Zatürre Hastalığının Tanısı

Grup Üyeleri:

Rahil IBRAHIMI: 181906127

Melike Can SULTAN: 203405022

Beyzanur GÜNBEGİ: 203405032

Yaren TEMİZEL: 193405064

1. Giriş

Bu proje, PyTorch ve diğer çeşitli Python kütüphanelerini kullanarak derin öğrenme modeli oluşturma, eğitim, doğrulama ve test işlemlerini gerçekleştirmektedir. Aşağıda her bir bileşenin ayrıntılı açıklaması ve kullanım amacı verilmiştir:

2. Kullanılan Kütüphaneler ve Araçlar

```
import torch
import numpy as np
import matplotlib.pyplot as plt
import os
from tqdm.notebook import tqdm
import zipfile
import opendatasets as od
import glob
import random
import matplotlib.pyplot as plt
from torchvision import transforms as T, datasets
from helper import show_image
from torch.utils.data import DataLoader
from torchvision.utils import make_grid
from helper import show_grid
import torchvision.datasets as dataset
from skimage.transform import AffineTransform, warp, resize
import cv2
#from torch.utils.data
from torch.utils.data import DataLoader
from torchvision.utils import make_grid
from torch import nn
import torch.nn.functional as F
import timm
from torchvision.datasets import CIFAR10
from torchvision.transforms import ToTensor
from torchsummary import summary
from helper import accuracy
```

3. Konfigürasyon Ayarları (cfg)

```
class CFG:

    epochs = 1 #training model
    lr = 0.001 #learning rate
    batch_size = 16 #batch size for ds
    model_name = 'tf_efficientnet_b6_ns' # model name
    img_size = 224

    DATA_DIR = "chest_xray_data/chest_xray/chest_xray"
    TRAIN = "train"
    VAL = "val"
    TEST = "test"
    img_size = 224


device = torch.device('cuda'if torch.cuda.is_available() else 'cpu')
print("ON WHICH DEVICE WE ARE ON : {}".format(device))
```

```
ON WHICH DEVICE WE ARE ON : cpu
```

4. Veri Dönüşümleri
Eğitim Dönüşümleri (train_transform)
Eğitim dönüşümleri, veri artırma teknikleri ile modelin genelleme yeteneğini artırmayı amaçlar. Örneğin, rastgele döndürme, kırpma veya yatay çevirme gibi işlemler uygulanabilir. Bu tür işlemler, modelin farklı veri varyasyonlarını görerek daha iyi genelleştirme yapmasını sağlar.

```
train_transform = T.Compose([
    T.Resize(size=(224, 224)),
    T.RandomRotation(degrees=(-20, 20)),
    T.ToTensor(),  # (H, W, C) -> (C, H, W)
    T.Normalize([0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

Doğrulama ve Test Dönüşümleri (valid_transform, test_transform)

Doğrulama ve test dönüşümleri, eğitim dönüşümlerinden daha basittir ve genellikle veri artırma içermez. Amaç, modelin performansını tutarlı bir şekilde değerlendirebilmektir.

```
valid_transform = T.Compose([
    T.Resize(size=(CFG.img_size, CFG.img_size)),
    T.ToTensor(),  # (H, W, C) -> (C, H, W)
    T.Normalize([0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

test_transform = T.Compose([
    T.Resize(size=(CFG.img_size, CFG.img_size)),
    T.ToTensor(),  # (H, W, C) -> (C, H, W)
    T.Normalize([0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

5. Veri Yükleyicileri

PyTorch'un datasets.ImageFolder sınıfı, belirli bir dizin yapısına sahip veri setlerini kolayca yüklemenizi sağlar. Bu dizin yapısı genellikle sınıf adlarına göre düzenlenmiş klasörler içerir ve her klasör içinde ilgili sınıfa ait görüntüler bulunur.

[5]:
```
train_path = os.path.join (CFG.DATA_DIR, CFG.TRAIN)
valid_path = os.path.join(CFG.DATA_DIR, CFG.VAL)
test_path = os.path. join (CFG.DATA_DIR, CFG. TEST)

trainset = datasets. ImageFolder(train_path,transform = train_transform)
validset = datasets. ImageFolder(valid_path,transform = valid_transform)
testset = datasets. ImageFolder(test_path,transform = test_transform)
```

6. Veri Yükleme ve DataLoader Kullanımı

Veri setleri yüklendikten sonra, torch.utils.data.DataLoader sınıfı kullanılarak veri yükleme işlemleri optimize edilir. DataLoader, veri setlerini mini-batch'lere böler ve eğitim sırasında veri yüklemesini hızlandırır.

