Skin Diseases Classification using CNN

In [1]:
```
#import data by kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

In [2]:
```
!kaggle datasets download -d ismailpromus/skin-diseases-image-datas
```

```
Warning: Your Kaggle API key is readable by other users on this sy
stem! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.jso
n'
Downloading skin-diseases-image-dataset.zip to /content
100% 5.19G/5.19G [04:09<00:00, 25.1MB/s]
100% 5.19G/5.19G [04:09<00:00, 22.4MB/s]
```

In [3]:
```
#file unzip
import zipfile
zip_ref = zipfile.ZipFile('/content/skin-diseases-image-dataset.zip
zip_ref.extractall('/content')
zip_ref.close()
```

In [6]:
```
import splitfolders
import os

os.makedirs('output')
os.makedirs('output/train')
os.makedirs('output/val')
os.makedirs('output/test')

loc = "/content/IMG_CLASSES/"

splitfolders.ratio(loc,output = "output",seed = 42,ratio = (0.80,.1
```

```
Copying files: 27153 files [00:55, 485.59 files/s]
```

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import os
import PIL
import pathlib

import tensorflow as tf
from tensorflow import keras
```

In [8]:
```
train_dir = pathlib.Path('/content/output/train')
val_dir = pathlib.Path('/content/output/val')
test_dir = pathlib.Path('/content/output/test')
train_image_count = len(list(train_dir.glob('*/*.jpg')))
train_image_count
```

Out[8]:
```
21719
```

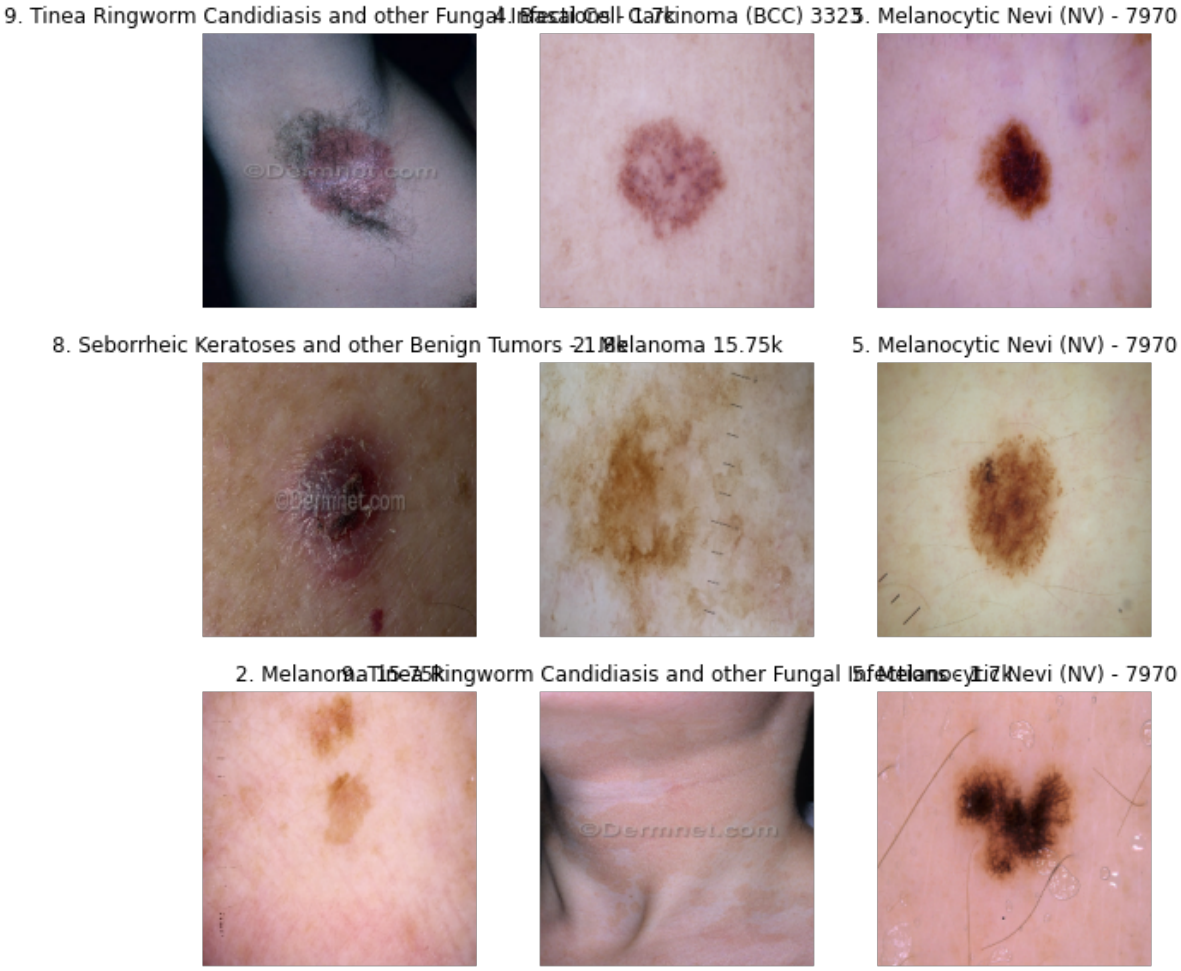
In [9]:
```
batch_size = 32
image_width = 224
image_height = 224

train_ds = keras.preprocessing.image_dataset_from_directory(
    train_dir,
    seed=123,
    #validation_split=0.5,
    #subset='training',
    image_size=(image_height, image_width),
    batch_size=batch_size
)

val_ds = keras.preprocessing.image_dataset_from_directory(
    val_dir,
    seed=123,
    image_size=(image_height, image_width),
    batch_size=batch_size
)

test_ds = keras.preprocessing.image_dataset_from_directory(
    test_dir,
    seed=123,
    image_size=(image_height, image_width),
    batch_size=batch_size
)
```

```
Found 21719 files belonging to 10 classes.
Found 2711 files belonging to 10 classes.
Found 2723 files belonging to 10 classes.
```

