rice-disease-resnet50

May 7, 2024

[6]:
```
import os
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, models, transforms
from torch.utils.data import DataLoader
```

[7]:
```
# Define device
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

# Define data directory paths
data_dir = "rice_diseases"
train_dir = os.path.join(data_dir, "train")
val_dir = os.path.join(data_dir, "val")

# Define data transforms
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

[8]:
```
# Create datasets
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),␣
 ↪data_transforms[x]) for x in ['train', 'val']}

# Create data loaders
```

```python
dataloaders = {x: DataLoader(image_datasets[x], batch_size=32, shuffle=True,␣
 ↪num_workers=4) for x in ['train', 'val']}

# Define the model architecture (using ResNet-50)
model = models.resnet50(pretrained=True)
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, len(image_datasets['train'].classes))

# Move model to device
model = model.to(device)

# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

```
c:\Users\fr5009tu\AppData\Local\Programs\Python\Python311\Lib\site-
packages\torchvision\models\_utils.py:208: UserWarning: The parameter
'pretrained' is deprecated since 0.13 and may be removed in the future, please
use 'weights' instead.
  warnings.warn(
c:\Users\fr5009tu\AppData\Local\Programs\Python\Python311\Lib\site-
packages\torchvision\models\_utils.py:223: UserWarning: Arguments other than a
weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed
in the future. The current behavior is equivalent to passing
`weights=ResNet50_Weights.IMAGENET1K_V1`. You can also use
`weights=ResNet50_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
```

[9]:
```python
# Train the model
def train_model(model, criterion, optimizer, num_epochs):
    for epoch in range(num_epochs):
        print(f'Epoch {epoch+1}/{num_epochs}')
        print('-' * 10)

        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()  # Set model to training mode
            else:
                model.eval()   # Set model to evaluate mode

            running_loss = 0.0
            running_corrects = 0

            # Iterate over data.
            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                # Zero the parameter gradients
                optimizer.zero_grad()

                # Forward pass
                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)

                    # Backward + optimize only if in training phase
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                # Statistics
                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            epoch_loss = running_loss / len(image_datasets[phase])
            epoch_acc = running_corrects.double() / len(image_datasets[phase])

            print(f'{phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}')
```

[10]:
```
# Train the model
train_model(model, criterion, optimizer, num_epochs=10)
```

```
Epoch 1/10
----------
train Loss: 0.8500 Acc: 0.5920
val Loss: 0.9071 Acc: 0.6667
Epoch 2/10
----------
train Loss: 0.5180 Acc: 0.7713
val Loss: 0.5162 Acc: 0.7793
Epoch 3/10
----------
train Loss: 0.4154 Acc: 0.8201
val Loss: 0.3825 Acc: 0.8161
Epoch 4/10
----------
train Loss: 0.4130 Acc: 0.8098
val Loss: 0.3478 Acc: 0.8437
Epoch 5/10
----------
train Loss: 0.3464 Acc: 0.8402
```

```
val Loss: 0.2900 Acc: 0.8713
Epoch 6/10
----------
train Loss: 0.3427 Acc: 0.8397
val Loss: 0.3329 Acc: 0.8414
Epoch 7/10
----------
train Loss: 0.3083 Acc: 0.8552
val Loss: 0.2331 Acc: 0.9034
Epoch 8/10
----------
train Loss: 0.3229 Acc: 0.8511
val Loss: 0.2488 Acc: 0.8805
Epoch 9/10
----------
train Loss: 0.3008 Acc: 0.8718
val Loss: 0.3104 Acc: 0.8598
Epoch 10/10
----------
train Loss: 0.2965 Acc: 0.8626
val Loss: 0.2231 Acc: 0.8966
```

[11]:
```
import pickle

# Save the entire model object
with open('rice_disease_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

[12]:
```
# Step 1: Load the model from pickle
with open('rice_disease_model.pkl', 'rb') as f:
    new_model = pickle.load(f)
```

[13]:
```
# Train the model
train_model(new_model, criterion, optimizer, num_epochs=10)
```

```
Epoch 1/10
----------
train Loss: 0.2880 Acc: 0.8632
val Loss: 0.2130 Acc: 0.8897
Epoch 2/10
----------
train Loss: 0.2900 Acc: 0.8644
val Loss: 0.2198 Acc: 0.8851
Epoch 3/10
----------
train Loss: 0.2908 Acc: 0.8770
val Loss: 0.2100 Acc: 0.8897
Epoch 4/10
```

