In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-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 list all files u

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 5GB to the current directory (/kaggle/working/) that gets preserved a
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of t
```

```
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_DC-14-9461
-400-025.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_DC-14-1618
8-400-003.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_DC-14-1671
6-400-018.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_DC-14-5695
-400-001.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_LC-14-1220
4-400-030.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_PC-14-9146
-400-009.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_PC-14-1944
0-400-013.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_PC-14-1570
4-400-028.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_DC-14-1231
2-400-026.png
/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_DC-14-5695
```

In [2]:

```
import pandas as pd
import numpy as np
import os
import tensorflow as tf
import keras
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam

image_size = [224,224]
data_path = '/content/output'

vgg = VGG16(input_shape= image_size+[3],weights='imagenet',include_top=False)

x = vgg.output
x = GlobalAveragePooling2D()(x)

x = Dense(1024,activation='relu')(x)
x = Dense(1024,activation='relu')(x)
x = Dense(512, activation='relu')(x)

preds = Dense(2,activation='softmax')(x)

model = Model(inputs = vgg.input,outputs=preds)

for layer in vgg.layers:
    layer.trainable = False

train_datagen=ImageDataGenerator(preprocessing_function=preprocess_input) #included in our
test_datagen=ImageDataGenerator(preprocessing_function=preprocess_input)
train_generator=train_datagen.flow_from_directory('/kaggle/input/breakhis-400x/BreaKHis 400
                                                  target_size=(224,224),
                                                  color_mode='rgb',
                                                  batch_size=32,
                                                  class_mode='categorical',
                                                  shuffle=True)
test_generator=test_datagen.flow_from_directory('/kaggle/input/breakhis-400x/BreaKHis 400X/
                                                  target_size=(224,224),
                                                  color_mode='rgb',
                                                  batch_size=32,
                                                  shuffle=False)

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


print(train_generator.n)
print(train_generator.batch_size)
print(746//32)

step_size_train=train_generator.n//train_generator.batch_size
r = model.fit_generator(generator=train_generator,
                    validation_data=test_generator,
                    steps_per_epoch=step_size_train,
                    epochs=50)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-app
lications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5 (http
s://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weig
hts_tf_dim_ordering_tf_kernels_notop.h5)
58892288/58889256 [==============================] - 1s 0us/step
Found 1148 images belonging to 2 classes.
Found 545 images belonging to 2 classes.
1148
32
23
Epoch 1/50
35/35 [==============================] - 33s 952ms/step - loss: 1.0704 -
 accuracy: 0.7455 - val_loss: 0.3637 - val_accuracy: 0.8422
Epoch 2/50
35/35 [==============================] - 32s 901ms/step - loss: 0.2945 -
 accuracy: 0.8871 - val_loss: 0.2841 - val_accuracy: 0.8771
Epoch 3/50
35/35 [==============================] - 31s 886ms/step - loss: 0.2004 -
 accuracy: 0.9247 - val_loss: 0.4188 - val_accuracy: 0.8642
```

In [3]:

```
acc=model.evaluate_generator(test_generator)
print(acc[1])
```

```
0.9266055226325989
```

In [4]:

```
import matplotlib.pyplot as plt
```

In [5]:

```
history=r
```

In [8]:

```
print ('Training Accuracy = ' + str(history.history['accuracy']))
print ('Validation Accuracy = ' + str(history.history['val_accuracy']))
```

