Deep Learning

Final Project

Dr. DavoodAbadi

Winter 2023

Sina Eskandari – Elnaz Rezaee – Hoorieh Sabzevari

طرح مسئله:

در این پروژه از ما خوسته شده تا با استفاده از روش‌های داده‌افزایی ارائه شده در مقاله ۲، متد انتقال یادگیری (transform learning) را بر روی معماری Resnet18 پیاده‌سازی کنیم.

شرح کارها:

در ابتدا دیتاست را از لینک داده شده دانلود کرده و پیش‌پردازش‌های اولیه‌ای داخل کد مقاله را اعمال کردیم.

```
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize(224),
        transforms.RandomResizedCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]) ,
    'test': transforms.Compose([
        transforms.Resize(224),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])}
image_datasets = {x: datasets.ImageFolder(os.path.join(dataset_path, x), data_transforms[x]) for x in ['train', 'test']}
class_names = image_datasets['train'].classes
```

transforms.RandomResizedCrop(224): این تابع یک پچ با اندازه (۲۲۴، ۲۲۴) را به صورت تصادفی از تصویر ورودی ما استخراج می‌کند. بنابراین، ممکن است این مسیر را از بالا، پایین سمت راست یا هر جای دیگری انتخاب کند. بنابراین، در این بخش افزایش داده انجام می‌شود.

transforms.RandomHorizontalFlip(): تصویر را به صورت افقی برمی‌گرداند. این بخش نوع دیگری از افزایش داده است.

transforms.ToTensor(): این تابع تصویر ورودی را به تنسور PyTorch تبدیل می‌کند.

transforms.Normalize(): این تابع برای scale داده‌های ورودی است و این مقادیر (میانگین و std) باید برای مجموعه‌ی داده‌ی ما از قبل محاسبه شده باشند.

سپس تصاویر خود را که به دو بخش train و test تقسیم شده‌اند، در دیکشنری image_datasets لود می‌کنیم. هایپرپارامترهای خود را نیز با تعداد ۱۰ ایپاک و اندازه دسته‌ی برابر با ۳۲ تعیین می‌کنیم.

۱) داده افزایی:

داده‌ها را با دو روش از روش‌های ارائه شده در مقاله دوم، افزایش می‌دهیم.

```
augmentation1 = transforms.Compose([
                    transforms.Resize(224),
                    transforms.RandomResizedCrop(224),
                    transforms.ToTensor(),
                    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
                    transforms.RandomRotation(degrees=(-10, 10)),
                    transforms.RandomHorizontalFlip(),
                    transforms.RandomAffine(degrees=0, translate=(0.2, 0.2))
                ])

augmentation2 = transforms.Compose([
                    transforms.Resize(224),
                    transforms.RandomResizedCrop(224),
                    transforms.ToTensor(),
                    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
                    transforms.RandomRotation(degrees=(-30, 30)),
                    transforms.RandomHorizontalFlip()
                ])
augmented_dataset1 = datasets.ImageFolder(os.path.join(dataset_path, 'train'), augmentation1)
augmented_dataset2 = datasets.ImageFolder(os.path.join(dataset_path, 'train'), augmentation2)
image_datasets['train'] = torch.utils.data.ConcatDataset([image_datasets['train'], augmented_dataset1, augmented_dataset2])
```

این روش‌ها عبارتند از تغییرسایز، کراپ تصویر، چرخش با زوایای رندوم، فلیپ تصویر و اعمال تبدیلات هندسی مانند تبدیل Affine. پس از اعمال این دو روش، تمام دیتاها را باهم concatenate می‌کنیم.

سپس دیتا را با استفاده از تابع DataLoader لود می‌کنیم.

۲) تعریف توابع:

توابع آموزش و ارزیابی مدل و یک تابع کمکی برای رسم نمودار احتمال‌های موجود در مقاله تعریف می‌کنیم.

۳) تعریف مدل:

از معماری Resnet18 همه‌ی لایه‌ها بجز لایه‌ی آخر FC و لایه‌ی آخر sequential را فریز می‌کنیم و از وزن‌های از پیش‌آموخته‌شده استفاده می‌کنیم.

از بهینه‌ساز adam استفاده می‌کنیم.

```
model_conv = torchvision.models.resnet18(pretrained=True)
for i, child in enumerate(model_conv.children()):
    if i >= 7 :
        for param in child.parameters():
            param.requires_grad = True
    else:
        for param in child.parameters():
            param.requires_grad = False

num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2)


model_conv = model_conv.to(device)
criterion = nn.CrossEntropyLoss()

optimizer_conv = optim.Adam(model_conv.fc.parameters())

summary(model_conv, (3, 224, 224), batch_size=BATCHSIZE)
```

۴) آموزش مدل:

حال مدل را آموزش می‌دهیم.

```
model_conv, train_acc, valid_acc = train_model(model_conv, criterion, optimizer_conv, BATCHSIZE, EPOCHS)
model_conv.eval()
torch.save(model_conv, './covid_resnet18.pt')
```

```
Training complete in 14m 34s
Best val acc= 0.9909677419354839 at Epoch: 7
```

