In [1]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
from torchvision import datasets, models, transforms
import numpy as np
import matplotlib.pyplot as plt
import cv2
import time
import os
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu") # devi
ce object
```

In [2]:
```
transforms_train = transforms.Compose([
    transforms.Resize((224, 224)),   #must same as here
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(), # data augmentation
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) # norma
lization
])
transforms_test = transforms.Compose([
    transforms.Resize((224, 224)),   #must same as here
     transforms.CenterCrop((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

In [3]:
```
train_dir = "data/train/"
test_dir = "data/test/"
train_classa_dir = "data/train/benign/"
train_classb_dir = "data/train/malignant/"
train_classb_dir = "data/train/normal/"
test_classa_dir = "data/test/benign/"
test_classb_dir = "data/test/malignant/"
test_classb_dir = "data/test/normal/"
```

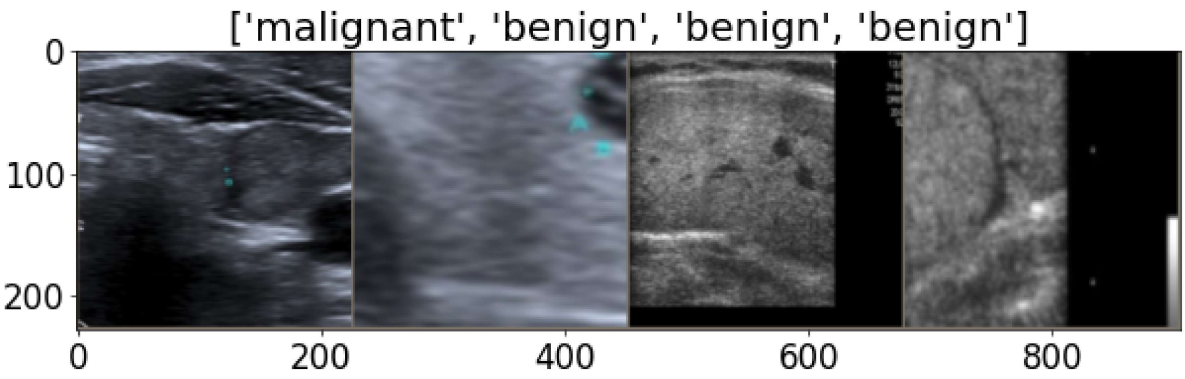
In [4]:
```
train_dataset = datasets.ImageFolder(train_dir, transforms_train)
test_dataset = datasets.ImageFolder(test_dir, transforms_test)
train_dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=12, s
huffle=True, num_workers=8)
test_dataloader = torch.utils.data.DataLoader(test_dataset, batch_size=12, shu
ffle=False, num_workers=8)
```

In [5]:
```
print('Train dataset size:', len(train_dataset))
print('Test dataset size:', len(test_dataset))
class_names = train_dataset.classes
print('Class names:', class_names)
```

```
Train dataset size: 1597
Test dataset size: 731
Class names: ['benign', 'malignant', 'normal']
```

In [6]:
```
plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams['figure.dpi'] = 60
plt.rcParams.update({'font.size': 20})
def imshow(input, title):
    # torch.Tensor => numpy
    input = input.numpy().transpose((1, 2, 0))
    # undo image normalization
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    input = std * input + mean
    input = np.clip(input, 0, 1)
    # display images
    plt.imshow(input)
    plt.title(title)
    plt.show()
```

In [7]:
```
# load a batch of train image
iterator = iter(train_dataloader)
# visualize a batch of train image
inputs, classes = next(iterator)
out = torchvision.utils.make_grid(inputs[:4])
imshow(out, title=[class_names[x] for x in classes[:4]])
```



Feature Fusion ResNet

In [8]:
```
model = models.resnet18(pretrained=True)   #load resnet18 model
num_features = model.fc.in_features     #extract fc layers features
model.fc = nn.Linear(num_features, 3)
model = model.to(device)
criterion = nn.CrossEntropyLoss()  #(set loss function)
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

In [9]:
```
num_epochs = 50   #(set no of epochs)
start_time = time.time() #(for showing time)
for epoch in range(num_epochs): #(loop for every epoch)
    print("Epoch {} running".format(epoch)) #(printing message)
    """ Training Phase """
    model.train()    #(training model)
    running_loss = 0.   #(set loss 0)
    running_corrects = 0
    # load a batch data of images
    for i, (inputs, labels) in enumerate(train_dataloader):
        inputs = inputs.to(device)
        labels = labels.to(device)
        # forward inputs and get output
        optimizer.zero_grad()
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)
        # get loss value and update the network weights
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)
    epoch_loss = running_loss / len(train_dataset)
    epoch_acc = running_corrects / len(train_dataset) * 100.
    print('[Train #{}] Loss: {:.4f} Acc: {:.4f}% Time: {:.4f}s'.format(epoch,
epoch_loss, epoch_acc, time.time() -start_time))

    """ Testing Phase """
    model.eval()
    with torch.no_grad():
        running_loss = 0.
        running_corrects = 0
        for inputs, labels in test_dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)
            loss = criterion(outputs, labels)
            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)
        epoch_loss = running_loss / len(test_dataset)
        epoch_acc = running_corrects / len(test_dataset) * 100.
        print('[Test #{}] Loss: {:.4f} Acc: {:.4f}% Time: {:.4f}s'.format(epoc
h, epoch_loss, epoch_acc, time.time()- start_time))
```

```
Epoch 0 running
[Train #0] Loss: 0.9347 Acc: 52.8491% Time: 143.0941s
[Test #0] Loss: 0.9227 Acc: 55.4036% Time: 170.4168s
Epoch 1 running
[Train #1] Loss: 0.8585 Acc: 57.9211% Time: 319.5970s
[Test #1] Loss: 0.9226 Acc: 50.0684% Time: 346.9357s
Epoch 2 running
[Train #2] Loss: 0.7883 Acc: 62.6174% Time: 506.3778s
[Test #2] Loss: 0.9329 Acc: 55.4036% Time: 534.6911s
Epoch 3 running
[Train #3] Loss: 0.7769 Acc: 62.9305% Time: 705.3093s
[Test #3] Loss: 0.9367 Acc: 60.6019% Time: 732.1740s
Epoch 4 running
[Train #4] Loss: 0.7208 Acc: 65.9987% Time: 873.6865s
[Test #4] Loss: 1.1901 Acc: 45.8276% Time: 900.4502s
Epoch 5 running
[Train #5] Loss: 0.7039 Acc: 67.3763% Time: 1044.1711s
[Test #5] Loss: 1.1345 Acc: 62.1067% Time: 1071.1723s
Epoch 6 running
[Train #6] Loss: 0.6578 Acc: 69.0044% Time: 1213.2905s
[Test #6] Loss: 1.2354 Acc: 61.5595% Time: 1240.0456s
Epoch 7 running
[Train #7] Loss: 0.6539 Acc: 68.4408% Time: 1382.1365s
[Test #7] Loss: 1.0500 Acc: 63.2011% Time: 1408.9198s
Epoch 8 running
[Train #8] Loss: 0.6396 Acc: 71.3838% Time: 1549.6999s
[Test #8] Loss: 1.5403 Acc: 57.5923% Time: 1576.2814s
Epoch 9 running
[Train #9] Loss: 0.6076 Acc: 71.6343% Time: 1717.9724s
[Test #9] Loss: 1.4179 Acc: 63.4747% Time: 1744.8109s
Epoch 10 running
[Train #10] Loss: 0.6122 Acc: 72.5736% Time: 1887.8819s
[Test #10] Loss: 1.7258 Acc: 60.6019% Time: 1915.0039s
Epoch 11 running
[Train #11] Loss: 0.5772 Acc: 74.2642% Time: 2056.0621s
[Test #11] Loss: 1.3257 Acc: 63.0643% Time: 2082.5987s
Epoch 12 running
[Train #12] Loss: 0.5792 Acc: 73.5755% Time: 2223.9632s
[Test #12] Loss: 1.5875 Acc: 59.2339% Time: 2250.5770s
Epoch 13 running
[Train #13] Loss: 0.5633 Acc: 74.2016% Time: 2391.8843s
[Test #13] Loss: 1.2039 Acc: 70.1778% Time: 2418.6706s
Epoch 14 running
[Train #14] Loss: 0.5271 Acc: 75.7671% Time: 2559.1938s
[Test #14] Loss: 1.8650 Acc: 63.7483% Time: 2585.7286s
Epoch 15 running
[Train #15] Loss: 0.5668 Acc: 72.8867% Time: 2725.9422s
[Test #15] Loss: 1.8959 Acc: 63.3379% Time: 2752.6806s
Epoch 16 running
[Train #16] Loss: 0.4632 Acc: 78.7727% Time: 2894.1459s
[Test #16] Loss: 1.5567 Acc: 66.2107% Time: 2920.7465s
Epoch 17 running
[Train #17] Loss: 0.5422 Acc: 74.7026% Time: 3063.0910s
[Test #17] Loss: 2.0225 Acc: 64.4323% Time: 3089.8950s
Epoch 18 running
```

```
[Train #18] Loss: 0.4892 Acc: 76.3306% Time: 3230.0971s
[Test #18] Loss: 1.6233 Acc: 68.1259% Time: 3256.9580s
Epoch 19 running
[Train #19] Loss: 0.4536 Acc: 79.9624% Time: 3397.4123s
[Test #19] Loss: 2.0753 Acc: 64.7059% Time: 3424.0406s
Epoch 20 running
[Train #20] Loss: 0.5078 Acc: 76.8942% Time: 3564.7796s
[Test #20] Loss: 1.5416 Acc: 58.2763% Time: 3591.9384s
Epoch 21 running
[Train #21] Loss: 0.4784 Acc: 78.3970% Time: 3733.8340s
[Test #21] Loss: 1.7099 Acc: 65.3899% Time: 3760.8558s
Epoch 22 running
[Train #22] Loss: 0.4671 Acc: 78.8979% Time: 3901.5212s
[Test #22] Loss: 2.1988 Acc: 61.1491% Time: 3928.1263s
Epoch 23 running
[Train #23] Loss: 0.4646 Acc: 79.0858% Time: 4069.8075s
[Test #23] Loss: 1.3829 Acc: 64.8427% Time: 4096.7120s
Epoch 24 running
[Train #24] Loss: 0.4494 Acc: 78.5848% Time: 4237.5830s
[Test #24] Loss: 2.0408 Acc: 63.7483% Time: 4264.5926s
Epoch 25 running
[Train #25] Loss: 0.4645 Acc: 78.5848% Time: 4404.5589s
[Test #25] Loss: 1.9913 Acc: 66.7579% Time: 4431.4306s
Epoch 26 running
[Train #26] Loss: 0.4131 Acc: 79.3989% Time: 4573.1731s
[Test #26] Loss: 1.9277 Acc: 64.9795% Time: 4600.0428s
Epoch 27 running
[Train #27] Loss: 0.4665 Acc: 77.9587% Time: 4742.6716s
[Test #27] Loss: 1.8126 Acc: 62.5171% Time: 4769.5879s
Epoch 28 running
[Train #28] Loss: 0.4796 Acc: 78.2091% Time: 4910.7943s
[Test #28] Loss: 1.9447 Acc: 63.6115% Time: 4937.7351s
Epoch 29 running
[Train #29] Loss: 0.4423 Acc: 79.1484% Time: 5078.7535s
[Test #29] Loss: 2.4034 Acc: 64.9795% Time: 5105.4466s
Epoch 30 running
[Train #30] Loss: 0.4242 Acc: 79.5241% Time: 5247.2210s
[Test #30] Loss: 2.0374 Acc: 68.1259% Time: 5274.0510s
Epoch 31 running
[Train #31] Loss: 0.4365 Acc: 78.7101% Time: 5414.9650s
[Test #31] Loss: 2.2136 Acc: 65.9371% Time: 5441.5269s
Epoch 32 running
[Train #32] Loss: 0.4042 Acc: 80.9643% Time: 5582.6675s
[Test #32] Loss: 2.5450 Acc: 65.1163% Time: 5609.7761s
Epoch 33 running
[Train #33] Loss: 0.4003 Acc: 80.4008% Time: 5751.0834s
[Test #33] Loss: 1.9661 Acc: 51.8468% Time: 5777.9996s
Epoch 34 running
[Train #34] Loss: 0.4141 Acc: 80.2129% Time: 5918.2767s
[Test #34] Loss: 2.5482 Acc: 66.2107% Time: 5945.4137s
Epoch 35 running
[Train #35] Loss: 0.3689 Acc: 81.8410% Time: 6088.0690s
[Test #35] Loss: 2.6225 Acc: 64.1587% Time: 6114.8356s
Epoch 36 running
[Train #36] Loss: 0.4048 Acc: 79.7746% Time: 6255.4617s
[Test #36] Loss: 3.1995 Acc: 61.5595% Time: 6282.0228s
```

