مبانی هوش محاسباتی

دانشگاه فردوسی مشهد

فاز چهارم

نیمسال دوم تحصیلی ۱۴۰۲-۱۴۰۱

گروه مهندسی کامپیوتر مهلت ارسال: ۱۶ تیر ساعت ۲۳:۵۹

در فاز آخر قصد داریم مدل شبکه عصبی برای طبقه بندی ارقام دیتاست "Digit-Five" طراحی کنیم. در ادامه در بخش اول دیتاست Digit-Five توضیح داده شده‌است، در بخش دوم معماری مدل مورد نظر بررسی شده ، در بخش سوم توضیحاتی مربوط به توابع زیان [۱] مدل آورده شده و در بخش چهارم مواردی که باید در این فاز پیاده‌سازی شود آورده شده است.

۱) دیتاست Digit-Five

دیتاست Digit-Five شامل ۵ دیتاست MNIST ، MNIST-M ، SVHN ، SYNthetic digits (SYN) و USPS می‌باشد که به هرکدام از این دیتاست‌ها یک دامین[۲] می‌گوییم. هر دامین شامل تصاویری از ارقام صفر تا نه می‌باشند که ابعاد این تصاویر $3 \times 32 \times 32$ است.[۳] هر داده از دیتاست Digit-Five شامل یک تصویر با ابعاد $3 \times 32 \times 32$ ، یک برچسب نشان دهنده رقم تصویر (عددی صحیح بین ۰ تا ۹) و یک برچسب نشان دهنده دامین تصویر (عددی صحیح بین ۰ تا ۴) [۴] می‌باشد. چندین نمونه از تصاویر هر دامین در شکل ۱ آورده شده‌است.

---

[۱] Loss Functions

[۲] Domain

[۳] تصاویر دیتاست MNIST رنگی نیستند و برخلاف چهار دیتاست دیگر، دارای یک چنل هستند. همچنین ابعاد تصاویر این دیتاست 28×28 است، اما در کدی که در اختیار شما قرار گرفته شده، تصاویر MNIST تغییر سایز داده شده‌اند و به ۳ چنل تبدیل شده‌اند و مانند چهار دیتاست دیگر دارای ابعاد $3 \times 32 \times 32$ هستند

[۴] ارقام ۰ تا ۴ به ترتیب نشان دهنده دامین‌های MNIST ، MNISTM ، SVHN ، SYN و USPS می‌باشند.

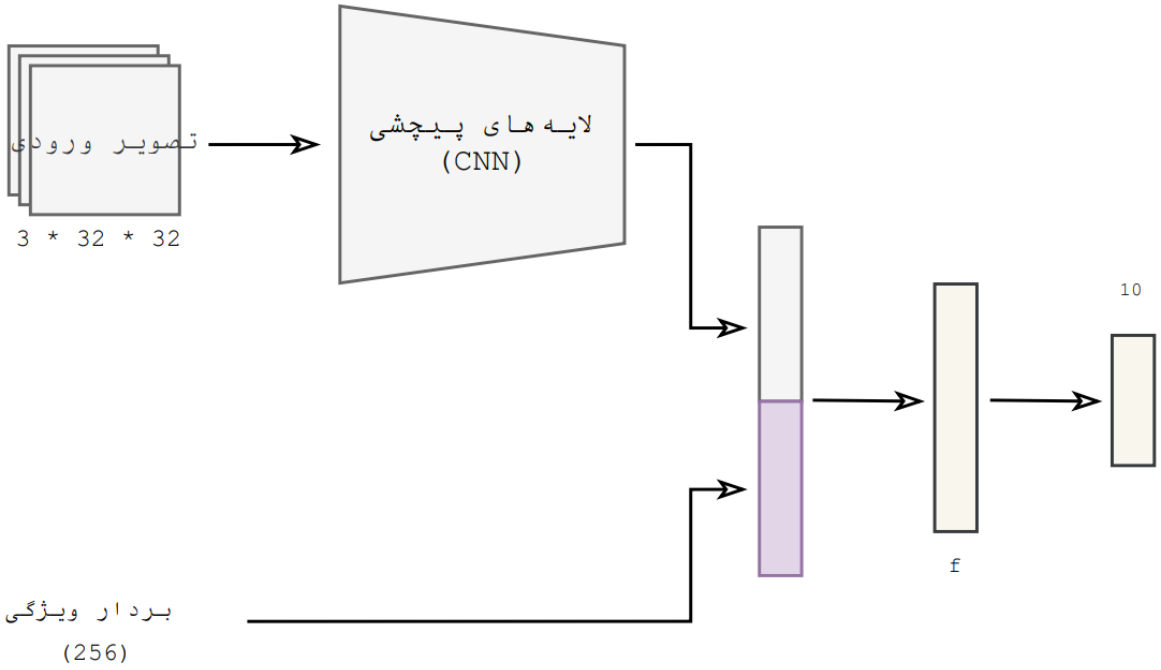
شکل ۱: هر سطر نشان دهنده ۵ نمونه داده یک رقم از هر دامین دیتاست Digit-Five است، به ترتیب از چپ به راست هر ۵ نمونه متعلق به MNIST ، MNIST-M ، SVHN ، SYN و USPS می‌باشد