۵) ارزیابی مدل:

معیارهای متفاوتی برای ارزیابی عملکرد مدل‌های classification استفاده می‌شود، مانند accuracy، sensitivity، specificity، precision، recall و F1-score. از آنجایی که مجموعه داده آزمایشی فعلی بسیار imbalance است(۱۰۰ تصویر COVID-19 و ۳۰۰۰

تصویر Non-COVID)، specificity و sensitivity معیارهای مناسبی هستند که در اینجا می‌توانند استفاده شوند. حال به توضیح مختصر هر یک از این ویژگی‌ها می‌پردازیم:

Sensitivity:

Sensitivity نسبت نمونه‌های مثبت واقعی (True Positive) که توسط مدل به درستی پیش‌بینی شده‌اند به مجموع نمونه‌های مثبت واقعی که مدل درست تشخیص داده است و نمونه‌های منفی که مدل اشتباه تشخیص داده است، می‌باشد. بنابراین sensitivity به صورت زیر تعریف می‌شود:

$$Sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Specificity:

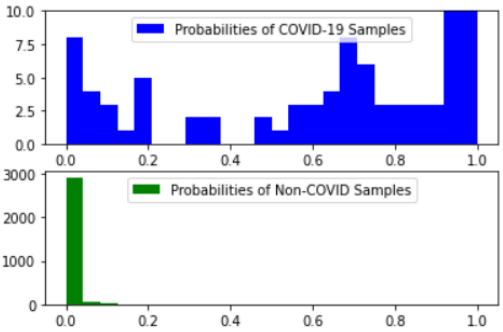
Specificity نسبت داده‌های منفی که به‌درستی توسط مدل تشخیص داده شده‌اند، به مجموع داده‌های منفی‌ای که مدل آن‌ها را درست تشخیص داده است و داده‌های مثبتی که مدل آن‌ها را اشتباه پیش‌بینی کرده است، می‌باشد. بنابراین داریم:

$$Specificity = \frac{True\ Negative}{True\ Negative + False\ Positive}$$

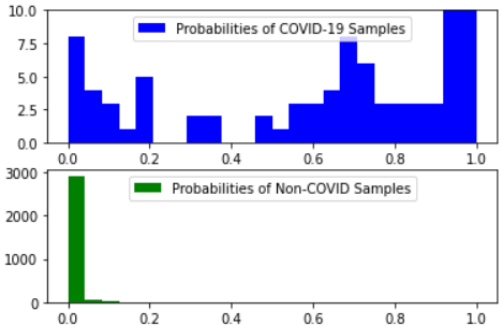
جدول نتایج:

Threshold	Sensitivity	Specificity
0.1	0.86	0.989
0.17	0.84	0.996
0.2	0.82	0.997
0.25	0.79	0.997
0.35	0.76	0.999

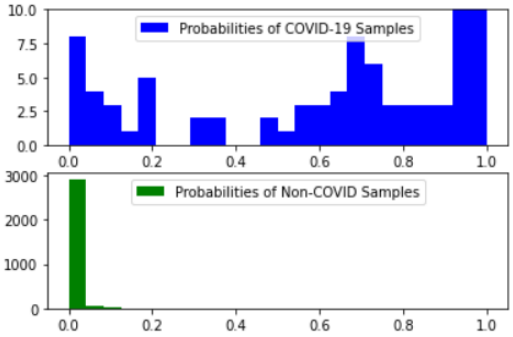
نمودارهای مربوط به احتمالات پیش‌بینی‌شده:



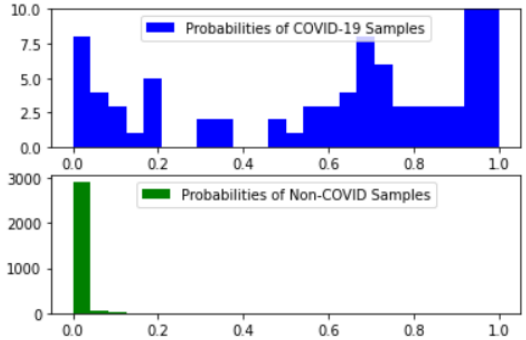
Threshold = 0.17



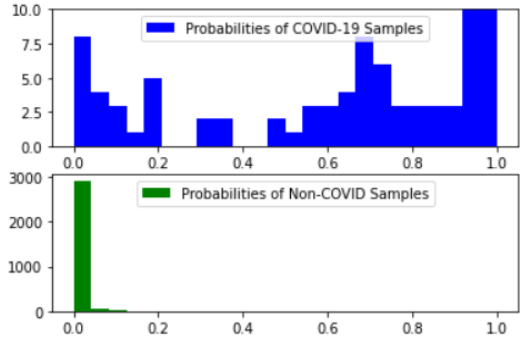
Threshold = 0.1



Threshold = 0.25



Threshold = 0.2



Threshold = 0.35