Project 3 : Fashion MNIST classifying project

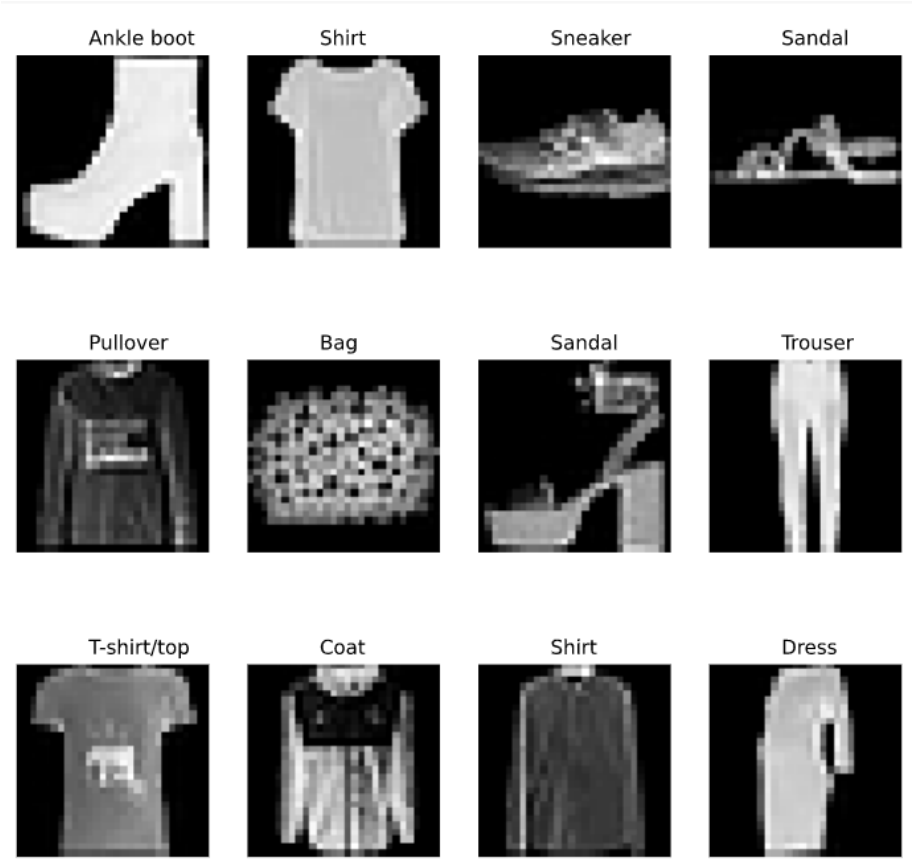
نام فایل

FMNIST_00.ipynb

در این پروژه دسته بندی تصاویر دیتاست Fashion MNIST که شامل تعداد ۶۰ هزار تصویر سیاه و سفید (یک کاناله) در قالب ۱۰ دسته کالا می باشند خواسته شده است.

دسته بندی کالا ها عبارتند از:

```
(0, 'T-shirt/top'),
(1, 'Trouser'),
(2, 'Pullover'),
(3, 'Dress'),
(4, 'Coat'),
(5, 'Sandal'),
(6, 'Shirt'),
(7, 'Sneaker'),
(8, 'Bag'),
(9, 'Ankle boot')
```



نتیجه اجرای پروژه با شرایط زیر:

- با سه لایه کانولوشن و مکس پولینگ و لیکی رلو =====> دقت ۹۴ درصد
- در این برنامه از GPU استفاده شده است.
- می توان در برنامه پس از اجرای مکس پولینگ داده ها را نرمال کرد (nn.BatchNorm2d) و سپس لیکی رلو را اجرا نمود.

وارد کردن داده های اولیه

برای گرفتن داده ها می توان از دیتاست های موجود در اینترنت **(گیت هاب)** یا استفاده از نسخه ای که در تورچ ویژن وجود دارد استفاده کرد.

لذا در برنامه هر دو روش ذکر شده است ولی از روش دانلود داده و قرار دادن آنها در گوگل درایو و سپس بارگذاری آن در برنامه استفاده شده است.