In [10]:
```
#check data class
class_names = train_ds.class_names
class_names
```

Out[10]:
```
['1. Eczema 1677',
 '10. Warts Molluscum and other Viral Infections - 2103',
 '2. Melanoma 15.75k',
 '3. Atopic Dermatitis - 1.25k',
 '4. Basal Cell Carcinoma (BCC) 3323',
 '5. Melanocytic Nevi (NV) - 7970',
 '6. Benign Keratosis-like Lesions (BKL) 2624',
 '7. Psoriasis pictures Lichen Planus and related diseases - 2k',
 '8. Seborrheic Keratoses and other Benign Tumors - 1.8k',
 '9. Tinea Ringworm Candidiasis and other Fungal Infections - 1.7k
']
```

In [11]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i+1)
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(class_names[labels[i]])
        plt.axis('off')
```



In [12]:
```
# Criando o modelo base em cima do modelo MobileNetV3
base_model = keras.applications.MobileNetV3Small(input_shape=(image
                                                  classes=400,
                                                  include_top=False,
                                                  weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v3/weights_mobilenet_v3_small_224_1.0_float_no_top_v2.h5
(https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v3/weights_mobilenet_v3_small_224_1.0_float_no_top_v2.h5)
4334752/4334752 [==============================] - 0s 0us/step
```

In [13]:
```
# Freeze convolutional base
base_model.trainable = False
base_model.summary()
```

```
 tf.math.multiply_12 (TFOpLambd  (None, 14, 14, 120)  0           [
're_lu_17[0][0]']
 a)

 multiply_8 (Multiply)          (None, 14, 14, 120)  0           [
'expanded_conv_6/depthwise/Batch
                                                                 N
orm[0][0]',

'tf.math.multiply_12[0][0]']

 expanded_conv_6/squeeze_excite  (None, 1, 1, 120)   0           [
'multiply_8[0][0]']
 /AvgPool (GlobalAveragePooling
 2D)

 expanded_conv_6/squeeze_excite  (None, 1, 1, 32)    3872        [
'expanded_conv_6/squeeze_excite/
 /Conv (Conv2D)                                                  A
vgPool[0][0]']
```

In [14]:
```
data_augmentation = keras.models.Sequential([
  keras.layers.RandomFlip('horizontal'),
  keras.layers.RandomRotation(0.2)
])
```

In [15]:
```
num_classes = len(class_names)# 7

inputs = keras.Input(shape=(image_width, image_height, 3))
#x = data_augmentation(inputs)
x = keras.applications.mobilenet_v3.preprocess_input(inputs)
x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)


outputs = keras.layers.Dense(num_classes, activation='softmax')(x)
model = keras.Model(inputs, outputs)
```

In [16]:
```
#compile model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

