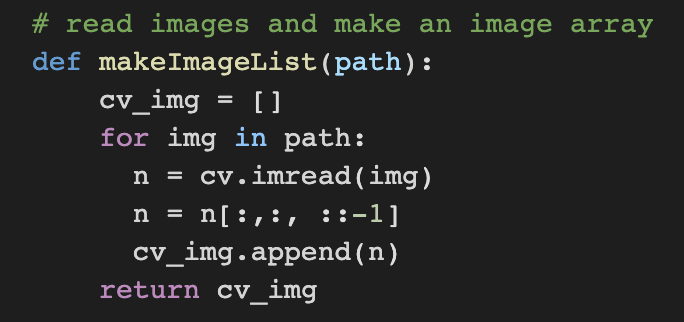
**گزارش پروژه اول هوش محاسباتی**

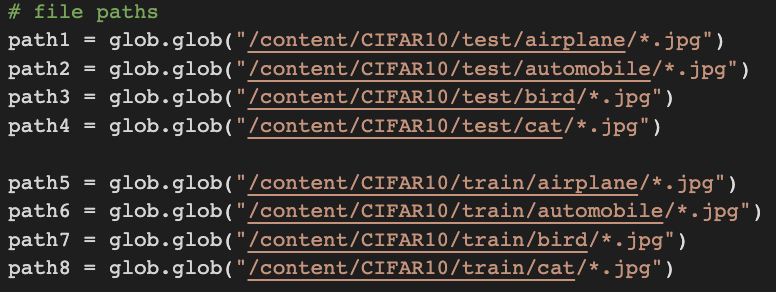
محمدسپهر توکلی کرمانی – ۹۸۳۱۱۱۱

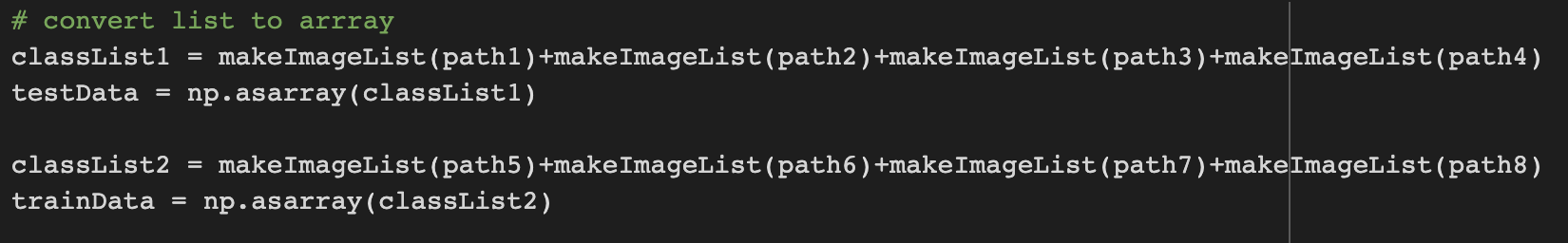
**۱ ) دریافت دیتاست و پیش پردازش**

در اولین قدم از پروژه باید دیتای مورد نیاز را دریافت و آنرا برای ورود به شبکه عصبی آماده کنیم. در گام اول دیتاست  CIFAR10 را دانلود کرده و با تابع زیر آنرا در یک آرایه جای میدهیم.

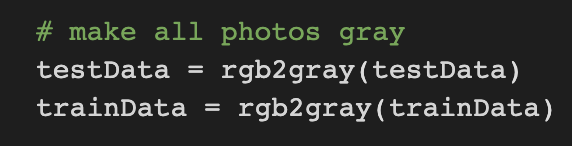


این کار را یکبار برای داده های تست و یکبار برای داده های ترین انجام میدهیم و ۲ آرایه testData و trainData را بوجود می آوریم.

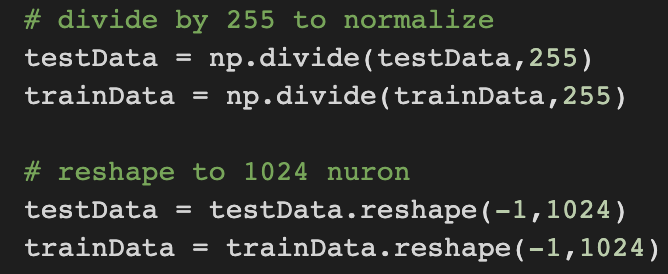




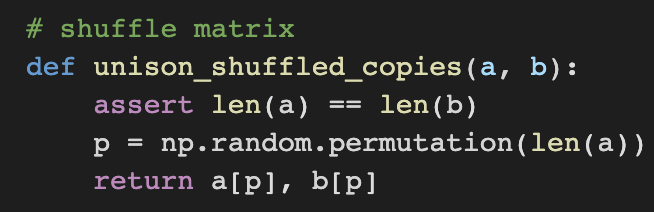
مثلا برای داده های تست ماتریس به ابعاد ۴۰۰۰ در ۳۲ در ۳۲ در ۳ خواهیم داشت. در مرحله بعدی تصاویر را با تابع زیر خاکستری می کنیم.



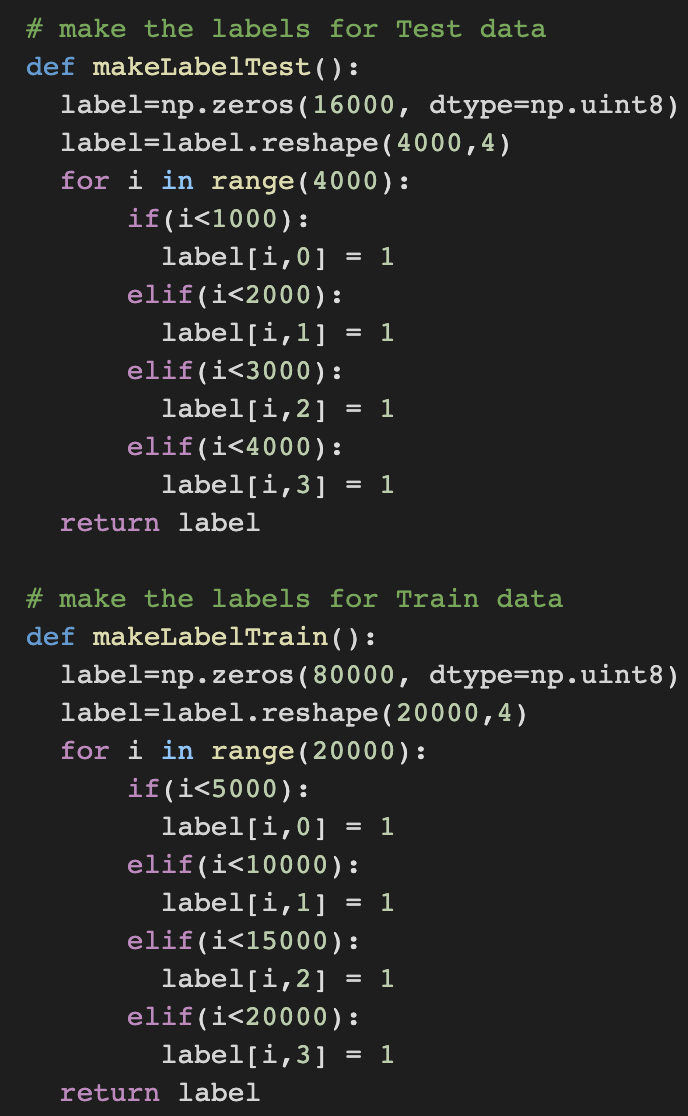
سپس نرمالسازی میکنیم تا مقادیر بین ۰ و۱ باشند و reshape می کنیم تا مطابق ورودی شبکه ۱۰۲۴ نورون باشند.

