گذارش کار پروژه:

این مدل با استفاده از کتابخانه pytorch روی مجموعه داده CIFAR10 که شامل 60هزار تصویر رنگی به اندازه 32*32 در 10 کلاس است. 50هزار عکس در train set و 10هزار عکس آن در test set قرار دارد. این مدل با استفاده از معماری residual neural network طراحی و از مدل از پیش تمرین داده شده ResNet50 روی مجموعه داده ImageNet برای ئیاده سازی استفاده شده است. برای اینکه مدل بتواند روی data set ما تمرین داده شود یک لایه fully connected به آن اضافه شد است تا سایز خروجی مدل ResNet50 را از 128 به 10 که تعداد کلاس های data set ما است برساند.

مراحل پیاده سازی:

ابتدا کتابخانه های مورد نیاز را وارد میکنیم

```
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
from PIL import Image
import torch
import torchvision
from torchvision import datasets, models, transforms
import torch.nn as nn
from torch.nn import functional as F
import torch.optim as optim
from torch.autograd import Variable
```

سپس train set و test set را آماده میکنیم

مرحله اول ترنسفورم های مربوطه را مینویسیم:

```
data_transforms = {
    'train':
    transforms.Compose([
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465),
(0.2023, 0.1994, 0.2010)),
```

```
    ]),
    'test':
    transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465),
(0.2023, 0.1994, 0.2010))
    ]),
}
```

سپس data set را از کتابخانه آماده pytorch دانلود کرده و ترنسفورم هارا روی تصاویر اعمال میکنیم:

```
data_sets = {
    'train': torchvision.datasets.CIFAR10(root='data',
download=True, train=True, transform=data_transforms['t
rain']),
    'test': torchvision.datasets.CIFAR10(root='data', d
ownload=True, train=False, transform=data_transforms['t
est'])
}
```

سپس data set را به کلاس دیتا لودر pytorch تبدیل میکنیم تا به اندازه batch sizeهایی که ما میخواهیم دیتا هارا دسته بندی کند:

```
dataloaders = {
    'train':
    torch.utils.data.DataLoader(data_sets['train'],
                                batch_size=batch_size,
                                shuffle=True,
                                num_workers=0),
    'test':
    torch.utils.data.DataLoader(data_sets['test'],
                                batch_size=batch_size,
                                shuffle=False,
                                num_workers=0)
```

```
}
```

حال نوبت آماده کردن مدلمان فرا میرسد

در این مرحله اول مدل را از مدل های آماده کتابخانه pytorch دانلود میکنیم:

```
model = models.resnet50(pretrained=True)
```

آنرا به کارت گرافیک منتقل میکنیم:

```
model = model.cuda() if use_cuda else model
```

سپس یک لایه فولی کانکتد به مدل اضافه میکنیم تا خروجی مدل را به اندازه کلاس های دسته بندی data setمان که ده تاست برسانیم:

```
num_ftrs = model.fc.in_features
model.fc = torch.nn.Linear(num_ftrs, len(data_sets['tra
in'].classes))
```

لایه اخر را هم به کارت گرافیک منتقل میکنیم:

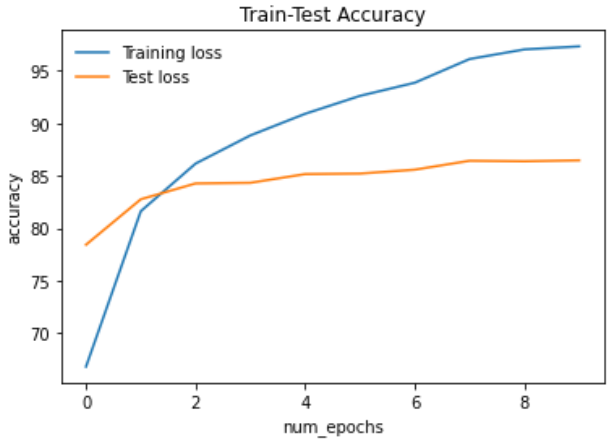
```
model.fc = model.fc.cuda() if use_cuda else model.fc
```

حال برای معیاراختلاف خروجی مدل از متد CrossEntropyLoss، برای بهینه سازی از متد SGD و از یک زمانبند که مقدار ریت یادگیری را در طول زمان کاهش دهد برای آموزش مدلمان بهره میگیریم:

```
criterion = nn.CrossEntropyLoss()
```

```
optimizer = optim.SGD(model.parameters(), lr=learning_r
ate, momentum=0.9, weight_decay=5e-4)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
step_size=7, gamma=0.1)
```

در نهایت مدل را به طول epoch 10 آموزش میدهیم و چک میکنیم تا در هر epoch به ازای بالاترین دقت بدست آمده تا به حال وزن های مدل را ذخیره میکنیم تا در آخرین مرحله مطمئن باشیم مدل با بالاترین دقت در test set ذخیره شود و پس از پایان آموزش نتایج زیر بدست میاید:



مدل در دسته داده های تمرینی (train set) به بالاترین دقت 97.33 درصد و در دسته داده های ازمایشی (test set) به بالاترین دقت 86.39 درصد میرسد.

حال برای آزمایش ده عکس را به صورت تصادفی از train set انتخاب کرده و آنهارا به مدل میدهیم تا دسته بندی آنهارا تشخیص دهد و از ده مورد مدل ما به نه مورد پاسخ صحیح میدهد:


dog: True
automobile: True
dog: True
truck: False
frog: True
cat: True
deer: True
airplane: True
automobile: True
truck: True