نمونه‌هایی از تصاویر هر دامین به تفکیک در قسمت ۵.۳ آورده شده‌اند. همچنین فایلی با نام exploring_data.py در اختیار شما قرار گرفته که با استفاده از آن می‌توانید نمونه‌های مختلفی از این دیتاست را مشاهده کنید.

۲) معماری مدل

مدلی که قصد طراحی آن را داریم، یک تصویر با ابعاد $3 \times 32 \times 32$ و یک بردار ویژگی[۵] با ابعاد 256 را به عنوان ورودی می‌گیرد و رقم مربوط به داده ورودی را پیشبینی می‌کند. از آنجایی که مدل دو ورودی دارد، باید به نحوی این دو ورودی را ترکیب کرد، برای این کار ابتدا فقط تصویر ورودی ($3 \times 32 \times 32$) را از چند لایه پیچشی[۶] می‌گذرانیم و یک بردار بازنمایی از تصویر ورودی بدست می‌آوریم. این بردار بازنمایی که خروجی چند لایه پیچشی می‌باشد را بردار بازنمایی تصویر می‌نامیم. حال برای ترکیب دو ورودی مدل، بردار ویژگی را با بردار بازنمایی تصویر Concatenate می‌کنیم. نمایی کلی از مدل مورد نظر در شکل ۲ آمده است.

[۵] بردار ویژگی مربوط به هر داده، از اعمال PCA بر تصویر داده‌های آموزشی بدست آمده.
[۶] Convolutional

