In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries instal
led
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/d
ocker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will lis
t all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that ge
ts preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved
outside of the current session
```

In [2]:
```
# importing libraries

import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
```

In [3]:
```
image_set =  "../input/lung-and-colon-cancer-histopathological-images/lung_col
on_image_set/lung_image_sets"
```

In [4]:
```
SIZE_X = SIZE_Y = 224

datagen = tf.keras.preprocessing.image.ImageDataGenerator(validation_split = 0.2)

train_set = datagen.flow_from_directory(image_set,
                                        class_mode = "categorical",
                                        target_size = (SIZE_X,SIZE_Y),
                                        color_mode="rgb",
                                        batch_size = 128,
                                        shuffle = False,
                                        subset='training',
                                        seed = 42)

validate_set = datagen.flow_from_directory(image_set,
                                        class_mode = "categorical",
                                        target_size = (SIZE_X, SIZE_Y),
                                        color_mode="rgb",
                                        batch_size = 128,
                                        shuffle = False,
                                        subset='validation',
                                        seed = 42)
```

In [5]:
```
IMAGE_SIZE = [224, 224]
```

In [6]:
```
resnet = ResNet50(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

In [7]:
```
# don't train existing weights
for layer in resnet.layers:
    layer.trainable = False
```

In [8]:
```
flatten = Flatten()(resnet.output)
dense = Dense(256, activation = 'relu')(flatten)
dense = Dense(128, activation = 'relu')(dense)
prediction = Dense(3, activation = 'softmax')(dense)
```

In [9]:
```
#creating a model
model = Model(inputs = resnet.input, outputs = prediction )
```

In [10]:
```
model.summary()
```

In [11]:
```
# COMPILING THE MODEL

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

In [12]:
```
#executing the model
history = model.fit_generator(train_set, validation_data = (validate_set), epochs = 5, verbose = 1)
```

In [13]:
```
# plotting the loss
plt.plot(history.history['loss'],label = 'train_loss')
plt.plot(history.history['val_loss'], label = 'testing_loss')
plt.title('loss')
plt.legend()
plt.show()
```

In [14]:
```
# Both Validation and Training accuracy is shown here

plt.plot(history.history['accuracy'], label='training_accuracy')
plt.plot(history.history['val_accuracy'], label='validation accuracy')
plt.title('Accuracy')
plt.legend()
plt.show()
```

In [15]:
```
# CHECKING THE CONFUSION MATRIX

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
Y_pred = model.predict_generator(validate_set)
y_pred = np.argmax(Y_pred ,axis =1)
print('Confusion Matrix')
confusion_matrix = confusion_matrix(validate_set.classes, y_pred)
print(confusion_matrix)
print('Classification Report')
target_names = ['aca','n', 'scc']
print(classification_report(validate_set.classes, y_pred, target_names=target_
names))
```

In [16]:
```
result = model.evaluate(validate_set,batch_size=128)
print("test_loss, test accuracy",result)
```

In [ ]:
```

```