[9]:
```
trainloader = DataLoader(trainset, batch_size=CFG.batch_size, shuffle=True)
validloader = DataLoader(validset, batch_size=CFG.batch_size, shuffle=True)
testloader = DataLoader(testset, batch_size=CFG.batch_size, shuffle=True)

print("No. of batches in trainloader: {}".format(len(trainloader)))
print("No. of Total examples: {}".format(len(trainloader.dataset)))
```

```
No. of batches in trainloader: 326
No. of Total examples: 5216
```

7. Model Eğitimi
Eğitim süreci, modelin ağırlıklarının optimizasyon algoritması (örneğin, Adam veya SGD) kullanılarak güncellenmesini içerir. Her epoch'ta model, tüm eğitim veri seti üzerinden geçer ve her mini-batch için kayıp hesaplanır, geriye yayılım yapılır ve ağırlıklar güncellenir.

```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(),lr = CFG.lr)
schedular = None

trainer = PneumoniaTrainer (criterion, optimizer, schedular)
trainer. fit (model, trainloader, validloader, epochs = CFG. epochs)
```

Error displaying widget: model not found

Error displaying widget: model not found

```
Valid loss decreased (inf --> 0.5697861909866333)
Epoch: 1 Train Loss: 0.3274 Train Acc: 0.8625
Epoch: 1 Valid Loss: 0.5698 Valid Acc: 0.6875
```

8. Doğrulama Süreci

Doğrulama süreci, modelin performansını eğitim sırasında izlemek için kullanılır. Model, doğrulama veri seti üzerinde test edilir ve doğruluk gibi performans metrikleri hesaplanır.

```
model.load_state_dict(torch.load('ColabPneumonia.pt', map_location=device))
model.to(device)
model.eval()
```

9. Test Süreci
Test süreci, modelin nihai performansını değerlendirmek için kullanılır. Model, test veri seti üzerinde test edilir ve performans metrikleri hesaplanır.

```
avg_test_acc , avg_test_loss = trainer.valid_batch_loop(model,testloader)

print("Test Acc : {}".format(avg_test_acc))
print("Test Loss : {}".format(avg_test_loss))
```

Error displaying widget: model not found

```
Test Acc : 0.8028846383094788
Test Loss : 0.42205347693883455
```

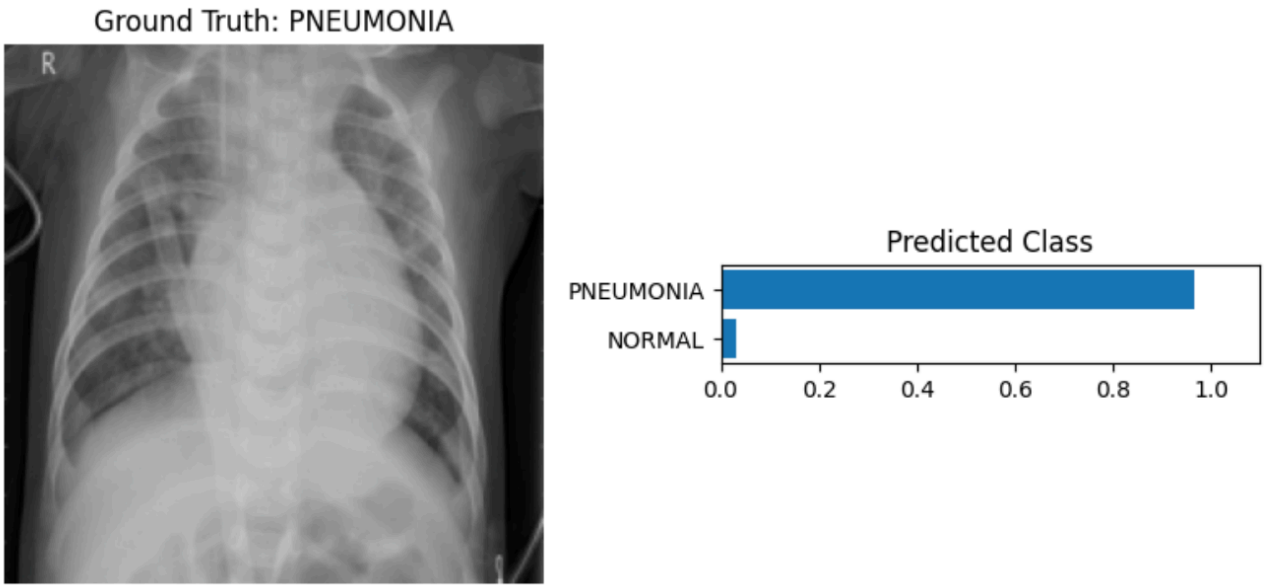
10. Sonuçlar ve Değerlendirme

Modelin eğitim ve doğrulama süreçlerinde elde edilen sonuçlar, modelin genelleme yeteneğini ve performansını değerlendirmek için kullanılır. Test sonuçları, modelin gerçek dünya verilerine karşı nasıl performans gösterdiğini gösterir.

```
image, label = testset[324]
image = image.unsqueeze(0).to(device)

logit = model(image)
ps = F.softmax(logit, dim=1)

view_classify(image.squeeze().cpu(), ps.cpu(), label)
```



Bu rapor, derin öğrenme projelerinde temel adımların anlaşılması ve uygulanması için bir rehber niteliği taşımaktadır. Gelecekteki projelerde, bu yapı ve teknikleri kullanarak daha karmaşık ve yüksek performanslı modeller geliştirebilirsiniz. Ayrıca, bu süreci kendi veri setleriniz ve özel gereksinimleriniz doğrultusunda özelleştirerek, derin öğrenme uygulamalarınızın etkinliğini artırabilirsiniz.

Teşekkürler
Bu projeyi başarıyla tamamlamamıza olanak tanıyan Ümit Şentürk hocamıza ve grup üyelerimiz (Rahil Ibrahimi, Melike Can sultan, Yaren Temizel ve Beyzanur Günbeygi) ve açık kaynak kütüphanelerin geliştiricilerine teşekkürlerimizi sunarız.