In [1]:
```
#importing libraries

from numpy.random import seed
seed(101)

import pandas as pd
import numpy as np

import tensorflow

#from tensorflow.keras.models import Sequential
from keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
import numpy as np
import pydot
import matplotlib.pyplot as plt
import h5py
from keras.utils import np_utils
from keras import backend as K
from keras.models import load_model
from keras.utils.vis_utils import plot_model
import cv2
import os
from sklearn.model_selection import train_test_split
```

```
2022-04-26 17:51:59.869688: W tensorflow/stream_executor/platform/defa
ult/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.
0'; dlerror: libcudart.so.11.0: cannot open shared object file: No suc
h file or directory; LD_LIBRARY_PATH: /home/ravina/catkin_ws/devel/li
b:/opt/ros/noetic/lib
2022-04-26 17:51:59.869709: I tensorflow/stream_executor/cuda/cudart_s
tub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up
on your machine.
```

In [2]:
```
# load data
numepochs=100
batchsize=128
folder_path = '/home/ravina/Desktop/DL/Dataset of Mammography with Benig
images = []
labels = []
class_label = 0
```

In [3]:
```
def load_images_from_folder(folder,class_label):
    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder, filename))
        if img is not None:
            img = cv2.resize(img,(140,92))
            img = img.reshape(92,140,3)
            images.append(img)
            labels.append(class_label)
    class_label=class_label+1
    return class_label
```

In [4]:
```
class_label = load_images_from_folder(folder_path+'benign',class_label)
class_label = load_images_from_folder(folder_path+'malignant',class_labe
```

In [5]:
```
Data = np.asarray(images)
Labels = np.asarray(labels)
```

In [6]:
```
X_train,X_test,y_train,y_test=train_test_split(Data,Labels,test_size=0.2
```

In [7]:
```
# normalize inputs from 0-255 to 0-1
X_train = X_train / 255
X_test = X_test / 255
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

In [8]:
```
#printing sizes
print ("train data shape:")
print (X_train.shape)
print ("test data shape:")
print (X_test.shape)
print ("train label shape:")
print (y_train.shape)
print ("test label shape:")
print (y_test.shape)
```

```
train data shape:
(3004, 92, 140, 3)
test data shape:
(752, 92, 140, 3)
train label shape:
(3004, 2)
test label shape:
(752, 2)
```

In [9]:
```
# define the larger model
def larger_model():
    # create model
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding="same",input_shape=(92,140,3),
    #model.add(Conv2D(32, (3, 3), activation='relu',padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(32, (3, 3), activation='relu',padding = 'same'))
    #model.add(Conv2D(64, (3, 3), activation='relu',padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu',padding = 'same'))
    #model.add(Conv2D(128, (3, 3), activation='relu',padding = 'same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    #model.add(Dense(50, activation='relu'))
    #model.add(Dropout(0.2))
    model.add(Dense(num_classes, activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', met
    return model
```

In [10]:
```
# build the model
model = larger_model()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 92, 140, 32)       896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 46, 70, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 46, 70, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 35, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 23, 35, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 17, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 11, 17, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 11968)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 11968)             0
_________________________________________________________________
dense (Dense)                (None, 64)                766016
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130
=================================================================
Total params: 798,946
Trainable params: 798,946
Non-trainable params: 0
_________________________________________________________________
```

```
2022-04-26 17:52:25.678823: W tensorflow/stream_executor/platform/def
ault/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1';
dlerror: libcuda.so.1: cannot open shared object file: No such file o
r directory; LD_LIBRARY_PATH: /home/ravina/.local/lib/python3.8/site-
packages/cv2/../../lib64:/home/ravina/catkin_ws/devel/lib:/opt/ros/no
etic/lib
2022-04-26 17:52:25.678863: W tensorflow/stream_executor/cuda/cuda_dr
iver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-04-26 17:52:25.678891: I tensorflow/stream_executor/cuda/cuda_di
agnostics.cc:156] kernel driver does not appear to be running on this
host (RavinakaXPS): /proc/driver/nvidia/version does not exist
2022-04-26 17:52:25.679306: I tensorflow/core/platform/cpu_feature_gu
ard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neur
al Network Library (oneDNN) to use the following CPU instructions in
performance-critical operations:  AVX2 FMA
```

```
To enable them in other operations, rebuild TensorFlow with the appro
priate compiler flags.
```

In [13]:
```
# Fit the model
hist=model.fit(X_train, y_train, validation_data=(X_test, y_test), epoch
```