```
Training Accuracy = [0.7455196976661682, 0.8870967626571655, 0.9247311949729
919, 0.934587836265564, 0.9722222089767456, 0.9820788502693176, 0.9722222089
767456, 0.9767025113105774, 0.975806474685669, 0.9811828136444092, 0.9937499
761581421, 0.990143358707428, 0.9767025113105774, 0.9838709831237793, 0.9946
236610412598, 0.9910394549369812, 0.9829748868942261, 0.9910394549369812, 0.
9937276244163513, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.
0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.
0, 1.0, 1.0, 1.0, 1.0]
Validation Accuracy = [0.842201828956604, 0.8770642280578613, 0.864220201969
1467, 0.8917431235313416, 0.9137614965438843, 0.9082568883895874, 0.89357799
29161072, 0.878899097442627, 0.8972477316856384, 0.8917431235313416, 0.92293
57838630676, 0.910091757774353, 0.8330275416374207, 0.910091757774353, 0.921
100914478302, 0.8733944892883301, 0.9009174108505249, 0.8697247505187988, 0.
9247706532478333, 0.9266055226325989, 0.9192660450935364, 0.926605522632598
9, 0.9247706532478333, 0.9266055226325989, 0.9247706532478333, 0.92477065324
78333, 0.9247706532478333, 0.9266055226325989, 0.9247706532478333, 0.9266055
226325989, 0.9266055226325989, 0.9247706532478333, 0.9247706532478333, 0.924
7706532478333, 0.9247706532478333, 0.9247706532478333, 0.9247706532478333,
0.9247706532478333, 0.9247706532478333, 0.9229357838630676, 0.92293578386306
76, 0.9229357838630676, 0.9247706532478333, 0.9229357838630676, 0.9247706532
478333, 0.9247706532478333, 0.9247706532478333, 0.9247706532478333, 0.924770
6532478333, 0.9266055226325989]
```

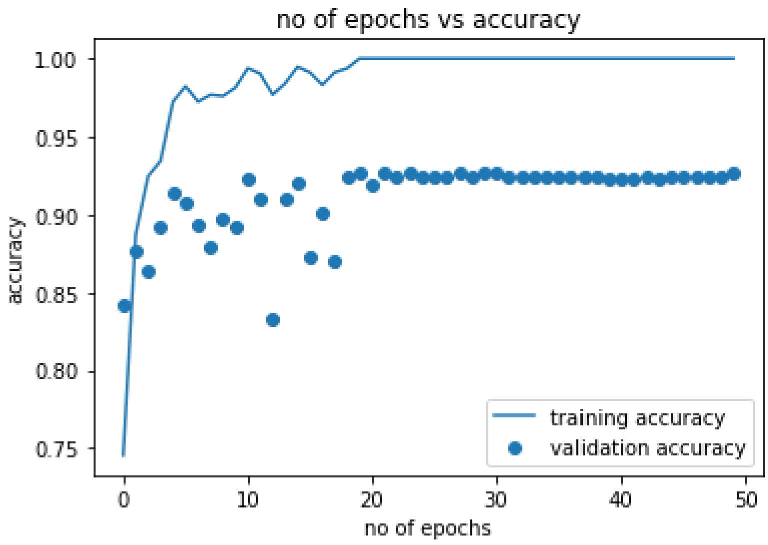
In [9]:

```
acc=history.history['accuracy']  ##getting  accuracy of each epochs
epochs_=range(0,50)
plt.plot(epochs_,acc,label='training accuracy')
plt.xlabel('no of epochs')
plt.ylabel('accuracy')

acc_val=history.history['val_accuracy']  ##getting validation accuracy of each epochs
plt.scatter(epochs_,acc_val,label="validation accuracy")
plt.title("no of epochs vs accuracy")
plt.legend()
```

Out[9]:

```
<matplotlib.legend.Legend at 0x7ff1a003b990>
```



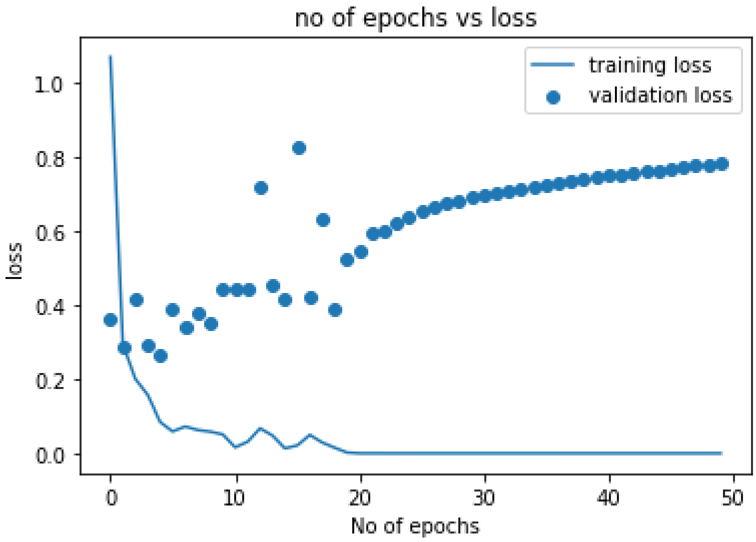
In [11]:

```
acc=history.history['loss']    ##getting  loss of each epochs
epochs_=range(0,50)
plt.plot(epochs_,acc,label='training loss')
plt.xlabel('No of epochs')
plt.ylabel('loss')

acc_val=history.history['val_loss']  ## getting validation loss of each epochs
plt.scatter(epochs_,acc_val,label="validation loss")
plt.title('no of epochs vs loss')
plt.legend()
```

Out[11]:

```
<matplotlib.legend.Legend at 0x7ff14c127650>
```



In [18]:

```
from keras import models
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import Model
import matplotlib.pyplot as plt
from numpy import expand_dims
```

In [19]:

```
from tensorflow.keras.preprocessing import image
```

In [14]:

```
pred=model.predict(test_generator,batch_size=32)
```

In [17]:

