Session4_CNN

Exercise *3*

Number of epochs	*frozen*	*unfrozen*
2 epochs	*Epoch 1/2, Loss: 0.8285023129885764* *Epoch 2/2, Loss: 0.6177520032047921*	*Epoch 1/2, Loss: 0.4440410536763918* *Epoch 2/2, Loss: 0.21804531144402217*
3 epochs	*Epoch 1/3, Loss: 0.8363932815506635* *Epoch 2/3, Loss: 0.6153141984625545* *Epoch 3/3, Loss: 0.5893527653516101*	*Epoch 1/3, Loss: 0.44071715576645665* *Epoch 2/3, Loss: 0.21380107268176574* *Epoch 3/3, Loss: 0.11855014333444293*
5 epochs	*Epoch 1/5, Loss: 0.8229046010833871* *Epoch 2/5, Loss: 0.6164884779535597* *Epoch 3/5, Loss: 0.5858670111431186* *Epoch 4/5, Loss: 0.5764337191191475* *Epoch 5/5, Loss: 0.5681294006726626*	*Epoch 1/5, Loss: 0.4460315287608625* *Epoch 2/5, Loss: 0.21330634143460742* *Epoch 3/5, Loss: 0.12214335390959707* *Epoch 4/5, Loss: 0.07992329932225253* *Epoch 5/5, Loss: 0.0595302439188761*

برای *unfreeze* کردن این قسمت کد را به این صورت تغییر میدهیم:

ابتدا همه را فریز می‌کنیم

```
for name, param in resnet_unfrozen.named_parameters():
    param.requires_grad = False
```

آزاد کردن *layer4* آخرین بلوک کانولوشنی

```
for param in resnet_unfrozen.layer4.parameters():
    param.requires_grad = True
```

آزاد کردن لایه*fully-connected*

```
for param in resnet_unfrozen.fc.parameters():
    param.requires_grad = True
```

وقتی از *ResNet* برای *CIFAR-10* استفاده می‌کنی، سه لایه مهم وجود دارند:

لایه‌های اولیه ⟶ لبه‌ها، خطوط، رنگ‌ها

لایه‌های میانی ⟶ اشکال کوچک، الگوها

لایه‌های پایانی ⟶ مفهوم اشیا*(dog, airplane, car, bird ...)*

- **لایه‌های اولیه عملاً ربطی به دیتاست ندارند**

این لایه‌ها فقط لبه و رنگ و *texture* یاد می‌گیرند و برای هر دیتاستی یکسان است. پس *Freeze* کردن آنها منطقی است.

- **لایه‌های آخر مخصوص *ImageNet* هستند**

این لایه‌ها ویژگی‌هایی مخصوص ۱۰۰۰ کلاس *ImageNet* تولید می‌کنند، مثلاً: گوش سگ، بدنه‌ی ماشین، بال هواپیما، شکل قاشق، چشم انسان و ... ولی *CIFAR-10* فقط شامل این ۱۰ کلاس است:

airplane, car, bird, cat, deer, dog, frog, horse, ship, truck

پس *ResNet* در ابتدا ویژگی‌هایی تولید می‌کند که برای *ImageNet* خوبند ولی برای *CIFAR-10* نه.

پس وقتی *Layer4* را *Unfreeze* می‌کنیم : لایه‌های قبلی ثابت می‌مانند و پس سرعت یادگیری کم نمی‌شود و فقط آخرین *Block*خودش را با ویژگی‌های *CIFAR-10* تطبیق می‌دهد

پس ویژگی‌های *ResNet* از ویژگی‌های *ImageNet* تبدیل می‌شود به ویژگی‌های*.CIFAR-10*

نتیجه:

شبکه دقیقاً برای دیتاست جدید *Fine-Tune* می‌شود و دقت بالاتر می‌رود.

جابه‌جایی *adam* با *sgd* و *momentum 0.9*

adam	*SGD with momentum 0.9*
Epoch 1/2, Loss: 0.8285023129885764 *Epoch 2/2, Loss: 0.6177520032047921*	*Epoch 1/2, Loss: 0.9758166646027504* *Epoch 2/2, Loss: 0.677689578336523*
Epoch 1/5, Loss: 0.8229046010833871 *Epoch 2/5, Loss: 0.6164884779535597* *Epoch 3/5, Loss: 0.5858670111431186* *Epoch 4/5, Loss: 0.5764337191191475* *Epoch 5/5, Loss: 0.5681294006726626*	*Epoch 1/5, Loss: 0.9771978472504774* *Epoch 2/5, Loss: 0.6757536112804852* *Epoch 3/5, Loss: 0.6284997473135019* *Epoch 4/5, Loss: 0.6062377333031286* *Epoch 5/5, Loss: 0.5908396911743047*

همانطور که مشاهده میکنیم سرعت همگرایی *SGD* آهسته‌تر از *ADAM* است.

جابه‌جایی *resnet* با *vgg16*

```
vgg16.classifier = nn.Sequential(
    nn.Linear(512 * 7 * 7, 4096),  # توجه: VGG16 ۵۱۲ خروجی دارد x7x7
    nn.ReLU(True),
    nn.Dropout(0.5),
    nn.Linear(4096, 4096),
    nn.ReLU(True),
    nn.Dropout(0.5),
    nn.Linear(4096, 10)  # 10 کلاس برای CIFAR-10
)
```

vgg16 = vgg16.to(device)

در *VGG16* احتمال *vanishing gradient* بیشتر است و لایه‌های زیادی که دارد باعث می‌شود برای آموزش سخت‌تر باشد. تعداد پارامتر *VGG16* حدودا ۱۳۸ میلیون است که برای *CIFAR10* ایکه داده کوچک است مدل‌های خیلی بزرگ مثل *VGG16* ، *Overfit* می‌کنند.

Epoch 1/2, Loss: 1.5621331770096898

Epoch 2/2, Loss: 1.162198828766718

این اعداد هم *loss* اجرای *vgg16* روی *cifar10* هستند.