

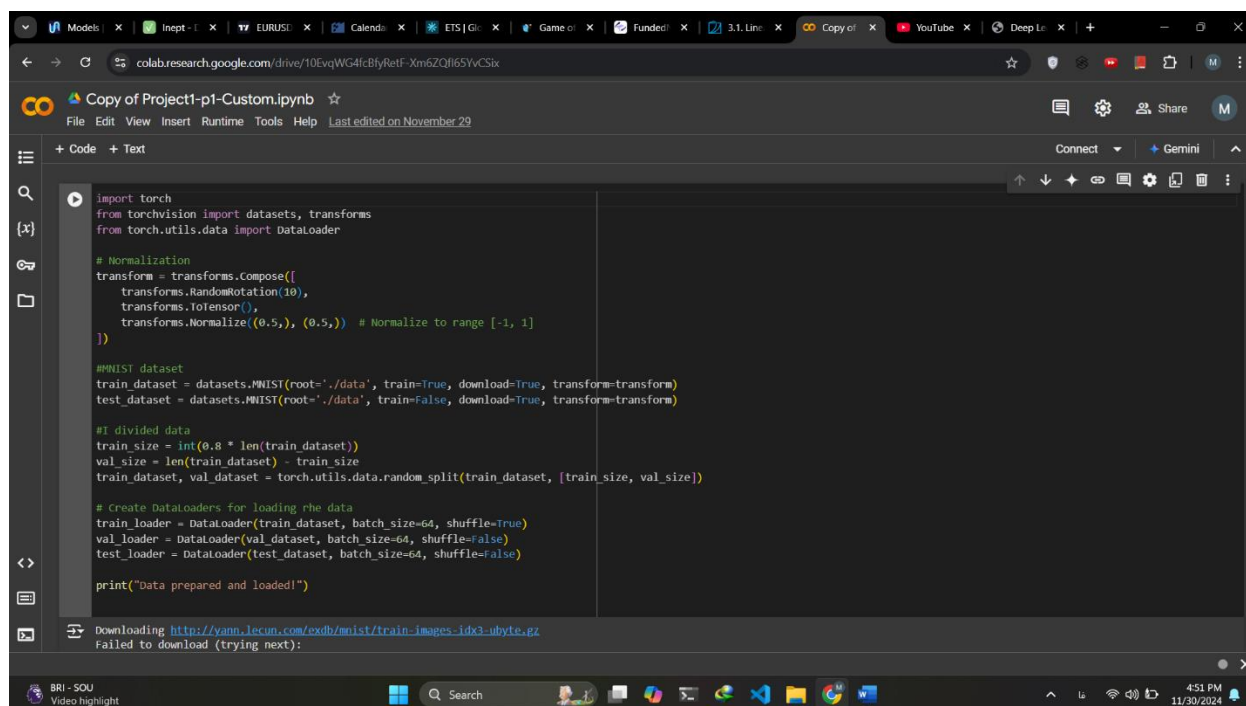
به نام خدا

پروژه اول دیپ لرنینگ

محمد محسن هایونی

بخش اول:

در بخش اول من دیتاست MNIST رو برای ترین کردن مدل دلخواه و مدل پری ترین شده انتخاب کردم. ابتدا دیتاست را دانلود نموده و دیتا ها را pre-process کردم.



```
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

# Normalization
transform = transforms.Compose([
    transforms.RandomRotation(10),
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,)) # Normalize to range [-1, 1]
])

#MNIST dataset
train_dataset = datasets.MNIST(root='./data', train=True, download=True, transform=transform)
test_dataset = datasets.MNIST(root='./data', train=False, download=True, transform=transform)

#I divided data
train_size = int(0.8 * len(train_dataset))
val_size = len(train_dataset) - train_size
train_dataset, val_dataset = torch.utils.data.random_split(train_dataset, [train_size, val_size])

# Create DataLoaders for loading the data
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

print("Data prepared and loaded!")
```

Downloading <http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz>
Failed to download (trying next):

دیتا را به سه بخش train, val, test تقسیم کردم و با کمک ترنسفورم آن را تبدیل به تنسور و سپس نرمال کردم.

با جستجو در وب به یک مدل پیشنهادی رسیدم و آن را ۱۰ ایپاک ترین کردم که به مرور دقت آن بهبود پیدا کرد و در ایپاک آخر به دقت ۹۸.۸۴٪ رسید.

برای ResNet ابتدا ورودی را از تصویر تک بعدی grayscale تبدیل به یک تصویر ۳ بعدی کردم که با ورودی معمول آن مطابقت داشته باشد. سپس Max-pooling را حذف کردم و خروجی آن را که در حالت عادی ۲۵۶ عدد است به ۱۰ عدد (برای اعداد ۰ تا ۹) تغییر دادم. در ResNet و در اولین تلاش accuracy خیلی پایینی داشتم در حدود 50% برای حل این موضوع ابتدا تصویر را در بخش resize به اندازه ۲۲۴*۲۲۴ که اندازه خود ResNet است در آوردم ولی در این حالت مشکلی که ایجاد میشد پر شدن سریع رم و از کار افتادن گوگل کولب بود. برای بهبود این شرایط ابتدا اندازه را ۶۴*۶۴ کردم سپس بچ سایز را از ۶۴ به ۱۶ کاهش دادم و تعداد ایپاک ها را از ۱۰ به ۵ کاهش دادم. پس از این تغییرات سرعت آموزش افزایش پیدا کرد و دقت هم به مقدار قابل توجهی بالا رفت. ولی به چیزی که من میخواستم نمی رسید برای همین مقدار learning rate را به طور قابل توجهی کاهش دادم. زمان آموزش بالاتر رفت ولی دقت به چیزی که مد نظر بود رسید. در آخرین ایپاک دقت مدل به ۹۹.۱۷٪ رسید.