```
accuracy: 0.9521 - val_loss: 0.2537 - val_accuracy: 0.9162
Epoch 95/100
24/24 [==============================] - 36s 2s/step - loss: 0.1191 -
accuracy: 0.9547 - val_loss: 0.2574 - val_accuracy: 0.9176
Epoch 96/100
24/24 [==============================] - 31s 1s/step - loss: 0.1324 -
accuracy: 0.9521 - val_loss: 0.2628 - val_accuracy: 0.9162

Epoch 97/100
24/24 [==============================] - 34s 1s/step - loss: 0.1391 -
accuracy: 0.9467 - val_loss: 0.2417 - val_accuracy: 0.9122
Epoch 98/100
24/24 [==============================] - 36s 2s/step - loss: 0.1148 -
accuracy: 0.9591 - val_loss: 0.2313 - val_accuracy: 0.9202
Epoch 99/100
24/24 [==============================] - 35s 1s/step - loss: 0.1099 -
accuracy: 0.9597 - val_loss: 0.2363 - val_accuracy: 0.9229
Epoch 100/100
24/24 [==============================] - 30s 1s/step - loss: 0.1137 -
accuracy: 0.9574 - val_loss: 0.2450 - val_accuracy: 0.9202
```

In [14]:
```
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=1,batch_size=batchsize)
```

```
6/6 [==============================] - 2s 401ms/step - loss: 0.2450 -
accuracy: 0.9202
```

In [15]:
```
model.save('model_breastCancerMIAS.h5')
```

In [16]:
```
print("Deep Net Accuracy: %.2f%%" % (scores[1]*100))
```

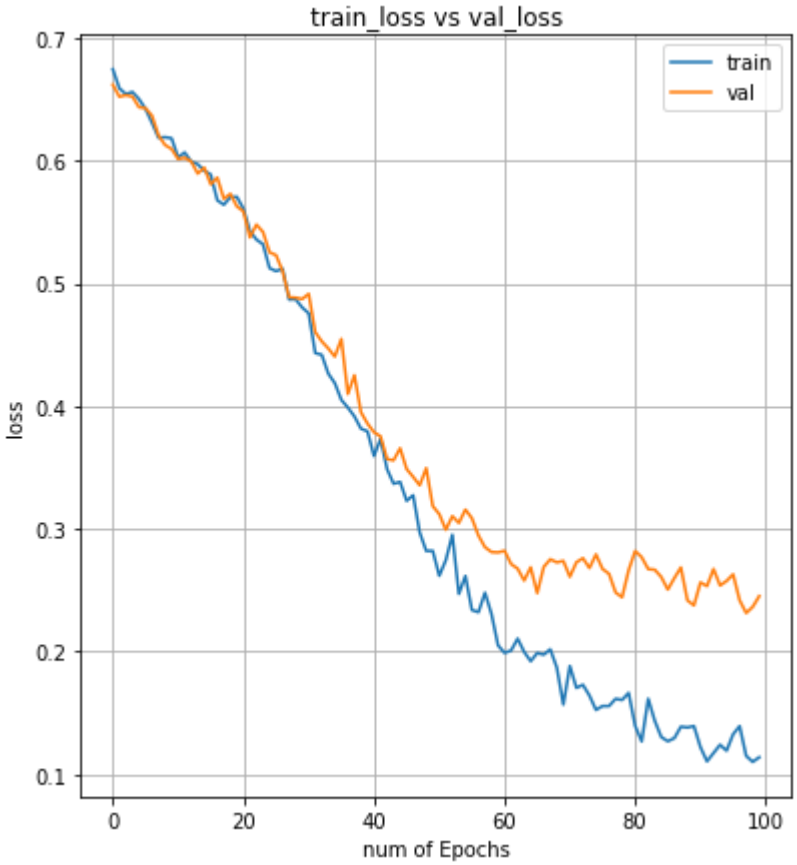
```
Deep Net Accuracy: 92.02%
```

In [17]:
```
#testing an image from the test set
print("\n\n***** TESTING AN IMAGE FROM TEST SET *****\n")
test_image = X_test[0:1]
print("Shape of test image 1:")
print (test_image.shape)
print("Predicted accuracies:")
print(model.predict(test_image))
predict_x=model.predict(test_image)
classes_x=np.argmax(predict_x,axis=1)
print("Predicted class:")
print(classes_x)
```

