تمرین دهم یادگیری عمیق

محمدعلی فراهت **۹۷۵۲۱۴۲۳**

سوال ۱)

الف) در این قسمت باید مدل را برای وزن‌های تصادفی در شبکه ResNet آموزش بدهیم.

برای این کار در هنگام استفاده از شبکه ResNet ، مقدار پارامتر pretrained را False در نظر می‌گیریم تا وزن‌دهی اولیه آن تصادفی باشد. به بقیه پارامترهای آن هم دست نمی‌زنیم و آموزش را در ۲۵ ایپاک انجام می‌دهیم.

* متاسفانه برای این قسمت نتوانستم خروجی نوتبوک را دانلود کنم، چون می‌خواستم با قسمت‌های بعد دانلود کنم ولی متاسفانه از دست رفت. ولی همان موقع از خروجی آخر عکس گرفته بودم و در زیر آن را می‌بینیم.

* این قسمت توسط نوت‌بوک اولیه انجام شده و برای همین دقت آن به شدت پایین است، اما مقدار loss را میتوان مشاهده کرد که درحال کاهش است.

```
Epoch 22/24
----------
Iterating through data...
train Loss: 113.9849 Acc: 0.0491
Iterating through data...
val Loss: 143.2343 Acc: 0.0244

Epoch 23/24
----------
Iterating through data...
train Loss: 113.1742 Acc: 0.0527
Iterating through data...
val Loss: 143.8439 Acc: 0.0246

Epoch 24/24
----------
Iterating through data...
train Loss: 113.7399 Acc: 0.0481
Iterating through data...
val Loss: 143.1192 Acc: 0.0261

Training complete in 122m 43s
Best val Acc: 0.026489
=> saving checkpoint
```

نتیجه‌گیری: وقتی می‌خواهیم وزن‌ها را از اول خودمان آموزش بدهیم، حتی اگر شبکه خوبی مانند resnet داشته باشیم، به زمان زیادی نیاز داریم تا بتوانیم به دقت خوبی برسیم، مثلا در اینجا بعد از ۱۲۲ دقیقه آموزش و در ۲۵ ایپاک با GPU در کولب، نتوانستیم دقتی بالاتر از ۵ درصد در آموزش و ۲.۶ درصد در تست را داشته باشیم. پس بهتر است از وزن‌های از پیش آموزش دیده استفاده کنیم که در قسمت بعدی می‌بینیم.

ب) در این قسمت از ما خواسته شده که لایه classifier را روی وزن‌های از پیش آموزش‌دیده‌ی ImageNet ، آموزش دهیم. برای این کار پارامتر pretrained را در مدل resnet به True تغییر می‌دهیم، همچنین چون دیگر نیازی به آموزش آن‌ها نداریم (در این قسمت) مقدار requires_grad را برای پارامترهای آن False می‌کنیم تا آن‌ها فریز شوند و آموزش نبینند.

```
def __init__(self, original_model, num_classes):
    super(ResnetModel, self).__init__()

    # Everything except the last linear layer
    self.features = nn.Sequential(*list(original_model.children())[:-1])
    for param in self.features.parameters():
        param.requires_grad = False
    self.classifier = nn.Sequential(
        nn.Linear(2048, num_classes),
    )
```

با اینکار تعداد پارامترهای قابل آموزش به شدت کم می‌شود که در زیر اعداد آن را می‌بینید:

```
pytorch_total_params=  23909636
pytorch_total_trainable_params=  401604
```

سپس مدل را در ۲۵ ایپاک آموزش می‌دهیم. البته ابتدا با نوت‌بوک اولیه این کار را انجام دادم و درصد دقت آن به شدت پایین بود که در زیر می‌بینید:

```
Epoch 22/24
----------
Iterating through data...
train Loss: 91.6327 Acc: 0.0920
Iterating through data...
val Loss: 119.7142 Acc: 0.0478

Epoch 23/24
----------
Iterating through data...
train Loss: 91.0091 Acc: 0.0941
Iterating through data...
val Loss: 119.5255 Acc: 0.0464

Epoch 24/24
----------
Iterating through data...
train Loss: 91.1127 Acc: 0.0903
Iterating through data...
val Loss: 119.7679 Acc: 0.0455

Training complete in 124m 58s
Best val Acc: 0.048004
=> saving checkpoint
```

سپس قسمتی از نوت‌بوک اولیه را تغییر دادم و تابع train را عوض کردم و بعد از تست کردن دیدم دقت بسیار بالاتر رفت، پس از همان کد استفاده کردم (نوت‌بوک ها ضمیمه شده اند)
خروجی آن‌را در زیر می‌بینید:

```
Epoch 22/24
----------
Iterating through data...
train Loss: 30.4304 Acc: 0.8468
Iterating through data...
val Loss: 78.3167 Acc: 0.4097

Epoch 23/24
----------
Iterating through data...
train Loss: 30.2598 Acc: 0.8482
Iterating through data...
val Loss: 78.4951 Acc: 0.4068

Epoch 24/24
----------
Iterating through data...
train Loss: 30.4341 Acc: 0.8471
Iterating through data...
val Loss: 78.1750 Acc: 0.4120

Training complete in 103m 13s
Best val Acc: 0.413879
=> saving checkpoint
```

* بعد از ۱۰ ایپاک واقعا دقت تست و ترین تغییر زیادی نمی‌کرد و بنظرم نیازی نبود بیش از ۱۰۰ دقیقه اجرا شود.
* دقت آموزش به ۸۴.۸ درصد و دقت تست به ۴۱ درصد رسید. این نشان‌دهنده‌ی این است که مدل overfit شده است و شاید بهتر بود از روش‌هایی مثل رگولایز کردن یا داده افزایی استفاده کنیم تا این اتفاق کمتر رخ بدهد. اما به دلیل زمان اجرای زیاد این کار مقدور نبود. ولی با این حال دقت تست بیش از ۴۱ درصد برای مسئله‌ای با ۱۹۶ کلاس، دقت بدی به نظر نمی‌رسد.