*مقایسه ی وزن های دولایه از هر مدل در نوت بوک مربوطه آورده شده است.

در کل مدل کاستومی که ساخته شد زمان بیشتری برای یادگیری داشت و در نهایت حتی با تعداد ایپاک بالاتر هم نتوانست به دقت مدل pre-train برسد. مدل Resnet با اینکه تعداد ایپاک کمتری داشت ولی به علت دارا بود وزن های قبلی توانست دقت بسیار بهتری ارائه دهد.

اوپتیمایز های هر دو مدل از نوع آدام استفاده شده است که به سریع تر شدن یادگیری با تغییر پارامتر کمک میکند.

بخش دوم:

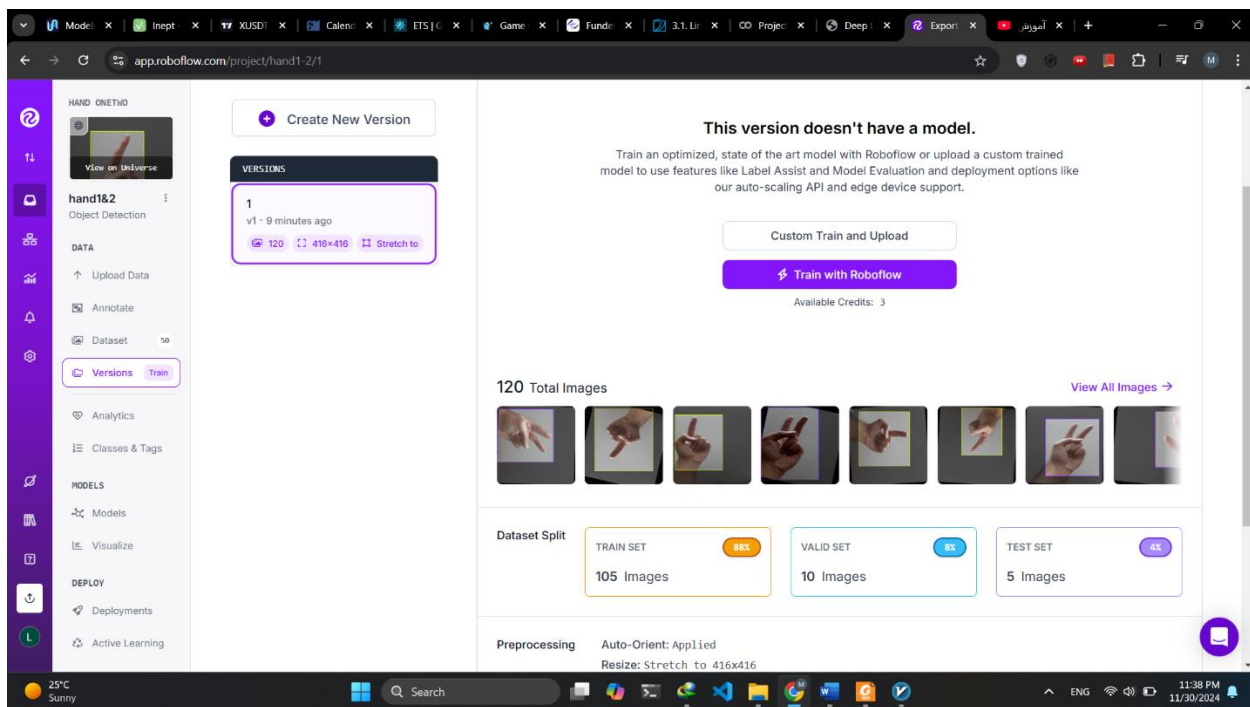
برای این بخش برای راحت تر شدن کار و استفاده از قابلیت های گوگل دیتاست را به صورت مستقیم بر روی درایو دانلود کردم و از آن استفاده کردم.

سپس با کمک کد گفته شده در ویدئو دیتاست را با کمک پسوند های فایل تقسیم بندی کردم و آنها را به سه بخش `test`, `train`, `val` تقسیم کردم .

در انتها و قبل از فاین تیون کردن مدل فایل `data.yaml` را نسبت به آدرس پوشه ها و کلاس های گفته شده در توضیحات دیتاست تغییر دادم و فایل جدید را در پوشه مربوطه قرار دادم.

برای مرحله `fine-tune` از مدل `s` در `yolo5` استفاده کردم چون دقت خیلی بالایی نمیخواستم. در ابتدا اندازه بچ من ۶۴ و تعداد اپیک ۵۰ بود که باعث میشد گوگل کولب سشن را قطع کند برای همین اندازه بچ را ۳۲ و اپیک را ۳۰ قرار دادمو مدل `fine-tune` شد و دقت قابل قبولی کسب کرد.

برای بخش بعدی که ساختن دیتاست بود حدود ۵۴ عکس از اعداد یک و دو که با دست نمایش داده شده بود گرفتم . ابتدا فرمت عکس ها را به `jpeg` تغییر دادم و آنها را به کمک `roboflow` لیبل زدم و ۴ نوع `data augmentation` انجام دادم تا هم تعداد داده ها بیشتر شود و هم دقت بالا تر برود.



در انتها دیتاست را به مدل یولو دادم و نتیجه مورد قبولی گرفتم. برای این نتیجه با مدل I یولو و با ۵۰ اپیک ترین کردم.