```
Epoch 37 running
[Train #37] Loss: 0.3895 Acc: 80.3381% Time: 6423.4325s
[Test #37] Loss: 2.2571 Acc: 66.8947% Time: 6450.1244s
Epoch 38 running
[Train #38] Loss: 0.3863 Acc: 81.4652% Time: 6590.4927s
[Test #38] Loss: 2.5875 Acc: 68.5363% Time: 6617.2364s
Epoch 39 running
[Train #39] Loss: 0.3924 Acc: 81.5279% Time: 6759.1276s
[Test #39] Loss: 2.0404 Acc: 69.3570% Time: 6785.8926s
Epoch 40 running
[Train #40] Loss: 0.3761 Acc: 82.0288% Time: 6927.4117s
[Test #40] Loss: 1.9420 Acc: 60.7387% Time: 6954.1284s
Epoch 41 running
[Train #41] Loss: 0.4086 Acc: 79.2736% Time: 7094.8732s
[Test #41] Loss: 2.1041 Acc: 68.8099% Time: 7122.2660s
Epoch 42 running
[Train #42] Loss: 0.3679 Acc: 81.8410% Time: 7264.4162s
[Test #42] Loss: 2.3857 Acc: 67.4419% Time: 7291.0023s
Epoch 43 running
[Train #43] Loss: 0.3919 Acc: 82.4045% Time: 7432.7260s
[Test #43] Loss: 1.8342 Acc: 59.7811% Time: 7459.2311s
Epoch 44 running
[Train #44] Loss: 0.3697 Acc: 81.5279% Time: 7601.2959s
[Test #44] Loss: 2.8621 Acc: 66.3475% Time: 7627.9621s
Epoch 45 running
[Train #45] Loss: 0.3667 Acc: 82.3419% Time: 7769.2306s
[Test #45] Loss: 3.4607 Acc: 66.4843% Time: 7796.1790s
Epoch 46 running
[Train #46] Loss: 0.3836 Acc: 81.5279% Time: 7937.1394s
[Test #46] Loss: 1.8534 Acc: 64.0219% Time: 7963.6031s
Epoch 47 running
[Train #47] Loss: 0.3533 Acc: 82.8428% Time: 8105.0127s
[Test #47] Loss: 2.1071 Acc: 67.5787% Time: 8132.2144s
Epoch 48 running
[Train #48] Loss: 0.3657 Acc: 80.6512% Time: 8274.4667s
[Test #48] Loss: 2.9081 Acc: 66.7579% Time: 8301.1645s
Epoch 49 running
[Train #49] Loss: 0.3533 Acc: 81.9662% Time: 8441.7583s
[Test #49] Loss: 2.9667 Acc: 66.4843% Time: 8468.7149s
```

In [10]:
```
save_path = 'resnet18.pth'
torch.save(model.state_dict(), save_path)
```

In [11]:
```
model = models.resnet18(pretrained=True)   #load resnet18 model
num_features = model.fc.in_features #extract fc layers features
model.fc = nn.Linear(num_features, 3)
model.load_state_dict(torch.load("resnet18.pth"))
model.to(device)
```

Out[11]:
```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), b
ias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_
stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_m
ode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_runn
ing_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
```

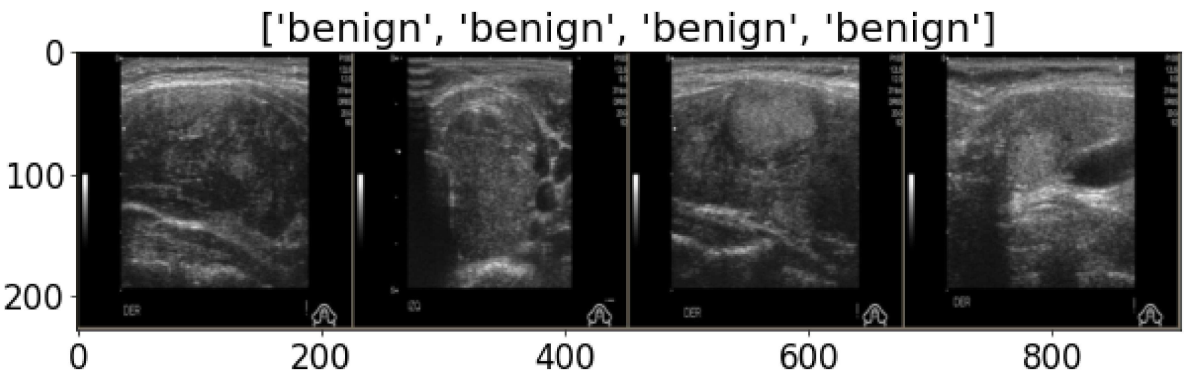
```
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
```

```
ning_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_run
ning_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=3, bias=True)
)
```

In [12]:
```
plt.rcParams['figure.figsize'] = [12, 8]
plt.rcParams['figure.dpi'] = 60
plt.rcParams.update({'font.size': 20})
def imshow(input, title):
    # torch.Tensor => numpy
    input = input.numpy().transpose((1, 2, 0))
    # undo image normalization
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    input = std * input + mean
    input = np.clip(input, 0, 1)
    # display images
    plt.imshow(input)
    plt.title(title)
    plt.show()
```

In [13]:
```
##Testing
model.eval()
start_time = time.time()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
with torch.no_grad():
    running_loss = 0.
    running_corrects = 0
    for i, (inputs, labels) in enumerate(test_dataloader):
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)
        running_loss += loss.item() * inputs.size(0)
        running_corrects += torch.sum(preds == labels.data)
        if i == 0:
            print('======>RESULTS<======')
            images = torchvision.utils.make_grid(inputs[:4])
            imshow(images.cpu(), title=[class_names[x] for x in labels[:4]])
    epoch_loss = running_loss / len(test_dataset)
    epoch_acc = running_corrects / len(test_dataset) * 100.
    print('[Test #{}] Loss: {:.4f} Acc: {:.4f}% Time: {:.4f}s'.
          format(epoch, epoch_loss, epoch_acc, time.time() - start_time))
```

```
======>RESULTS<======
```




```
[Test #49] Loss: 2.9667 Acc: 66.4843% Time: 27.4110s
```

Feature Fusion VGG16

In [14]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from matplotlib import image as mp_image
import seaborn as sns

# Required magic to display matplotlib plots in notebooks
%matplotlib inline

from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split

import os
import shutil
```

```
C:\Users\Public\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the pu
blic API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [15]:
```
# The images are in a folder named 'input/natural-images/natural_images'
training_folder_name = 'data/train'

# All images are 128x128 pixels
img_size = (128,128)

# The folder contains a subfolder for each class of shape
classes = sorted(os.listdir(training_folder_name))
print(classes)
```

```
['benign', 'malignant', 'normal']
```

In [16]:
```
# Import PyTorch libraries
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
```

In [17]:
```
from PIL import Image

# function to resize image
def resize_image(src_image, size=(128,128), bg_color="white"):
    from PIL import Image, ImageOps

    # resize the image so the longest dimension matches our target size
    src_image.thumbnail(size, Image.ANTIALIAS)

    # Create a new square background image
    new_image = Image.new("RGB", size, bg_color)

    # Paste the resized image into the center of the square background
    new_image.paste(src_image, (int((size[0] - src_image.size[0]) / 2), int((s
ize[1] - src_image.size[1]) / 2)))

    # return the resized image
    return new_image
```

In [18]:
```
training_folder_name = 'data/train'

# New location for the resized images
train_folder = 'dataset/train'


# Create resized copies of all of the source images
size = (128,128)

# Create the output folder if it doesn't already exist
if os.path.exists(train_folder):
    shutil.rmtree(train_folder)

# Loop through each subfolder in the input folder
print('Transforming images...')
for root, folders, files in os.walk(training_folder_name):
    for sub_folder in folders:
        print('processing folder ' + sub_folder)
        # Create a matching subfolder in the output dir
        saveFolder = os.path.join(train_folder,sub_folder)
        if not os.path.exists(saveFolder):
            os.makedirs(saveFolder)
        # Loop through the files in the subfolder
        file_names = os.listdir(os.path.join(root,sub_folder))
        for file_name in file_names:
            # Open the file
            file_path = os.path.join(root,sub_folder, file_name)
            #print("reading " + file_path)
            image = Image.open(file_path)
            # Create a resized version and save it
            resized_image = resize_image(image, size)
            saveAs = os.path.join(saveFolder, file_name)
            #print("writing " + saveAs)
            resized_image.save(saveAs)

print('Done.')
```

```
Transforming images...
processing folder benign
processing folder malignant
processing folder normal
Done.
```

In [19]:
```
def load_dataset(data_path):
    import torch
    import torchvision
    import torchvision.transforms as transforms
    # Load all the images
    transformation = transforms.Compose([
        # Randomly augment the image data
            # Random horizontal flip
        transforms.RandomHorizontalFlip(0.5),
            # Random vertical flip
        transforms.RandomVerticalFlip(0.3),
        # transform to tensors
        transforms.ToTensor(),
        # Normalize the pixel values (in R, G, and B channels)
        transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
    ])

    # Load all of the images, transforming them
    full_dataset = torchvision.datasets.ImageFolder(
        root=data_path,
        transform=transformation
    )


    # Split into training (70% and testing (30%) datasets)
    train_size = int(0.7 * len(full_dataset))
    test_size = len(full_dataset) - train_size

    # use torch.utils.data.random_split for training/test split
    train_dataset, test_dataset = torch.utils.data.random_split(full_dataset,
[train_size, test_size])

    # define a loader for the training data we can iterate through in 50-image
batches
    train_loader = torch.utils.data.DataLoader(
        train_dataset,
        batch_size=50,
        num_workers=0,
        shuffle=False
    )

    # define a loader for the testing data we can iterate through in 50-image
 batches
    test_loader = torch.utils.data.DataLoader(
        test_dataset,
        batch_size=50,
        num_workers=0,
        shuffle=False
    )

    return train_loader, test_loader



# Recall that we have resized the images and saved them into
```

```
train_folder = 'dataset/train'

# Get the iterative dataloaders for test and training data
train_loader, test_loader = load_dataset(train_folder)
batch_size = train_loader.batch_size
print("Data loaders ready to read", train_folder)
```