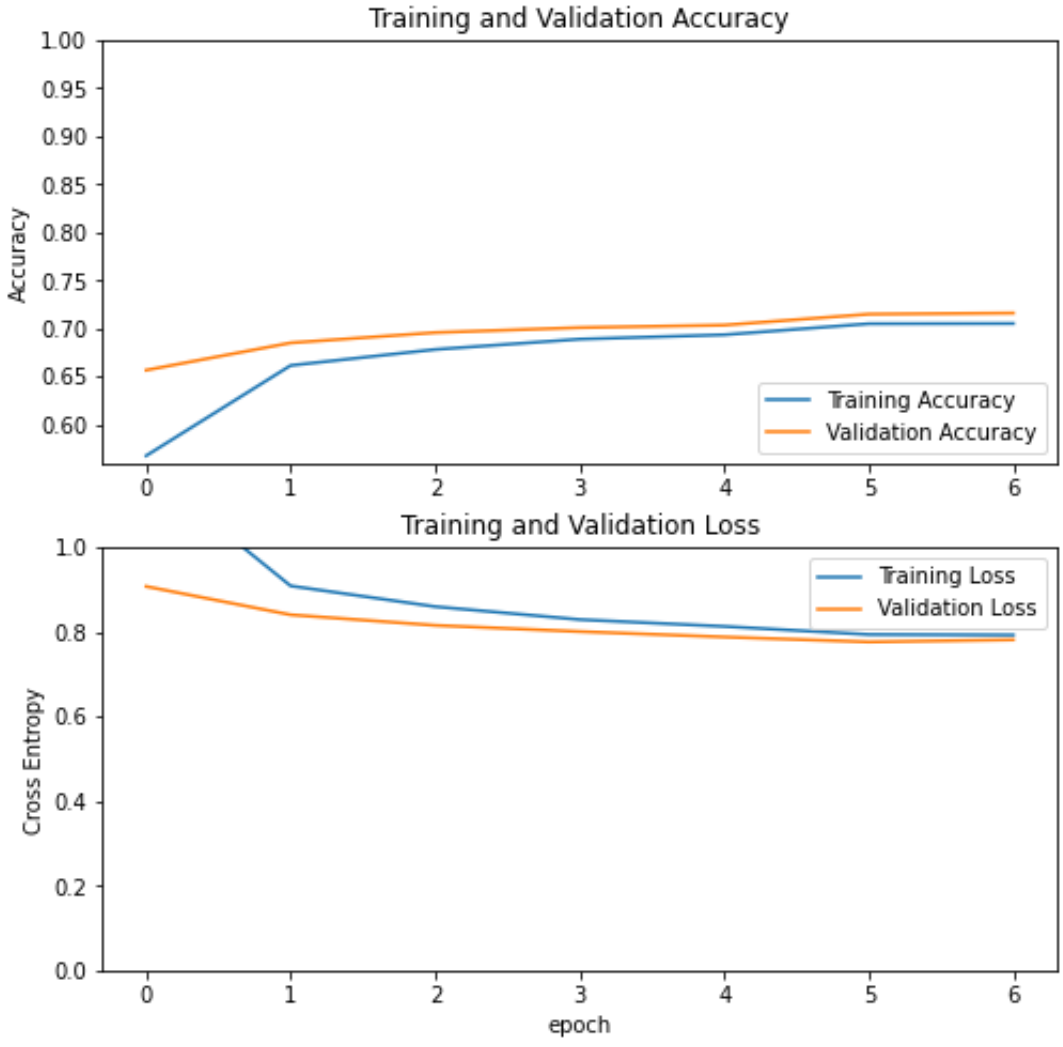
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 MobilenetV3small (Functiona  (None, 7, 7, 576)        939120
 l)

