/step - loss: 1.064
0 - accuracy: 0.5879 - val_loss: 1.2041 - val_accuracy: 0.5820
Epoch 23/30
169/169 [==============================] - 11s 62ms/step - loss: 1.091
2 - accuracy: 0.5742 - val_loss: 1.1819 - val_accuracy: 0.5397
Epoch 24/30
169/169 [==============================] - 11s 65ms/step - loss: 1.063
0 - accuracy: 0.5861 - val_loss: 1.2573 - val_accuracy: 0.5182
Epoch 25/30
169/169 [==============================] - 11s 65ms/step - loss: 1.049
6 - accuracy: 0.5877 - val_loss: 1.1296 - val_accuracy: 0.5724
Epoch 26/30
169/169 [==============================] - 11s 64ms/step - loss: 1.046
3 - accuracy: 0.5912 - val_loss: 1.1584 - val_accuracy: 0.5568
Epoch 27/30
169/169 [==============================] - 12s 66ms/step - loss: 1.049
7 - accuracy: 0.5836 - val_loss: 1.1599 - val_accuracy: 0.5568
Epoch 28/30
169/169 [==============================] - 11s 64ms/step - loss: 1.048
4 - accuracy: 0.5922 - val_loss: 1.1680 - val_accuracy: 0.5516
Epoch 29/30
169/169 [==============================] - 12s 66ms/step - loss: 1.009
6 - accuracy: 0.6037 - val_loss: 1.1037 - val_accuracy: 0.5969
Epoch 30/30
169/169 [==============================] - 11s 64ms/step - loss: 1.004
2 - accuracy: 0.5966 - val_loss: 1.2232 - val_accuracy: 0.5405
```

In [34]:

```
# Plot the training curves
plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)

#Plot Model Accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'val'], loc='upper left')

#Plot Model Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



After adding more images with Augmentor, dense layers the train accuracy, test accuracy are closer. There is an improvement in accuracy as well as reduction in overfitting

Conclusion: When the CNN was trained on the images the accuracy of the train phase was low. At the same time there was huge difference in train and val accuracy. After implementing transformations, Augmentor and dropout the train accuracy improved and also the overfitting is reduced.

In [ ]:

```

```