```
Data loaders ready to read dataset/train
```

In [20]:
```
# Create a neural net class
class VGG16(nn.Module):


    # Defining the Constructor
    def __init__(self, num_classes=3):
        super(VGG16, self).__init__()

        # In the init function, we define each layer we will use in our model

        # Our images are RGB, so we have input channels = 3.
        # We will apply 12 filters in the first convolutional layer
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=12, kernel_size=3,
stride=1, padding=1)

        # A second convolutional layer takes 12 input channels, and generates
 24 outputs
        self.conv2 = nn.Conv2d(in_channels=12, out_channels=24, kernel_size=3,
stride=1, padding=1)

        # We in the end apply max pooling with a kernel size of 2
        self.pool = nn.MaxPool2d(kernel_size=2)

        # A drop layer deletes 20% of the features to help prevent overfitting
        self.drop = nn.Dropout2d(p=0.2)

        # Our 128x128 image tensors will be pooled twice with a kernel size of
2. 128/2/2 is 32.
        # This means that our feature tensors are now 32 x 32, and we've gener
ated 24 of them

        # We need to flatten these in order to feed them to a fully-connected
 layer
        self.fc = nn.Linear(in_features=32 * 32 * 24, out_features=num_classes
)

    def forward(self, x):
        # In the forward function, pass the data through the layers we defined
in the init function

        # Use a ReLU activation function after layer 1 (convolution 1 and poo
l)
        x = F.relu(self.pool(self.conv1(x)))

        # Use a ReLU activation function after layer 2
        x = F.relu(self.pool(self.conv2(x)))

        # Select some features to drop to prevent overfitting (only drop durin
g training)
        x = F.dropout(self.drop(x), training=self.training)

        # Flatten
        x = x.view(-1, 32 * 32 * 24)
        # Feed to fully-connected layer to predict class
        x = self.fc(x)
```

```
        # Return class probabilities via a log_softmax function
        return torch.log_softmax(x, dim=1)

device = "cpu"
if (torch.cuda.is_available()):
    # if GPU available, use cuda (on a cpu, training will take a considerable
 length of time!)
    device = "cuda"

# Create an instance of the model class and allocate it to the device
model = VGG16(num_classes=len(classes)).to(device)

print(model)
```

```
VGG16(
  (conv1): Conv2d(3, 12, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(12, 24, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode
=False)
  (drop): Dropout2d(p=0.2, inplace=False)
  (fc): Linear(in_features=24576, out_features=3, bias=True)
)
```

In [21]:
```
def train(model, device, train_loader, optimizer, epoch):
    # Set the model to training mode
    model.train()
    train_loss = 0
    print("Epoch:", epoch)
    # Process the images in batches
    for batch_idx, (data, target) in enumerate(train_loader):
        # Use the CPU or GPU as appropriate
        # Recall that GPU is optimized for the operations we are dealing with
        data, target = data.to(device), target.to(device)

        # Reset the optimizer
        optimizer.zero_grad()

        # Push the data forward through the model layers
        output = model(data)

        # Get the loss
        loss = loss_criteria(output, target)

        # Keep a running total
        train_loss += loss.item()

        # Backpropagate
        loss.backward()
        optimizer.step()

        # Print metrics so we see some progress
        print('\tTraining batch {} Loss: {:.6f}'.format(batch_idx + 1, loss.it
em()))

    # return average loss for the epoch
    avg_loss = train_loss / (batch_idx+1)
    print('Training set: Average loss: {:.6f}'.format(avg_loss))
    return avg_loss
```

In [22]:
```
def test(model, device, test_loader):
    # Switch the model to evaluation mode (so we don't backpropagate or drop)
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        batch_count = 0
        for data, target in test_loader:
            batch_count += 1
            data, target = data.to(device), target.to(device)

            # Get the predicted classes for this batch
            output = model(data)

            # Calculate the loss for this batch
            test_loss += loss_criteria(output, target).item()

            # Calculate the accuracy for this batch
            _, predicted = torch.max(output.data, 1)
            correct += torch.sum(target==predicted).item()

    # Calculate the average loss and total accuracy for this epoch
    avg_loss = test_loss / batch_count
    print('Validation set: Average loss: {:.6f}, Accuracy: {}/{} ({:.0f}%)\n'.
format(
        avg_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))

    # return average loss for the epoch
    return avg_loss
```

In [23]:
```
# Use an "Adam" optimizer to adjust weights
optimizer = optim.Adam(model.parameters(), lr=0.01)

# Specify the loss criteria
loss_criteria = nn.CrossEntropyLoss()

# Track metrics in these arrays
epoch_nums = []
training_loss = []
validation_loss = []


epochs = 10
print('Training on', device)
for epoch in range(1, epochs + 1):
        train_loss = train(model, device, train_loader, optimizer, epoch)
        test_loss = test(model, device, test_loader)
        epoch_nums.append(epoch)
        training_loss.append(train_loss)
        validation_loss.append(test_loss)
```

```
Training on cpu
Epoch: 1
        Training batch 1 Loss: 1.139699
        Training batch 2 Loss: 6.896463
        Training batch 3 Loss: 6.442767
        Training batch 4 Loss: 1.163173
        Training batch 5 Loss: 1.038831
        Training batch 6 Loss: 1.082490
        Training batch 7 Loss: 1.047917
        Training batch 8 Loss: 0.891837
        Training batch 9 Loss: 1.090592
        Training batch 10 Loss: 0.938712
        Training batch 11 Loss: 1.057281
        Training batch 12 Loss: 1.062923
        Training batch 13 Loss: 1.085912
        Training batch 14 Loss: 1.073299
        Training batch 15 Loss: 1.078177
        Training batch 16 Loss: 1.074605
        Training batch 17 Loss: 1.080554
        Training batch 18 Loss: 1.073086
        Training batch 19 Loss: 1.065296
        Training batch 20 Loss: 1.084582
        Training batch 21 Loss: 1.062273
        Training batch 22 Loss: 1.059003
        Training batch 23 Loss: 1.077982
Training set: Average loss: 1.550759
Validation set: Average loss: 1.055466, Accuracy: 188/480 (39%)

Epoch: 2
        Training batch 1 Loss: 1.050342
        Training batch 2 Loss: 1.063581
        Training batch 3 Loss: 1.054520
        Training batch 4 Loss: 1.046869
        Training batch 5 Loss: 1.038971
        Training batch 6 Loss: 1.048035
        Training batch 7 Loss: 1.027439
        Training batch 8 Loss: 1.036911
        Training batch 9 Loss: 1.034046
        Training batch 10 Loss: 1.036242
        Training batch 11 Loss: 1.050571
        Training batch 12 Loss: 1.011031
        Training batch 13 Loss: 1.062329
        Training batch 14 Loss: 1.001940
        Training batch 15 Loss: 0.993129
        Training batch 16 Loss: 0.998835
        Training batch 17 Loss: 1.033735
        Training batch 18 Loss: 1.024234
        Training batch 19 Loss: 0.991122
        Training batch 20 Loss: 1.053218
        Training batch 21 Loss: 0.995840
        Training batch 22 Loss: 1.007513
        Training batch 23 Loss: 1.032857
Training set: Average loss: 1.030144
Validation set: Average loss: 0.999519, Accuracy: 245/480 (51%)
```

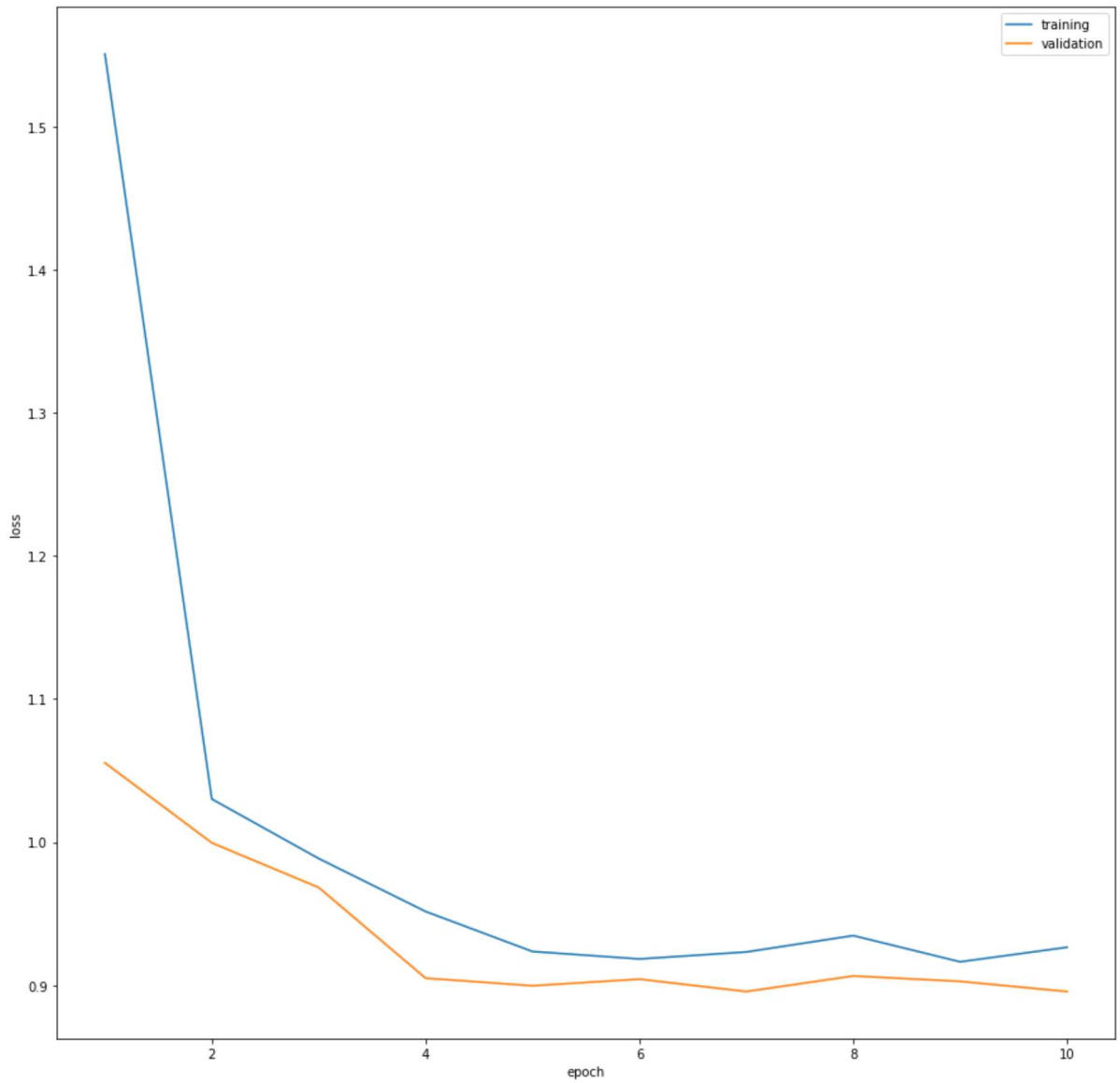
```
Epoch: 3
        Training batch 1 Loss: 0.994336
        Training batch 2 Loss: 1.015679
        Training batch 3 Loss: 1.011284
        Training batch 4 Loss: 1.001605
        Training batch 5 Loss: 0.987831
        Training batch 6 Loss: 1.001736
        Training batch 7 Loss: 0.971192
        Training batch 8 Loss: 0.981424
        Training batch 9 Loss: 0.976838
        Training batch 10 Loss: 0.993768
        Training batch 11 Loss: 1.028214
        Training batch 12 Loss: 0.952608
        Training batch 13 Loss: 1.053875
        Training batch 14 Loss: 0.956918
        Training batch 15 Loss: 0.927816
        Training batch 16 Loss: 0.948291
        Training batch 17 Loss: 1.009922
        Training batch 18 Loss: 0.999387
        Training batch 19 Loss: 0.943528
        Training batch 20 Loss: 1.040727
        Training batch 21 Loss: 0.953204
        Training batch 22 Loss: 0.977799
        Training batch 23 Loss: 1.008695
Training set: Average loss: 0.988551
Validation set: Average loss: 0.968334, Accuracy: 245/480 (51%)