```
----------
train Loss: 0.2650 Acc: 0.8741
val Loss: 0.2118 Acc: 0.8851
Epoch 5/10
----------
train Loss: 0.2995 Acc: 0.8609
val Loss: 0.2094 Acc: 0.8851
Epoch 6/10
----------
train Loss: 0.2951 Acc: 0.8678
val Loss: 0.2048 Acc: 0.8897
Epoch 7/10
----------
train Loss: 0.2876 Acc: 0.8747
val Loss: 0.2127 Acc: 0.8874
Epoch 8/10
----------
train Loss: 0.3078 Acc: 0.8546
val Loss: 0.2053 Acc: 0.8897
Epoch 9/10
----------
train Loss: 0.2956 Acc: 0.8603
val Loss: 0.2044 Acc: 0.8851
Epoch 10/10
----------
train Loss: 0.3032 Acc: 0.8690
val Loss: 0.2173 Acc: 0.8943
```

[14]:
```python
# Save the entire model object
with open('rice_disease_model.pkl', 'wb') as f:
    pickle.dump(new_model, f)
```

[15]:
```python
# Step 1: Load the model from pickle
with open('rice_disease_model.pkl', 'rb') as f:
    new_model = pickle.load(f)
```

[16]:
```python
# Train the model
train_model(new_model, criterion, optimizer, num_epochs=10)
```

```
Epoch 1/10
----------
train Loss: 0.2894 Acc: 0.8770
val Loss: 0.2091 Acc: 0.8874
Epoch 2/10
----------
train Loss: 0.2736 Acc: 0.8810
val Loss: 0.2096 Acc: 0.8920
Epoch 3/10
```

```
----------
train Loss: 0.2726 Acc: 0.8810
val Loss: 0.2122 Acc: 0.8874
Epoch 4/10
----------
train Loss: 0.3106 Acc: 0.8609
val Loss: 0.2081 Acc: 0.8897
Epoch 5/10
----------
train Loss: 0.2931 Acc: 0.8667
val Loss: 0.2013 Acc: 0.8897
Epoch 6/10
----------
train Loss: 0.2706 Acc: 0.8764
val Loss: 0.2127 Acc: 0.8897
Epoch 7/10
----------
train Loss: 0.2879 Acc: 0.8701
val Loss: 0.2165 Acc: 0.8805
Epoch 8/10
----------
train Loss: 0.2919 Acc: 0.8730
val Loss: 0.2161 Acc: 0.8897
Epoch 9/10
----------
train Loss: 0.2761 Acc: 0.8718
val Loss: 0.2024 Acc: 0.8897
Epoch 10/10
----------
train Loss: 0.2899 Acc: 0.8678
val Loss: 0.2069 Acc: 0.8897
```

[17]:
```python
# Save the entire model object
with open('rice_disease_model.pkl', 'wb') as f:
    pickle.dump(new_model, f)
```

[18]:
```python
# Step 1: Load the model from pickle
with open('rice_disease_model.pkl', 'rb') as f:
    new_model = pickle.load(f)
```

[19]:
```python
# Train the model
train_model(new_model, criterion, optimizer, num_epochs=10)
```

```
Epoch 1/10
----------
train Loss: 0.2970 Acc: 0.8603
val Loss: 0.2102 Acc: 0.8920
Epoch 2/10
```

```
----------
train Loss: 0.3027 Acc: 0.8580
val Loss: 0.2216 Acc: 0.8943
Epoch 3/10
----------
train Loss: 0.2952 Acc: 0.8655
val Loss: 0.2078 Acc: 0.8920
Epoch 4/10
----------
train Loss: 0.2989 Acc: 0.8632
val Loss: 0.2142 Acc: 0.8920
Epoch 5/10
----------
train Loss: 0.2939 Acc: 0.8649
val Loss: 0.2115 Acc: 0.8897
Epoch 6/10
----------
train Loss: 0.2983 Acc: 0.8724
val Loss: 0.2084 Acc: 0.8989
Epoch 7/10
----------
train Loss: 0.2948 Acc: 0.8707
val Loss: 0.2134 Acc: 0.8920
Epoch 8/10
----------
train Loss: 0.2694 Acc: 0.8805
val Loss: 0.2148 Acc: 0.8851
Epoch 9/10
----------
train Loss: 0.2865 Acc: 0.8649
val Loss: 0.2074 Acc: 0.8943
Epoch 10/10
----------
train Loss: 0.2949 Acc: 0.8695
val Loss: 0.2104 Acc: 0.8874
```

[20]:
```
# Save the entire model object
with open('rice_disease_model.pkl', 'wb') as f:
    pickle.dump(new_model, f)
```