 global_average_pooling2d (G  (None, 576)              0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 576)               0

 dense (Dense)               (None, 10)                5770

=================================================================
Total params: 944,890
Trainable params: 5,770
Non-trainable params: 939,120
_________________________________________________________________
```

In [17]:
```
#fitting model
initial_epochs = 10

early_stop = keras.callbacks.EarlyStopping(patience=1, restore_best

history = model.fit(train_ds,
          validation_data=val_ds,
          epochs=initial_epochs,
          callbacks=[early_stop])
```

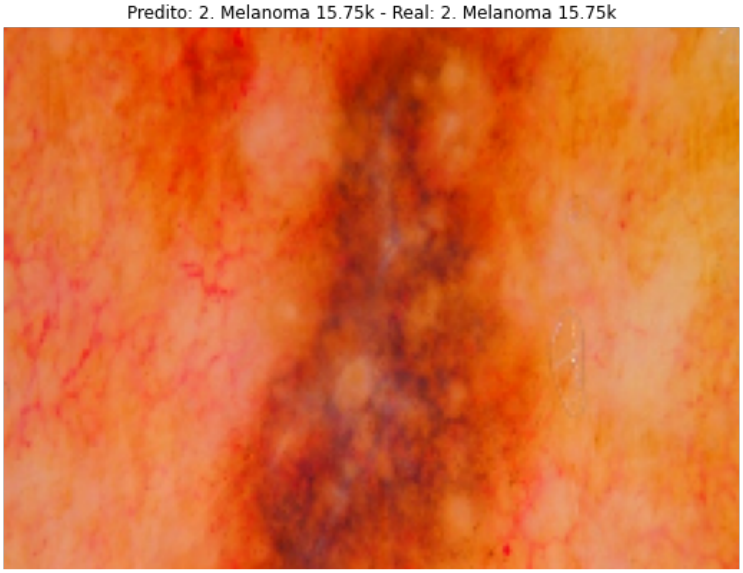
```
Epoch 1/10
679/679 [==============================] - 132s 177ms/step - loss:
1.1856 - accuracy: 0.5676 - val_loss: 0.9078 - val_accuracy: 0.656
6
Epoch 2/10
679/679 [==============================] - 117s 171ms/step - loss:
0.9090 - accuracy: 0.6615 - val_loss: 0.8407 - val_accuracy: 0.685
0
Epoch 3/10
679/679 [==============================] - 102s 149ms/step - loss:
0.8600 - accuracy: 0.6781 - val_loss: 0.8157 - val_accuracy: 0.695
7
Epoch 4/10
679/679 [==============================] - 98s 143ms/step - loss:
0.8294 - accuracy: 0.6888 - val_loss: 0.8010 - val_accuracy: 0.700
8
Epoch 5/10
679/679 [==============================] - 99s 144ms/step - loss:
0.8132 - accuracy: 0.6934 - val_loss: 0.7882 - val_accuracy: 0.703
4
Epoch 6/10
679/679 [==============================] - 98s 143ms/step - loss:
0.7940 - accuracy: 0.7049 - val_loss: 0.7766 - val_accuracy: 0.714
9
Epoch 7/10
679/679 [==============================] - 99s 145ms/step - loss:
0.7929 - accuracy: 0.7051 - val_loss: 0.7816 - val_accuracy: 0.716
0
```

In [18]:
```
```

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

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

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```



In [19]:
```
plt.figure(figsize=(10, 100))
for images, labels in test_ds.take(1):
    prediction = model.predict(images,batch_size=32)
    for i in range(9):
        ax = plt.subplot(9, 1, i+1)
        pred = np.argmax(prediction[i])
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(f'Predito: {class_names[pred]} - Real: {class_nam
        plt.axis('off')
```



In [20]:
```
#Visualize the result
results = model.evaluate(test_ds, verbose=0)
```

In [22]:
```
print("Test Loss: {:.5f}".format(results[0]))
print("Accuracy on the test set: {:.2f}%".format(results[1] * 100))
```

```
Test Loss: 0.75583
Accuracy on the test set: 71.58%
```

In [23]:
```
model.save('/content/Virus.h5')
```

In [ ]:
```
```