Epoch: 4
        Training batch 1 Loss: 0.964604
        Training batch 2 Loss: 0.991592
        Training batch 3 Loss: 0.989584
        Training batch 4 Loss: 0.982001
        Training batch 5 Loss: 0.959337
        Training batch 6 Loss: 0.977046
        Training batch 7 Loss: 0.939636
        Training batch 8 Loss: 0.947451
        Training batch 9 Loss: 0.938653
        Training batch 10 Loss: 0.960661
        Training batch 11 Loss: 1.013195
        Training batch 12 Loss: 0.891105
        Training batch 13 Loss: 1.129086
        Training batch 14 Loss: 0.881569
        Training batch 15 Loss: 0.844952
        Training batch 16 Loss: 0.886647
        Training batch 17 Loss: 0.982460
        Training batch 18 Loss: 0.919104
        Training batch 19 Loss: 0.904690
        Training batch 20 Loss: 1.021703
        Training batch 21 Loss: 0.860197
        Training batch 22 Loss: 0.912573
        Training batch 23 Loss: 0.987971
Training set: Average loss: 0.951557
Validation set: Average loss: 0.904952, Accuracy: 247/480 (51%)

Epoch: 5
        Training batch 1 Loss: 0.890474
```

```
        Training batch 2 Loss: 0.935182
        Training batch 3 Loss: 0.942090
        Training batch 4 Loss: 0.972836
        Training batch 5 Loss: 0.913812
        Training batch 6 Loss: 0.932891
        Training batch 7 Loss: 0.891546
        Training batch 8 Loss: 0.894713
        Training batch 9 Loss: 0.891181
        Training batch 10 Loss: 0.925913
        Training batch 11 Loss: 0.955470
        Training batch 12 Loss: 0.857447
        Training batch 13 Loss: 1.062887
        Training batch 14 Loss: 0.903586
        Training batch 15 Loss: 0.828151
        Training batch 16 Loss: 0.908733
        Training batch 17 Loss: 0.964687
        Training batch 18 Loss: 0.932260
        Training batch 19 Loss: 0.829319
        Training batch 20 Loss: 1.013061
        Training batch 21 Loss: 0.870510
        Training batch 22 Loss: 0.935786
        Training batch 23 Loss: 0.989924
Training set: Average loss: 0.923585
Validation set: Average loss: 0.899660, Accuracy: 243/480 (51%)

Epoch: 6
        Training batch 1 Loss: 0.877857
        Training batch 2 Loss: 0.922191
        Training batch 3 Loss: 0.943797
        Training batch 4 Loss: 0.924589
        Training batch 5 Loss: 0.916004
        Training batch 6 Loss: 0.915510
        Training batch 7 Loss: 0.888234
        Training batch 8 Loss: 0.888425
        Training batch 9 Loss: 0.868337
        Training batch 10 Loss: 0.910656
        Training batch 11 Loss: 0.975796
        Training batch 12 Loss: 0.875953
        Training batch 13 Loss: 0.996527
        Training batch 14 Loss: 0.930859
        Training batch 15 Loss: 0.803886
        Training batch 16 Loss: 0.885981
        Training batch 17 Loss: 0.948836
        Training batch 18 Loss: 0.970482
        Training batch 19 Loss: 0.848507
        Training batch 20 Loss: 1.035215
        Training batch 21 Loss: 0.867724
        Training batch 22 Loss: 0.937450
        Training batch 23 Loss: 0.990197
Training set: Average loss: 0.918392
Validation set: Average loss: 0.904263, Accuracy: 246/480 (51%)

Epoch: 7
        Training batch 1 Loss: 0.887182
        Training batch 2 Loss: 0.952768
        Training batch 3 Loss: 0.998636
```

```
        Training batch 4 Loss: 0.916222
        Training batch 5 Loss: 0.902469
        Training batch 6 Loss: 0.942148
        Training batch 7 Loss: 0.891457
        Training batch 8 Loss: 0.885179
        Training batch 9 Loss: 0.888133
        Training batch 10 Loss: 0.929504
        Training batch 11 Loss: 0.988721
        Training batch 12 Loss: 0.846622
        Training batch 13 Loss: 1.040564
        Training batch 14 Loss: 0.860249
        Training batch 15 Loss: 0.839093
        Training batch 16 Loss: 0.873625
        Training batch 17 Loss: 0.998354
        Training batch 18 Loss: 0.909111
        Training batch 19 Loss: 0.847384
        Training batch 20 Loss: 0.982581
        Training batch 21 Loss: 0.865908
        Training batch 22 Loss: 0.939627
        Training batch 23 Loss: 1.049190
Training set: Average loss: 0.923249
Validation set: Average loss: 0.895608, Accuracy: 247/480 (51%)

Epoch: 8
        Training batch 1 Loss: 0.874805
        Training batch 2 Loss: 0.924720
        Training batch 3 Loss: 0.943519
        Training batch 4 Loss: 0.945917
        Training batch 5 Loss: 0.939919
        Training batch 6 Loss: 0.934571
        Training batch 7 Loss: 0.895993
        Training batch 8 Loss: 0.896708
        Training batch 9 Loss: 0.950069
        Training batch 10 Loss: 0.922997
        Training batch 11 Loss: 0.986954
        Training batch 12 Loss: 0.847706
        Training batch 13 Loss: 1.022613
        Training batch 14 Loss: 0.896389
        Training batch 15 Loss: 0.818286
        Training batch 16 Loss: 0.859656
        Training batch 17 Loss: 0.965619
        Training batch 18 Loss: 0.925676
        Training batch 19 Loss: 0.839019
        Training batch 20 Loss: 1.004312
        Training batch 21 Loss: 0.860743
        Training batch 22 Loss: 0.886322
        Training batch 23 Loss: 1.355627
Training set: Average loss: 0.934702
Validation set: Average loss: 0.906482, Accuracy: 244/480 (51%)

Epoch: 9
        Training batch 1 Loss: 0.844865
        Training batch 2 Loss: 0.895654
        Training batch 3 Loss: 0.942454
        Training batch 4 Loss: 0.927773
        Training batch 5 Loss: 0.920645
```

```
        Training batch 6 Loss: 0.925621
        Training batch 7 Loss: 0.883580
        Training batch 8 Loss: 0.873131
        Training batch 9 Loss: 0.929771
        Training batch 10 Loss: 0.939808
        Training batch 11 Loss: 1.003423
        Training batch 12 Loss: 0.890052
        Training batch 13 Loss: 1.006557
        Training batch 14 Loss: 0.891008
        Training batch 15 Loss: 0.824627
        Training batch 16 Loss: 0.883660
        Training batch 17 Loss: 0.966037
        Training batch 18 Loss: 0.939456
        Training batch 19 Loss: 0.876201
        Training batch 20 Loss: 1.026388
        Training batch 21 Loss: 0.838235
        Training batch 22 Loss: 0.902818
        Training batch 23 Loss: 0.945097
Training set: Average loss: 0.916385
Validation set: Average loss: 0.902716, Accuracy: 247/480 (51%)

Epoch: 10
        Training batch 1 Loss: 0.839526
        Training batch 2 Loss: 0.917476
        Training batch 3 Loss: 0.978043
        Training batch 4 Loss: 1.014013
        Training batch 5 Loss: 0.920749
        Training batch 6 Loss: 0.934658
        Training batch 7 Loss: 0.875565
        Training batch 8 Loss: 0.907640
        Training batch 9 Loss: 0.891620
        Training batch 10 Loss: 0.930371
        Training batch 11 Loss: 0.993154
        Training batch 12 Loss: 0.900367
        Training batch 13 Loss: 1.033256
        Training batch 14 Loss: 0.886373
        Training batch 15 Loss: 0.834219
        Training batch 16 Loss: 0.852029
        Training batch 17 Loss: 0.946649
        Training batch 18 Loss: 0.931650
        Training batch 19 Loss: 0.803738
        Training batch 20 Loss: 1.024390
        Training batch 21 Loss: 0.831973
        Training batch 22 Loss: 0.869175
        Training batch 23 Loss: 1.193535
Training set: Average loss: 0.926529
Validation set: Average loss: 0.895606, Accuracy: 251/480 (52%)
```

In [24]:
```
plt.figure(figsize=(15,15))
plt.plot(epoch_nums, training_loss)
plt.plot(epoch_nums, validation_loss)
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['training', 'validation'], loc='upper right')
plt.show()
```



In [30]:
```
from tensorflow.keras.layers import Dense, Flatten, Input, Lambda
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import preprocess_input
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from keras.models import Sequential
from glob import glob
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
```

In [26]:
```
# Resizinig all the images to (224,224)
IMAGE_SIZE = [224,224]

train_path1 = 'data/train'
test_path1 = 'data/test'
```

In [27]:
```
# Scaling all the images between 0 to 1

train_datagen1 = ImageDataGenerator(rescale = 1./255, shear_range=0.2, zoom_ra
nge=0.2, horizontal_flip=False)

# Performing only scaling on the test dataset

test_datagen1 = ImageDataGenerator(rescale=1./255)
```

In [28]:
```
train_set1 = train_datagen1.flow_from_directory(train_path1,
                                                target_size=(224,224),
                                                batch_size=32,
                                                class_mode = 'categorical')

test_set1 = test_datagen1.flow_from_directory(test_path1,
                                              target_size=(224,224),
                                              batch_size=32,
                                              class_mode='categorical')
```

```
Found 1597 images belonging to 3 classes.
Found 731 images belonging to 3 classes.
```

ResNet

In [31]:
```
resnet = ResNet50(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_
top=False)
```

In [32]:
```
x = Flatten()(resnet.output)
prediction = Dense(3, activation='softmax')(x)
model = Model(inputs = resnet.inputs, outputs = prediction)
model.summary()
```

```
Model: "functional_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_1[0]
[0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0]
[0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv
[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0]
[0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu
[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0]
[0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool
[0][0]
__________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1
_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1
_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block1
_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1
_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1
_2_bn[0][0]
```

```
__________________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool
[0][0]
__________________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block1
_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1
_0_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block1
_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_add (Add)          (None, 56, 56, 256)  0           conv2_block1
_0_bn[0][0]
                                                                 conv2_block1
_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_out (Activation)   (None, 56, 56, 256)  0           conv2_block1
_add[0][0]
__________________________________________________________________________________________________
conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block1
_out[0][0]
__________________________________________________________________________________________________
conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2
_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0           conv2_block2
_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block2
_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block2
_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0           conv2_block2
_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block2
_2_relu[0][0]
__________________________________________________________________________________________________
```

