

**دانشگاه تهران**

**دانشکده‌ علوم و فنون نوین**

**تمرین پنج**

|  |  |
| --- | --- |
| فاطمه چیت ساز | نام و نام خانوادگی |
| 830402092 | شماره‌ دانشجویی |
| 17 دی 1402 | تاریخ ارسال گزارش |

Contents

[تمرین یک: تخمین بیشینه شباهت 1](#_Toc155960494)

[تمرین دو : روش گشتاور ها 3](#_Toc155960495)

[تمرین سه : روش گشتاور ها 3](#_Toc155960496)

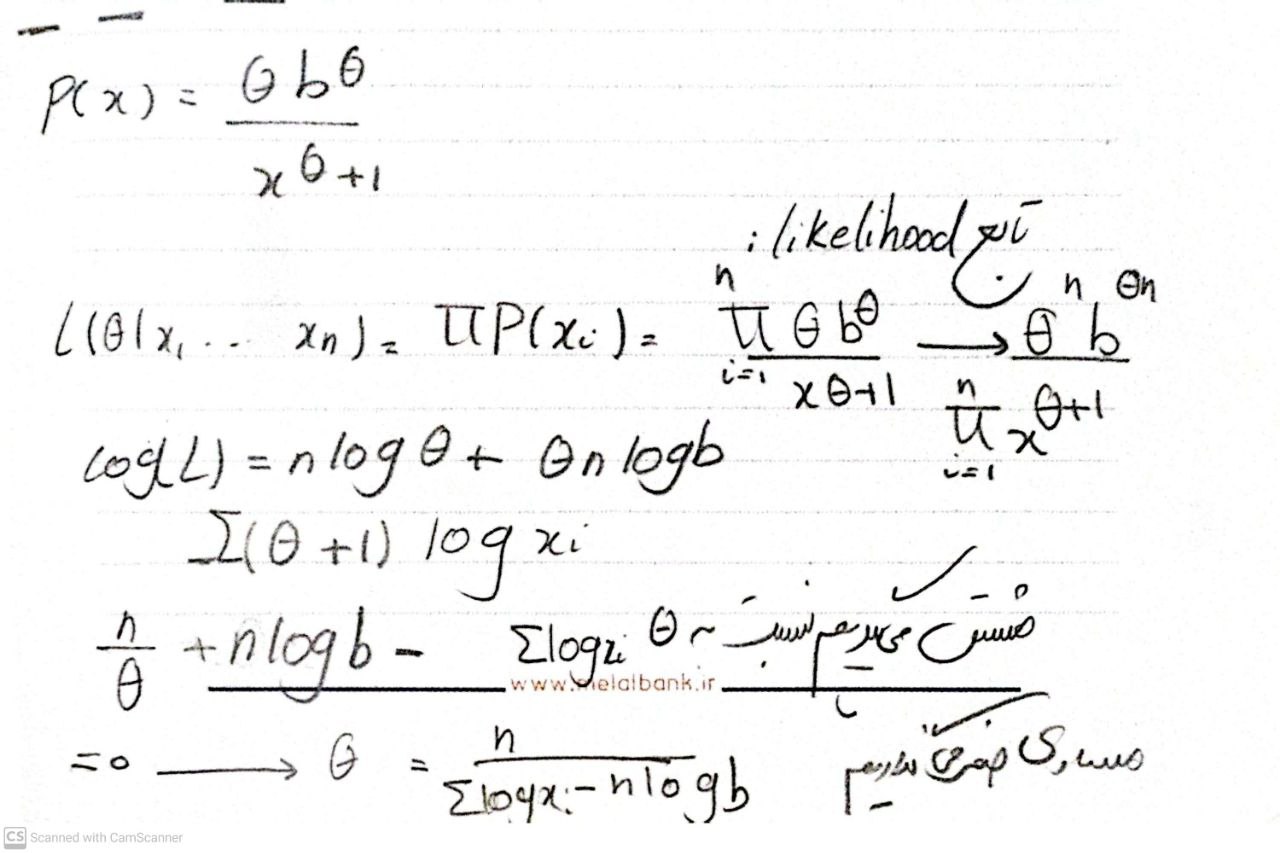
[تمرین چهار: مدل تشخیص گفتار با استفاده از پرسپترون چند لایه 5](#_Toc155960497)

[تمرین پنج: خوشه بندی 17](#_Toc155960498)

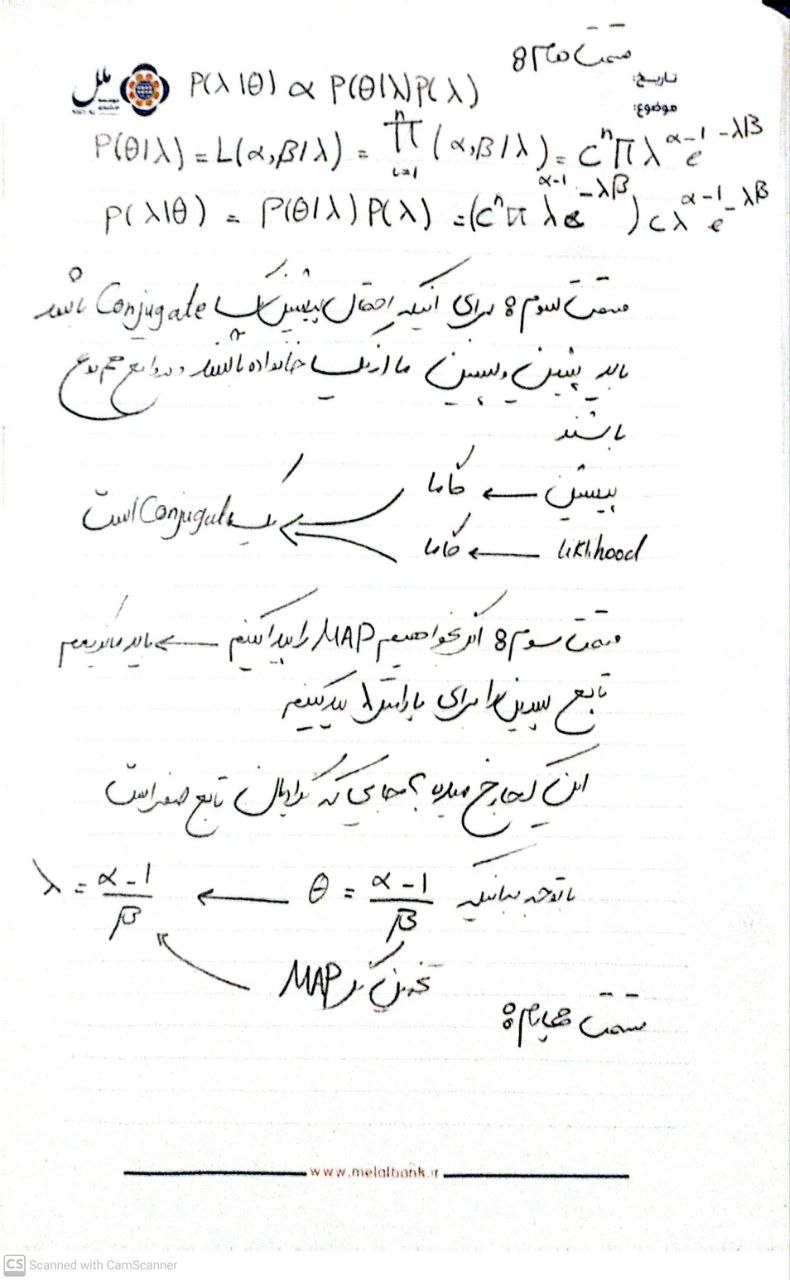
[تمرین شش: k means 19](#_Toc155960499)

# تمرین یک: تخمین بیشینه شباهت

قسمت اول:



قسمت دو وسه :



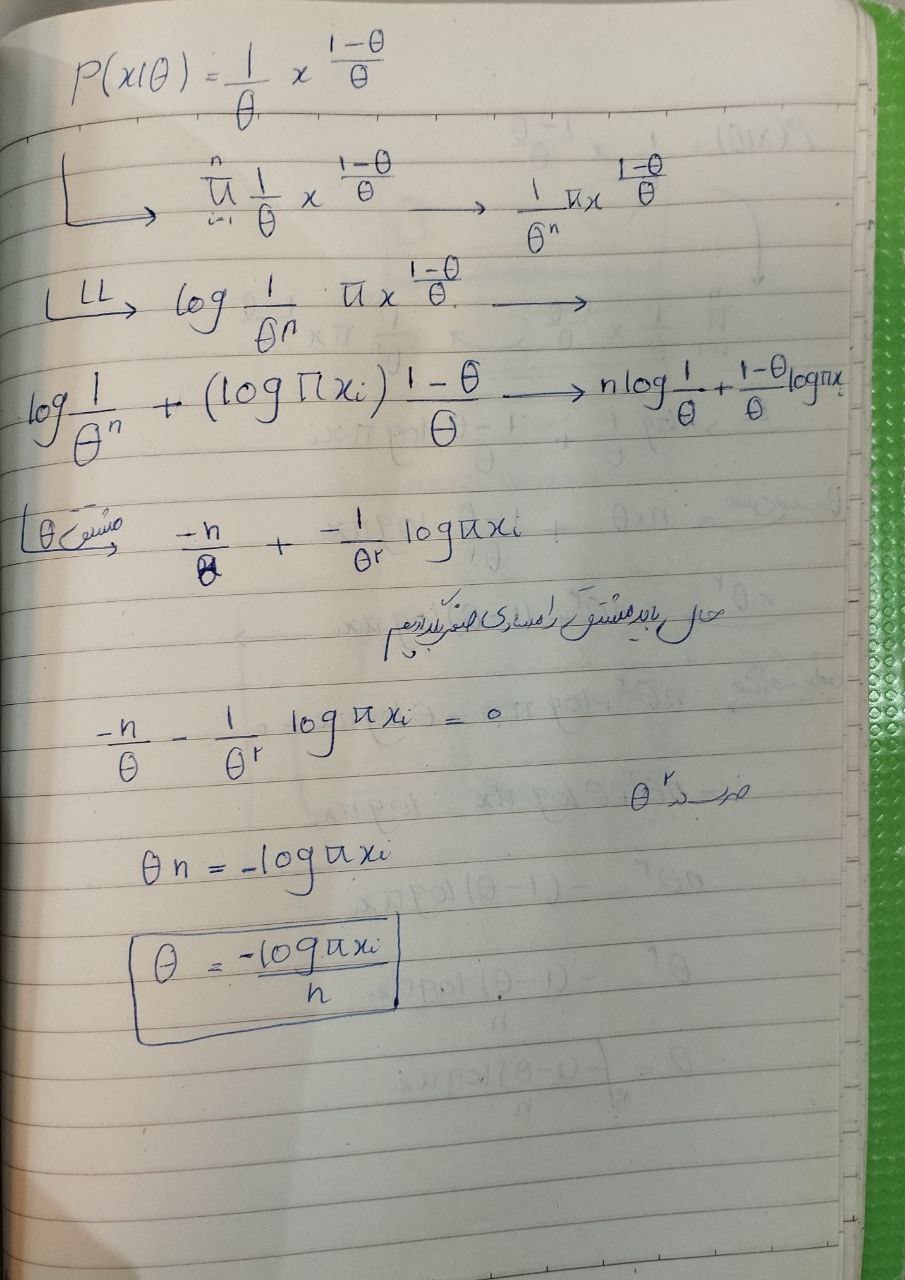
قسمت چهار:

اگر پارامترهای توزیع پیشین به بی‌نهایت بزرگ شوند (به سوی بی‌نهایت)، تخمین‌گر MAP به تخمین‌گر ML میل می‌کند. دلیل این امر این است که تأثیر توزیع پیشین با تعداد نمونه‌ها متناسب است و با افزایش تعداد نمونه‌ها، تأثیر توزیع پیشین اهمیت کمتری پیدا می‌کند و تخمین‌گر MAP به سمت تخمین‌گر ML همگرا می‌شود.

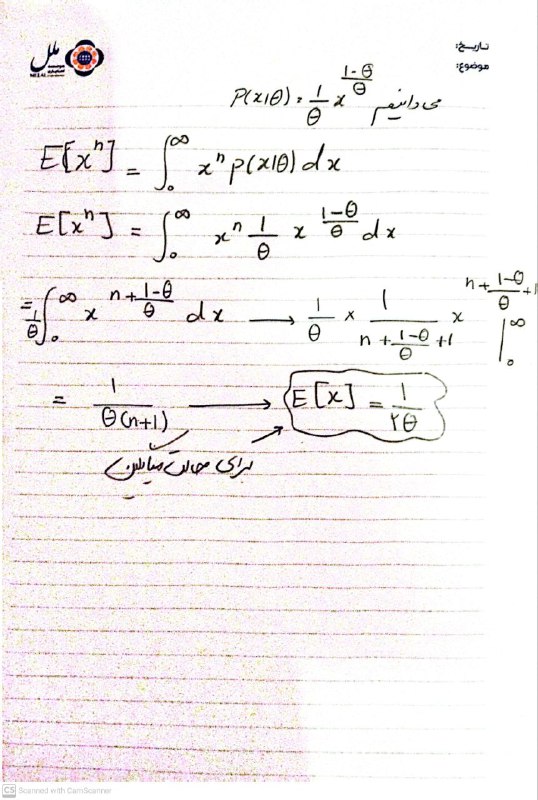
# تمرین دو : روش گشتاور ها

# تمرین سه : روش گشتاور ها

قسمت اول :



قسمت دوم :



# تمرین چهار: مدل تشخیص گفتار با استفاده از پرسپترون چند لایه

هدف چیست؟

در این کد، یک مدل شبکه عصبی برای تشخیص گفتار سطح فریم (Frame-Level Speech Recognition) آموزش داده می‌شود. ورودی‌های این مدل اطلاعات MFCC (Mel-Frequency Cepstral Coefficients) برای هر فریم می‌باشند و هدف آن تشخیص فونم‌های حاضر در هر فریم است.

مراحل اجرا:

1. پیش‌پردازش داده :داده‌های آموزش و اعتبارسنجی از دو دسته مجزا از دایرکتوری‌های مختلف برای MFCC و ترنسکریپت‌ها بارگیری می‌شوند. سپس یک مرحله پیش‌پردازش شامل نرمالیزیشن سپترال بر روی داده‌های MFCC صورت می‌گیرد.

2. ساختار مدل :یک مدل MLP ساده با دو لایه خطی و یک لایه ReLU برای تصمیم‌گیری از معماری استفاده می‌کند.

3. تعریف توابع آموزش و اعتبارسنجی:\*\* توابع جداگانه برای مراحل آموزش و اعتبارسنجی تعریف شده‌اند. این توابع شامل مراحل انتقال داده به دستگاه GPU (اگر ممکن باشد)، محاسبه خطا، انجام مراحل بهینه‌سازی، و ثبت نتایج هر ایپوک می‌شوند.

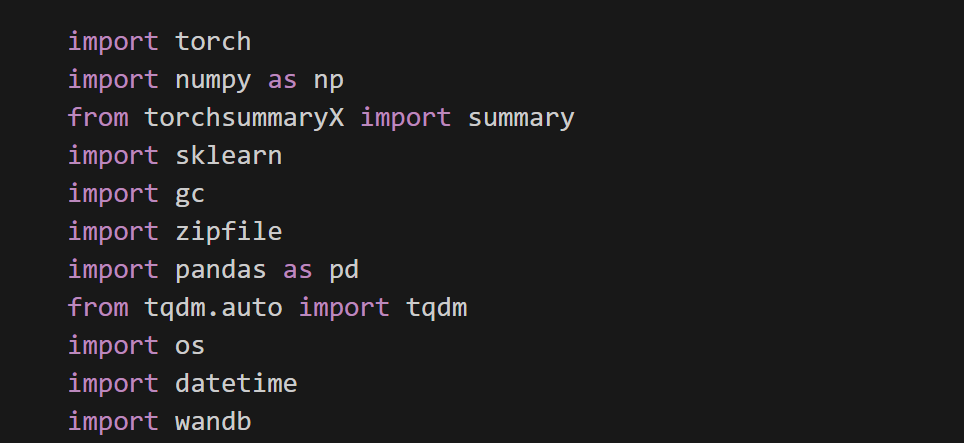
4. آموزش مدل :مدل با تعداد ایپوک‌های مشخص شده آموزش داده می‌شود و نتایج مربوط به هر ایپوک چاپ می‌شوند.

5. گزارش و نتیجه‌گیری :گزارش کاملی از نتایج آموزش و اعتبارسنجی ارائه شده و به توضیحات و جزئیات هر بخش پرداخته می‌شود.

هدف پروژه :هدف اصلی این پروژه بهینه‌سازی یک مدل برای تشخیص گفتار بر اساس اطلاعات MFCC و ترنسکریپت‌هاست تا مدل بتواند بهترین عملکرد را در تعیین فونم‌های هر فریم از ورودی ارائه دهد.

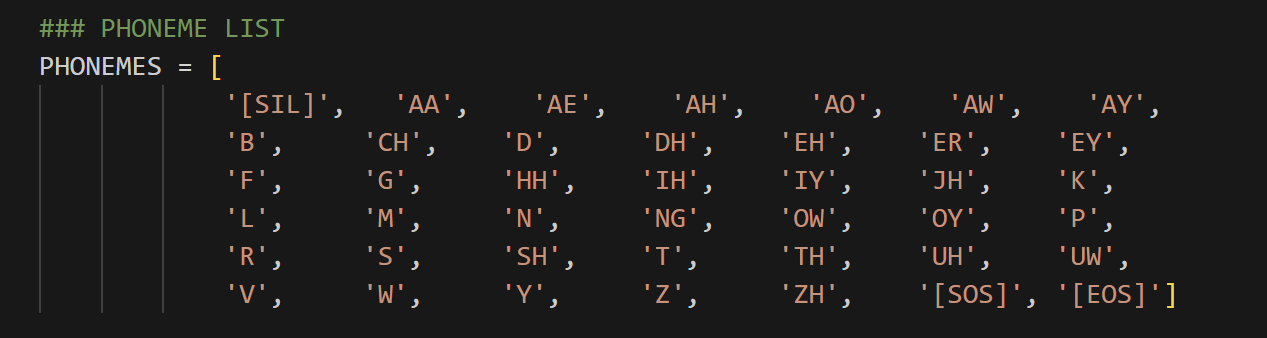
خب حال میتونیم به توضیح کد بپردازیم :

ابتدا کتابخانه‌های لازم نصب و وارد می‌شوند.

