به نام خدا

گزارش پروژه اختیاری

درس داده کاوی

استاد درس: دکتر حاجی محمدی

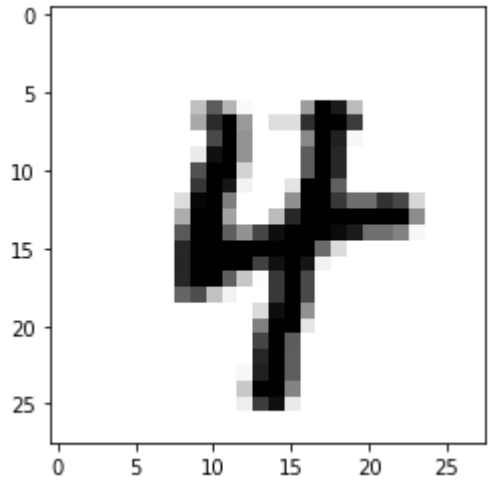
نام و شماره دانشجویی: مهکامه عرب گری ۹۷۱۳۰۵۴

پروژه اختیاری: فرآیند ساخت یک شبکه عصبی، آموزش آن، آزمایش و در نهایت ذخیره آن
هدف تشخیص رقم دست‌نویس با استفاده از PyTorch است و در طول مسیر یک طبقه بندی کننده رقمی دست نویس را از صفر توسعه می دهیم.

مرحله ی اول شناساندن دیتاست است. برای این پروژه، ما از پایگاه داده MNIST استفاده خواهیم کرد. این مجموعه ای از ۷۰۰۰۰ رقم دست نویس است که به ترتیب به مجموعه آموزشی و ۶۰۰۰۰ و آزمایشی ۱۰۰۰۰ تصویر تقسیم شده است. بخش مهمی از کار کلین کردن دیتا است که ،
PyTorch برای ما پیاده‌سازی آسانی برای دانلود داده‌های تمیز شده و آماده شده با استفاده از چند خط کد ارائه می‌کند. قبل از شروع، باید تمام importهای لازم را انجام دهیم. سپس نوعی ویرایش سفارشی با استفاده از torchvision.transforms بر روی تصاویر انجام می دهیم تا همه تصاویر دارای ابعاد و ویژگی های یکسان باشند.
تابع transforms.ToTensor() تصویر را به اعدادی تبدیل می کند که توسط سیستم قابل درک است. این تصویر را به سه کانال رنگی (تصاویر جداگانه) جدا می کند: قرمز، سبز و آبی. سپس پیکسل‌های هر تصویر را به روشنایی رنگشان بین 0 تا 255 تبدیل می‌کند. سپس این مقادیر به محدوده‌ای بین ۰ تا ۱ کاهش می‌یابند. اکنون تصویر به یک Torch Tensor تبدیل می‌شود.
تابعtransforms.Normalize() تنسور را با میانگین و انحراف استاندارد نرمال می کند که به ترتیب دو پارامتر است.
اکنون در نهایت مجموعه داده ها را دانلود می کنیم، آنها را به هم می زنیم و هر یک از آنها را تبدیل می کنیم. مجموعه داده‌ها را دانلود کرده و در DataLoader بارگذاری می‌کنیم، که مجموعه داده و نمونه‌گر را ترکیب می‌کند و تکرارکننده‌های تک یا چند فرآیندی را روی مجموعه داده ارائه می‌کند.
batch size اندازه دسته ای از تعداد عکسهایی است که می خواهیم یکباره بخوانیم.

مرحله دوم شناخت بهتر دیتاست است. در این مرحله، ما برخی از تجزیه و تحلیل داده های اکتشافی را روی تصاویر و تنسور انجام می دهیم و شکل عکسها و برچسب ها را بررسی کنیم.در هر دسته ۶۴ تصویر وجود دارد و هر تصویر دارای ابعاد ۲۸*۲۸ پیکسل است. این ۶۴ تصویر باید به ترتیب دارای ۶۴ برچسب باشند. حال با دستور زیر یک عکس(مثلا اولی) از مجموعه آموزشی نمایش می دهیم.

```
plt.imshow(images[0].numpy().squeeze(), cmap='gray_r');
```



حال چند تصویر دیگر برای درک بهتر دیتاست نمایش می دهیم. کد زیر شبکه ای از تصاویر را به ترتیب تصادفی ایجاد می کند.

```
figure = plt.figure()
num_of_images = 60
for index in range(1, num_of_images + 1):
  plt.subplot(6, 10, index)
  plt.axis('off')
  plt.imshow(images[index].numpy().squeeze(), cmap='gray_r')
```

مرحله سوم ساخت شبکه عصبی است که شامل یک لایه ورودی (لایه اول)، یک لایه خروجی از ده نورون (یا واحدها، دایره ها) و دو لایه پنهان در بین آنها است. با ماژول torch.nn در PyTorch شبکه را ساده سازی می کنیم.

برای رپ کردن لایه های شبکه از nn.Sequential استفاده می کنیم. با تابع فعال ساز ReLU سه لایه یافت می شود. لایه خروجی یک لایه خطی با تابع فعال ساز LogSoftmax می باشد چون این یک مشکل classification است.
حال برای تعریف negative log-likelihood loss با دو تابع ()LogSoftmax و ()NLLLoss به عنوان از cross-entropy loss عمل می کنند.
در لایه اول ۷۸۴ (همان ۲۸*۲۸) واحد داریم چون هر عکس را قبل از ارسال به داخل شبکه عصبی flatten می کنیم.
مرحله چهارم تنظیم weight است. قبل از backward pass,، وزن های مدل روی مقادیر none دیفالت تنظیم می شوند. با تابع ()backward ، بروزرسانی weightها را انجام می دهیم.

مرحله پنجم فرآیند اصلی ترینینگ است. شبکه ی عصبی ما پس از تکرار روی ترینینگ ست و آپدیت کردن weight ها، با استفاده از ماژول torch.optim، مدل را بهینه سازی می کنیم و gradient descent انجام می دهیم و وزن ها را با back-propagation آپدیت می کنیم. پس در هر epoch یک کاهش تدریجی روی training loss را شاهد هستیم. (که ۳.۱۵ دقیقه طول کشید.)

مرحله ششم تست و ارزیابی است. تقریباً کار ما تمام شده و مدل آماده است و باید آن را ارزیابی کنیم. با تابع ()view_classify احتمالات تصویر و کلاس پیش بینی شده را نشان می دهیم.عکسی را از مجموعه validation set که قبلا ساختیم به مدل آموزش دیده می فرستیم تا ببینیم مدل چطور کار می کند. حال با مجموعه validation set تعداد کل پیش بینی های صحیح و دقت را محاسبه می کنیم.

```
Number Of Images Tested = 10000
```

```
Model Accuracy = 0.974
```

که دقت ما ۹۷.۴٪ است چون مجموعه داده‌های ما تمیز بود، تصاویر متنوعی زیادی داشت که به خوبی در هم ریخته شده بودند.

در **مرحله ی هفتم** و در واقع آخرین مرحله، مدل را سیو می کنیم.