```
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block2
_3_conv[0][0]
__________________________________________________________________________
_____________________
conv2_block2_add (Add)          (None, 56, 56, 256)  0           conv2_block1
_out[0][0]
                                                                 conv2_block2
_3_bn[0][0]
__________________________________________________________________________
_____________________
conv2_block2_out (Activation)   (None, 56, 56, 256)  0           conv2_block2
_add[0][0]
__________________________________________________________________________
_____________________
conv2_block3_1_conv (Conv2D)    (None, 56, 56, 64)   16448       conv2_block2
_out[0][0]
__________________________________________________________________________
_____________________
conv2_block3_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block3
_1_conv[0][0]
__________________________________________________________________________
_____________________
conv2_block3_1_relu (Activation (None, 56, 56, 64)   0           conv2_block3
_1_bn[0][0]
__________________________________________________________________________
_____________________
conv2_block3_2_conv (Conv2D)    (None, 56, 56, 64)   36928       conv2_block3
_1_relu[0][0]
__________________________________________________________________________
_____________________
conv2_block3_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block3
_2_conv[0][0]
__________________________________________________________________________
_____________________
conv2_block3_2_relu (Activation (None, 56, 56, 64)   0           conv2_block3
_2_bn[0][0]
__________________________________________________________________________
_____________________
conv2_block3_3_conv (Conv2D)    (None, 56, 56, 256)  16640       conv2_block3
_2_relu[0][0]
__________________________________________________________________________
_____________________
conv2_block3_3_bn (BatchNormali (None, 56, 56, 256)  1024        conv2_block3
_3_conv[0][0]
__________________________________________________________________________
_____________________
conv2_block3_add (Add)          (None, 56, 56, 256)  0           conv2_block2
_out[0][0]
                                                                 conv2_block3
_3_bn[0][0]
__________________________________________________________________________
_____________________
conv2_block3_out (Activation)   (None, 56, 56, 256)  0           conv2_block3
_add[0][0]
__________________________________________________________________________
_____________________
```

```
conv3_block1_1_conv (Conv2D)    (None, 28, 28, 128)  32896       conv2_block3
_out[0][0]
__________________________________________________________________________
_____________________
conv3_block1_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block1
_1_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block1_1_relu (Activation (None, 28, 28, 128)  0           conv3_block1
_1_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block1_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block1
_1_relu[0][0]
__________________________________________________________________________
_____________________
conv3_block1_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block1
_2_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block1_2_relu (Activation (None, 28, 28, 128)  0           conv3_block1
_2_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block1_0_conv (Conv2D)    (None, 28, 28, 512)  131584      conv2_block3
_out[0][0]
__________________________________________________________________________
_____________________
conv3_block1_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block1
_2_relu[0][0]
__________________________________________________________________________
_____________________
conv3_block1_0_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block1
_0_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block1_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block1
_3_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block1_add (Add)          (None, 28, 28, 512)  0           conv3_block1
_0_bn[0][0]
                                                                 conv3_block1
_3_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block1_out (Activation)   (None, 28, 28, 512)  0           conv3_block1
_add[0][0]
__________________________________________________________________________
_____________________
conv3_block2_1_conv (Conv2D)    (None, 28, 28, 128)  65664       conv3_block1
_out[0][0]
__________________________________________________________________________
_____________________
conv3_block2_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block2
_1_conv[0][0]
```

```
__________________________________________________________________________________________________
__________________
conv3_block2_1_relu (Activation (None, 28, 28, 128)  0           conv3_block2
_1_bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block2
_1_relu[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block2
_2_conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_2_relu (Activation (None, 28, 28, 128)  0           conv3_block2
_2_bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block2
_2_relu[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block2
_3_conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_add (Add)          (None, 28, 28, 512)  0           conv3_block1
_out[0][0]
                                                                 conv3_block2
_3_bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block2_out (Activation)   (None, 28, 28, 512)  0           conv3_block2
_add[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_1_conv (Conv2D)    (None, 28, 28, 128)  65664       conv3_block2
_out[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block3
_1_conv[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_1_relu (Activation (None, 28, 28, 128)  0           conv3_block3
_1_bn[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block3
_1_relu[0][0]
__________________________________________________________________________________________________
__________________
conv3_block3_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block3
_2_conv[0][0]
__________________________________________________________________________________________________
__________________
```

```
conv3_block3_2_relu (Activation (None, 28, 28, 128)  0           conv3_block3
_2_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block3_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block3
_2_relu[0][0]
__________________________________________________________________________
_____________________
conv3_block3_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block3
_3_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block3_add (Add)          (None, 28, 28, 512)  0           conv3_block2
_out[0][0]
                                                                 conv3_block3
_3_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block3_out (Activation)   (None, 28, 28, 512)  0           conv3_block3
_add[0][0]
__________________________________________________________________________
_____________________
conv3_block4_1_conv (Conv2D)    (None, 28, 28, 128)  65664       conv3_block3
_out[0][0]
__________________________________________________________________________
_____________________
conv3_block4_1_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block4
_1_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block4_1_relu (Activation (None, 28, 28, 128)  0           conv3_block4
_1_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block4_2_conv (Conv2D)    (None, 28, 28, 128)  147584      conv3_block4
_1_relu[0][0]
__________________________________________________________________________
_____________________
conv3_block4_2_bn (BatchNormali (None, 28, 28, 128)  512         conv3_block4
_2_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block4_2_relu (Activation (None, 28, 28, 128)  0           conv3_block4
_2_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block4_3_conv (Conv2D)    (None, 28, 28, 512)  66048       conv3_block4
_2_relu[0][0]
__________________________________________________________________________
_____________________
conv3_block4_3_bn (BatchNormali (None, 28, 28, 512)  2048        conv3_block4
_3_conv[0][0]
__________________________________________________________________________
_____________________
conv3_block4_add (Add)          (None, 28, 28, 512)  0           conv3_block3
_out[0][0]
```

```
                                                                 conv3_block4
_3_bn[0][0]
__________________________________________________________________________
_____________________
conv3_block4_out (Activation)   (None, 28, 28, 512)  0           conv3_block4
_add[0][0]
__________________________________________________________________________
_____________________
conv4_block1_1_conv (Conv2D)    (None, 14, 14, 256)  131328      conv3_block4
_out[0][0]
__________________________________________________________________________
_____________________
conv4_block1_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block1
_1_conv[0][0]
__________________________________________________________________________
_____________________
conv4_block1_1_relu (Activation (None, 14, 14, 256)  0           conv4_block1
_1_bn[0][0]
__________________________________________________________________________
_____________________
conv4_block1_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block1
_1_relu[0][0]
__________________________________________________________________________
_____________________
conv4_block1_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block1
_2_conv[0][0]
__________________________________________________________________________
_____________________
conv4_block1_2_relu (Activation (None, 14, 14, 256)  0           conv4_block1
_2_bn[0][0]
__________________________________________________________________________
_____________________
conv4_block1_0_conv (Conv2D)    (None, 14, 14, 1024) 525312      conv3_block4
_out[0][0]
__________________________________________________________________________
_____________________
conv4_block1_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block1
_2_relu[0][0]
__________________________________________________________________________
_____________________
conv4_block1_0_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block1
_0_conv[0][0]
__________________________________________________________________________
_____________________
conv4_block1_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block1
_3_conv[0][0]
__________________________________________________________________________
_____________________
conv4_block1_add (Add)          (None, 14, 14, 1024) 0           conv4_block1
_0_bn[0][0]
                                                                 conv4_block1
_3_bn[0][0]
__________________________________________________________________________
_____________________
conv4_block1_out (Activation)   (None, 14, 14, 1024) 0           conv4_block1
_add[0][0]
```

```
__________________________________________________________________________________________________
conv4_block2_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block1
_out[0][0]
__________________________________________________________________________________________________
conv4_block2_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block2
_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_1_relu (Activation (None, 14, 14, 256)  0           conv4_block2
_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block2
_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block2_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block2
_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_2_relu (Activation (None, 14, 14, 256)  0           conv4_block2
_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block2
_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block2_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block2
_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_add (Add)          (None, 14, 14, 1024) 0           conv4_block1
_out[0][0]
                                                                 conv4_block2
_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_out (Activation)   (None, 14, 14, 1024) 0           conv4_block2
_add[0][0]
__________________________________________________________________________________________________
conv4_block3_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block2
_out[0][0]
__________________________________________________________________________________________________
conv4_block3_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3
_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_1_relu (Activation (None, 14, 14, 256)  0           conv4_block3
_1_bn[0][0]
__________________________________________________________________________________________________
```

```
conv4_block3_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block3
_1_relu[0][0]
__________________________________________________________________________________
__________________
conv4_block3_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3
_2_conv[0][0]
__________________________________________________________________________________
__________________
conv4_block3_2_relu (Activation (None, 14, 14, 256)  0           conv4_block3
_2_bn[0][0]
__________________________________________________________________________________
__________________
conv4_block3_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block3
_2_relu[0][0]
__________________________________________________________________________________
__________________
conv4_block3_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block3
_3_conv[0][0]
__________________________________________________________________________________
__________________
conv4_block3_add (Add)          (None, 14, 14, 1024) 0           conv4_block2
_out[0][0]
                                                                 conv4_block3
_3_bn[0][0]
__________________________________________________________________________________
__________________
conv4_block3_out (Activation)   (None, 14, 14, 1024) 0           conv4_block3
_add[0][0]
__________________________________________________________________________________
__________________
conv4_block4_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block3
_out[0][0]
__________________________________________________________________________________
__________________
conv4_block4_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block4
_1_conv[0][0]
__________________________________________________________________________________
__________________
conv4_block4_1_relu (Activation (None, 14, 14, 256)  0           conv4_block4
_1_bn[0][0]
__________________________________________________________________________________
__________________
conv4_block4_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block4
_1_relu[0][0]
__________________________________________________________________________________
__________________
conv4_block4_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block4
_2_conv[0][0]
__________________________________________________________________________________
__________________
conv4_block4_2_relu (Activation (None, 14, 14, 256)  0           conv4_block4
_2_bn[0][0]
__________________________________________________________________________________
__________________
conv4_block4_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block4
_2_relu[0][0]
```

```
__________________________________________________________________________________________________
____________________
conv4_block4_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block4
_3_conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block4_add (Add)          (None, 14, 14, 1024) 0           conv4_block3
_out[0][0]
                                                                 conv4_block4
_3_bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block4_out (Activation)   (None, 14, 14, 1024) 0           conv4_block4
_add[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block4
_out[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block5
_1_conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_1_relu (Activation (None, 14, 14, 256)  0           conv4_block5
_1_bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block5
_1_relu[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block5
_2_conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_2_relu (Activation (None, 14, 14, 256)  0           conv4_block5
_2_bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block5
_2_relu[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block5
_3_conv[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_add (Add)          (None, 14, 14, 1024) 0           conv4_block4
_out[0][0]
                                                                 conv4_block5
_3_bn[0][0]
__________________________________________________________________________________________________
____________________
conv4_block5_out (Activation)   (None, 14, 14, 1024) 0           conv4_block5
_add[0][0]
```

```
__________________________________________________________________________________________________
conv4_block6_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block5
_out[0][0]
__________________________________________________________________________________________________
conv4_block6_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block6
_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_1_relu (Activation (None, 14, 14, 256)  0           conv4_block6
_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block6
_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block6_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block6
_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_2_relu (Activation (None, 14, 14, 256)  0           conv4_block6
_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block6
_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block6_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block6
_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_add (Add)          (None, 14, 14, 1024) 0           conv4_block5
_out[0][0]
                                                                 conv4_block6
_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_out (Activation)   (None, 14, 14, 1024) 0           conv4_block6
_add[0][0]
__________________________________________________________________________________________________
conv5_block1_1_conv (Conv2D)    (None, 7, 7, 512)    524800      conv4_block6
_out[0][0]
__________________________________________________________________________________________________
conv5_block1_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block1
_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_1_relu (Activation (None, 7, 7, 512)    0           conv5_block1
_1_bn[0][0]
__________________________________________________________________________________________________
```