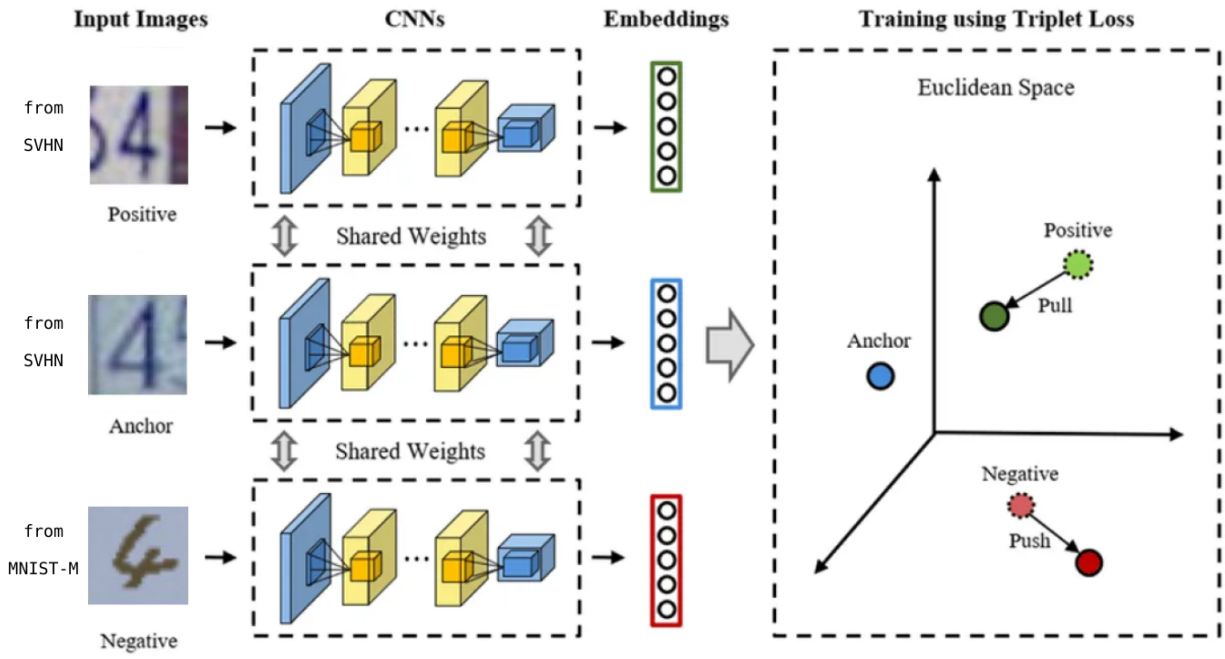


شکل ۲: نمایی کلی از مدل طبقه‌بندی ارقام

۳) توابع زیان

در مسائل طبقه‌بندی، رایج ترین تابع زیانی که استفاده می‌شود Cross Entropy است. در گام اول مدل خواسته شده که در قسمت دوم توضیح داده شد را با تابع زیان Cross Entropy برای طبقه‌بندی ارقام پیاده‌سازی می‌کنیم. در گام بعدی می‌خواهیم تابع زیان دومی به مدل اضافه کنیم. برای اینکار می‌خواهیم Triplet Loss **بر اساس برچسب دامین** تصاویر تعریف کرده و از لایه ماقبل آخر، آن را بر مدل اعمال کنیم.

ایده کلی این تابع زیان به این صورت است که اگر داده ورودی در فضایی مانند $f(.)$ بازنمایی شود، برای هر داده ورودی (که آن را Anchor می‌نامیم)، ترجیح می‌دهیم داده‌هایی که ماهیت یکسانی با داده Anchor دارند (در ساده ترین حالت، داده‌هایی که برچسب یکسانی با Anchor دارند) در فضای $f(.)$ به داده Anchor نزدیک باشند (به این داده‌ها، داده‌های مثبت برای Anchor می‌گوییم). همچنین ترجیح می‌دهیم داده‌هایی که ماهیت متفاوتی با داده Anchor دارند (در ساده‌ترین حالت داده‌هایی که برچسب متفاوتی دارند) در فضای $f(.)$ از داده Anchor دور باشند (به این داده‌ها، داده‌های منفی برای Anchor می‌گوییم).



شکل ۳: شمای کلی از Triplet Loss (با معیار فاصله اقلیدسی) . دقت کنید همه داده‌های ورودی از یک شبکه عبور می‌کنند و در شکل برای خوانایی بیشتر ۳ مدل جدا کشیده شده، اما در حقیقت هر ۳ داده از شبکه یکسانی عبور می‌کنند.

برای اینکار، فرض کنید در هر batch ، M داده وجود دارد و همانطور که بالاتر گفته شد، می‌خواهیم از لایه ماقبل آخر مدل شکل ۲، Triplet loss را اعمال کنیم، پس فضای بازنمایی داده‌ها (همان $f(.)$) در واقع خروجی لایه ماقبل آخر است.

به این صورت عمل می‌کنیم که بعد از دادن همه M داده یک batch به مدل، بازنمایی‌های $f(x_1), f(x_2), ..., f(x_M)$ را از لایه ماقبل آخر می‌گیریم.

- ابتدا باید دورترین داده مثبت (P) و نزدیکترین داده منفی (N) را برای هر داده (A) مشخص کنیم. (دقت کنید دورترین و نزدیکترین در فضای $f(.)$ مد نظر است، همچنین P و N برای هر داده (Anchor یا A) از بین $M-1$ داده دیگر در batch فعلی انتخاب می‌شود) به عبارتی باید $N_1, N_2, ..., N_M$ و $P_1, P_2, ..., P_M$ را پیدا کنیم، به طوری که

$$\begin{aligned} P_i &= \arg\max_{x_k} \quad d(f(x_k), f(x_i)) \quad \text{where} \quad label(x_k) = label(x_i) \\ N_i &= \arg\min_{x_k} \quad d(f(x_k), f(x_i)) \quad \text{where} \quad label(x_k) \neq label(x_i) \end{aligned} \tag{۱}$$

- با داشتن P و N برای همه داده‌های batch فعلی، فاصله بازنمایی هر داده (A_i) را با دورترین مثبت (P_i) و

نزدیکترین منفی (N_i) خود محاسبه می‌کنیم.

$$d_{P_i} = distance(f(A_i), f(P_i))$$
$$d_{N_i} = distance(f(A_i), f(N_i)) \quad (۲)$$

- نهایتا مقدار Triplet loss برای یک batch از داده‌ها به صورت زیر تعریف می‌شود:

$$L_{Triplet}(A, P, N) = \frac{1}{M} \sum_{i=1}^{M} max(d_{P_i} - d_{N_i} + \alpha, 0) \quad (۳)$$

که می‌خواهیم این مقدار را کاهش دهیم.