در این روش یعنی خواندن داده ها از یک دیتاست عادی باید گام های زیر انجام شوند:

۱- وارد کردن داده ها : برای این امر از دو تابع استفاده می شود که منبع داده ها با پسوند gz را خوانده و در برنامه باز می کند و امکان استفاده از داده ها را فراهم می کند.
۲- پس از وارد کردن داده ها و کنترل ابعاد داده ها، نرمال سازی آنها را انجام می دهیم. این کار با تقسیم کردن داده ها به ماکزیموم آنها صورت می گیرد. در این حالت حداقل مقدار صفر و حداکثر آن یک خواهد بود.
۳- داده های نرمال شده را به تنسور تبدیل می کنیم.
۴- پس از آن دسته بندی آموزش و تست انجام می شود (train_test_split)
۵- سپس داده های آموزش و لیبل های آن (traindata, trainlabel) و
داده های تست و لیبل های آن testdata, testlabel) بصورت جداگانه تبدیل به دیتاست می گردند.
۶- در گام بعدی دیتاست های ایجاد شده به دیتالودر تبدیل می گردد تا در آموزش مدل استفاده شود.

نوشتن تابع ایجاد مدل شبکه عصبی کانولوشن

در این مدل از سه لایه کانولوشن و مکس پولینگ و تابع فعال ساز لیکی رلو استفاده می گردد.

گام های نوشتن تابع به شرح زیر است:

۱- ایجاد یک کلاس با ارث بری از nn.Module
۲- تعریف موارد زیر در تابع __init__ کلاس تعریف شده بالا:

a. لایه کانولوشن اول که ماتریس تصویر با یک کانال را به ۱۰ کانال (۱۰ ماتریس کرنل) با سایز ۵ در ۵ تبدیل کند (استراید و پدینگ مساوی یک)
b. لایه کانولوشن دوم که ماتریس تصویر با یک کانال را به ۲۰ کانال (۲ ماتریس کرنل) با سایز ۵ در ۵ تبدیل کند (استراید و پدینگ مساوی یک)
c. لایه کانولوشن سوم که ماتریس تصویر با یک کانال را به ۴۰ کانال (۲ ماتریس کرنل) با سایز ۲ در ۲ تبدیل کند (استراید و پدینگ مساوی یک)
d. محاسبه تعداد عناصر داده ها در ماتریس خروجی جهت تعریف بردار ورودی به لایه های FFN
e. تعریف لایه یک شبکه عصبی پیش بینی کننده خروجی با توجه به تعداد ورودی (40x2x2 = 160)
f. تعریف لایه دوم با ۸۰ نود
g. تعریف لایه سون با ۵۰ نود و ۱۰ خروجی (تعداد دسته بندی های مورد نیاز) ۱۶۰-> ۸۰ -> ۵۰ -> ۱۰

۳- تعریف تابع forward:
a. وارد کردن ماتریس های داده و اجرای کانولوشن ۱ و سپس لیکی رلو
b. وارد کردن ماتریس های داده و اجرای کانولوشن ۲ و سپس لیکی رلو
c. وارد کردن ماتریس های داده و اجرای کانولوشن ۲ و سپس لیکی رلو
d. وارد کردن ماتریس داده و اجرای لایه اول شبکه عصبی و لیکی رلو
e. وارد کردن ماتریس داده و اجرای لایه دوم شبکه عصبی و لیکی رلو
f. وارد کردن ماتریس داده و اجرای لایه سوم شبکه عصبی و ذخیره خروجی
g. برگرداندن خروجی بعنوان خروجی تابع فوروارد

۴- استفاده از CrossEntropyLoss بعنوان تابع خطا

۵- استفاده از بهینه ساز Adam با نرخ یادگیری ۰,۰۰۱ و weight_decay = 0.00001

در ادامه برنامه تابعی که آموزش و تست مدل را انجام می دهد نوشته شده است که با مدل تعریف شده توسط تابع بالا مقادیر هدف را پیش بینی و با مقایسه با لیبل مقدار خطا و دقت مدل را بر می گرداند.