```
conv5_block1_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block1
_1_relu[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block1
_2_conv[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_2_relu (Activation (None, 7, 7, 512)    0           conv5_block1
_2_bn[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_0_conv (Conv2D)    (None, 7, 7, 2048)   2099200     conv4_block6
_out[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block1
_2_relu[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_0_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block1
_0_conv[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block1
_3_conv[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_add (Add)          (None, 7, 7, 2048)   0           conv5_block1
_0_bn[0][0]
                                                                 conv5_block1
_3_bn[0][0]
__________________________________________________________________________________
_____________________
conv5_block1_out (Activation)   (None, 7, 7, 2048)   0           conv5_block1
_add[0][0]
__________________________________________________________________________________
_____________________
conv5_block2_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_block1
_out[0][0]
__________________________________________________________________________________
_____________________
conv5_block2_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block2
_1_conv[0][0]
__________________________________________________________________________________
_____________________
conv5_block2_1_relu (Activation (None, 7, 7, 512)    0           conv5_block2
_1_bn[0][0]
__________________________________________________________________________________
_____________________
conv5_block2_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block2
_1_relu[0][0]
__________________________________________________________________________________
_____________________
conv5_block2_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block2
_2_conv[0][0]
```

```
__________________________________________________________________________________________________
conv5_block2_2_relu (Activation (None, 7, 7, 512)    0           conv5_block2
_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block2
_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block2
_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_add (Add)          (None, 7, 7, 2048)   0           conv5_block1
_out[0][0]
                                                                 conv5_block2
_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_out (Activation)   (None, 7, 7, 2048)   0           conv5_block2
_add[0][0]
__________________________________________________________________________________________________
conv5_block3_1_conv (Conv2D)    (None, 7, 7, 512)    1049088     conv5_block2
_out[0][0]
__________________________________________________________________________________________________
conv5_block3_1_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3
_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_1_relu (Activation (None, 7, 7, 512)    0           conv5_block3
_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_2_conv (Conv2D)    (None, 7, 7, 512)    2359808     conv5_block3
_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, 7, 7, 512)    2048        conv5_block3
_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 7, 7, 512)    0           conv5_block3
_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624     conv5_block3
_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192        conv5_block3
_3_conv[0][0]
__________________________________________________________________________________________________
```

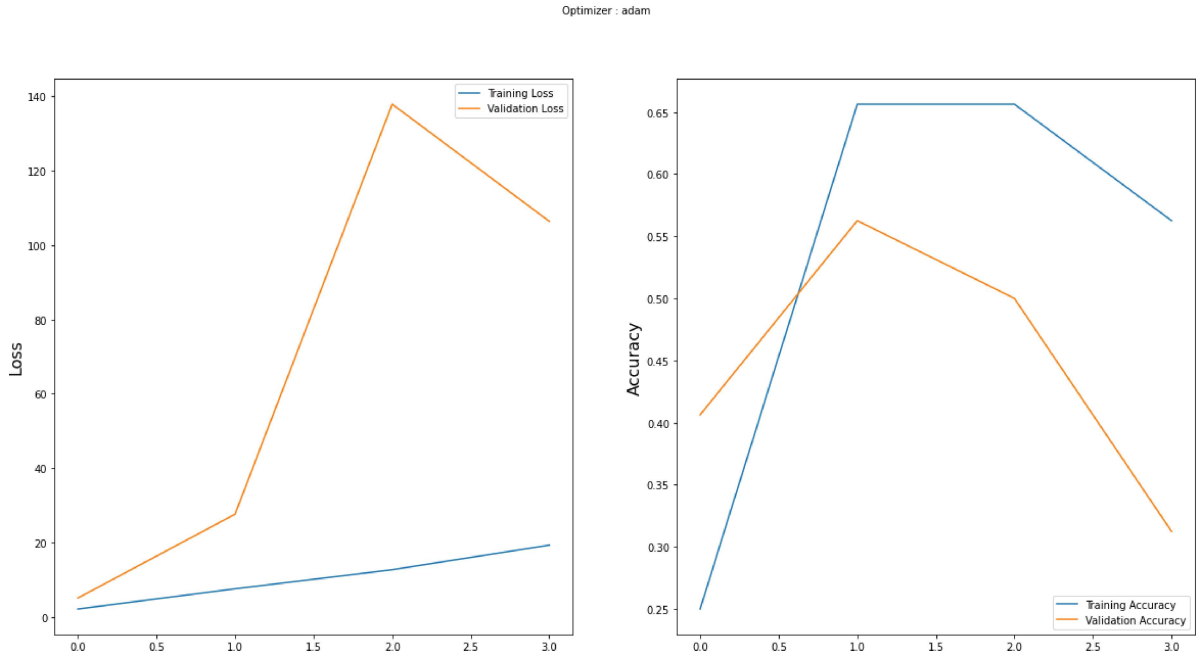
```
conv5_block3_add (Add)          (None, 7, 7, 2048)   0           conv5_block2
_out[0][0]
                                                                 conv5_block3
_3_bn[0][0]
__________________________________________________________________________________
_____________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0           conv5_block3
_add[0][0]
__________________________________________________________________________________
_____________________
flatten (Flatten)               (None, 100352)       0           conv5_block3
_out[0][0]
__________________________________________________________________________________
_____________________
dense (Dense)                   (None, 3)            301059      flatten[0]
[0]
==================================================================================
====================
Total params: 23,888,771
Trainable params: 23,835,651
Non-trainable params: 53,120
__________________________________________________________________________________
_____________________
```

In [33]:
```
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['a
ccuracy'])
```

In [34]:
```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
hist = model.fit_generator(train_set1, validation_data=test_set1, epochs=20, s
teps_per_epoch=1, validation_steps=1,callbacks=[callback])
```

```
WARNING:tensorflow:From <ipython-input-34-8210a643700c>:2: Model.fit_generato
r (from tensorflow.python.keras.engine.training) is deprecated and will be re
moved in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/20
1/1 [==============================] - 2s 2s/step - loss: 2.2548 - accuracy:
0.2500 - val_loss: 5.2061 - val_accuracy: 0.4062
Epoch 2/20
1/1 [==============================] - 2s 2s/step - loss: 7.6832 - accuracy:
0.6562 - val_loss: 27.7021 - val_accuracy: 0.5625
Epoch 3/20
1/1 [==============================] - 2s 2s/step - loss: 12.8062 - accuracy:
0.6562 - val_loss: 137.7536 - val_accuracy: 0.5000
Epoch 4/20
1/1 [==============================] - 2s 2s/step - loss: 19.3936 - accuracy:
0.5625 - val_loss: 106.3089 - val_accuracy: 0.3125
```

In [35]:
```
import matplotlib.pyplot as plt

x=hist
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



VGG16

In [36]:
```
from tensorflow.keras.applications.vgg16 import VGG16
vgg16=VGG16(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

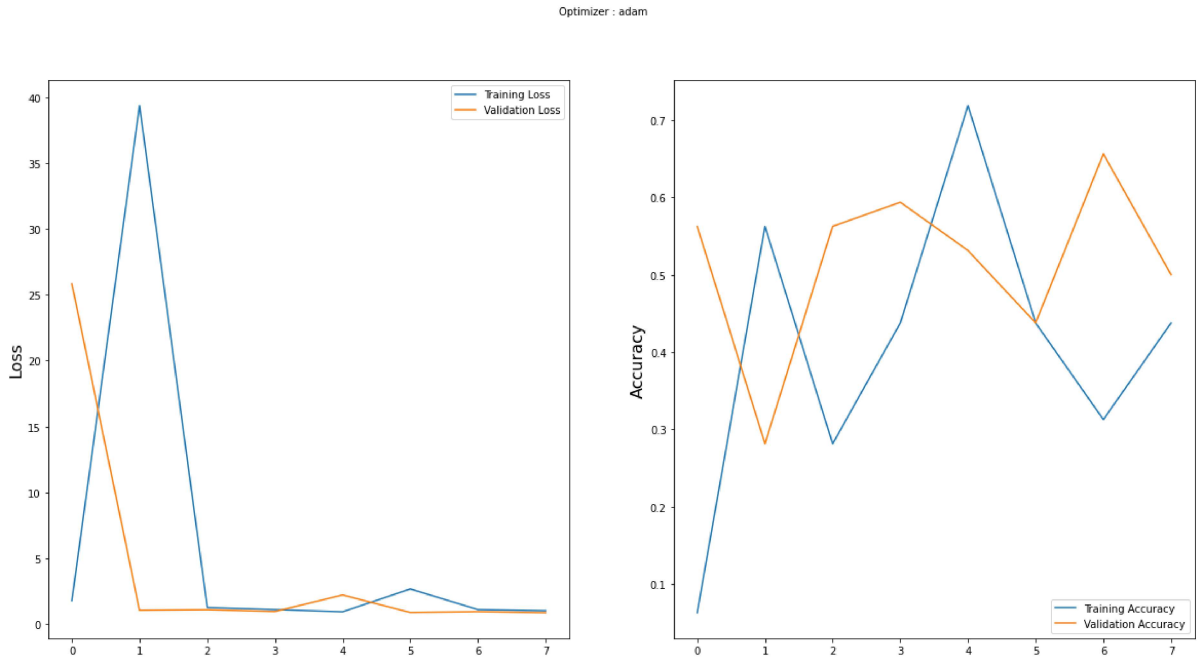
In [37]:
```
x1= Flatten()(vgg16.output)
prediction1 = Dense(3, activation='softmax')(x1)
model1 = Model(inputs = vgg16.inputs, outputs = prediction1)
model1.summary()
model1.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=[
'accuracy'])
```

```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 75267
=================================================================
Total params: 14,789,955
Trainable params: 14,789,955
Non-trainable params: 0
_________________________________________________________________
```

In [38]:
```
r1 = model1.fit_generator(train_set1, validation_data=test_set1, epochs=20, st
eps_per_epoch=1, validation_steps=1,callbacks=[callback])
```

```
Epoch 1/20
1/1 [==============================] - 3s 3s/step - loss: 1.7697 - accuracy:
0.0625 - val_loss: 25.8421 - val_accuracy: 0.5625
Epoch 2/20
1/1 [==============================] - 3s 3s/step - loss: 39.3716 - accuracy:
0.5625 - val_loss: 1.0446 - val_accuracy: 0.2812
Epoch 3/20
1/1 [==============================] - 3s 3s/step - loss: 1.2524 - accuracy:
0.2812 - val_loss: 1.0819 - val_accuracy: 0.5625
Epoch 4/20
1/1 [==============================] - 3s 3s/step - loss: 1.0954 - accuracy:
0.4375 - val_loss: 0.9470 - val_accuracy: 0.5938
Epoch 5/20
1/1 [==============================] - 3s 3s/step - loss: 0.9251 - accuracy:
0.7188 - val_loss: 2.2079 - val_accuracy: 0.5312
Epoch 6/20
1/1 [==============================] - 3s 3s/step - loss: 2.6633 - accuracy:
0.4375 - val_loss: 0.8720 - val_accuracy: 0.4375
Epoch 7/20
1/1 [==============================] - 3s 3s/step - loss: 1.0931 - accuracy:
0.3125 - val_loss: 0.9329 - val_accuracy: 0.6562
Epoch 8/20
1/1 [==============================] - 3s 3s/step - loss: 1.0042 - accuracy:
0.4375 - val_loss: 0.8525 - val_accuracy: 0.5000
```

In [39]:
```
import matplotlib.pyplot as plt

x=r1
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



MobileNet

In [40]:
```
from tensorflow.keras.applications import MobileNet, MobileNetV2
mob = MobileNet(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_to
p=False)
```