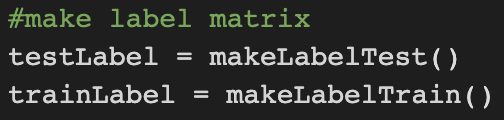


برای شافل کردن دیتا و لیبل ها از تابع زیر استفاده می کنیم:

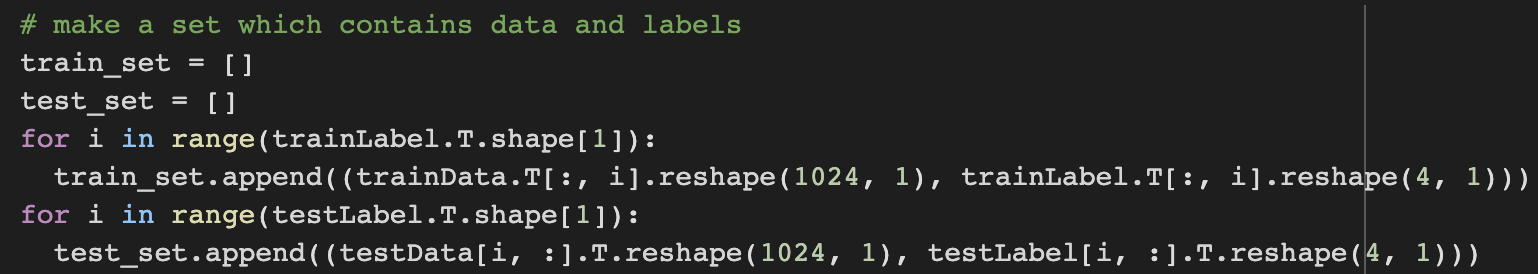


و برای ساخت لیبل ها از ماتریس one-hot به شکل زیر استفاده می کنیم:

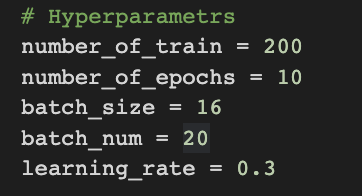




در آخر یک مجموعه برای تست و یک مجموعه برای ترین شامل دیتا و لبیل برای هر قسمت در نظر می گیریم :



هایپرپارامتر ها را نیز به شکل زیر در ابتدا تعریف میکنیم :

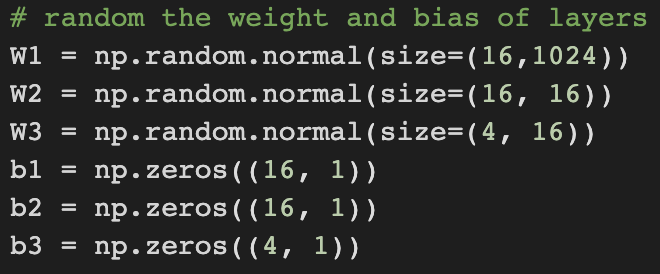


حالا دیتاست ما آماده ورود به شبکه عصبی است.

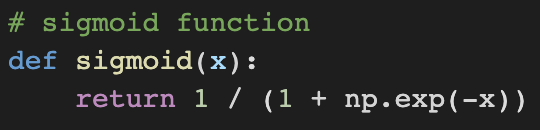
**۲ ) feed forward**

در این گام ابتدا ٢٠٠ داده اول را جدا میکنیم و سپس خروجی را برای آن ها حساب میکنیم.

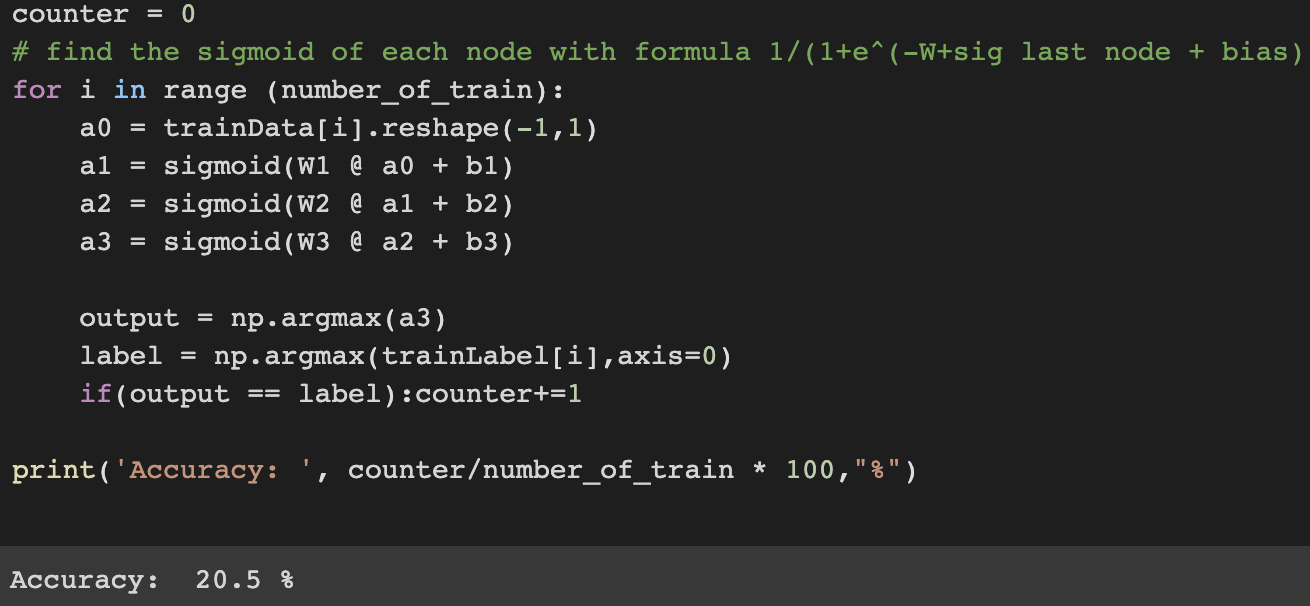
وزن ها را اعدادی رندوم و بایاس ها را صفر initialize میکنیم.



تابع سیگموید نیز به شکل زیر تعریف می شود :

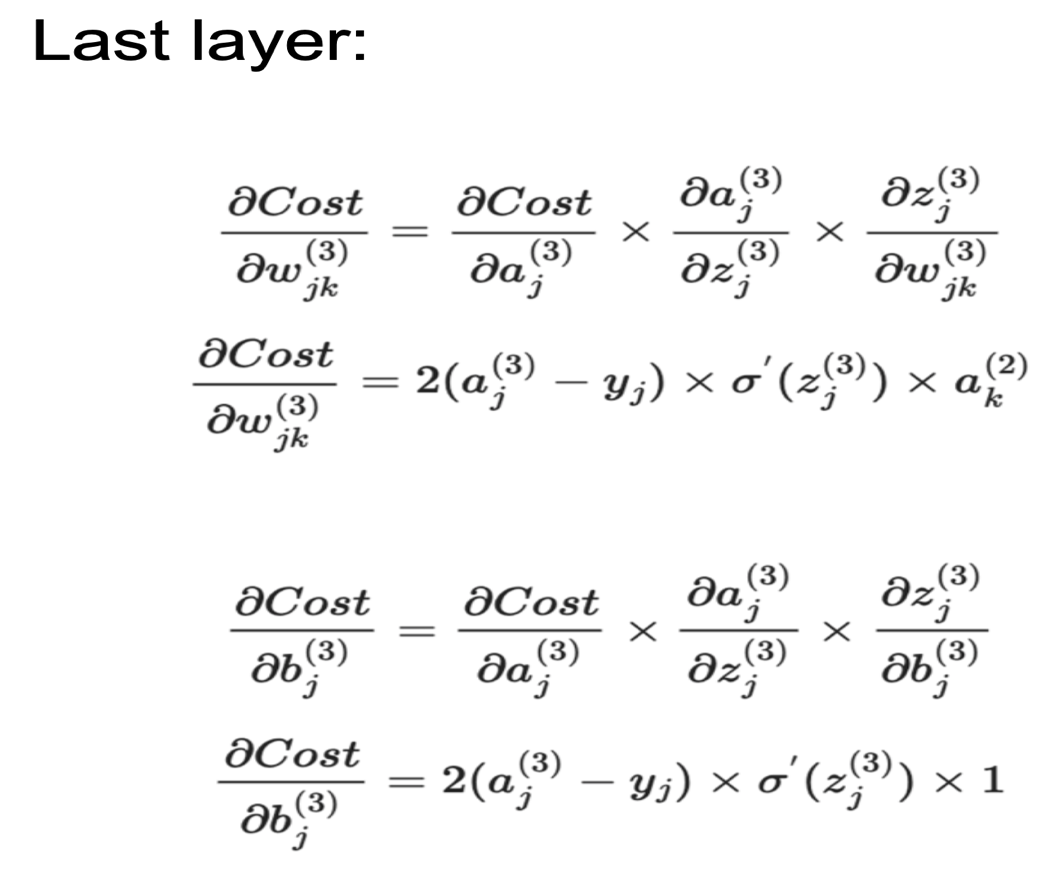


دقت را نیز در آخر حساب میکنیم و میبینم که حدود ۲۰ درصد خواهد بود :

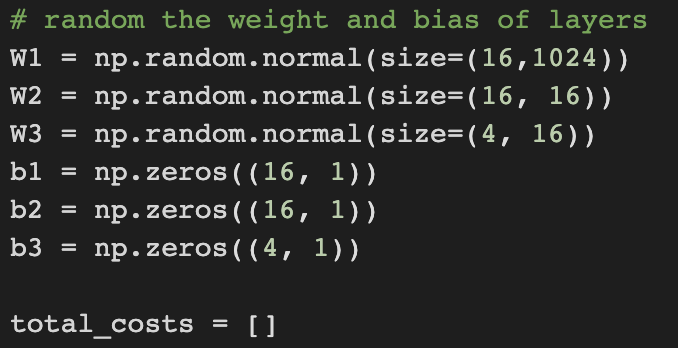


**۳ ) back propagation**

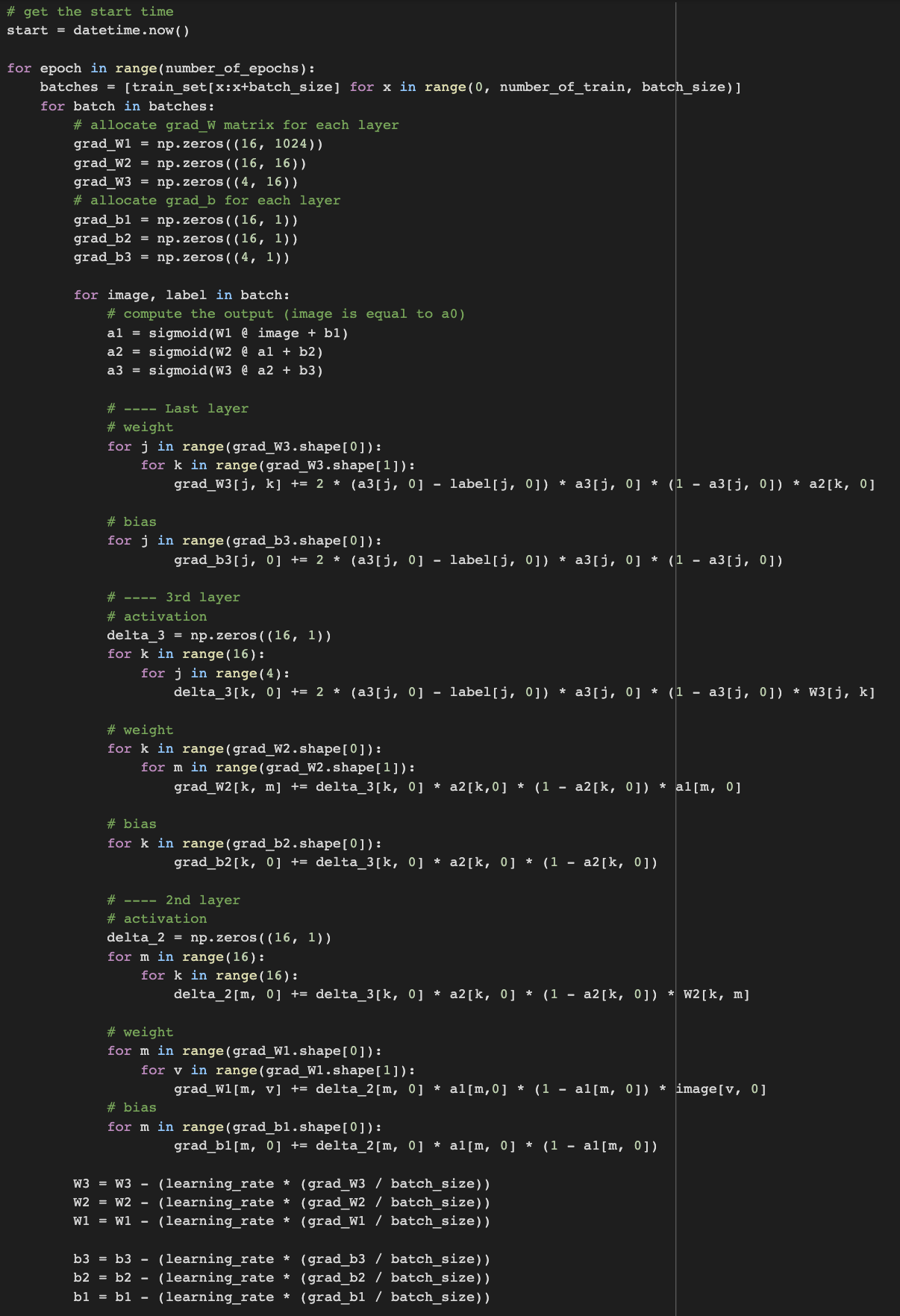
در این قسمت الگوریتم gradient descend را پیاده می کنیم و مشتقات جزیی را به شکل زیر به دست می آوریم و هدف کمینه کردن تابع هزینه است :

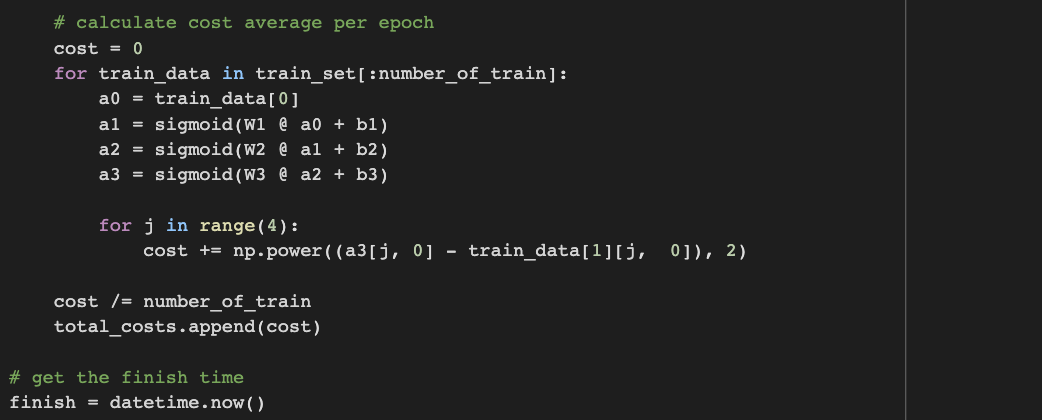


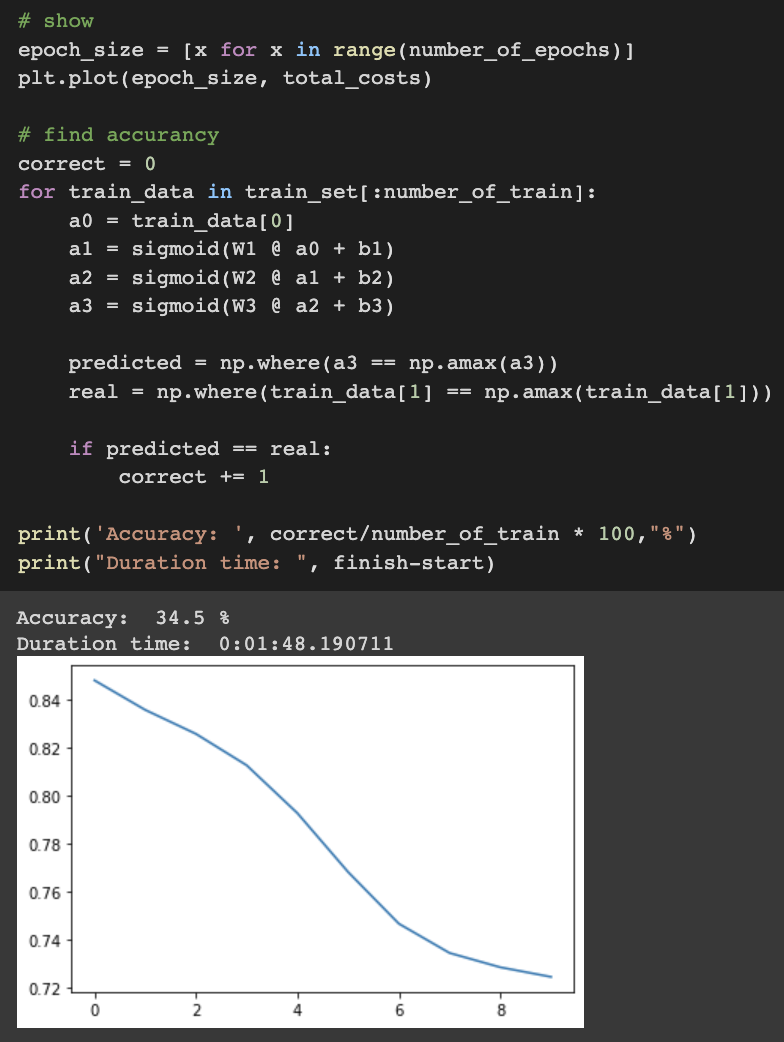
همانند بخش قبلی ابتدا مقادیر وزن و بایاس ها را مقدار دهی اولیه می کنیم :



در آخر دقت را حساب میکنیم که حدود ۳۴ درصد است ، نمودار میانگین cost نمونه ها در هر epoch را رسم می کنیم که به شکل نزولی است. واضح است که به دلیل حجم محاسبات زمان اجرا طولانی است.



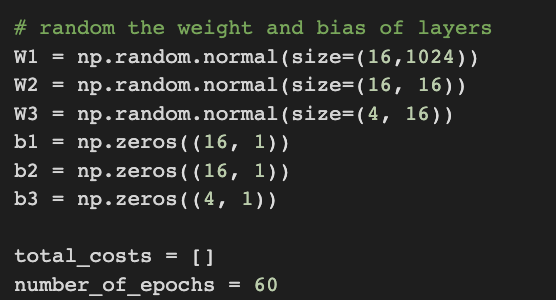




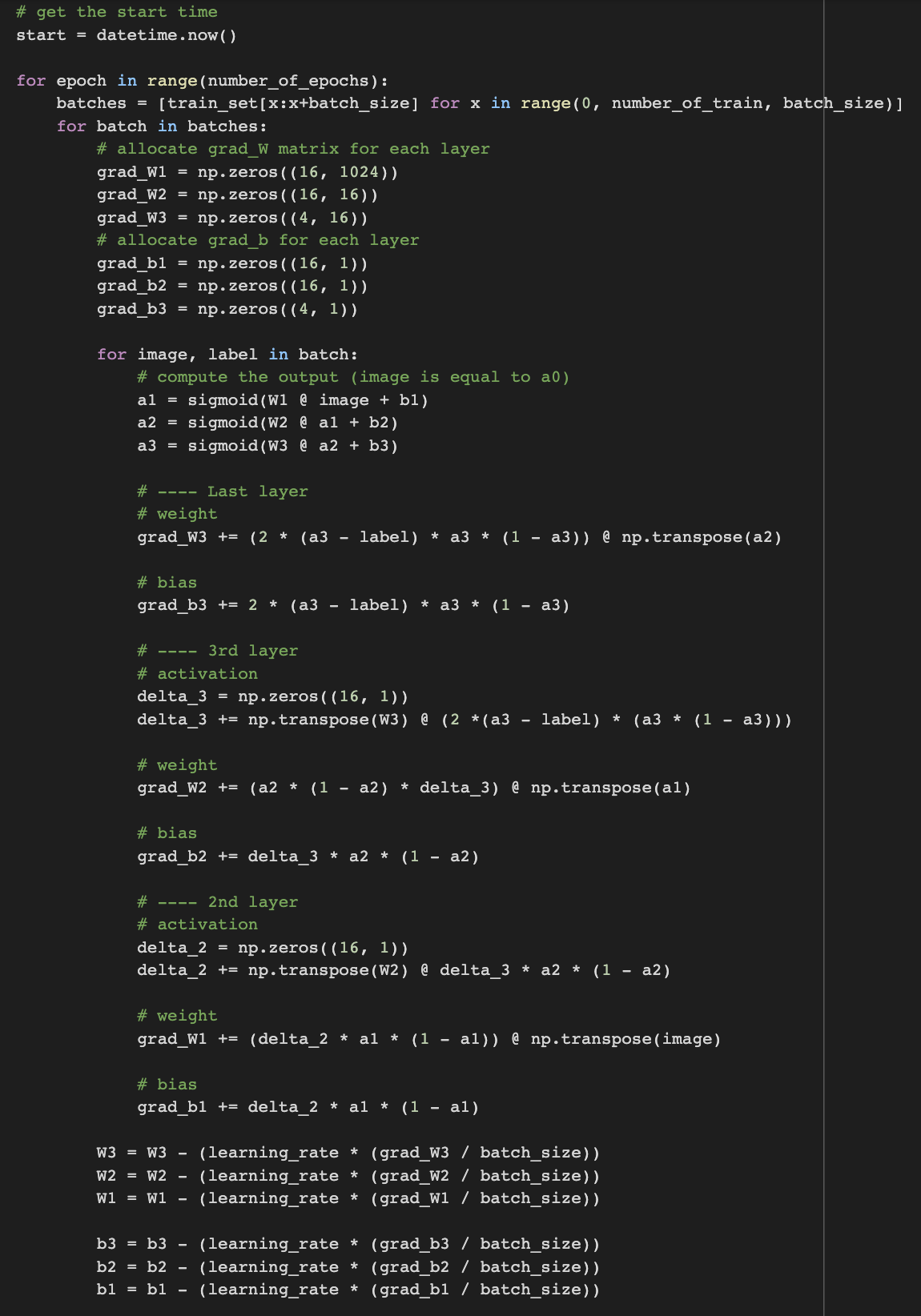
**۴ ) Vectorization**

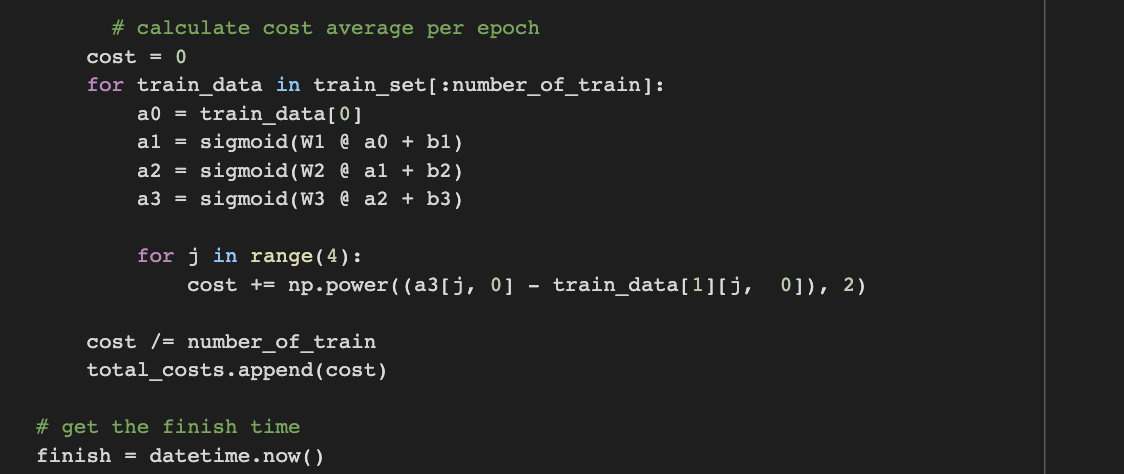
در این قسمت با استفاده از عملیات روی ماتریس ها سعی می کنیم که سرعت اجرا را کاهش دهیم تا بتوانیم از نمونه های بیشتری برای آموزش شبکه استفاده کنیم و زمان منطقی داشته باشد.

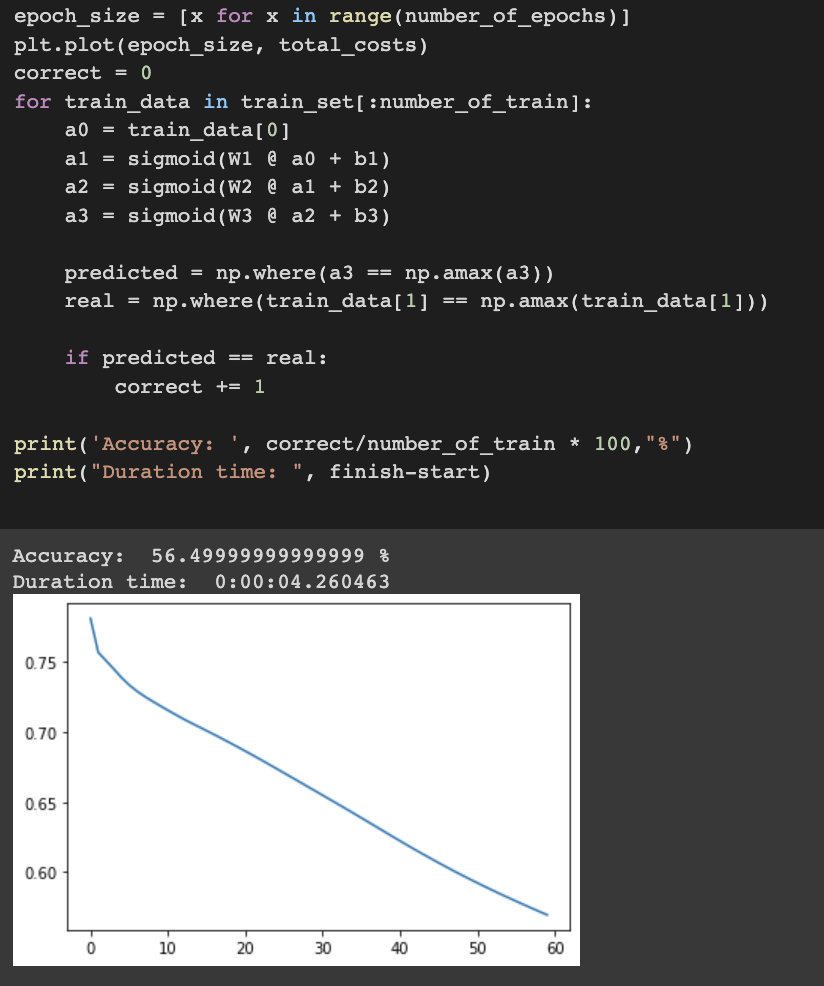
برای نتیجه گیری بهتر تعداد ایپاک ها را ۶۰ قرار می دهیم.



دقت را نیز مانند قبل محاسبه می کنیم که حدود ۵۶ است و زمان اجرا نیز از بخش قبلی خیلی کوتاه تر شده است.

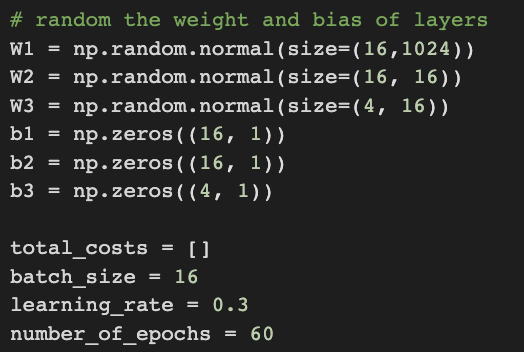




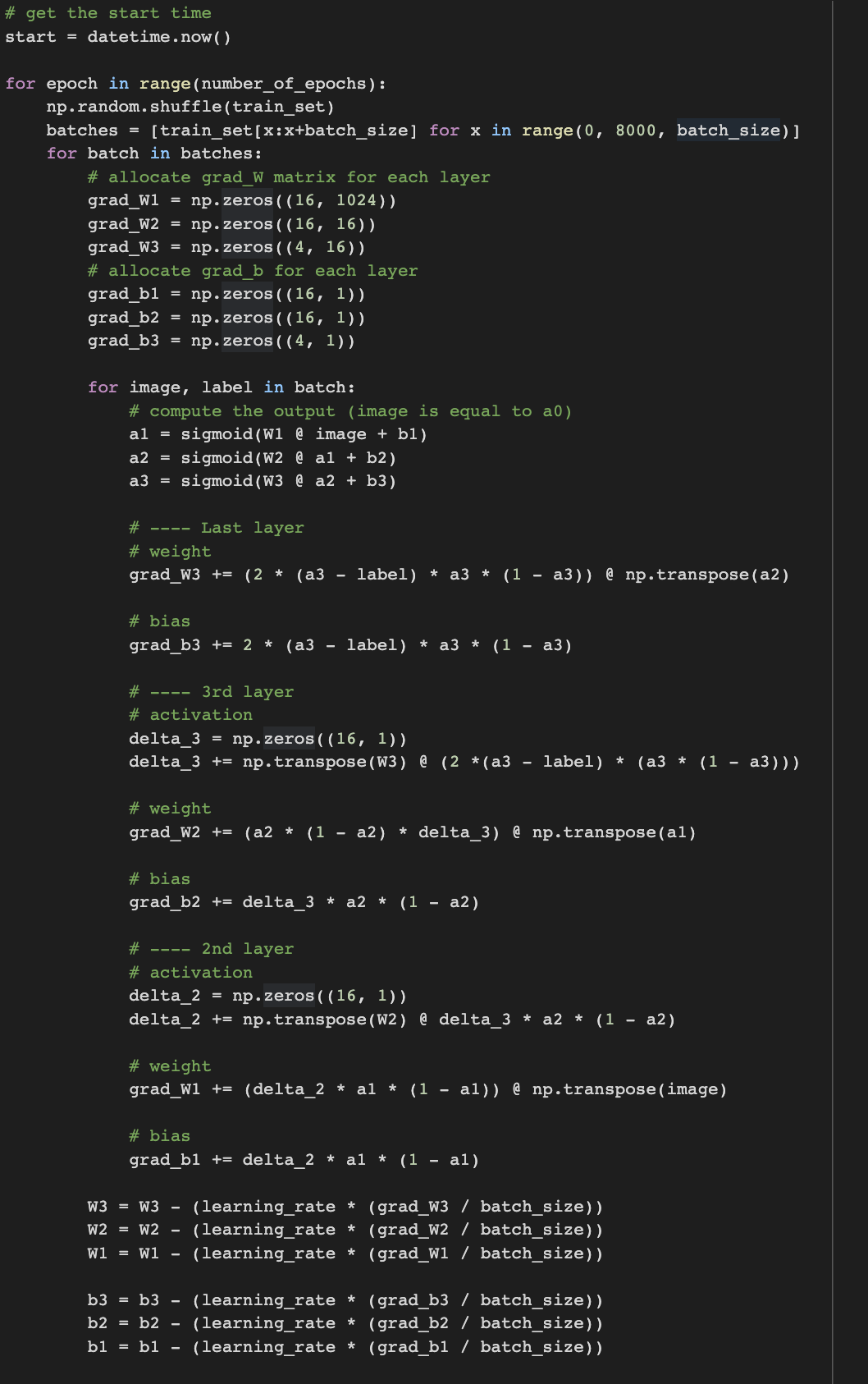


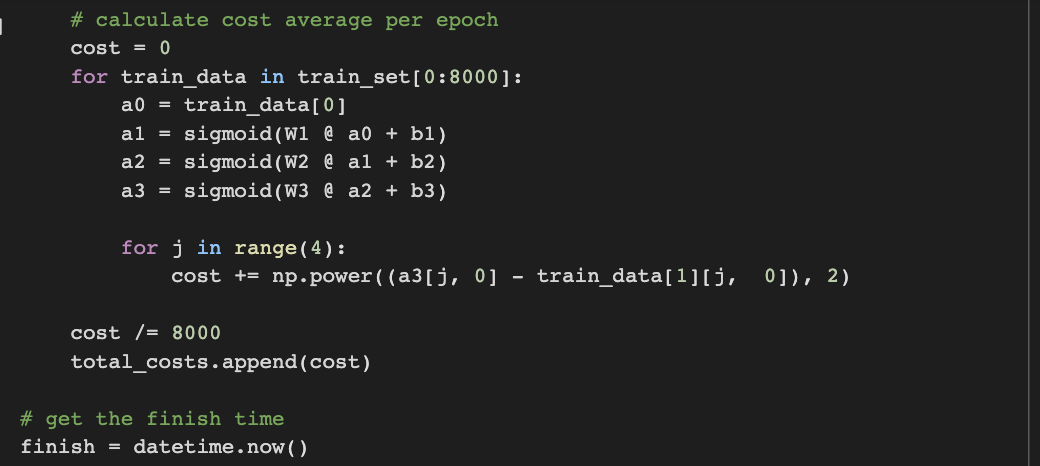
**۵ ) تست مدل**

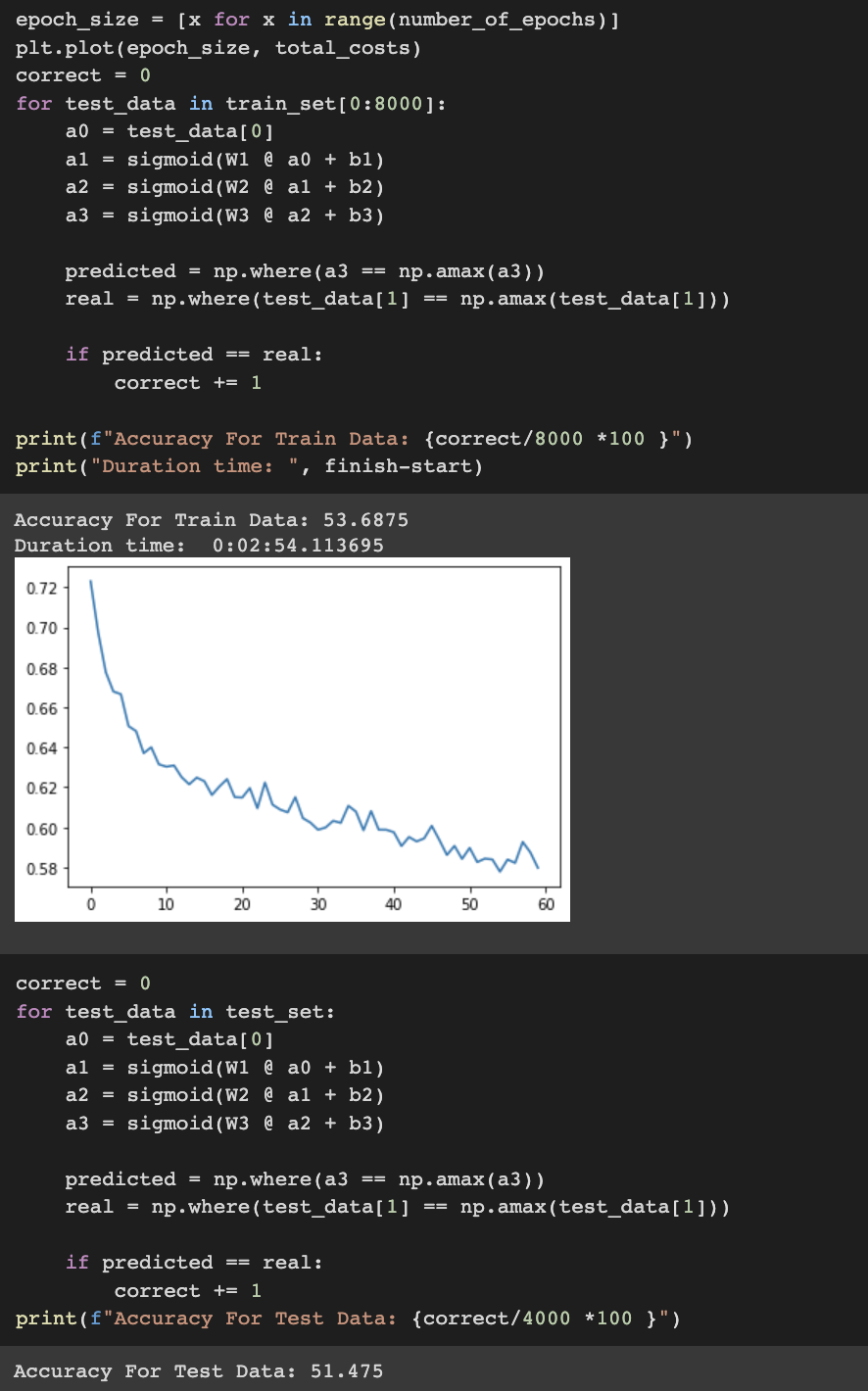
در این قسمت از همه ۸۰۰۰ داده ترین استفاده می کنیم و مدل را تست می کنیم :



در آخر دقت برای داده های تست حدود ۵۱ و داده های ترین ۵۳ خواهد بود.





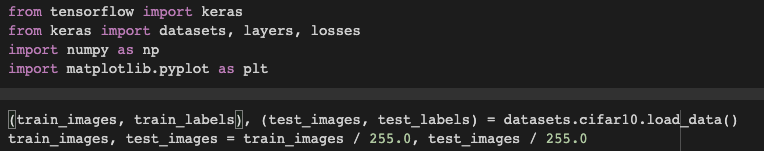


**بخش امتیازی**

**بخش اول**

۱ )

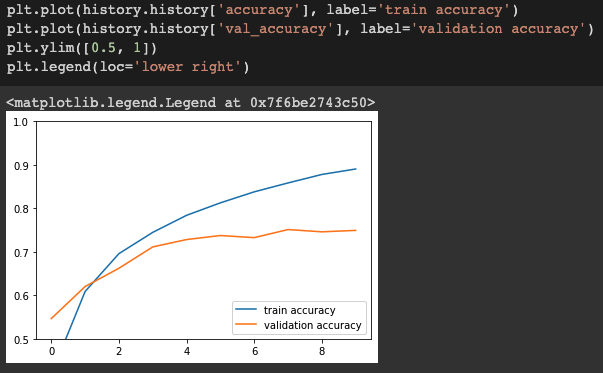
برای پیاده سازی این شبکه CNN ابتدا مانند قبل باید دیتاست را آمادی کنیم که این کار را با استفاده از keras و tensorflow انجام می دهیم :



سپس معماری داده شده را پیاده می کنیم :



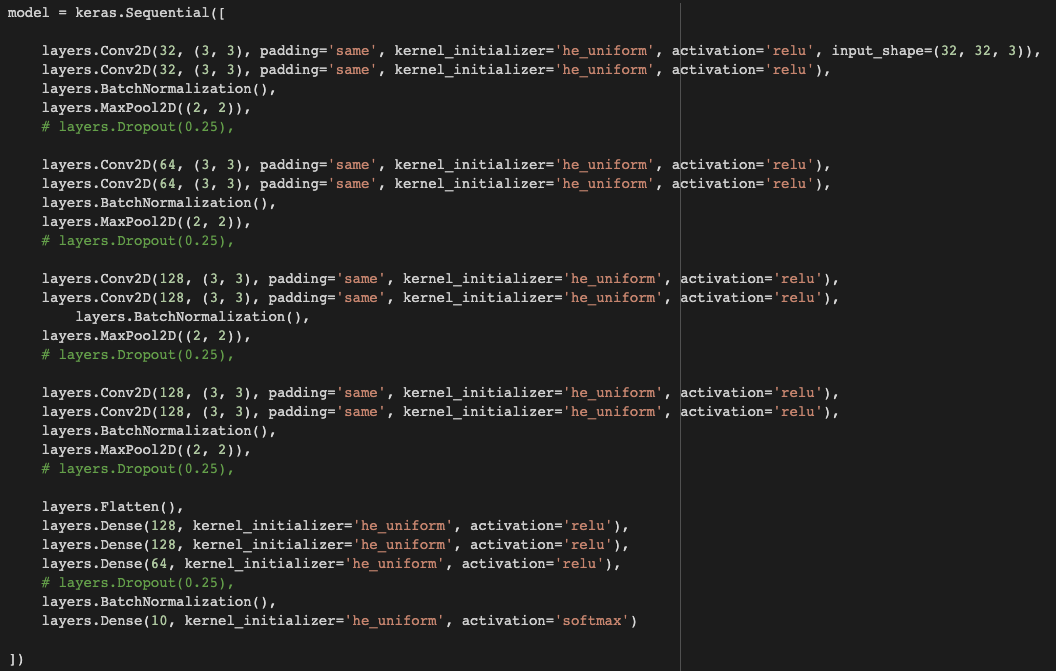
و بعد مدل را آموزش می دهیم و با داده های تست آنرا ارزیابی می کنیم.



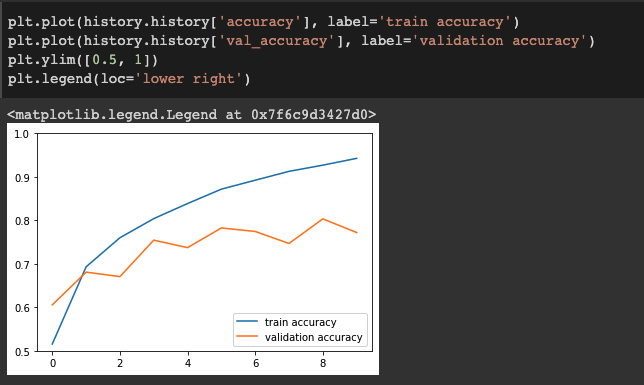
مشاهده می شود که مدل به خوبی عمل نمی کند و دچار overfitting شده است.

۲ )

Batch normalization : نرمال‌سازی دسته‌ای تکنیکی برای آموزش شبکه‌های عصبی بسیار عمیق است که ورودی‌های یک لایه را برای هر mini batch استاندارد می‌کند به این معنی که میانگین صفر و انحراف معیار یک خواهند داشت. این امر باعث تثبیت فرآیند یادگیری و کاهش چشمگیر تعداد دوره های آموزشی( epoch ) مورد نیاز برای آموزش شبکه های عمیق می شود. به این شکل آنرا به مدل اضافه می کنیم :

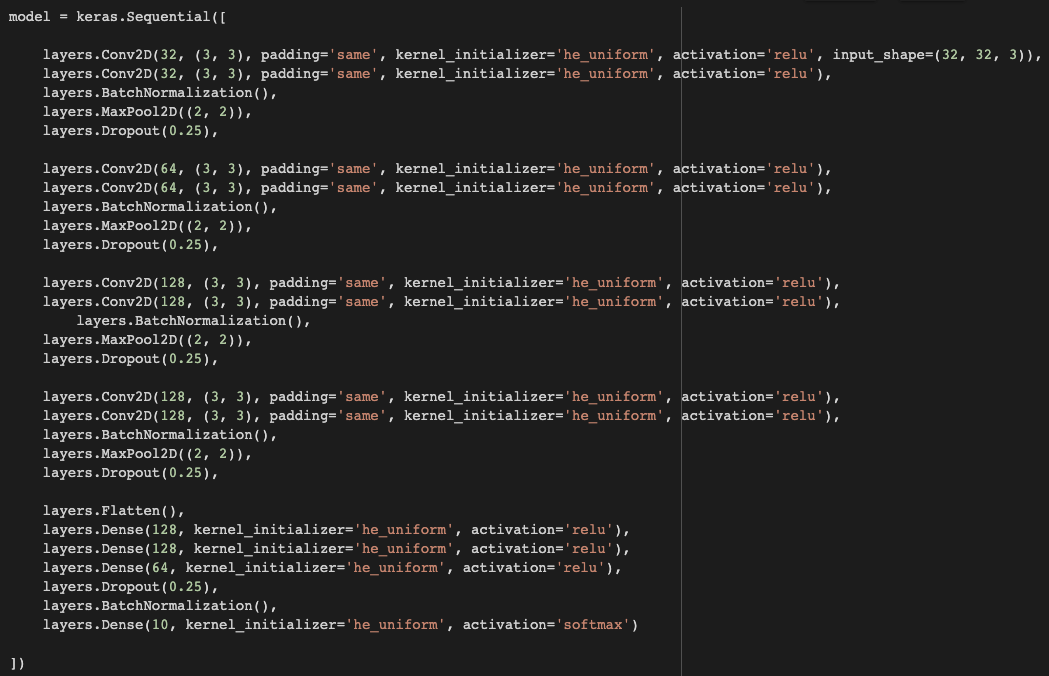


مشاهده میشود که سرعت آموزش کمی بهتر شده و هم چنین دقت داده های تست بهتر شده است :



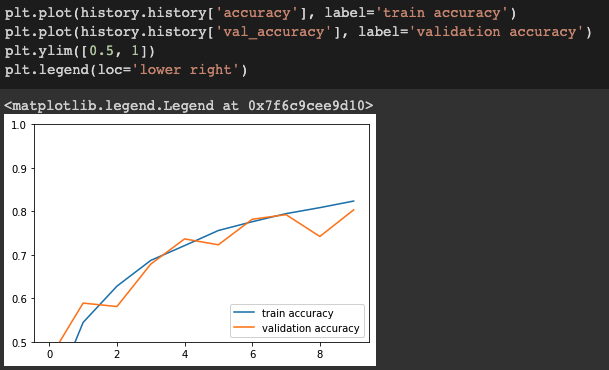
۳ )

dropout تکنیکی است که در آن نورون‌های انتخابی به شکل تصادفی در طول تمرین نادیده گرفته می‌شوند. این به این معنی است که سهم آنها در فعال سازی نورون های لایه های قبل به طور موقت در گذر رو به جلو حذف می شود و هرگونه به روز رسانی وزنی برای نورون در گذر به عقب اعمال نمی شود. در این مدل با استفاده از keras آن ها را در جای مناسب اضافه می کنیم :



مشاهده می شود که خطا تا کمتر از ۱ درصد در ۱۰ ایپاک رسیده و دقت که در مرحله قبلی حدود ۷۴ بود ه بالای ۸۰ درصد رسیده است و مشکل overfitting بر طرف شده است :





۴ )

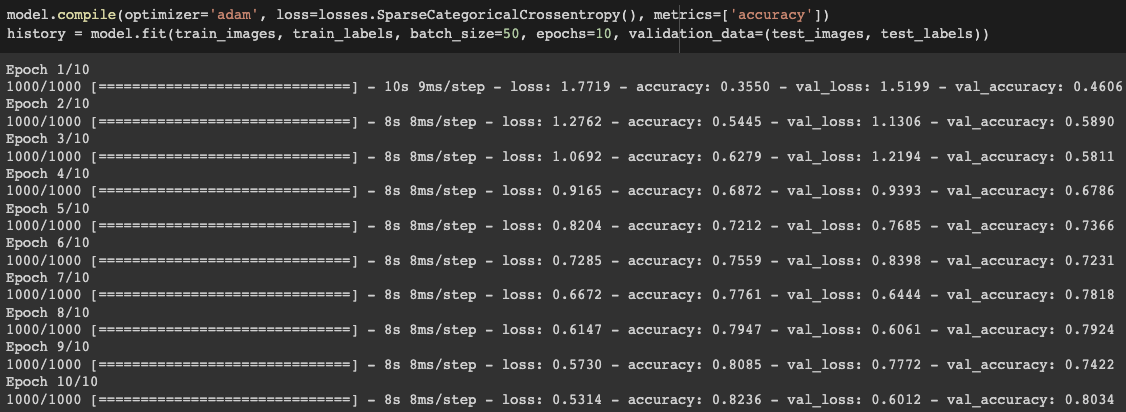
بهینه‌سازها کلاس‌ها یا متد هایی هستند که برای تغییر ویژگی‌های مدل یادگیری عمیق ماشین شما مانند وزن‌ها و نرخ یادگیری به منظور کاهش خطاها استفاده می‌شوند. بهینه سازها به دریافت سریعتر نتایج کمک می کنند.

انواع مختلفی از optimizer ها داریم که چند مورد از آن ها را در اینا توضیح می دهیم :

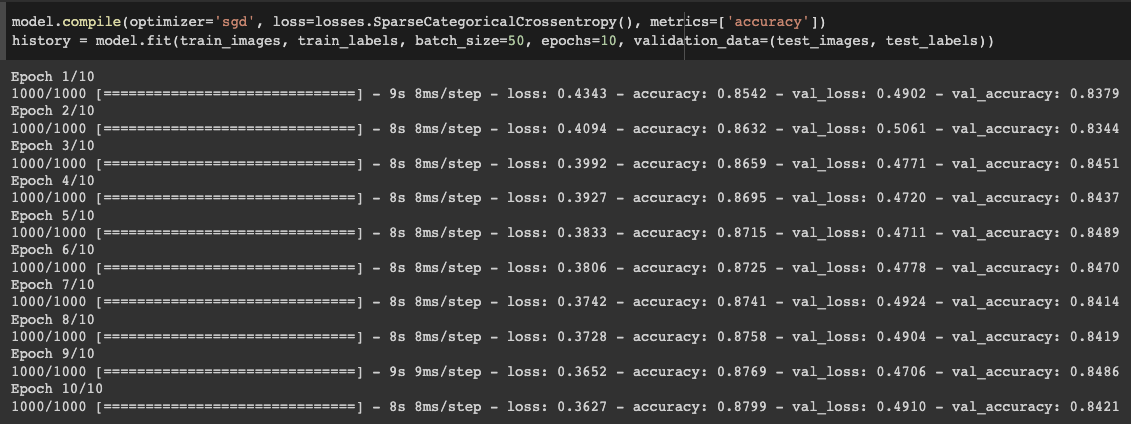
* Adadelta: Optimizer that implements the Adadelta algorithm.
* Adagrad: Optimizer that implements the Adagrad algorithm.
* Adam: Optimizer that implements the Adam algorithm.
* Adamax: Optimizer that implements the Adamax algorithm.
* Ftrl: Optimizer that implements the FTRL algorithm.
* Nadam: Optimizer that implements the NAdam algorithm.
* Optimizer class: Base class for Keras optimizers.
* RMSprop: Optimizer that implements the RMSprop algorithm.
* SGD: Gradient descent (with momentum) optimizer.

در اینجا ما بهیه ساز های adam و sgd و adadelta را با هم بررسی و مقایسه می کنیم:

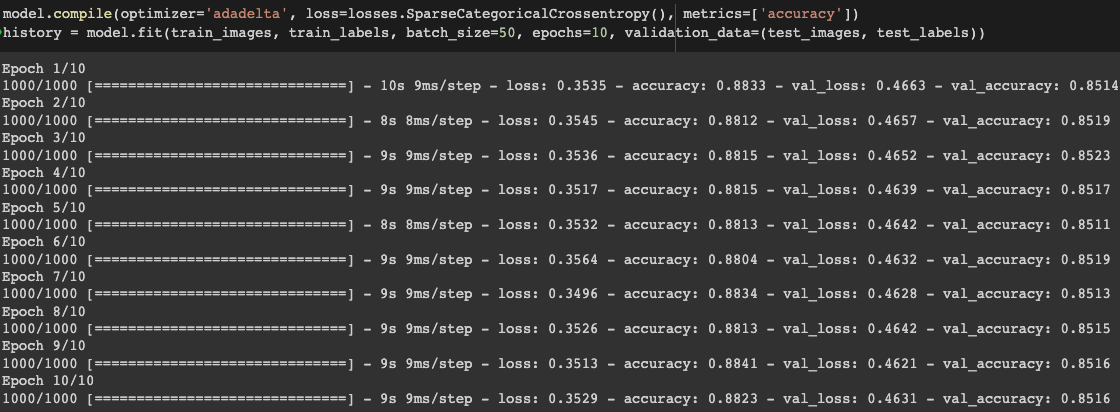
1) Adam



2) SGD



3) Adadelta



به نظر می رسد که adadelta بهتر از بقیه عمل کرده و دقت به حدود ۸۵ رسیده است هم چنین میزان خطا کاهش بیشتری داشته است .

۵ )

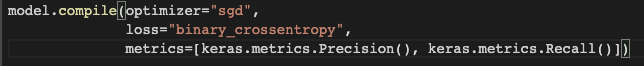
مهمترین معیار های ارزیابی در مسائل کلاس بندی ، accuracy, precision, و recallهستند.

accuracy:‌ به عنوان درصد پیش بینی های صحیح برای داده های تست تعریف می شود. می توان آن را به راحتی با تقسیم تعداد پیش بینی های صحیح بر تعداد کل پیش بینی ها محاسبه کرد.

precision:‌ به عنوان کسری از مثال‌های مرتبط (مثبت‌های واقعی) در بین همه نمونه‌هایی که پیش‌بینی شده بود به یک کلاس خاص تعلق دارند، تعریف می‌شود.

recall: یادآوری به عنوان کسری از مثال‌ها تعریف می‌شود که پیش‌بینی می‌شد با توجه به همه نمونه‌هایی که واقعاً به کلاس تعلق دارند، متعلق به یک کلاس باشند.

F1 : که میانگین هارمونیک از recall و precision است.



به این صورت و در آرایه metrics می توانیم مقادیر آن ها را به دست آوریم.

**بخش دوم**

۱ )

افزایش داده ها ( data augmentation ) فرآیند اصلاح یا "افزایش" یک مجموعه داده با داده های اضافی است. این داده های اضافی می تواند هر چیزی از تصویر گرفته تا متن باشد و استفاده از آن در الگوریتم های یادگیری ماشینی به بهبود عملکرد آنها کمک می کند.

مثلاً می‌خواهیم مدلی برای طبقه‌بندی نژادهای سگ بسازیم و از اکثر نژادها به جز پاگ، تصاویر زیادی داریم. در نتیجه، مدل نمی تواند پاگ ها را به خوبی طبقه بندی کند. ما می‌توانیم داده‌ها را با افزودن برخی تصاویر (واقعی یا جعلی) از پاگ‌ها، یا با ضرب کردن تصاویر پاگ موجود خود (مثلاً با تکثیر و تحریف آنها برای منحصربه‌فرد ساختن آنها) افزایش دهیم.