```
!pip install keract
```

```
Collecting keract
  Downloading keract-4.3.2-py3-none-any.whl (11 kB)
Requirement already satisfied: numpy>=1.18.5 in /opt/conda/lib/python3.7/sit
e-packages (from keract) (1.18.5)
Installing collected packages: keract
Successfully installed keract-4.3.2
```

In [18]:

```
def preprocess_image(img_path, model=None, rescale=255, resize=(256, 256)):
    """
    Preprocesses a given image for prediction with a trained model, with rescaling and resi

    Arguments:
            img_path: The path to the image file
            rescale: A float or integer indicating required rescaling.
                    The image array will be divided (scaled) by this number.
            resize: A tuple indicating desired target size.
                    This should match the input shape as expected by the model
    Returns:
            img: A processed image.
    """
    from keras.preprocessing.image import img_to_array, load_img
    import cv2
    import numpy as np

    assert type(img_path) == str, "Image path must be a string"
    assert (
        type(rescale) == int or type(rescale) == float
    ), "Rescale factor must be either a float or int"
    assert (
        type(resize) == tuple and len(resize) == 2
    ), "Resize target must be a tuple with two elements"

    img = load_img(img_path)
    img = img_to_array(img)
    img = img / float(rescale)
    img = cv2.resize(img, resize)
    if model != None:
        if len(model.input_shape) == 4:
            img = np.expand_dims(img, axis=0)

    return img
```

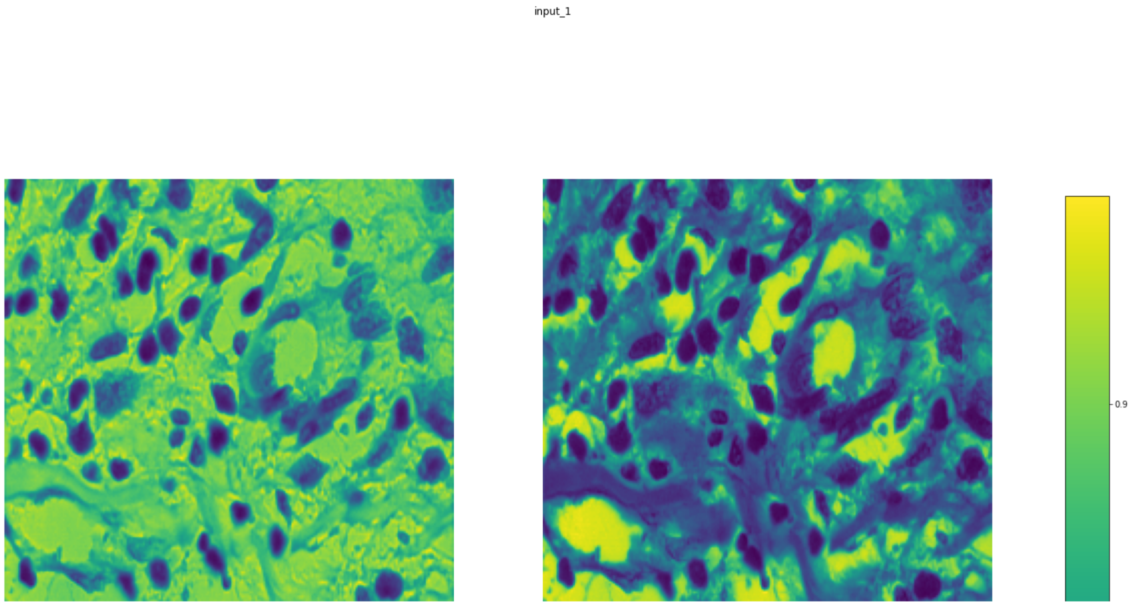
In [19]:

```
from keract import display_activations,get_activations
# The image path
img_path = '/kaggle/input/breakhis-400x/BreaKHis 400X/test/malignant/SOB_M_DC-14-11520-400-
# Preprocessing the image for the model
x = preprocess_image(img_path=img_path,model=model,resize=(224,224))
# Generate the activations
activations = get_activations(model, x)
```

In [20]:

```
display_activations(activations, save=False)
```

```
input_1 (1, 224, 224, 3)
```



In [21]:

```
#Benign
from keract import display_activations,get_activations
# The image path
img_path = '/kaggle/input/breakhis-400x/BreaKHis 400X/test/benign/SOB_B_PT-14-22704-400-011
# Preprocessing the image for the model
x = preprocess_image(img_path=img_path,model=model,resize=(224,224))
# Generate the activations
activations = get_activations(model, x)
```

In [22]:

```
display_activations(activations, save=False)#Benign
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-22-40b0536c0d97> in <module>
----> 1 display_activations(activations[0], save=False)#Benign

KeyError: 0
```

In [ ]:

```

```