In [41]:
```
x1= Flatten()(mob.output)
prediction1 = Dense(3, activation='softmax')(x1)
model12 = Model(inputs = mob.inputs, outputs = prediction1)
model12.summary()
model12.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=[
'accuracy'])
```

Model: "functional_5"

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 224, 224, 3)]	0
conv1_pad (ZeroPadding2D)	(None, 225, 225, 3)	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0
conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 64)	576
conv_dw_2_bn (BatchNormaliza	(None, 56, 56, 64)	256
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8192
conv_pw_2_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_pw_2_relu (ReLU)	(None, 56, 56, 128)	0
conv_dw_3 (DepthwiseConv2D)	(None, 56, 56, 128)	1152
conv_dw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_dw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pw_3 (Conv2D)	(None, 56, 56, 128)	16384
conv_pw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_pw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pad_4 (ZeroPadding2D)	(None, 57, 57, 128)	0
conv_dw_4 (DepthwiseConv2D)	(None, 28, 28, 128)	1152

conv_dw_4_bn (BatchNormaliza	(None, 28, 28, 128)	512
conv_dw_4_relu (ReLU)	(None, 28, 28, 128)	0
conv_pw_4 (Conv2D)	(None, 28, 28, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_4_relu (ReLU)	(None, 28, 28, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, 28, 28, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_dw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pw_5 (Conv2D)	(None, 28, 28, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pad_6 (ZeroPadding2D)	(None, 29, 29, 256)	0
conv_dw_6 (DepthwiseConv2D)	(None, 14, 14, 256)	2304
conv_dw_6_bn (BatchNormaliza	(None, 14, 14, 256)	1024
conv_dw_6_relu (ReLU)	(None, 14, 14, 256)	0
conv_pw_6 (Conv2D)	(None, 14, 14, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_6_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_7 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_8 (Conv2D)	(None, 14, 14, 512)	262144

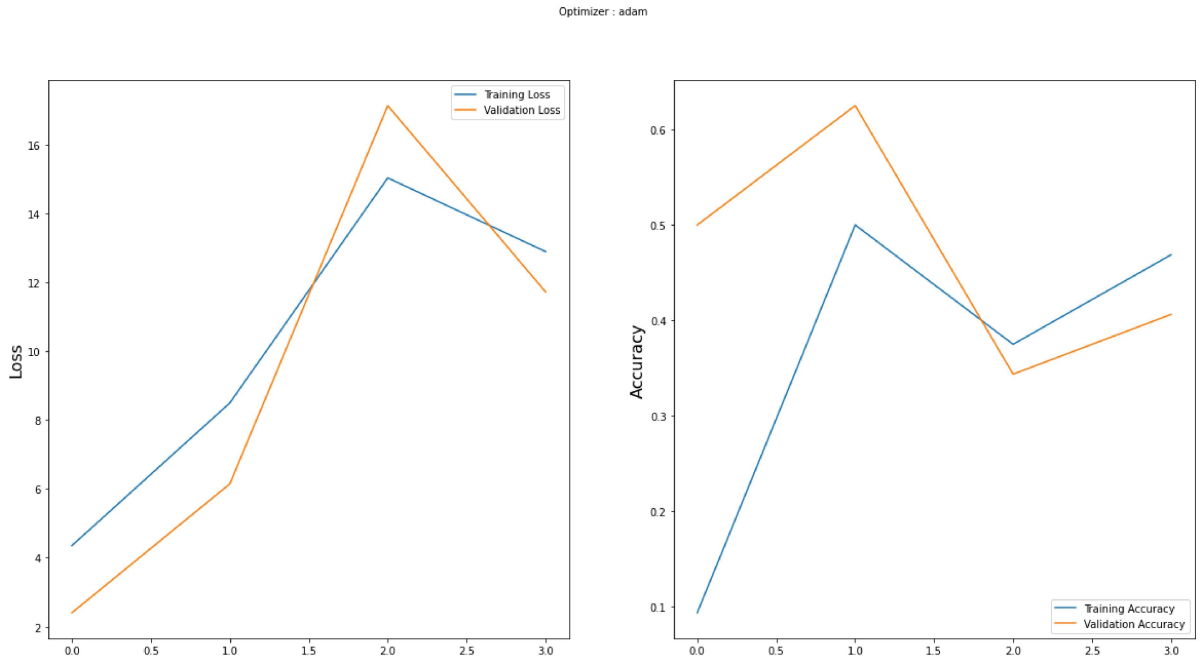
conv_pw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_9 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_10_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_dw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_10 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_10_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_10_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_11 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_dw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_11 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_11_bn (BatchNormaliz	(None, 14, 14, 512)	2048
conv_pw_11_relu (ReLU)	(None, 14, 14, 512)	0
conv_pad_12 (ZeroPadding2D)	(None, 15, 15, 512)	0
conv_dw_12 (DepthwiseConv2D)	(None, 7, 7, 512)	4608
conv_dw_12_bn (BatchNormaliz	(None, 7, 7, 512)	2048
conv_dw_12_relu (ReLU)	(None, 7, 7, 512)	0
conv_pw_12 (Conv2D)	(None, 7, 7, 1024)	524288
conv_pw_12_bn (BatchNormaliz	(None, 7, 7, 1024)	4096
conv_pw_12_relu (ReLU)	(None, 7, 7, 1024)	0
conv_dw_13 (DepthwiseConv2D)	(None, 7, 7, 1024)	9216

```
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
conv_dw_13_relu (ReLU)       (None, 7, 7, 1024)        0
conv_pw_13 (Conv2D)          (None, 7, 7, 1024)        1048576
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0
flatten_2 (Flatten)          (None, 50176)             0
dense_2 (Dense)              (None, 3)                 150531
=================================================================
Total params: 3,379,395
Trainable params: 3,357,507
Non-trainable params: 21,888
```

In [42]:
```
r1 = model12.fit_generator(train_set1, validation_data=test_set1, epochs=20, s
teps_per_epoch=1, validation_steps=1,callbacks=[callback])
```

```
Epoch 1/20
1/1 [==============================] - 1s 864ms/step - loss: 4.3529 - accurac
y: 0.0938 - val_loss: 2.4019 - val_accuracy: 0.5000
Epoch 2/20
1/1 [==============================] - 1s 511ms/step - loss: 8.4989 - accurac
y: 0.5000 - val_loss: 6.1466 - val_accuracy: 0.6250
Epoch 3/20
1/1 [==============================] - 1s 512ms/step - loss: 15.0379 - accura
cy: 0.3750 - val_loss: 17.1378 - val_accuracy: 0.3438
Epoch 4/20
1/1 [==============================] - 1s 527ms/step - loss: 12.8950 - accura
cy: 0.4688 - val_loss: 11.7228 - val_accuracy: 0.4062
```

In [43]:
```
import matplotlib.pyplot as plt

x=r1
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(x.history['accuracy'], label='Training Accuracy')
plt.plot(x.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



In [44]:
```
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import LeakyReLU
from keras.layers import Dropout
from keras.layers import Flatten
from keras.optimizers import Adam
from keras.utils.vis_utils import plot_model
```

In [45]:
```
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential, layers
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
```

GAN

In [46]:
```
def define_discriminator(in_shape=(224,224,3)):
        # image input
        in_image = Input(shape=in_shape)
        # downsample
        fe = Conv2D(128, (3,3), strides=(2,2), padding='same')(in_image)
        fe = LeakyReLU(alpha=0.2)(fe)
        # downsample
        fe = Conv2D(128, (3,3), strides=(2,2), padding='same')(fe)
        fe = LeakyReLU(alpha=0.2)(fe)
        # downsample
        fe = Conv2D(128, (3,3), strides=(2,2), padding='same')(fe)
        fe = LeakyReLU(alpha=0.2)(fe)
        # flatten feature maps
        fe = Flatten()(fe)
        # dropout
        fe = Dropout(0.4)(fe)
        # output layer
        d_out_layer = Dense(3, activation='sigmoid')(fe)
        # define and compile discriminator model
        d_model = Model(in_image, d_out_layer)
        d_model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0
002, beta_1=0.5))
        return d_model
```

In [47]:
```
model = define_discriminator()
```

In [48]:
```
model.summary()
```

```
Model: "functional_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 112, 112, 128)     3584
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 112, 112, 128)     0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 56, 56, 128)       147584
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 128)       147584
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 28, 28, 128)       0
_________________________________________________________________
flatten_3 (Flatten)          (None, 100352)            0
_________________________________________________________________
dropout (Dropout)            (None, 100352)            0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 301059
=================================================================
Total params: 599,811
Trainable params: 599,811
Non-trainable params: 0
_________________________________________________________________
```

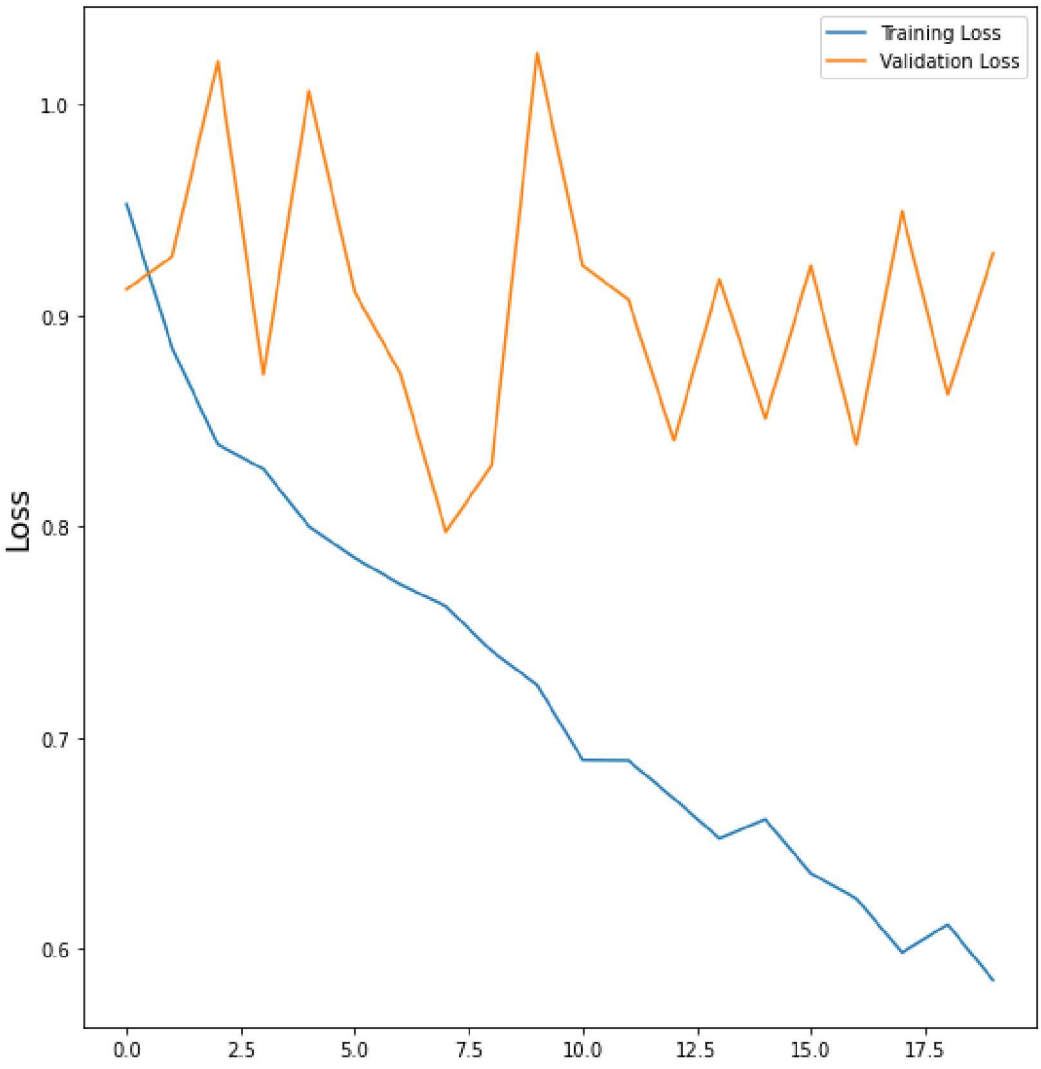
In [49]:
```
r1 = model.fit(train_set1, validation_data=test_set1, epochs=20)
```