پ) در این قسمت از سوال باید کلاس‌بندی را با روش SVM انجام دهیم. پس ابتدا باید feature ها را با استفاده از همان مدل resnet استخراج کنیم. سپس این ویژگی‌ها را به همراه label هایی که از دیتای اولیه داشتیم، به SVM بدهیم تا خودش را fit کند. سپس دقت را برای تست و ترین محاسبه می‌کنیم. روش انجام این کار را هم در زیر می‌بینید:

ابتدا لایه‌ی classifier را از مدل حذف می‌کنیم:

```python
class ResnetModel(nn.Module):
    def __init__(self, original_model, num_classes):
        super(ResnetModel, self).__init__()

        # Everything except the last linear layer
        self.features = nn.Sequential(*list(original_model.children())[:-1])
        for param in self.features.parameters():
            param.requires_grad = False

    def forward(self, x):
        f = self.features(x)
        f = f.view(f.size(0), -1)
        # y = self.classifier(f)
        return f
```

سپس تابعی برای گرفتن ویژگی‌ها می‌سازیم (هم تست هم آموزش) :

```python
def get_features_labels_test(model):
    features = []
    final_labels = []
    for inputs, labels in test_data_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        # print(outputs.shape)
        for i in range(outputs.cpu().shape[0]):
            features.append(np.array(outputs[i].cpu()))
            final_labels.append(np.array(labels[i].cpu()))
        # print(outputs)

    return (features, final_labels)
```

```python
def get_features_labels_train(model):
    features = []
    final_labels = []
    for inputs, labels in train_data_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        # print(outputs.shape)
        for i in range(outputs.cpu().shape[0]):
            features.append(np.array(outputs[i].cpu()))
            final_labels.append(np.array(labels[i].cpu()))
        # print(outputs)

    return (features, final_labels)
```

سپس مدل svm را با همان ویژگی‌هایی که استخراج کردیم آموزش می‌دهیم:

```python
clf = svm.SVC(kernel='linear')
clf.fit(features, labels)
```

در زیر دقت این روش را می‌بینیم:

```
[ ]  1 pred_train = clf.predict(features)
     2 print("Train Accuracy:",metrics.accuracy_score(labels, pred_train))
     3
```

```
Train Accuracy: 1.0
```

```
[ ]  1 pred_test = clf.predict(test_features)
     2 print("Test Accuracy:",metrics.accuracy_score(test_labels, pred_test))
     3 D
```

```
Test Accuracy: 0.3513244621315757
```

می‌بینیم که به صورت واضح مدل ما overfit شده ، و از قبل هم شدید تر است، در حدی که همه‌ی داده‌های آموزش را حفظ کرده ولی روی داده‌های تست را دقت ۳۵ درصدی دارد.

ت) مانند دستورالعملی که در اسلایدها آمده، ابتدا مانند قسمت ب ، classifier را آموزش می‌دهیم (۲۵ ایپاک) سپس چند لایه آخر از resnet را از حالت freeze خارج می‌کنیم تا بتوانیم آن ها را هم آموزش دهیم. بعد از آن هر دو را باهم، به اندازه ۱۵ ایپاک دیگر آموزش می‌دهیم (classifier و لایه‌های آزاد شده آخر). تا بتوانیم دقت را افزایش دهیم.

در زیر کد قسمتی را میبینید که لایه‌های freeze شده را trainable می‌کند.

```
x = 0
unfreezed = 135
for p in model.features.parameters():
  if(x >= unfreezed ):
    p.requires_grad = True
  x += 1

print("Total number of layers= ", x)
print("number of unfreezed layers= ", x - unfreezed)

pytorch_total_params = sum(p.numel() for p in model.parameters())
pytorch_total_trainable_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print('pytorch_total_params= ', pytorch_total_params )
print('pytorch_total_trainable_params= ', pytorch_total_trainable_params)
```

```
Total number of layers=  159
number of unfreezed layers=  24
pytorch_total_params=  23909636
pytorch_total_trainable_params=  12480708
```

می‌بینیم که با خارج کردن ۲۴ لایه آخر از حالت فریز، تعداد پارامترهای قابل آموزش به ۱۲ میلیون رسیده، در حالی که قبل از آن ۴ میلیون بود.
سپس آموزش را شروع می‌کنیم:

```
Epoch 14/14
----------
Iterating through data...
train Loss: 28.4621 Acc: 0.8582
Iterating through data...
val Loss: 76.4261 Acc: 0.4213

Training complete in 61m 36s
Best val Acc: 0.427434
=> saving checkpoint
```

این مدل ابتدا با خارج کردن ۱۵ لایه از حالت فریز آموزش دید، اما پیشرفت چشمگیری نداشت، سپس یکبار دیگه با ۲۴ لایه تست کردم که باز هم نتایج خیلی جالب نبود اما بهتر از قبل شد. تصویر بالا بعد از پایان هر دو آموزش است.
می‌بینیم که دقت در آموزش و تست بالاتر از قسمت ب است و پیشرفت داشتیم. اما همچنان overfitting وجود دارد.

ث) این قسمت هنوز کامل نشده، با تاخیر بیشتر ارسال خواهد شد.

با تشکر