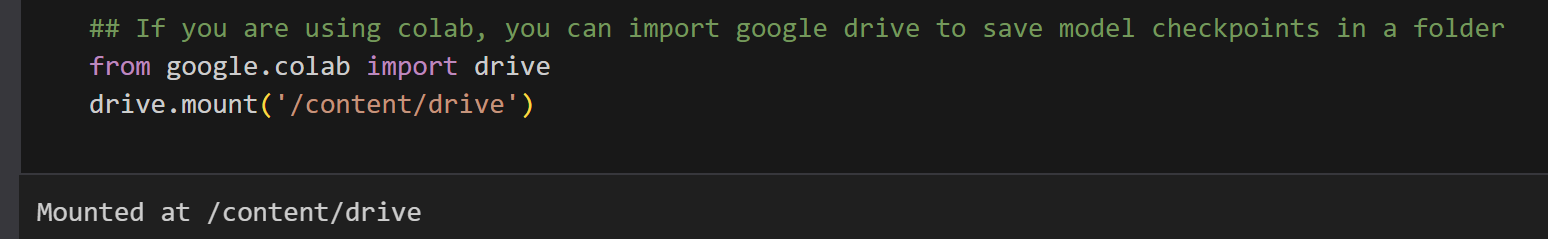


البته لازم به ذکره چندتا از کتابخونه ها را با pip نصب میکنیم

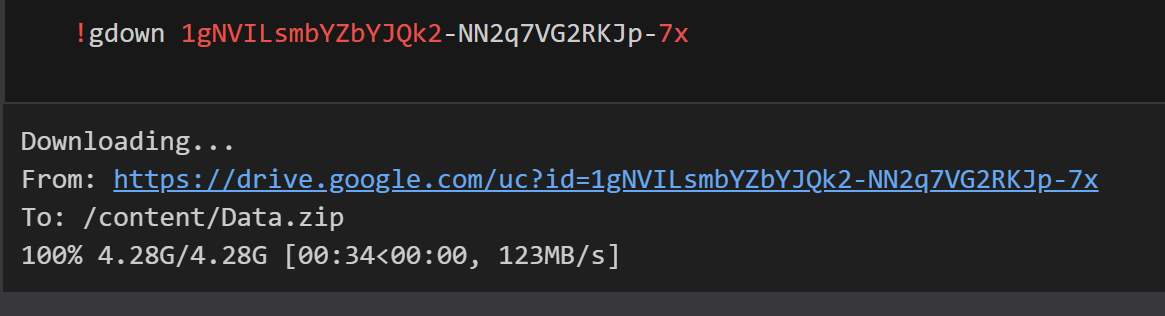
یک لیست از فونم‌ها تعریف می‌شود



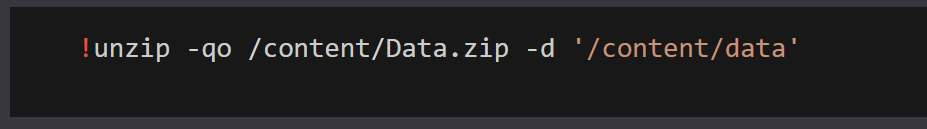
به گوگل درایو وصل میشیم برای دیتاست بزرگ عزیززز



دیتاست بزرگوار رو دانلود میکنیم



میذاریمش تو پوشه content/data و ان زیپ میفرماییم



در این بخش کلاس مجموعه داده برای داده‌های گفتار تعریف می‌شود

خب بیاید ببینیم این لودرمون چه ها میکنه

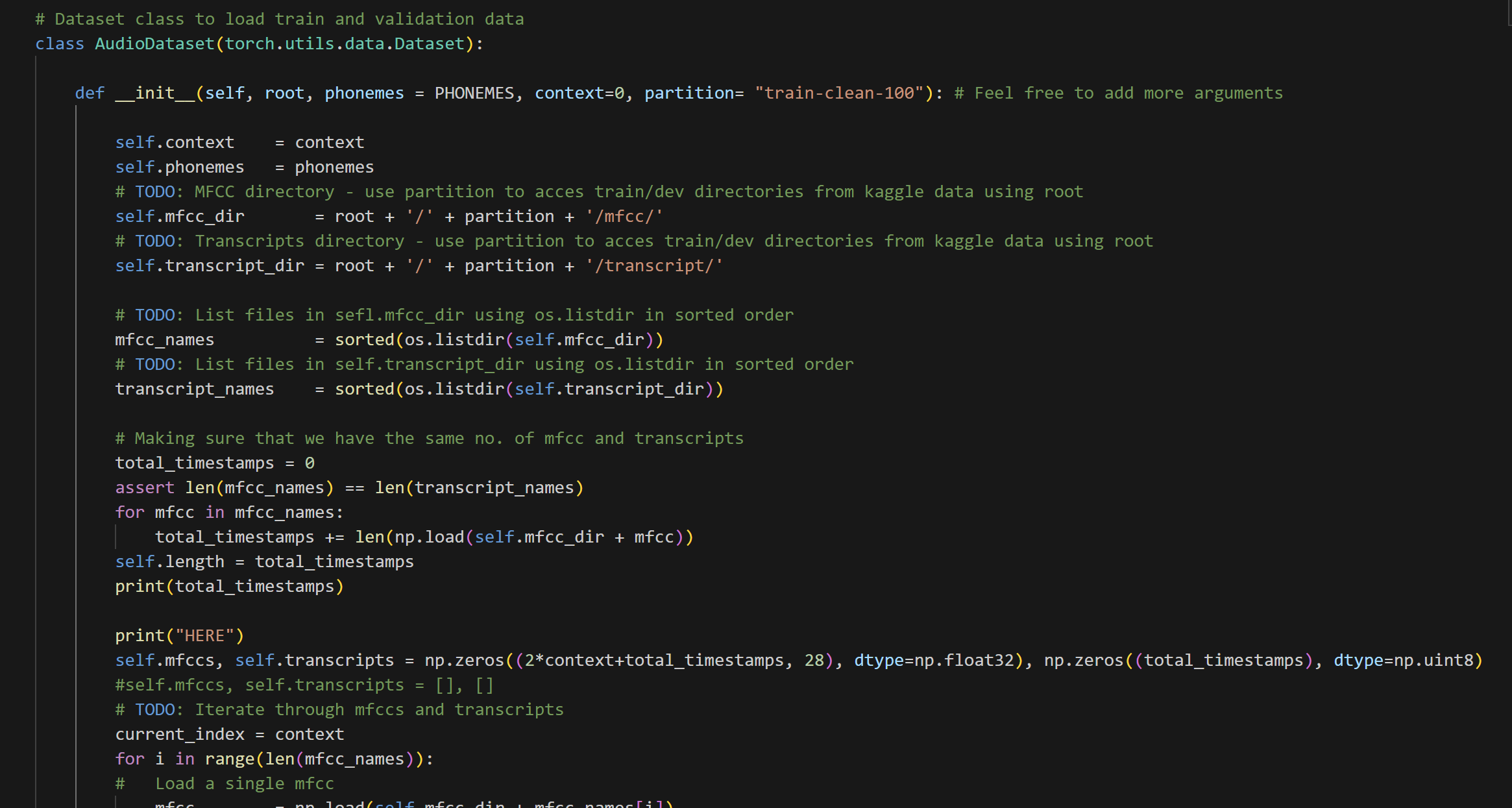
مسئولیت اول: تعیین دو دایرکتوری برای داده‌ها. یکی برای MFCC و دیگری برای ترنسکریپت. این دو دایرکتوری با استفاده از root و partition مشخص میشن. مثل اینه که می‌گیم "داده‌ها اینجا هستن!"

مسئولیت دوم: لیستی از نام فایل‌ها رو به ترتیب میگیریم. اول از همه، نام فایل‌های MFCC و ترنسکریپت رو مرتب می‌کنیم.

مسئولیت سوم: محاسبه میانگین و انحراف معیار برای هر MFCC. این یه جورایی یه کار تنظیمیه که داده‌ها رو به یه حالت خوب برسونه.

مسئولیت چهارم: تعداد کل فریم‌ها محاسبه میشه و مطمئن میشیم که تعداد فایل‌ها یکسانه.

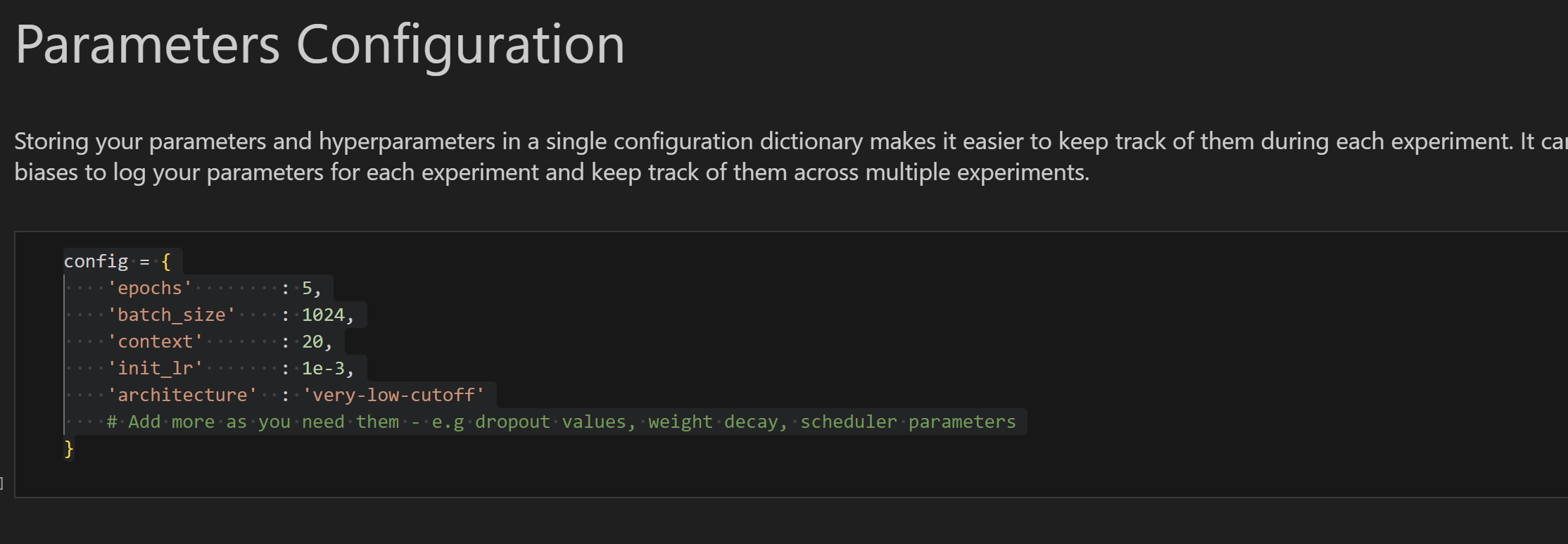
مسئولیت پنجم: یک ماتریس برای MFCC و یک آرایه برای ترنسکریپت ایجاد میشه. اینجا داده‌ها به ماتریس و آرایه مربوطه اضافه می‌شن.