```
***** TESTING AN IMAGE FROM TEST SET *****

Shape of test image 1:
(1, 92, 140, 3)
Predicted accuracies:
[[9.9984705e-01 1.5292587e-04]]
Predicted class:
[0]
```

In [19]:
```
xc=range(numepochs)
plt.figure(1,figsize=(14,7))
#plt.figure(1)
plt.subplot(121)
plt.plot(xc,train_loss)
plt.plot(xc,val_loss)
plt.xlabel('num of Epochs')
plt.ylabel('loss')
plt.title('train_loss vs val_loss')
plt.grid(True)
plt.legend(['train','val'])
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])
```



In [18]:
```
# visualizing losses and accuracy
train_loss=hist.history['loss']
val_loss=hist.history['val_loss']
train_acc=hist.history['accuracy']
val_acc=hist.history['val_accuracy']
```

In [22]:
```
#testing any image
print("\n\n***** TESTING ANY RANDOM IMAGE *****\n")
test_image = cv2.imread('/home/ravina/Desktop/DL/Dataset of Mammography
test_image= cv2.resize(test_image,(140,92))
#test_image = test_image.reshape(92,140,3)
test_image = np.array(test_image)
test_image = test_image.astype('float32')
test_image /= 255
test_image= np.expand_dims(test_image, axis=0)
print("Shape of test image 2:")
print (test_image.shape)
print("Predicted accuracies:")
print((model.predict(test_image)))
predict_x=model.predict(test_image)
classes_x=np.argmax(predict_x,axis=1)
print("Predicted class:")
print(classes_x)
```

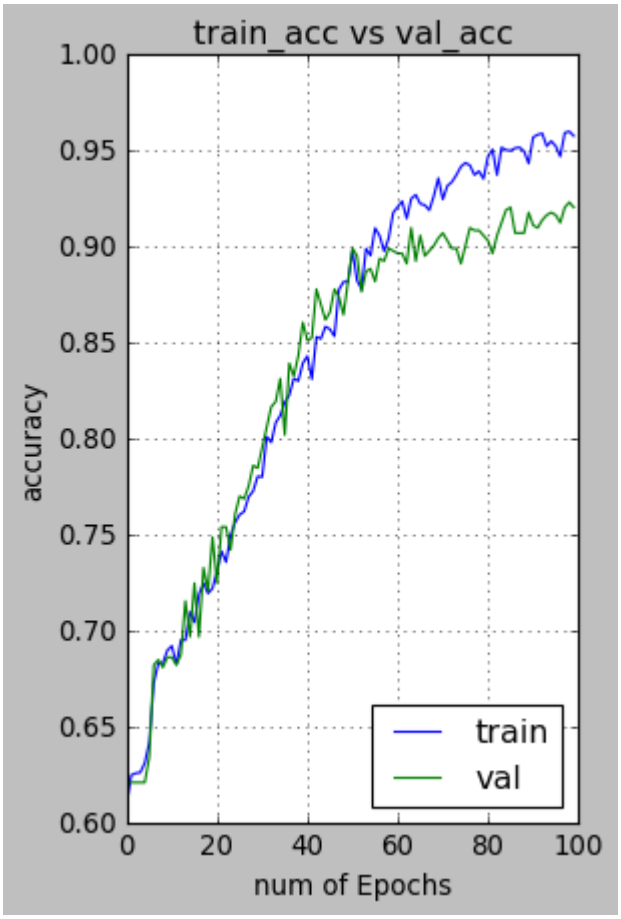
```
***** TESTING ANY RANDOM IMAGE *****

Shape of test image 2:
(1, 92, 140, 3)
Predicted accuracies:
[[0.95418817 0.04581186]]
Predicted class:
[0]
```

In [23]:
```
#plt.figure(2,figsize=(7,5))
plt.subplot(122)
plt.plot(xc,train_acc)
plt.plot(xc,val_acc)
plt.xlabel('num of Epochs')
plt.ylabel('accuracy')
plt.title('train_acc vs val_acc')
plt.grid(True)

plt.legend(['train','val'],loc=4)
#print plt.style.available # use bmh, classic,ggplot for big pictures
plt.style.use(['classic'])

plt.show()
```



In [41]:
```
y_pred = model.predict(X_test, verbose=0)
yhat_classes =np.argmax(y_pred,axis=1)
```

In [42]:
```
# reduce to 1d array
y_pred = y_pred[:, 0]
```

In [48]:
```
y_test = np.argmax(y_test,axis=1)
```

In [49]:
```
from sklearn.metrics import confusion_matrix
```

In [50]:
```
cm = confusion_matrix(y_test, yhat_classes)
```

In [51]:
```
from sklearn.metrics import classification_report
```

In [52]:
```
print(classification_report(y_test,yhat_classes))
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       467
           1       0.92      0.87      0.89       285

    accuracy                           0.92       752
   macro avg       0.92      0.91      0.91       752
weighted avg       0.92      0.92      0.92       752
```

In [ ]:
```

```