۱.(۳ هایپرپارامتر α

این پارامتر در رابطه (۳) مقدار حاشیه [۷] را مشخص می‌کند. مشابه مفهوم حاشیه در SVM ، هرچه مقدار حاشیه بیشتر باشد، مدل مجبور می‌شود $d_P - d_N$ را بیشتر کند و بازنمایی متراکم تری بدست آورد (داده‌های مثبت به هم نزدیک‌تر و داده‌های منفی دورتر)

۲.(۳ چگونگی پیاده‌سازی Triplet loss

پیاده سازی این تابع زیان عینا طبق توضیحات گفته شده در قسمت قبل است، فقط اینکه معیار فاصله که در معادله (۱) و (۲) ذکر شد را فاصله اقلیدسی در نظر می‌گیریم. به عبارتی داریم:

$$d_{P_i} = ||f(A_i) - f(P_i))||_2$$
$$d_{N_i} = ||f(A_i) - f(N_i)||_2 \quad (۴)$$

کد کلاس triplet_loss در فایل triplet_loss.py در اختیار شما قرار گرفته شده، تابع batch_hard_triplet_loss() از این کلاس تا قسمت معادله (۲) از توضیحات قسمت قبل را انجام می‌دهد. شما فقط باید از این تابع استفاده کرده و معادله (۳) را در تابع forward() پیاده سازی کنید و نهایتا از کلاس triplet_loss برای اعمال این تابع زیان در مدل خود استفاده کنید. توضیحات تکمیلی در مورد تابع batch_hard_triplet_loss() در قسمت ۵.۱ قرار گرفته است.

۳.(۳ تابع زیان نهایی

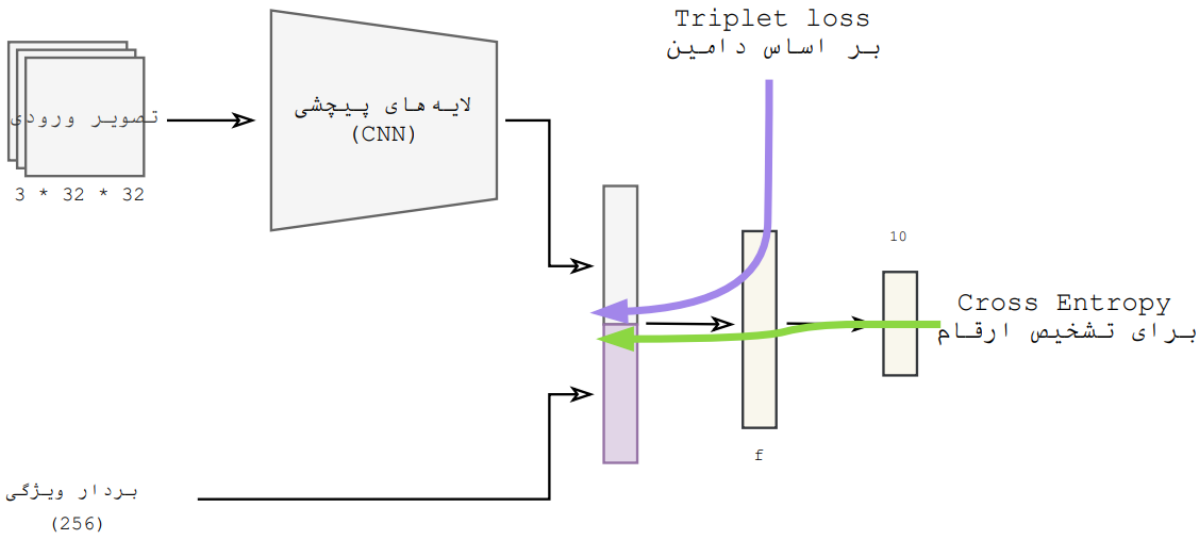
همانطور که قبلا گفته شد، علاوه بر triplet loss برحسب برچسب دامین، می‌خواهیم تابع زیان Cross Entropy را هم بر لایه آخر برای تشخیص رقم تصاویر اعمال کنیم. در نتیجه تابع زیان کل مدل برای یک batch از داده‌ها به صورت

[۷] margin

زیر محاسبه می‌شود:

$$L_T = L_{CrossEntropy} + \lambda L_{triplet}(A, P, N) \quad (۵)$$

که هایپرپارامتر λ عددی ثابت است و میزان تاثیر گذاری Triplet loss را تعیین می‌کند (اگر داشته باشیم $\lambda = 0$ در واقع Triplet loss اصلا اعمال نمی‌شود و فقط تابع زیان Cross Entropy در یادگیری پارامترهای شبکه تاثیر می‌گذارد). دقت کنید که تابع زیان Cross Entropy از لایه آخر نشر می‌کند[۸] و Triplet loss از لایه ماقبل آخر نشر می‌کند[۹] (شکل ۴)



شکل ۴: نمایی کلی از مدل به همراه توابع زیان

۴) چه چیزی باید پیاده سازی شود؟

موارد خواسته شده را با pytorch پیاده سازی کنید. (ویدئو آموزشی برای pytorch در سامانه آموزش مجازی دانشگاه قرار گرفته)

- ابتدا مدل خواسته شده که در قسمت ۲ توضیح داده شد را پیاده سازی کنید. انتخاب پارامترهای لایه‌های پیچشی، ابعاد لایه‌های میانی، توابع فعالیت[۱۰] و ... بر عهده شما است.
- پیاده سازی triplet loss را تکمیل کنید. این تابع زیان در قسمت ۳ توضیح داده شده است. و همانطور که

[۸] Backpropagation
[۹] زیرا Triplet Loss بردارهای بازنمایی که استفاده می‌کند را از لایه ماقبل آخر می‌گیرد
[۱۰] Activation Functions

گفته شد بخشی از این تابع زیان در قالب کلاس triplet_loss پیاده سازی شده و باید تابع forward آن را پیاده سازی کنید.

- در گام بعدی باید مدل طراحی شده و توابع زیان را کنار هم قرار دهید و با توجه به رابطه (۵) مدل را آموزش دهید. دقت کنید که تابع زیان Cross Entropy برای تشخیص رقم تصاویر، و Triplet Loss بر اساس دامین تصاویر می‌باشد (به این معنا که جفت مثبت و منفی که برای هر anchor تعریف می‌شود باید بر اساس برچسب دامین باشد) برای آموزش مدل از DataLoader داده‌های آموزشی که در اختیار شما قرار گرفته استفاده کنید (کد مربوط به DataLoader که در اختیار شما قرار گرفته در قسمت ۵.۳ توضیح داده شده است). همچنین Dataloader داده‌های تست هم در اختیار شما قرار گرفته، در هر مرحله[11] از آموزش، مقادیر Accuracy ارقام پیشبینی شده ، Triplet Loss ، Cross entropy Loss و Total Loss (معادله ۵) را برای داده آموزشی و داده تست گزارش کنید.

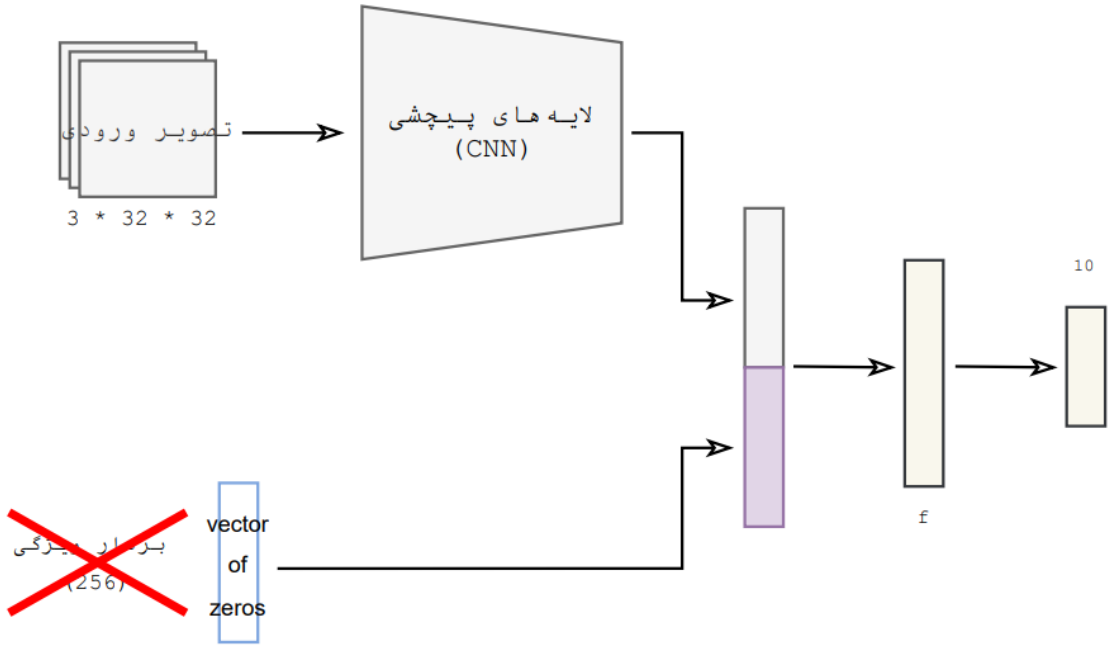
- تاثیر مقادیر مختلف پارامترهای λ و α (معادله ۵) را در آزمایشات خود بررسی کنید. برای این کار مقادیر مختلف این پارامترها را بررسی کرده، پارامتری که با آن مدلی که طراحی کرده‌اید به بهترین دقت خودش می‌رسد را گزارش دهید. توضیح دهید چرا با انتخاب این مقدار برای پارامتر λ به بهترین دقت می رسید.

۱.۴) نمره اضافی:

یک DataLoader دیگر با عنوان "test_missing" در اختیار شما قرار گرفته (توضیحات بیشتر در ۵.۲)، داده‌های این دیتاست عینا همان داده‌های دیتاست تست هستند، با این تفاوت که فرض شده بردار ویژگی در این دیتاست وجود ندارد. وقتی مدلی که در قسمت‌های قبل با داده آموزشی (که هم حاوی تصاویر بود و هم برداروِیژگی برای هر تصویر) آموزش داده‌اید را در هر مرحله از آموزش، با دیتاست "test" آزمایش کنید، از آنجایی که دیتاست "test" هم حاوی تصاویر و هم حاوی برداروِیژگی‌ها است، انتظار می‌رود هم در "train" و هم در "test" دقت خوبی بگیرید.

اما اگر همان مدل را با دیتاست "test_missing" (که در آن بردار ویژگی همه تصاویر ، یک بردار تماما صفر است، چون فرض شده بردار ویژگی تصاویر موجود نیست) آزمایش کنید می‌بینید که مدل نمی‌تواند ارقام را به خوبی تشخیص دهد. برای قسمت نمره اضافی پروژه، با روشی خلاقانه مدل را فقط با همان دیتاست آموزشی اولیه آموزش دهید به طوری که دقت خوبی در هر سه دیتاست "train" ، "test" و "test_missing" داشته باشد. در واقع در زمان آموزش به بردارهای ویژگی دسترسی داریم (شکل ۲) ، اما زمان تست فقط به تصاویر دسترسی داریم (شکل ۵)

[11] Epoch



شکل ۵: برای دیتاست "test_missing" در زمان تست فقط به تصاویر دسترسی داریم و بردار ویژگی برای هرتصویر موجود نیست

۵) ضمیمه

۱.۵) توضیحات تکمیلی درمورد پیاده‌سازی Triplet Loss

برای پیاده‌سازی triplet loss کد کلاس triplet_loss در اختیار شما قرار گرفته. تابع batch_hard_triplet_loss() تا قسمت معادله (۲) از مراحل triplet loss که در قسمت ۳ توضیح داده شده را انجام می‌دهد. این تابع با گرفتن بازنمایی‌ها ($f(x_1), f(x_2), f(x_3), ..., f(x_M)$) و برچسب‌ها، فاصله هر داده با دورترین مثبت خود و نزدیکترین منفی خود را در غالب دو متغیر dpو dn خروجی می‌دهد. در واقع این دو متغیر سمت چپ معادله (۲) از قسمت ۳ می‌باشد.

فلذا شما نیازی به پیاده سازی **پیدا کردن فاصله هر Anchor با نزدیک ترین جفت منفی و دورترین جفت مثبت** ندارید و برای اینکار از تابع batch_hard_triplet_loss() استفاده کنید.

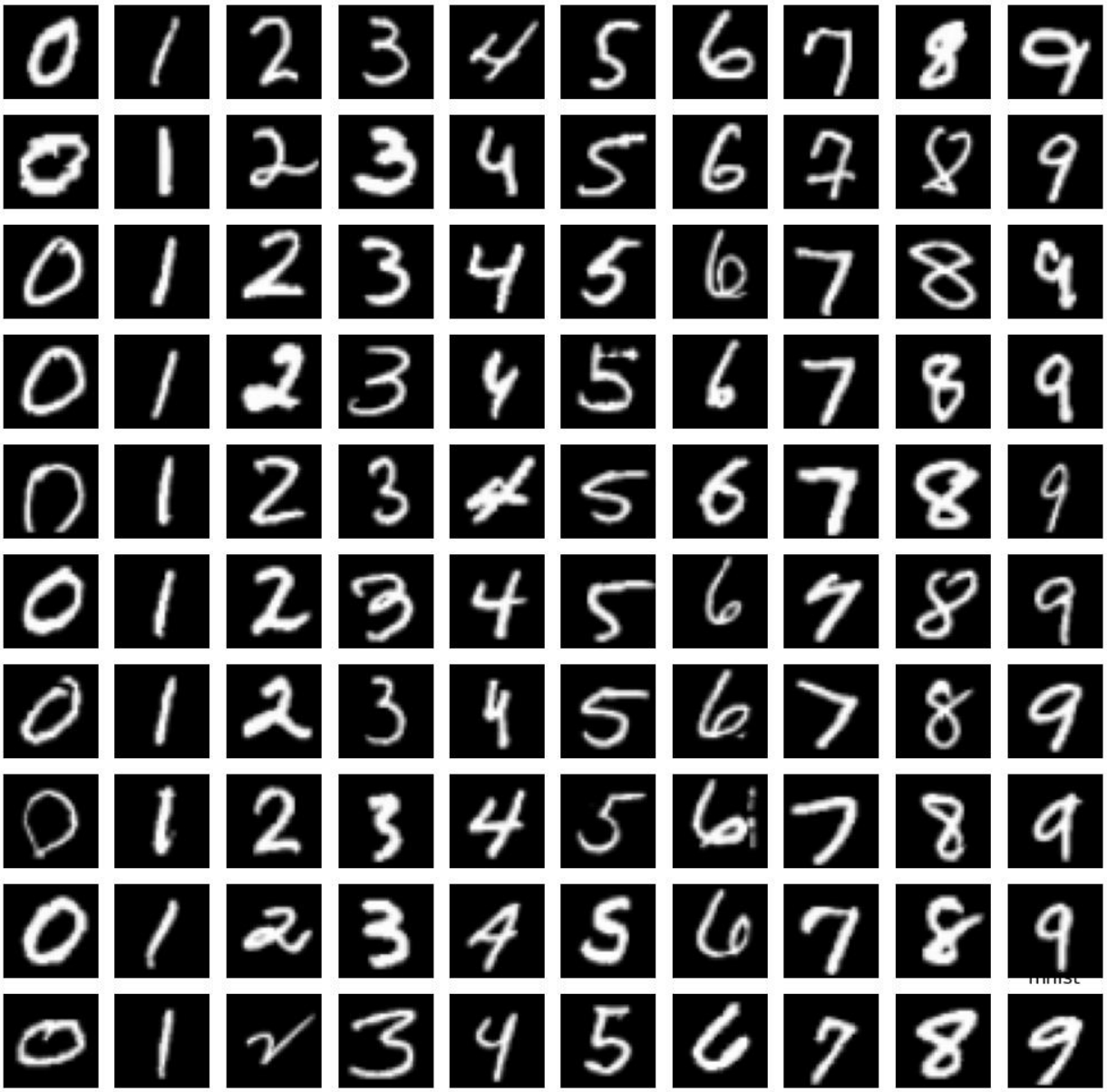
def batch_hard_triplet_loss(embeddings, labels)
این تابع بازنمایی و برچسب ورودی‌ها را در قالب متغیرهای embeddings ($M \times d$ که d ابعاد بازنمایی است) و labels ($M \times 1$) به عنوان ورودی می‌گیرد و دو آرایه که هرکدام به ترتیب فاصله اقلیدسی دورترین جفت مثبت و نزدیکترین جفت منفی برای **هر** M داده ورودی می‌باشد را خروجی می‌دهد.

۲.(۵ Dataloaders

تابع get_data_loaders در فایل utils.py در اختیار شما قرار گرفته که دیکشنری حاوی سه DataLoader برای داده آموزشی (”train”) ، داده تست (”test”) و داده تست بدون بردارهای ویژگی (”test_missing”) را خروجی می‌دهد. فایل dataloader_demo.ipynb نحوه استفاده از این DataLoader ها را نشان می‌دهد. لینک فایل داده‌ها و کدها:

Google Drive

۳.(۵ نمونه تصاویر Digit-five

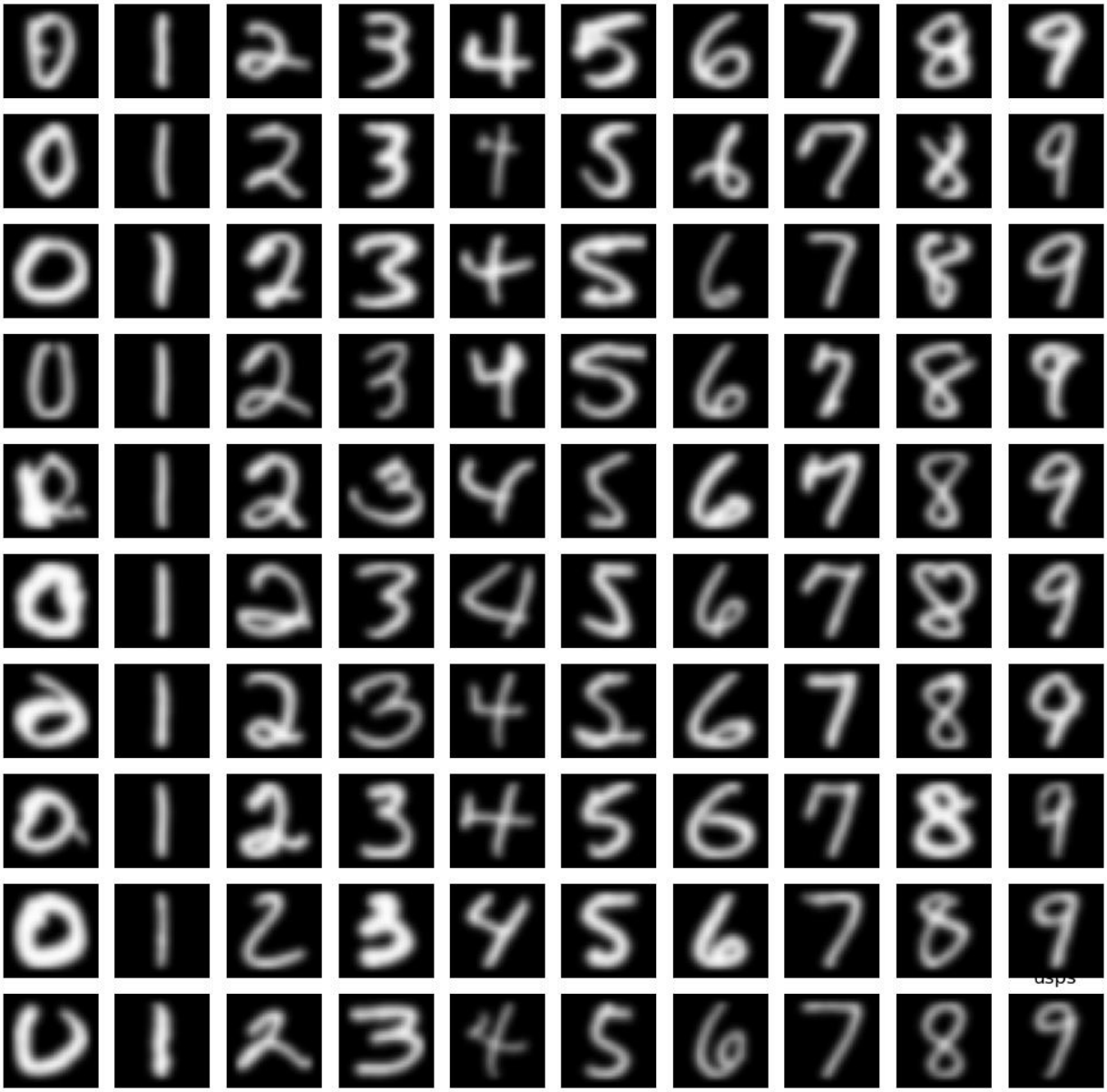


شکل ۶: نمونه تصاویر دامین MNIST

شکل ۷: نمونه تصاویر دامین MNIST-M

شکل ۸: نمونه تصاویر دامین SVHN

شکل ۹: نمونه تصاویر دامین SYN



شکل ۱۰: نمونه تصاویر دامین USPS