Flower category Identification

Yuganthi Liyanage

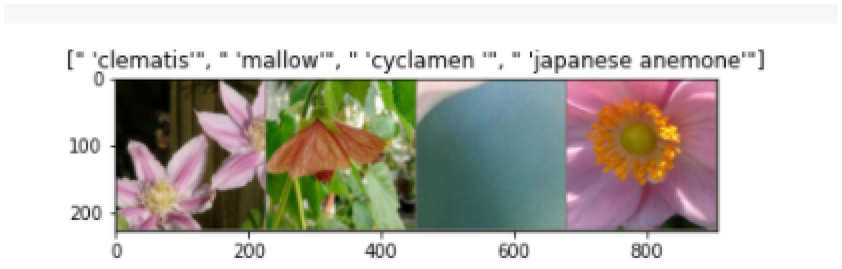
December 12, 2020

Abstract

In this report, we introduced a method to classify the images into several groups using an image classifier.

1 Introduction

First, we download the image set with their labels to our google drive. After that, we apply a data augmentation technique to increase the image size. Finally, 6552 train images and 818 testing images were used to our analysis.


[" 'clematis'", " 'mallow'", " 'cyclamen '", " 'japanese anemone'"]
0
100
200
0
200
400
600
800

2 Procedure

We made a simple classifier to make classification using pytorch.Model is based on cross-entropy loss and stochastic gradient decent algorithm used to find out the points which have minimum loss.After that resnet18 weights used as the weight of our classifier.After running several epochs,accuracy values for training and validation have recorded to check the performance of the model.

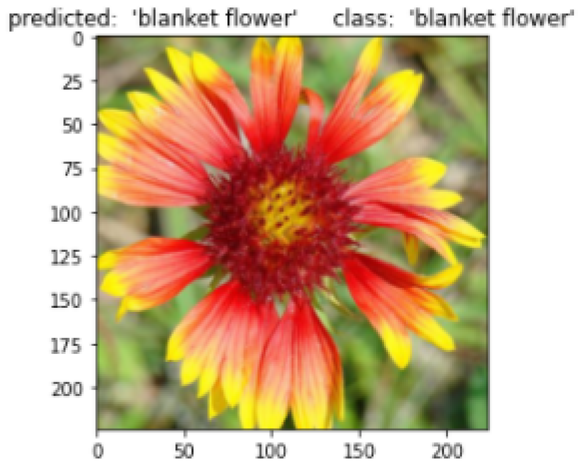
```
def train_model(model, num_epochs=25):

    model = model.to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
    scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)
```

```
for epoch in range(num_epochs):

    print('Epoch: ',epoch+1,'/',num_epochs)

    ###Train
    model.train()
    running_corrects = 0
    for inputs, labels in Bar(dataloaders['train']):
        inputs = inputs.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()

        outputs = model(inputs)

        preds = torch.max(outputs, 1)[1]
        running_corrects += torch.sum(preds == labels.data)

        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

    print("Train ", 'Acc: {:.2f}'.format(running_corrects.double()/datase

    scheduler.step()

    ###Val
    model.eval()
    running_corrects = 0
    for inputs, labels in Bar(dataloaders['valid']):
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        preds = torch.max(outputs, 1)[1]
        running_corrects += torch.sum(preds == labels.data)

    print("Valid ", 'Acc: {:.2f}'.format(running_corrects.double()/datase
    print("######################")
return model
```

3 Prediction using Resnet18

Finally we check the testing image with their actual and predicted class.

```
model = train_model(model, num_epochs=3)#train the model

def visualize_model(model, num_images=16):
    model.eval()
    index = 0
    for i, (inputs, labels) in enumerate(dataloaders['valid']):
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)

        preds = torch.max(outputs, 1)[1]

for j in range(inputs.size()[0]):
index += 1
title1 = 'predicted: ' +dataset_labels[int(class_names[preds[j]])-1] + '
class: ' + dataset_labels[int(class_names[labels[j]])-1]
imshow(inputs.cpu().data[j],title1)

if index == num_images:
return
```



3.1 Prediction for new image

```
image = io.imread('https://images-na.ssl-images-
amazon.com/images/I/51dZp-%2B4W9L._AC_.jpg')
plt.imshow(image);
img = apply_transforms(image).
clone().detach().requires_grad_(True).to(device)
outputs = model(img)
preds = torch.max(outputs, 1)[1]
print('predicted: ' + dataset_labels[int(class_names[preds])-1])
```

0
50
100
150
200
250
300
350
0
100
200
300
400