مسئولیت ششم: تابع getitem تعریف می‌کنیم که یک بخش از داده رو به عنوان ورودی برای مدل میاره. اینجا با مراعات فریم‌های اطراف و با توجه به شماره فریم، یک زوج از داده‌های MFCC و ترنسکریپت برمی‌گردونه. 

خب مرحله بعد ست کردن تنظیماته :

این تیکه از کد یک فایل تنظیمات به نام config ایجاد می‌کنه که تمام پارامترها و تنظیمات مربوط به آموزش یک مدل صوتی رو در خودش نگه می‌داره. مثلاً تعداد اپوک‌ها (تعداد دورهای آموزش)، اندازه دسته (تعداد نمونه‌ها در هر مرحله)، محدوده متن (تعداد فریم‌های اطراف هر فریم) و نرخ یادگیری اولیه از جمله پارامترهای مهم هستند.

در واقع مثل اینه یک دفترچه یادداشت داشته باشیم که تمام جزئیات لازم برای هر آزمایش رو در اختیار داره.



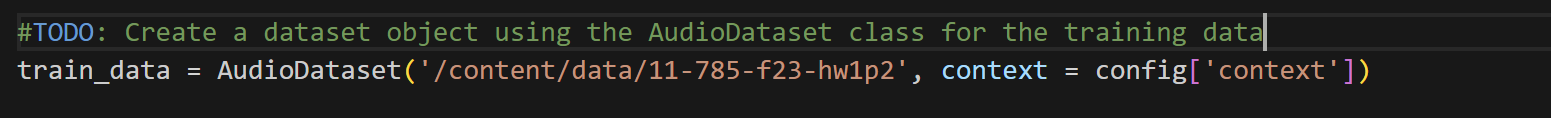
و بالاخره درست کردن دیتاها

1. ساخت آبجکت دیتاست:

این خط یک آبجکت از کلاس AudioDataset با نام train\_data ایجاد می‌کند.

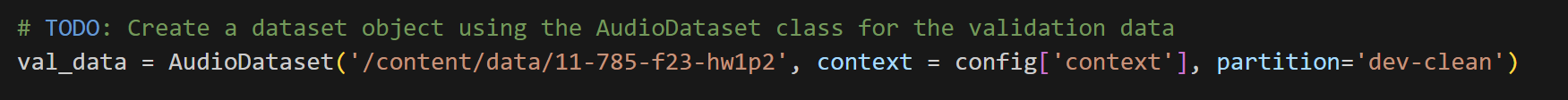
پارامتر اول آدرس دایرکتوری حاوی داده‌های آموزشی است.

پارامتر دوم مقدار context را از دیکشنری config دریافت می‌کند. context مشخص می‌کند که چند فریم قبل و بعد از هر فریم هدف باید برای آموزش مدل در نظر گرفته شود.



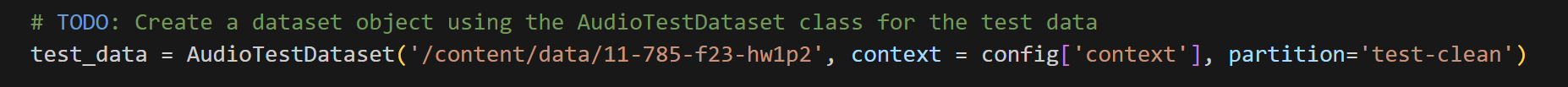
2. ساخت آبجکت دیتاست برای اعتبارسنجی:

مشابه train\_data، اما از پارتیشن dev-clean برای بارگیری داده‌های اعتبارسنجی استفاده می‌کند.

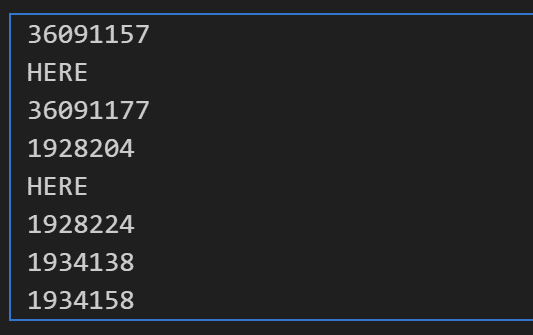


3. ساخت آبجکت دیتاست برای تست:

در اینجا یک شی از کلاس AudioTestDataset برای داده‌های تست ایجاد شده است. همچنین از مسیر دیتاست و مقدار context از فایل تنظیمات config استفاده شده‌اند. این دیتاست مربوط به قسمت تست (partition='test-clean') است.



خروجی :



اینجا دوتا چیز اول برای ترین بعد validation و بعد تست چاپ میشه

current\_index:

این متغیر نشان‌دهنده‌ی اندیس فریم فعلی در آرایه‌ی self.mfccs است

در ابتدای حلقه‌ی for مقدار آن context است

با پیمایش حلقه و اضافه کردن فریم‌های جدید به آرایه، مقدار آن افزایش می‌یابه

در نهایت، مقدار آن برابر با اندیس آخرین فریم به همراه فریم‌های زمینه‌ای هس

total\_timestamps:

این متغیر نشان‌دهنده‌ی تعداد کل فریم‌های صوتی در دیتاست است

با جمع کردن طول هر فایل MFCC به دست میاد

خب حالا می‌خوایم داده‌ها رو دسته‌بندی کنیم تا مدل یادگیری ماشینی‌مون بتونه راحت‌تر باهاشون کار کنه.

این کدها یه چیزی درست می‌کنن به اسم DataLoader که وظیفه‌ش اینه که داده‌های آموزشی، اعتبارسنجی و تست رو برامون دسته‌بندی کنه.

چیکار می‌کنه:

برای هر کدوم از سه تا دیتاست، یه DataLoader می‌سازه:

یکی برای داده‌های آموزشی: train\_loader

یکی برای داده‌های اعتبارسنجی: val\_loader

یکی برای داده‌های تست: test\_loader

به DataLoader می‌گه که:

از کدوم دیتاست استفاده کنه

چند تا کارگر بذاره برای بارگذاری داده‌ها

هر دسته چقدر نمونه داشته باشه

داده‌های آموزشی رو قاطی کنه

بعدش یه سری اطلاعات چاپ می‌کنه:

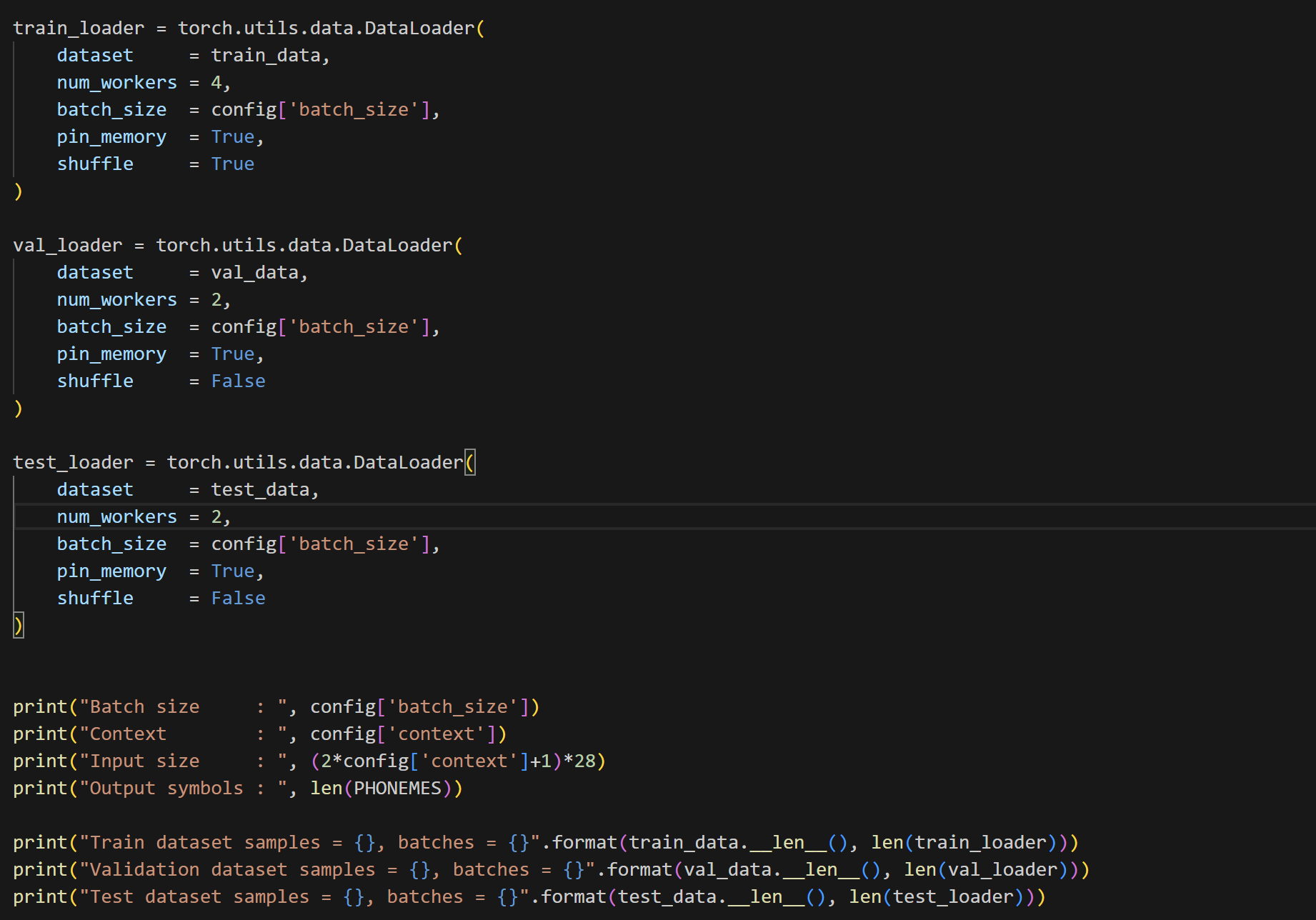
اندازه‌ی دسته‌ها

تعداد فریم‌های قبلی و بعدی که مدل برای پیش‌بینی هر فریم در نظر می‌گیره (مثل اینکه بهش بگی قبل از اینکه پیش‌بینی کنی، به چند تا فریم قبلی و بعدی نگاه کن)

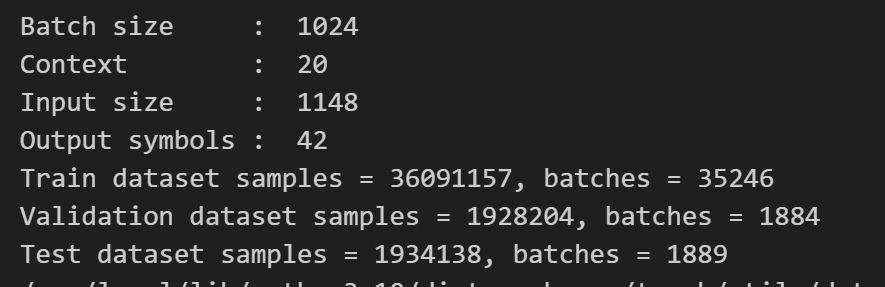
اندازه‌ی ورودی مدل (تعداد کل ویژگی‌هایی که مدل باید یاد بگیره)

تعداد نمادهای خروجی

تعداد نمونه‌ها و دسته‌ها توی هر دیتاست



خروجی :



حالا تست کوچیک انجام می‌دیم تا مطمئن شیم که DataLoaderها درست کار می‌کنه.

کار کد اینه که:

یه دور می‌زنه روی اولین دسته از داده‌های آموزشی:



داده‌های هر دسته رو باز می‌کنه:

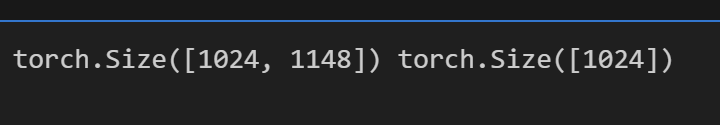


شکل داده‌های صدا و برچسب‌ها رو چاپ می‌کنه:

بعد از اولین دسته دیگه ادامه نمی‌ده:



خروجی :

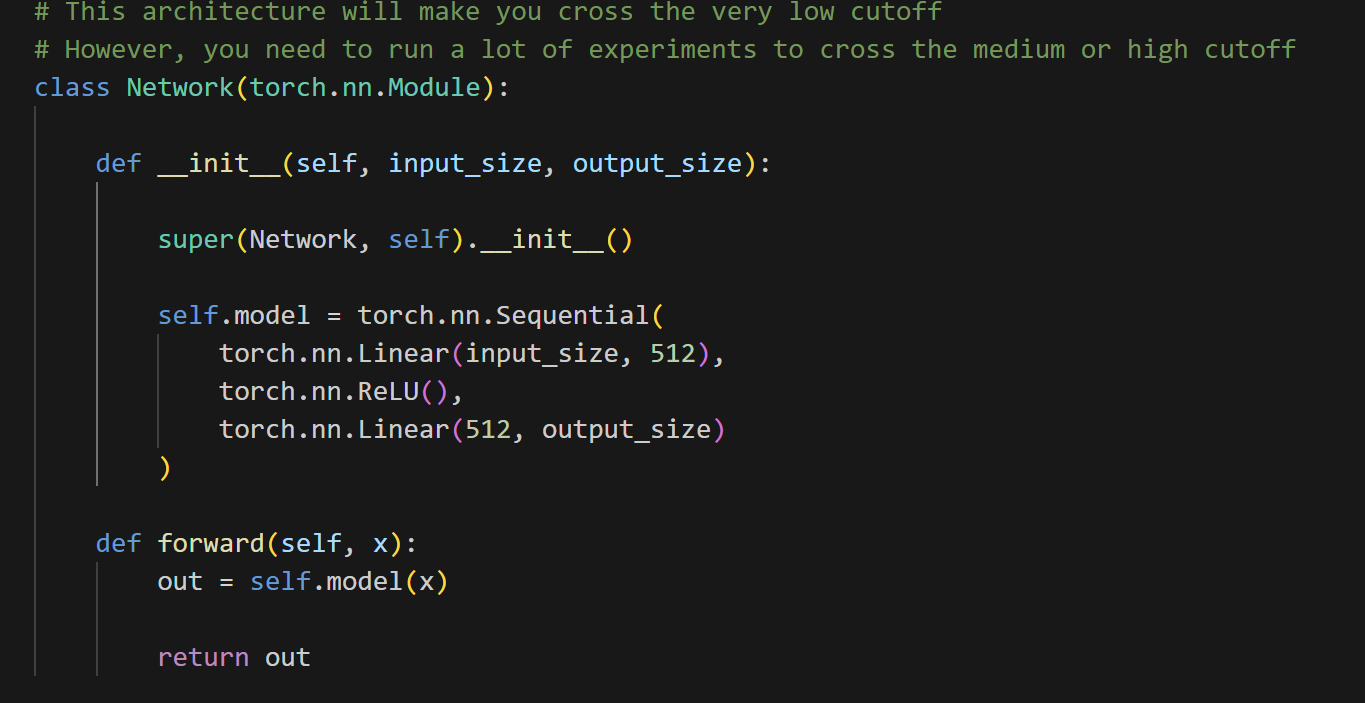


خب حال شبکه عصبی بزرگوار رو تعریف کنیم :

این شبکه یه ورودی می‌گیره (یه سری عدد که ویژگی‌های یه صدا رو نشون می‌دن)

از یه سری لایه‌های خطی و ReLU ردش می‌کنه (مثل اینکه یه سری فیلتر و پردازش روش انجام می‌ده)

یه خروجی تولید می‌کنه



حالا چندتا کار جالب کنیم قبل ترین کردن :

اندازه‌ی ورودی رو مشخص می‌کنن:

INPUT\_SIZE = (2\*config['context'] + 1) \* 28

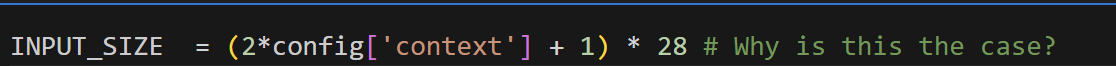
اینجوری حسابش می‌کنن:

هر فریم صوتی 28 تا ویژگی داره (مثلاً 28 تا عدد).

می‌خوان برای پیش‌بینی هر فریم، علاوه بر خودش، config['context'] فریم قبلی و config['context'] فریم بعدی رو هم در نظر بگیرن.

پس در کل (2\*config['context'] + 1) فریم رو با هم ترکیب می‌کنن.

به همین خاطر، اندازه‌ی ورودی میشه حاصل‌ضرب تعداد فریم‌ها (2\*config['context'] + 1) در تعداد ویژگی‌های هر فریم (28).



مدل رو می‌سازن و می‌فرستن روی کارت گرافیک (اگه موجود باشه):

مدل رو با اندازه‌ی ورودی مشخص‌شده و تعداد نمادهای خروجی می‌سازن.

بعدش هم می‌فرستنش روی کارت گرافیک تا سریع‌تر کار کنه.



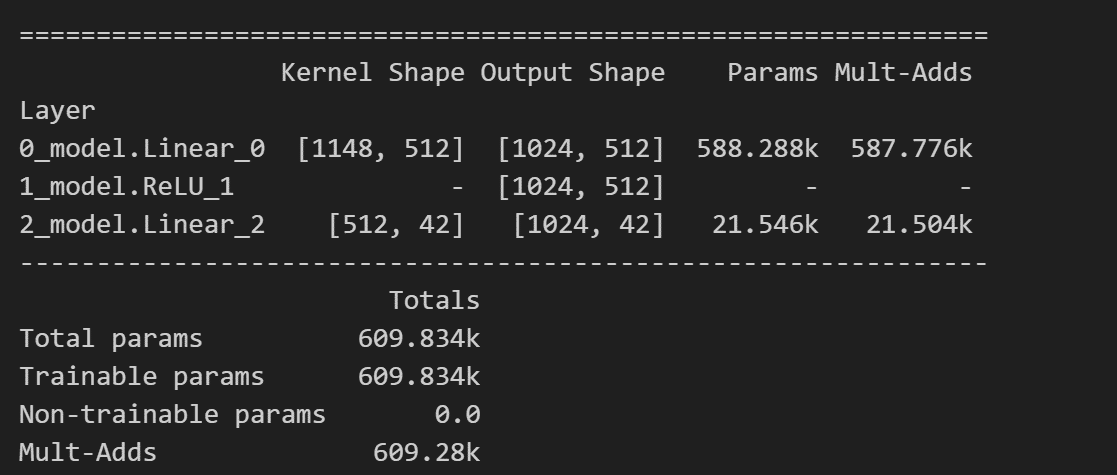
خلاصه‌ای از مدل رو نشون می‌دن:



این تابع ساختار مدل و تعداد پارامترهاش رو نشون می‌ده.

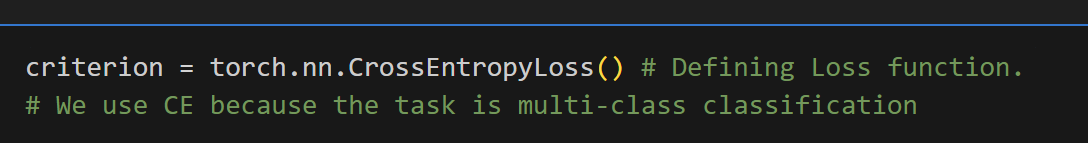
یه نمونه از داده‌ها رو هم بهش می‌دن تا بتونه محاسباتش رو انجام بده.

خروجی:



تعریف تابع خطا:

که صد البته میزان اختلاف بین پیش‌بینی مدل و جواب درست رو اندازه می‌گیره.

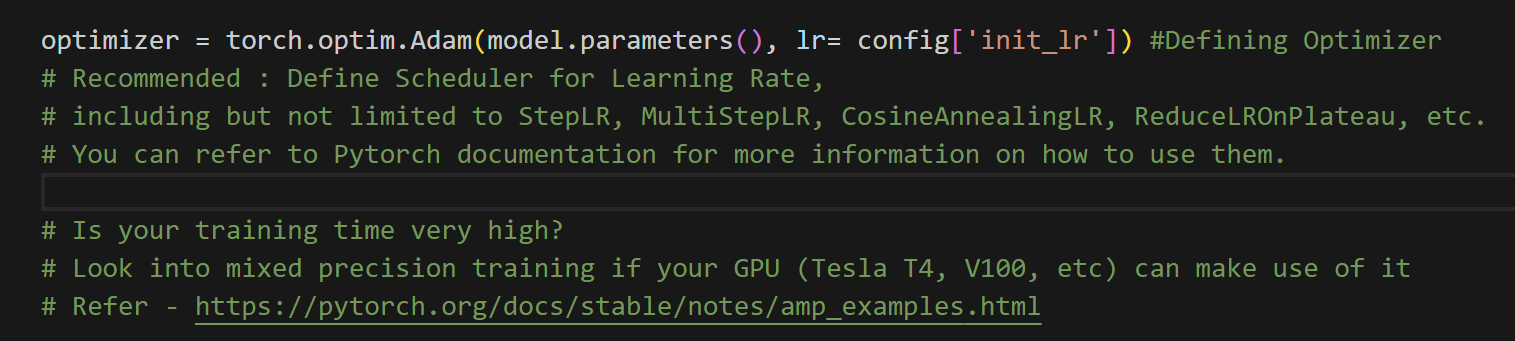


تعریف بهینه‌ساز:

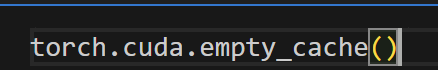
بهینه‌ساز به مدل کمک می‌کنه که با توجه به تابع خطا، پارامتراش رو تنظیم کنه و یاد بگیره.

اینجا از بهینه‌ساز Adam استفاده می‌کنن که معمولاً عملکرد خوبی داره.

lr هم نرخ یادگیریه که سرعت یادگیری مدل رو کنترل می‌کنه.



چندتا بهینه سازی و مراقبت



حافظه‌ی کش کارت گرافیک رو خالی می‌کنه.

این کار باعث می‌شه که فضا برای محاسبات جدید باز بشه و عملکرد کارت گرافیک بهتر بشه.



زباله‌های حافظه‌ی اصلی رو جمع‌آوری می‌کنه.

این کار باعث می‌شه که حافظه‌ی اصلی خالی بشه و از کمبود حافظه جلوگیری بشه.

حالا میریم سراغ ترین و این داستانامون :

تابع train:

مدل در حالت آموزش قرار می‌گیرد.

برای هر دسته از داده‌های آموزشی:

گرادیان‌ها صفر می‌شوند.

داده‌ها به کارت گرافیک منتقل می‌شوند (در صورت وجود).

پیش‌بینی انجام می‌شود.

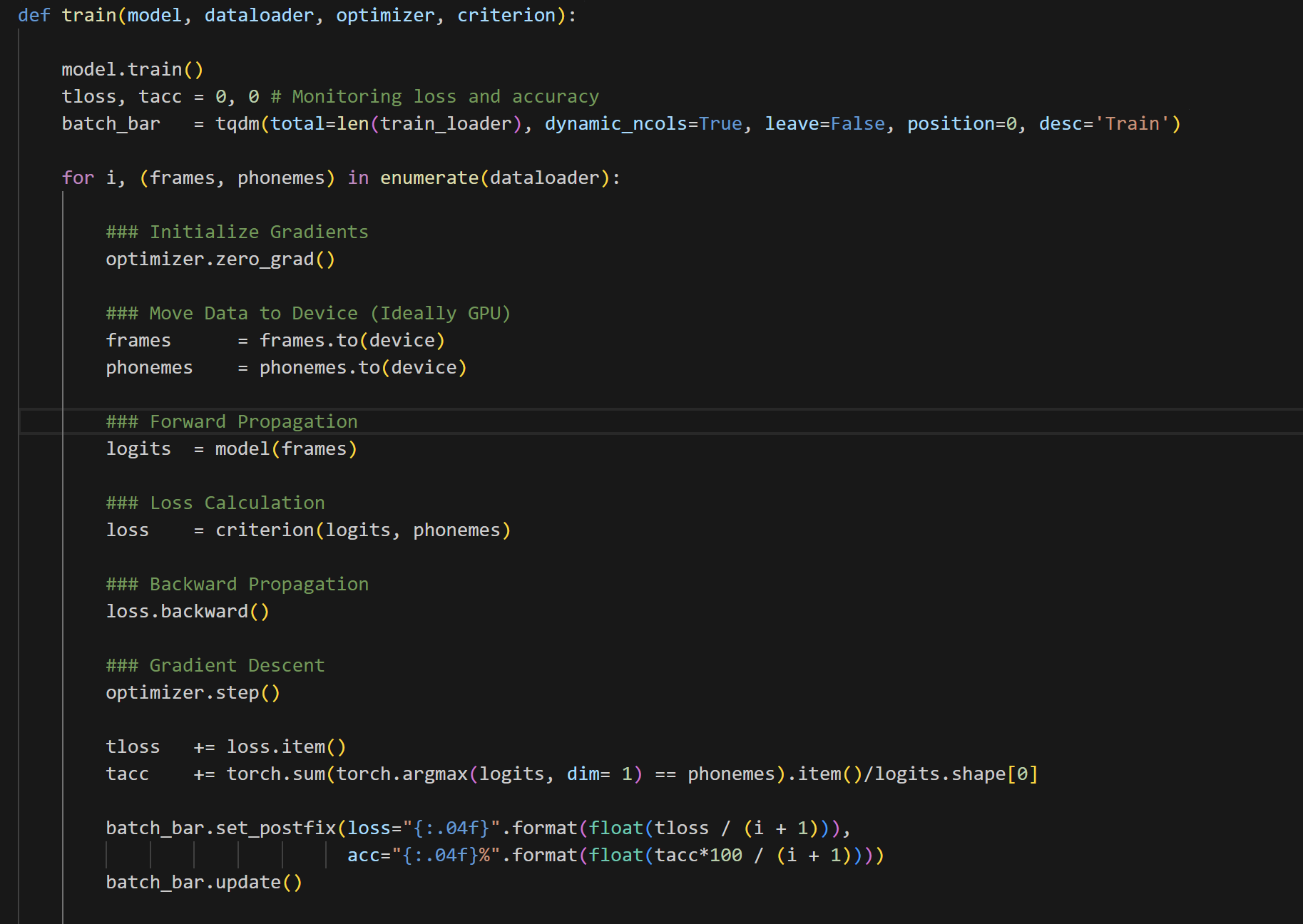
خطا محاسبه می‌شود.

گرادیان‌ها محاسبه می‌شوند.

پارامترهای مدل به‌روزرسانی می‌شوند.

خطا و دقت محاسبه و نمایش داده می‌شوند.

حافظه آزاد می‌شود.



تابع eval:

مدل در حالت ارزیابی قرار می‌گیرد.

برای هر دسته از داده‌های اعتبارسنجی:

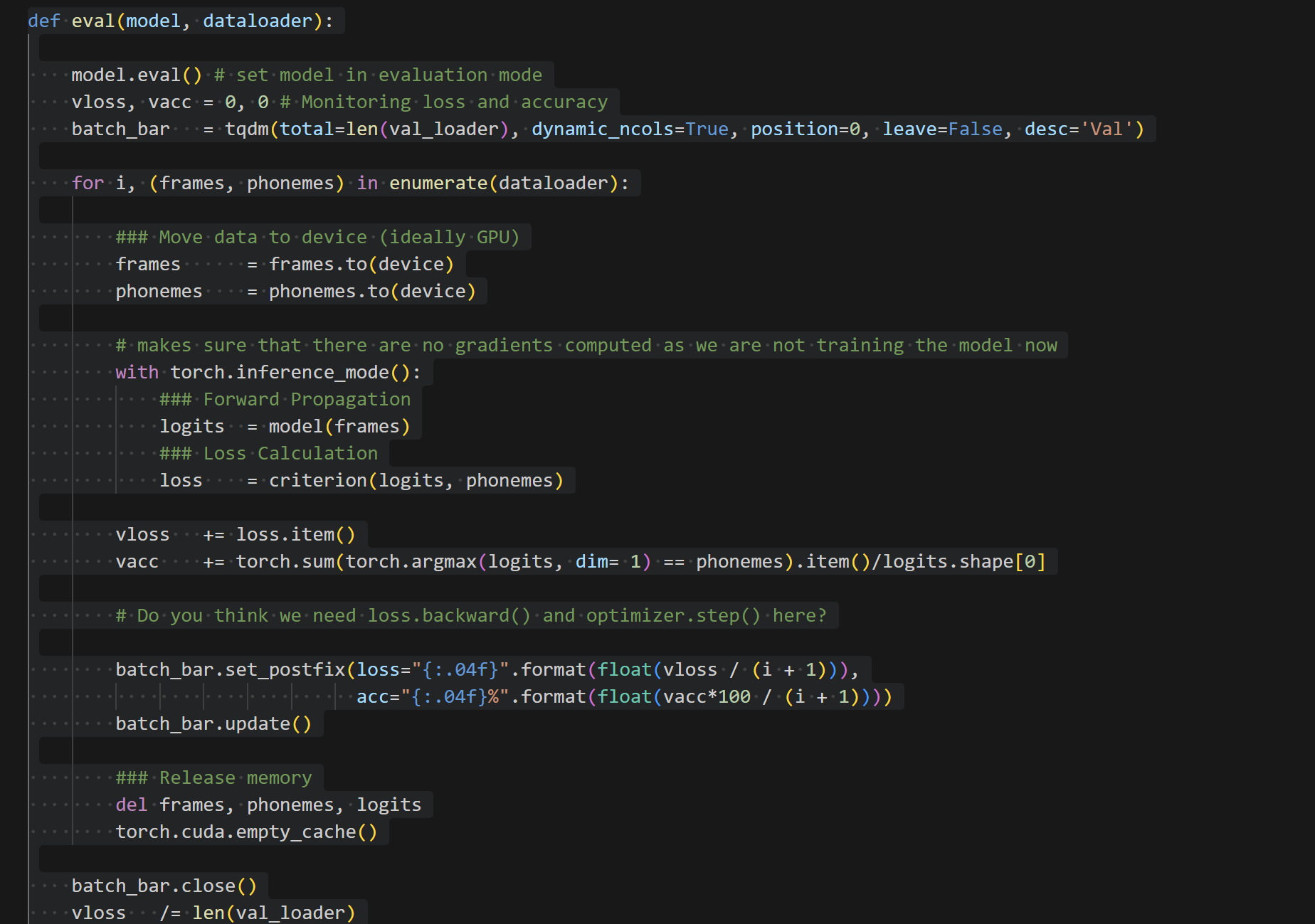
داده‌ها به کارت گرافیک منتقل می‌شوند.

پیش‌بینی انجام می‌شود (بدون محاسبه گرادیان).

خطا محاسبه می‌شود.

خطا و دقت محاسبه و نمایش داده می‌شوند.

حافظه آزاد می‌شود.



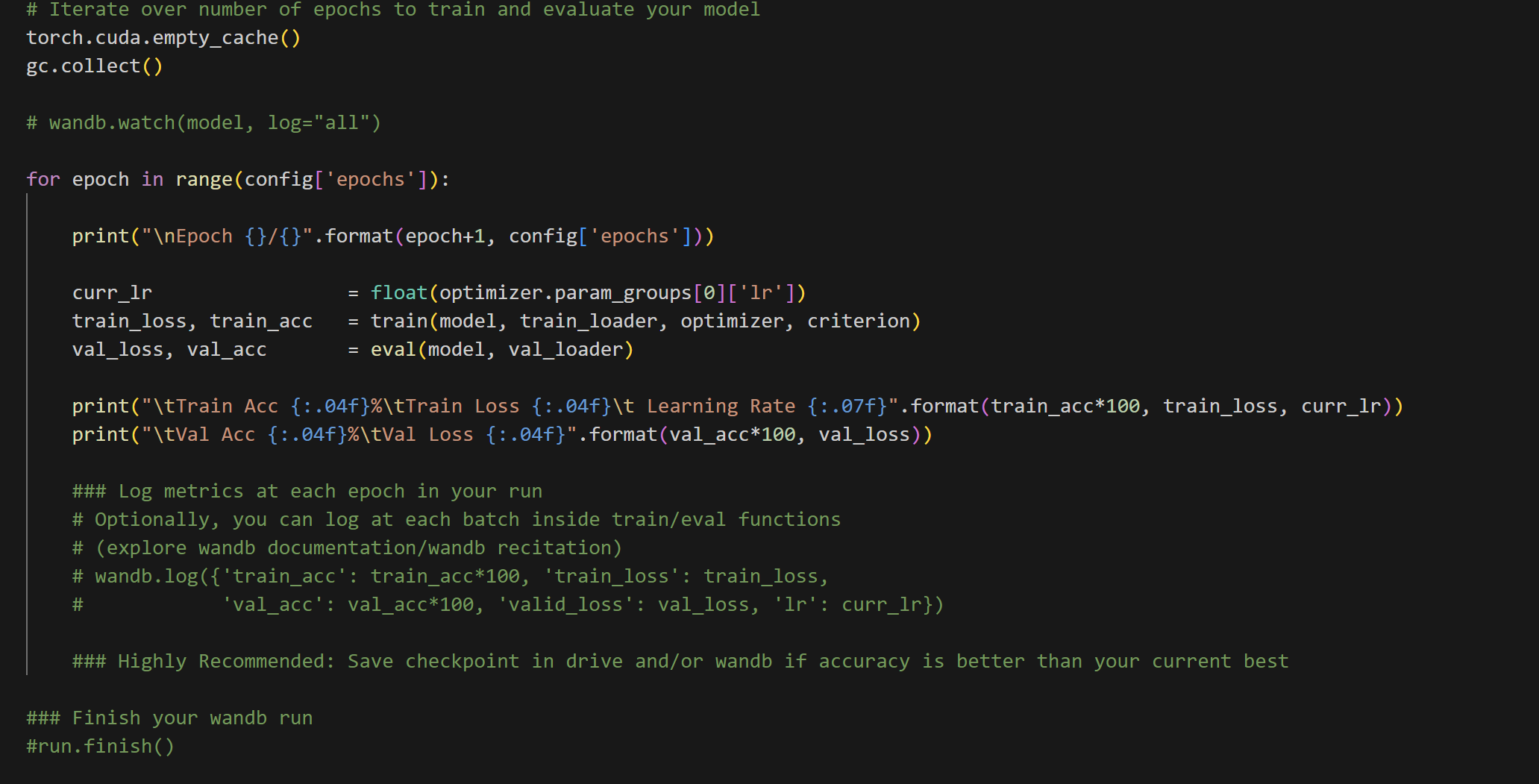
حلقه‌ی اصلی آموزش:

برای تعداد مشخص‌شده‌ی ایپاکا:

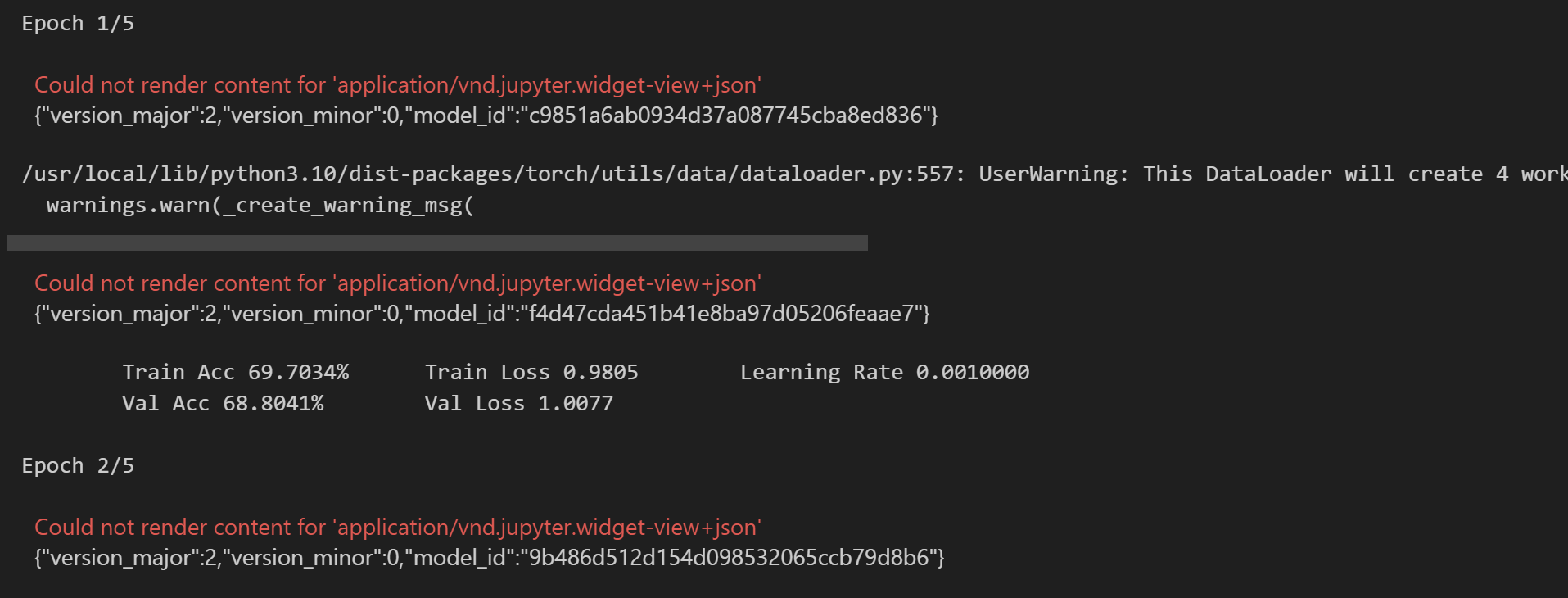
نرخ یادگیری فعلی نمایش داده می‌شود.

مدل روی داده‌های آموزشی آموزش داده می‌شود.

مدل روی داده‌های اعتبارسنجی ارزیابی می‌شود.



خروجی :



# تمرین پنج: خوشه بندی

قسمت اول :

افزایش ϵ :

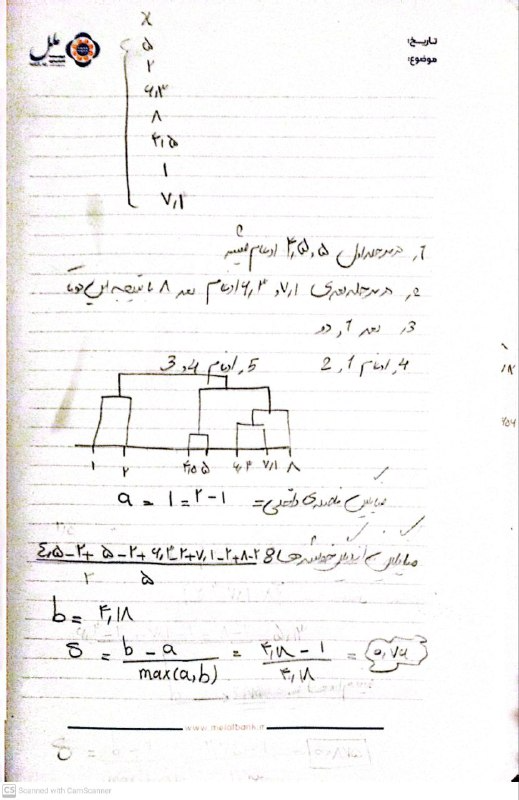
با افزایش اندازه شعاع همسایگی، نقاط بیشتری در همسایگی یک نقطه قرار می‌گیرند. این می‌تواند منجر به افزایش تعداد نقاط در یک خوشه شود و در نتیجه کاهش تعداد خوشه‌ها. همچنین، با افزایش

ϵ، تعداد نقاطی که به عنوان نویز شناخته می‌شوند کاهش می‌یابد، زیرا نقاط بیشتری قادر به رسیدن به حداقل تعداد نقاط مورد نیاز (MinPts) در شعاع ϵ خود هستند.

کاهش ϵ: با کاهش اندازه شعاع همسایگی، کمتر نقطه در همسایگی یک نقطه قرار می‌گیرد. این می‌تواند منجر به کاهش تعداد نقاط در یک خوشه شود و در نتیجه افزایش تعداد خوشه‌ها. همچنین، با کاهش

ϵ، تعداد نقاطی که به عنوان نویز شناسایی می‌شوند افزایش می‌یابد، زیرا کمتر نقطه قادر به رسیدن به حداقل تعداد نقاط مورد نیاز (MinPts) در شعاع ϵ خود هستند.

قسمت دوم :



# تمرین شش: k means

قسمت یک :

اثبات همگرایی:

برای اثبات همگرایی الگوریتم K-means، باید نشان دهیم که تابع هزینه الگوریتم در هر تکرار کاهش می‌یابد. تابع هزینه الگوریتم K-means مجموع مربعات فاصله هر نقطه داده از مرکز خوشه آن است.

مرحله 1:

در مرحله انتساب، هر نقطه داده به نزدیک‌ترین مرکز خوشه اختصاص داده می‌شود. این امر باعث می‌شود که تابع هزینه کاهش یابد، زیرا هر نقطه داده به خوشه‌ای اختصاص داده می‌شود که مرکز آن به آن نقطه نزدیک‌تر است.

مرحله 2:

در مرحله به‌روزرسانی، مرکز هر خوشه به عنوان میانگین نقاط داده‌ای که به آن خوشه اختصاص داده شده‌اند، محاسبه می‌شود. این امر نیز باعث می‌شود که تابع هزینه کاهش یابد، زیرا مرکز جدید خوشه به طور متوسط ​​به تمام نقاط داده‌ای که به آن خوشه اختصاص داده شده‌اند، نزدیک‌تر است.

نتیجه:

از آنجایی که تابع هزینه در هر تکرار الگوریتم K-means کاهش می‌یابد، و تابع هزینه یک تابع کراندار است، الگوریتم K-means باید همگرا شود.

قسمت دو :

به دلیل ماهیت ناحیه‌های متمرکز و ناحیه‌های با چگالی کمتر، احتمالاً در نهایت مراکز خوشه‌ها به صورت یکنواخت بین دو ناحیه توزیع نمی‌شوند. اگر دو ناحیه جدا از یکدیگر هستند و داده‌ها در هر ناحیه به صورت تقریباً یکنواخت توزیع شده‌اند، مراکز خوشه‌ها نیز به یکنواختی در امتداد هر ناحیه توزیع خواهند شد

اگر k کمتر از تعداد واقعی خوشه ها باشد، مراکز خوشه ها به طور مساوی بین دو ناحیه توزیع نمی شوند. به احتمال زیاد مراکز بیشتری در ناحیه متمرکز با چگالی داده بیشتر وجود خواهد داشت.

اگر k بیشتر از تعداد واقعی خوشه ها باشد، ممکن است مراکز خوشه ها در هر دو ناحیه به طور مساوی توزیع شوند.

ی سری چیز مهمه :

معیار فاصله:

انتخاب معیار فاصله مناسب می تواند بر توزیع مراکز خوشه ها تأثیر بگذارد.

پارامترهای دیگر:

پارامترهای دیگر الگوریتم K-means مانند epsilon (معیار توقف) نیز می توانند بر توزیع مراکز خوشه ها تأثیر بگذارند.

در نهایت، بدون اطلاعات بیشتر در مورد مجموعه داده خاصمون و پارامترهای انتخابی الگوریتم K-means، نمی توان با قاطعیت گفت که مراکز خوشه ها در نهایت چگونه توزیع می شوند.

قسمت سه :

این فرمول به این معنی است که نقاط داده‌ای که از مراکز موجود دورتر هستند، احتمال بیشتری برای انتخاب شدن به عنوان مرکز بعدی دارند.

این روش مشابه الگوریتم K-means++ است که هدف آن ارائه یک نقطه شروع بهتر برای K-means است تا منجر به خوشه‌بندی بهینه‌تری شود.

هنگام به‌روزرسانی مراکز (در مراحل بعدی K-means)، این روش دوباره مرکز جدید را بر اساس همان فرمول احتمال انتخاب می‌کند. این یک تفاوت با K-means استاندارد است که در آن مرکز جدید معمولاً میانگین نقاط داده در خوشه انتخاب می‌شود. این نوع الگوریتم به نظر می‌رسد هنگام به‌روزرسانی مراکز، وزن بیشتری به نقاط داده‌ای می‌دهد که از مرکز فعلی دورتر هستند.

الگوریتم K-means++ به طور معمول بهبودهایی در سرعت همگرایی به نسبت K-means ساده دارد. این بهبود اصلی به دلیل انتخاب هوشمندانه‌تر مراکز اولیه است.

در K-means++، انتخاب مراکز بر اساس احتمالات صورت می‌گیرد تا از ایجاد خوشه‌های شروع با فاصله بسیار بزرگ یا بسیار کوچک جلوگیری شود. این انتخاب هوشمندانه مراکز ، معمولاً منجر به کاهش تعداد مراکز به روز شده و در نتیجه به سرعت بهبود می‌بخشد.

در K-means ساده، انتخاب تصادفی مراکز اولیه ممکن است باعث ایجاد شروع نامناسبی برای خوشه‌بندی شود، که موجب نیاز به تعداد بیشتری تکرار (iteration) برای همگرایی مطلوب می‌شود.

بنابراین، کلیتاً می‌توان گفت که K-means++ ممکن است به سرعت بهتری در همگرایی نسبت به K-means ساده داشته باشد، به خصوص در مواقعی که داده‌ها حجم بالایی دارند.

قسمت چهار:

عیار فاصله کوسینوسی:

معیار فاصله کوسینوسی شباهت بین دو بردار را بر اساس زاویه بین آنها اندازه گیری می کند.

این معیار به مقیاس داده ها حساس نیست، که می تواند یک مزیت در برخی موارد باشد.

مناسب بودن برای داده های سری زمانی:

معیار فاصله کوسینوسی برای داده های سری زمانی که ترتیب زمانی داده ها مهم است، مناسب نیست.

دلیل این امر این است که معیار فاصله کوسینوسی به ترتیب زمانی داده ها توجهی نمی کند.

معیارهای مناسب تر:

معیار فاصله اقلیدسی: این معیار فاصله بین دو نقطه را به عنوان طول کوتاهترین خط مستقیم بین آنها اندازه گیری می کند.

معیار فاصله دینامیک زمانی (DTW): این معیار فاصله بین دو سری زمانی را با در نظر گرفتن ترتیب زمانی داده ها اندازه گیری می کند.

انتخاب معیار مناسب:

انتخاب معیار مناسب به نوع داده ها و هدف از خوشه بندی بستگی دارد.

برای داده های سری زمانی، معیارهای فاصله اقلیدسی یا DTW مناسب تر از معیار فاصله کوسینوسی هستند.

مثال:

فرض کنید دو سری زمانی داریم که هر کدام 10 نقطه داده دارند.

اگر داده ها به طور تصادفی مرتب شوند، معیار فاصله کوسینوسی ممکن است شباهت زیادی بین دو سری زمانی گزارش دهد.

با این حال، اگر داده ها بر اساس ترتیب زمانی مرتب شوند، معیار فاصله اقلیدسی یا DTW ممکن است شباهت کمتری بین دو سری زمانی گزارش دهد.

نتیجه:

استفاده از معیار فاصله کوسینوسی برای داده های سری زمانی مناسب نیست.

معیارهای فاصله اقلیدسی یا DTW برای داده های سری زمانی مناسب تر هستند.