```
Epoch 1/20
50/50 [==============================] - 56s 1s/step - loss: 0.9529 - val_los
s: 0.9124
Epoch 2/20
50/50 [==============================] - 55s 1s/step - loss: 0.8847 - val_los
s: 0.9280
Epoch 3/20
50/50 [==============================] - 54s 1s/step - loss: 0.8389 - val_los
s: 1.0205
Epoch 4/20
50/50 [==============================] - 54s 1s/step - loss: 0.8273 - val_los
s: 0.8721
Epoch 5/20
50/50 [==============================] - 54s 1s/step - loss: 0.8000 - val_los
s: 1.0064
Epoch 6/20
50/50 [==============================] - 54s 1s/step - loss: 0.7852 - val_los
s: 0.9111
Epoch 7/20
50/50 [==============================] - 54s 1s/step - loss: 0.7727 - val_los
s: 0.8726
Epoch 8/20
50/50 [==============================] - 54s 1s/step - loss: 0.7622 - val_los
s: 0.7975
Epoch 9/20
50/50 [==============================] - 55s 1s/step - loss: 0.7411 - val_los
s: 0.8290
Epoch 10/20
50/50 [==============================] - 54s 1s/step - loss: 0.7249 - val_los
s: 1.0244
Epoch 11/20
50/50 [==============================] - 54s 1s/step - loss: 0.6894 - val_los
s: 0.9237
Epoch 12/20
50/50 [==============================] - 54s 1s/step - loss: 0.6892 - val_los
s: 0.9076
Epoch 13/20
50/50 [==============================] - 54s 1s/step - loss: 0.6711 - val_los
s: 0.8408
Epoch 14/20
50/50 [==============================] - 55s 1s/step - loss: 0.6522 - val_los
s: 0.9172
Epoch 15/20
50/50 [==============================] - 55s 1s/step - loss: 0.6613 - val_los
s: 0.8511
Epoch 16/20
50/50 [==============================] - 58s 1s/step - loss: 0.6357 - val_los
s: 0.9236
Epoch 17/20
50/50 [==============================] - 56s 1s/step - loss: 0.6238 - val_los
s: 0.8388
Epoch 18/20
50/50 [==============================] - 55s 1s/step - loss: 0.5982 - val_los
s: 0.9494
Epoch 19/20
```

```
50/50 [==============================] - 54s 1s/step - loss: 0.6115 - val_los
s: 0.8626
Epoch 20/20
50/50 [==============================] - 54s 1s/step - loss: 0.5851 - val_los
s: 0.9296
```

In [50]:
```
import matplotlib.pyplot as plt

x= r1
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(x.history['loss'], label='Training Loss')
plt.plot(x.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')
```

Out[50]: <matplotlib.legend.Legend at 0x21351603e88>



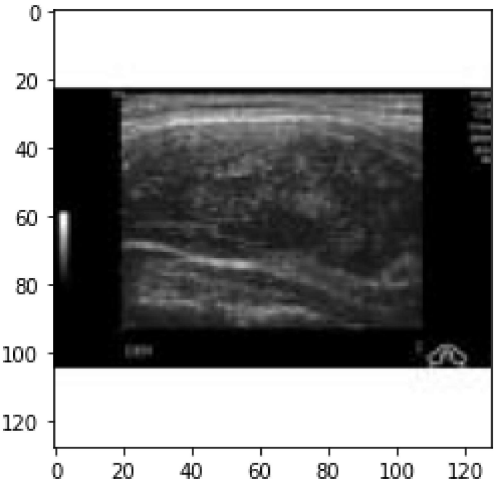
Machine Learning

In [51]:
```
import numpy as np # linear algebra
import pandas as pd
```

In [52]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import cv2
import os
from tqdm import tqdm
```

In [53]:
```
DATADIR = 'dataset/train'
CATEGORIES = ['benign','malignant','normal']
IMG_SIZE=100
```

In [54]:
```
for category in CATEGORIES:
    path=os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array=cv2.imread(os.path.join(path,img))
        plt.imshow(img_array)
        plt.show()
        break
    break
```



In [55]:
```
training_data=[]
def create_training_data():
    for category in CATEGORIES:
        path=os.path.join(DATADIR, category)
        class_num=CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array=cv2.imread(os.path.join(path,img))
                new_array=cv2.resize(img_array,(IMG_SIZE,IMG_SIZE))
                training_data.append([new_array,class_num])
            except Exception as e:
                pass
create_training_data()
```

In [56]:
```
lenofimage = len(training_data)
```

In [57]:
```
X=[]
y=[]

for categories, label in training_data:
    X.append(categories)
    y.append(label)
X= np.array(X).reshape(lenofimage,-1)
```

In [58]:
```
X = X/255.0
```

In [59]:
```
y=np.array(y)
```

In [60]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

In [61]:
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

In [62]:
```
kf = KFold(n_splits=5,shuffle=True, random_state=123)
```

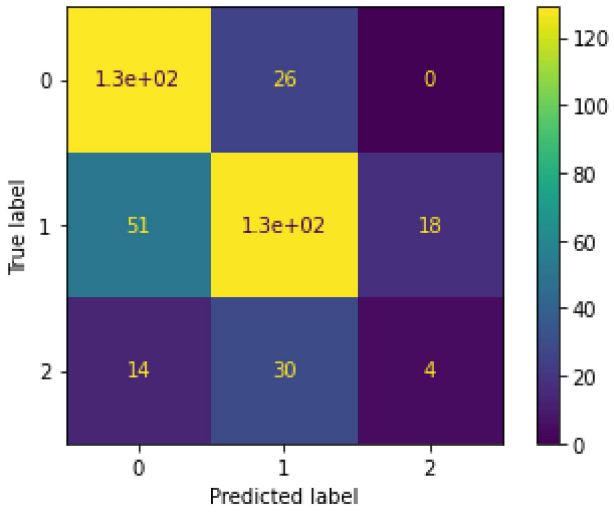
KNeighbors Classifier

In [63]:
```
from sklearn.neighbors import KNeighborsClassifier
clf1 = KNeighborsClassifier(n_neighbors=3)
cv_scores = cross_val_score(clf1,X_train,y_train,cv=kf)
cv_scores.mean()
```

Out[63]: 0.6416213389121339

In [64]:
```
clf1.fit(X_train, y_train)
predictions = clf1.predict(X_test)
print("*Confusion Matrix for KNN Classifier: ")
print(confusion_matrix(y_test, predictions))
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
cm = confusion_matrix(y_test, predictions)
plot_confusion_matrix(clf1, X_test, y_test)
plt.show()
```

```
*Confusion Matrix for KNN Classifier:
[[129  26   0]
 [ 51 128  18]
 [ 14  30   4]]
```



Logistic Regression

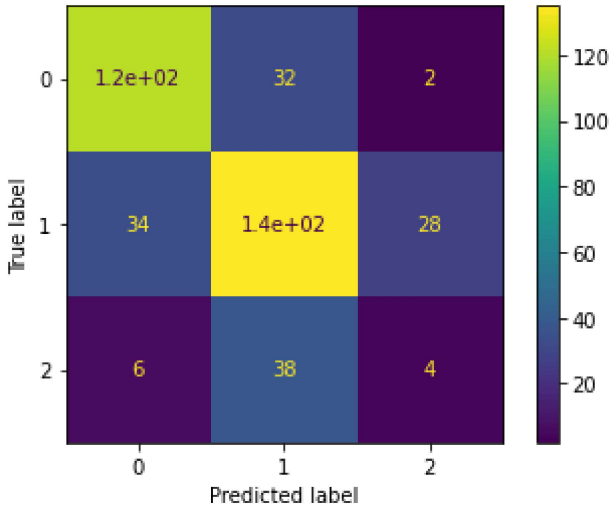
In [65]:
```
from sklearn.linear_model import LogisticRegression
clf1 = LogisticRegression(random_state=0)
cv_scores = cross_val_score(clf1,X_train,y_train,cv=kf)
cv_scores.mean()
```

```
C:\Users\Public\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:939: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\Users\Public\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:939: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\Users\Public\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:939: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\Users\Public\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:939: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\Users\Public\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:939: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[65]: 0.6800418410041841

In [66]:
```
clf1.fit(X_train, y_train)
predictions = clf1.predict(X_test)
print("*Confusion Matrix for LR Classifier: ")
print(confusion_matrix(y_test, predictions))
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
cm = confusion_matrix(y_test, predictions)
plot_confusion_matrix(clf1, X_test, y_test)
plt.show()
```

```
*Confusion Matrix for LR Classifier:
[[121  32   2]
 [ 34 135  28]
 [  6  38   4]]

C:\Users\Public\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:939: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html.
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```



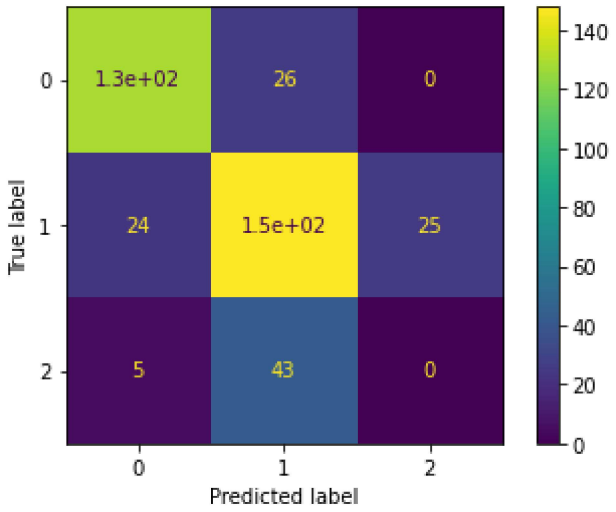
Voting Classifier

In [67]:
```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
clf1 = SVC(gamma='auto')
clf2 = RandomForestClassifier(n_estimators=50, random_state=1)
clf3 = DecisionTreeClassifier()
eclf1 = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('dt', clf3
)], voting='hard')
cv_scores = cross_val_score(eclf1,X_train,y_train,cv=kf)
cv_scores.mean()
```

Out[67]: 0.7384867503486751

In [68]:
```
eclf1.fit(X_train, y_train)
predictions = eclf1.predict(X_test)
print("*Confusion Matrix for Voting Classifier: ")
print(confusion_matrix(y_test, predictions))
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
cm = confusion_matrix(y_test, predictions)
plot_confusion_matrix(eclf1, X_test, y_test)
plt.show()
```

```
*Confusion Matrix for Voting Classifier:
[[129  26   0]
 [ 24 148  25]
 [  5  43   0]]
```



In [ ]: