**a**

**دانشگاه علم و صنعت**

**تمرین ششم مبانی بینایی کامپیوتر**

**نام و نام خانوادگی:**

فرناز خوش دوست آزاد

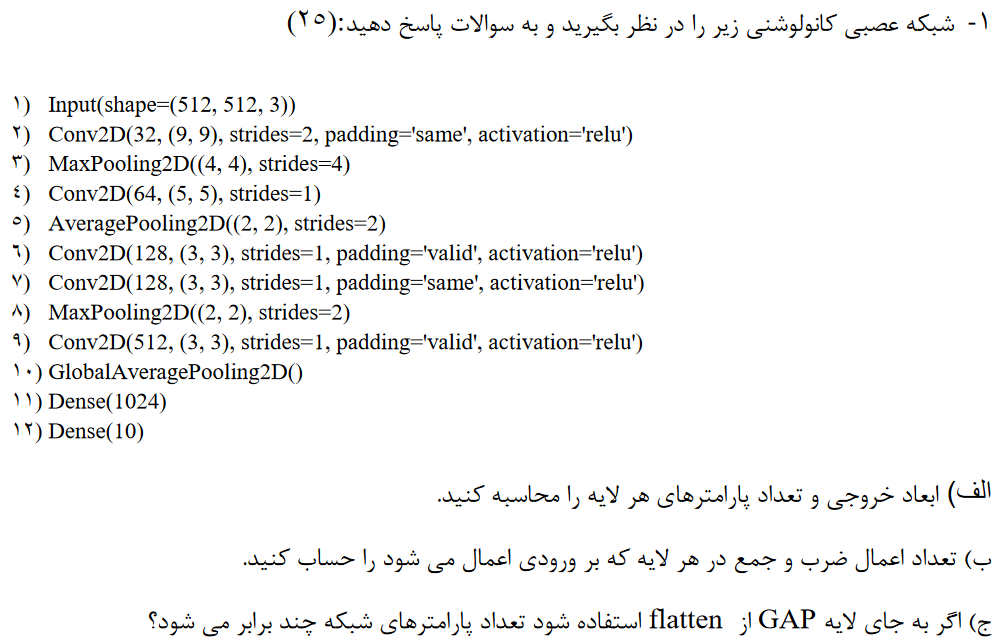
**شماره دانشجویی:**

99521253

**نام استاد:**

دکتر محمدرضا محمدی

1)



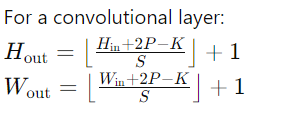
پاسخ:

الف)

برای هر عملگر بر روی ورودی تعداد پارامتر ها و ابعاد خروجی را با استفاده از فرمول های زیر که در اسلاید ها نیز موجود هستند، به دست می آوریم.

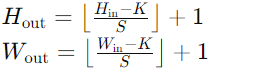
**Convolutional Layer Output Dimensions:**

For a convolutional layer:



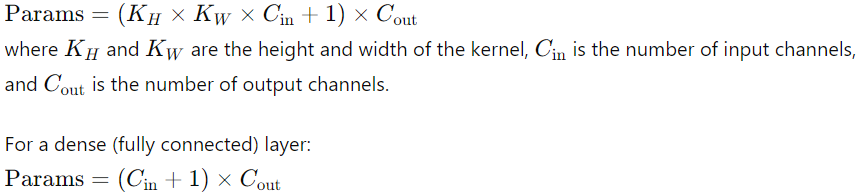
**Pooling Layer Output Dimensions**

For a pooling layer:

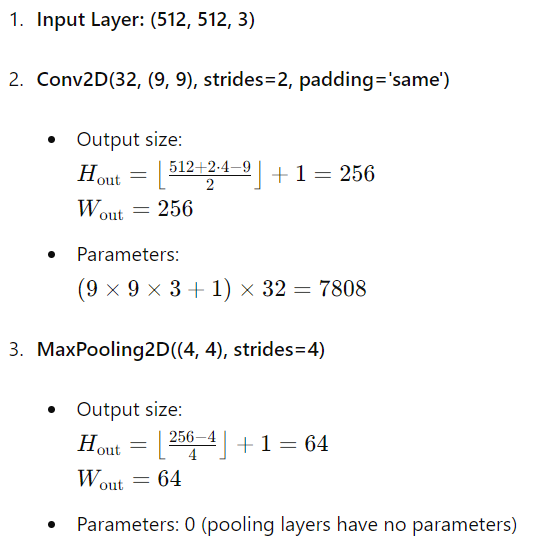


**Number of Parameters**

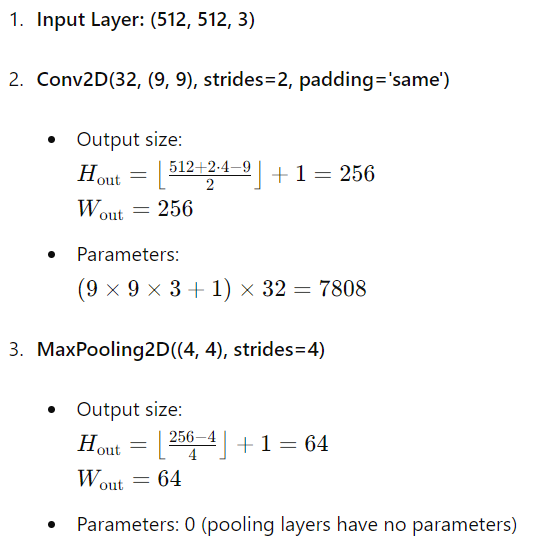
For a convolutional layer:



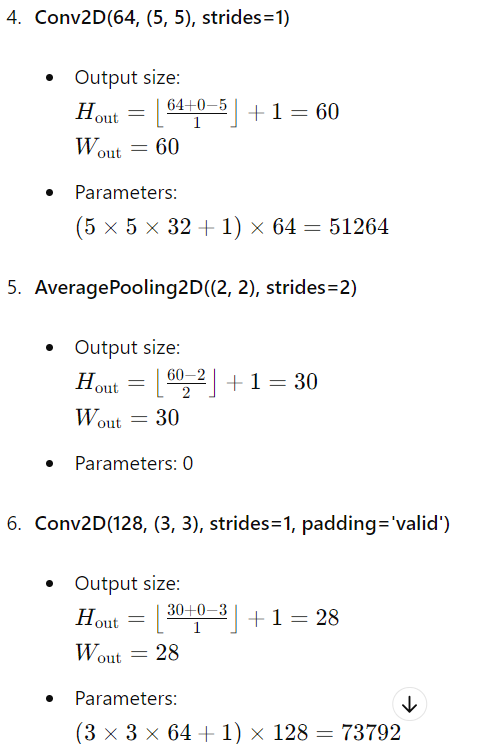
حال به محاسبات مورد نظر برای هر لایه می پردازیم.



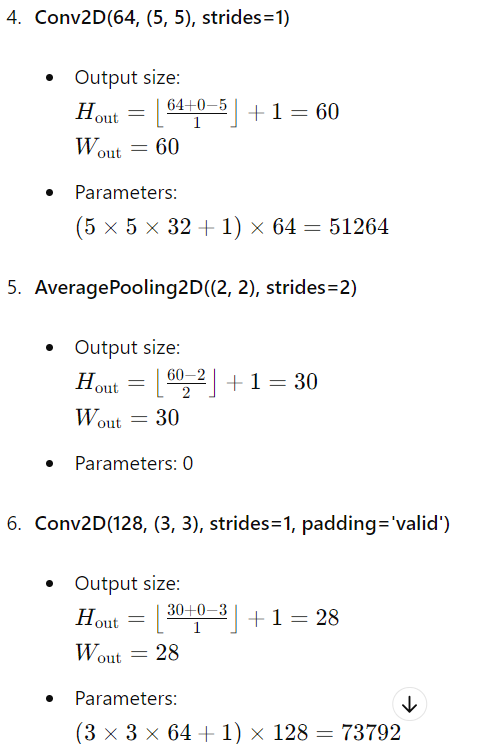
لایه ی اول که ورودی است و لذا همان ابعاد وردی را داریم و پارامتر ها نیز ثابت هستند. در لایه ی دوم چون padding = 'same' را داریم بدان معناست که ابعاد ورودی و خروجی یکی هستند، با این تفاوت که چون stride داریم، باید ابعاد ورودی را بر stride تقسیم کنیم.



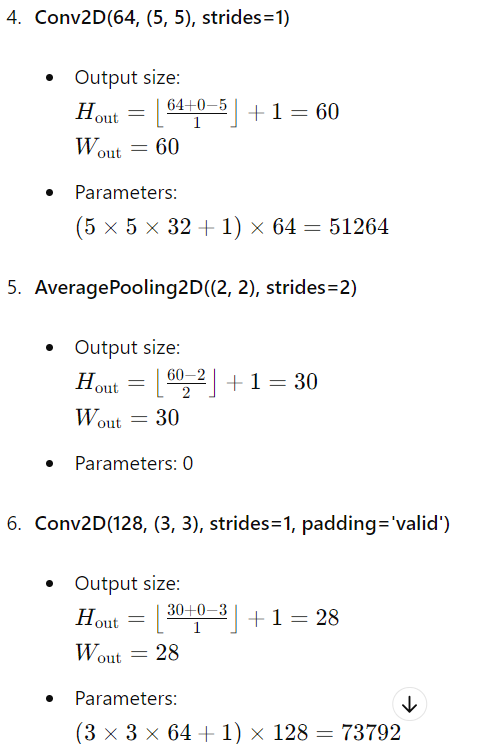
چون عرض و طول محاسبات یکسانی دارند، از نوشتن آنها پرهیز کردم.



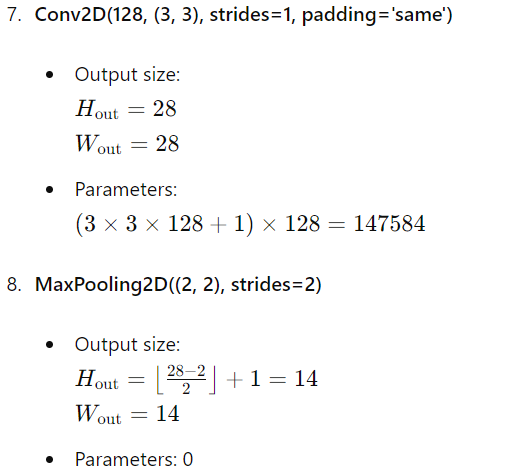
چون از قبل 32 تا لایه فیلتر داریم لذا باید برای تعداد پارامترها آن را ضربدر 32 کنیم و سپس به علاوه ی بایاس کنیم.



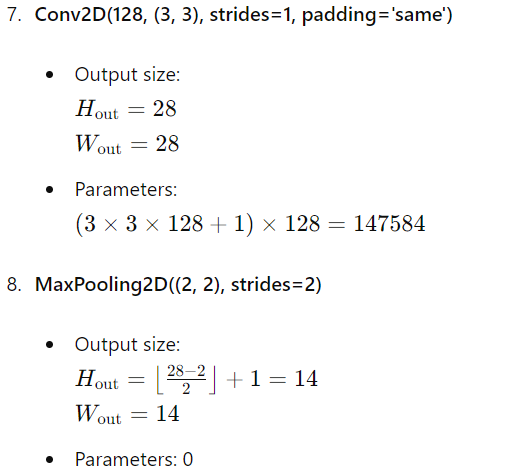
مانند حالت قبل برخورد می کنیم.

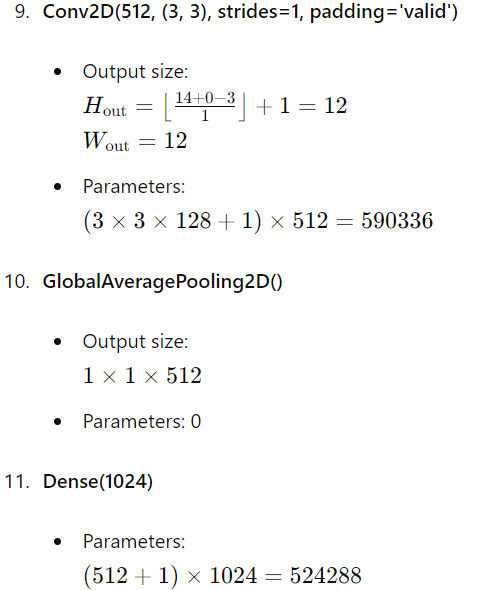


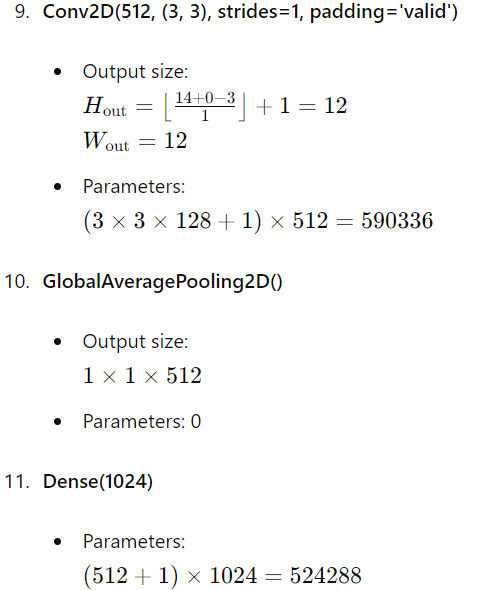
حال می دانیم از لایه ی قبل 64 تا فیلتر داریم، لذا از آن در فرمول parameters استفاده می کنیم.

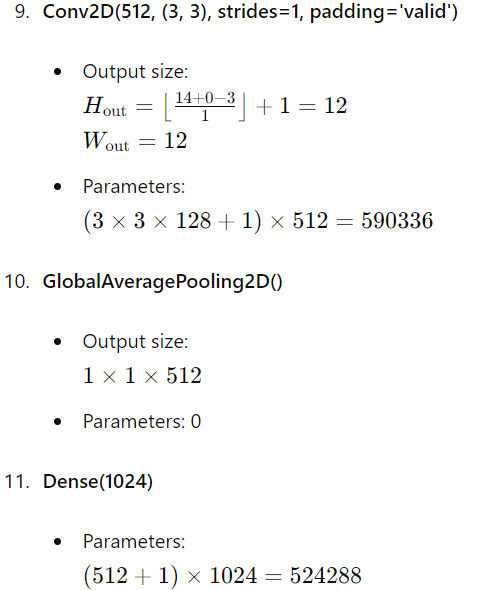


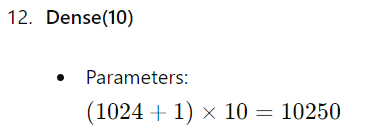
مانند حالت های بالا برخورد می کنیم.





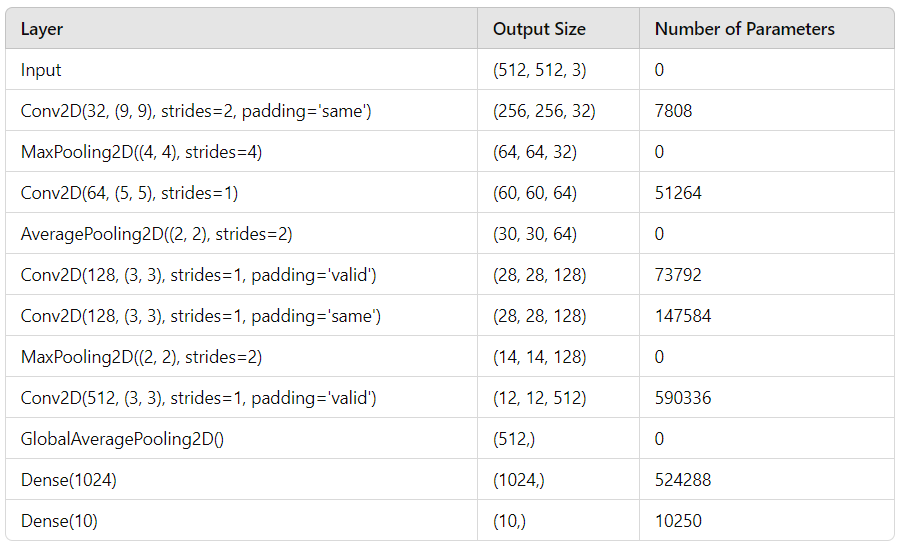






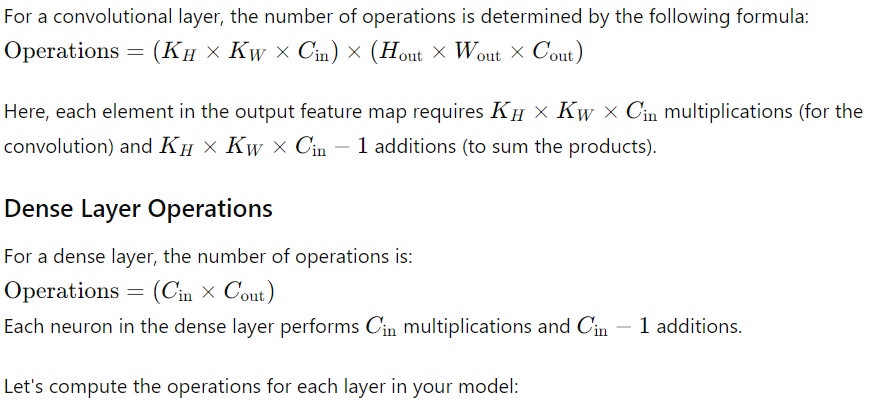
**Summary**

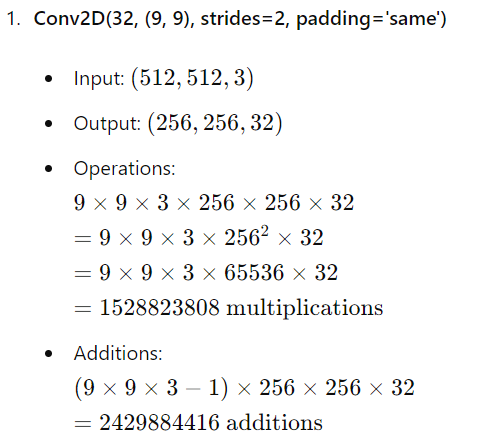
به اختصار جدول زیر محاسبات بالا را نشان می دهد.

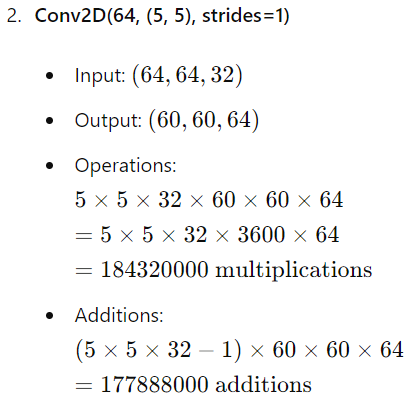


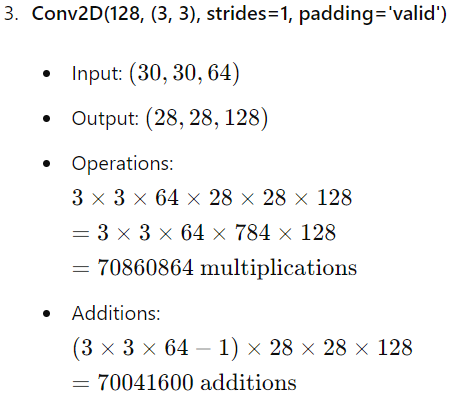
Total number of parameters: ( 7808 + 51264 + 73792 + 147584 + 590336 + 524288 + 10250 = 1,405,322).

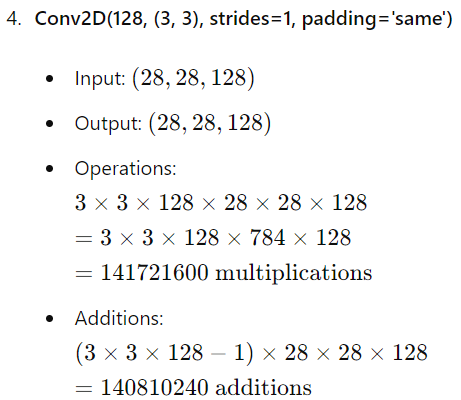
ب)

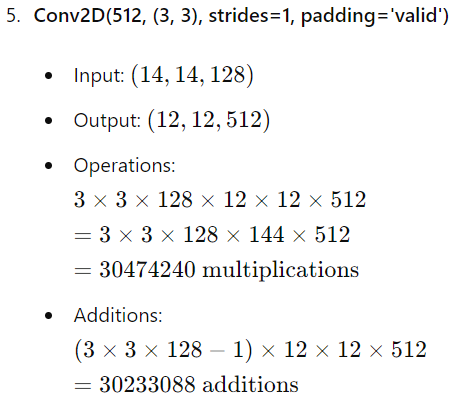


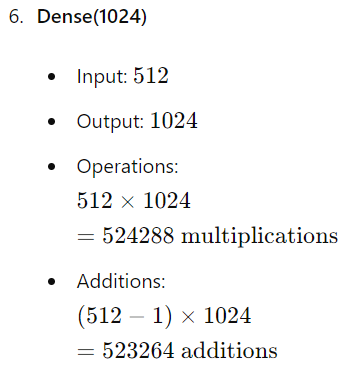


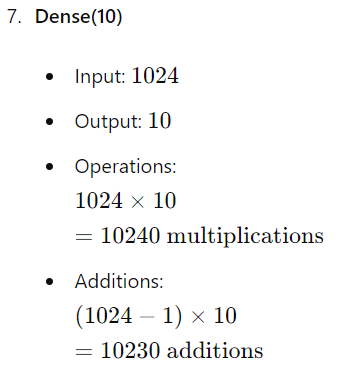


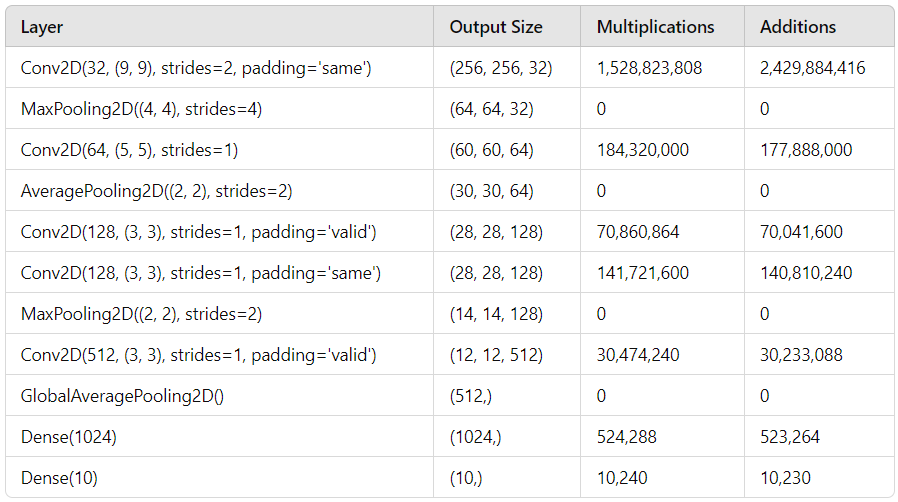






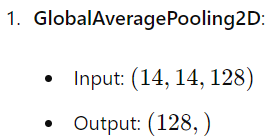


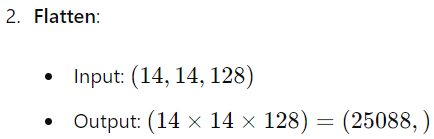


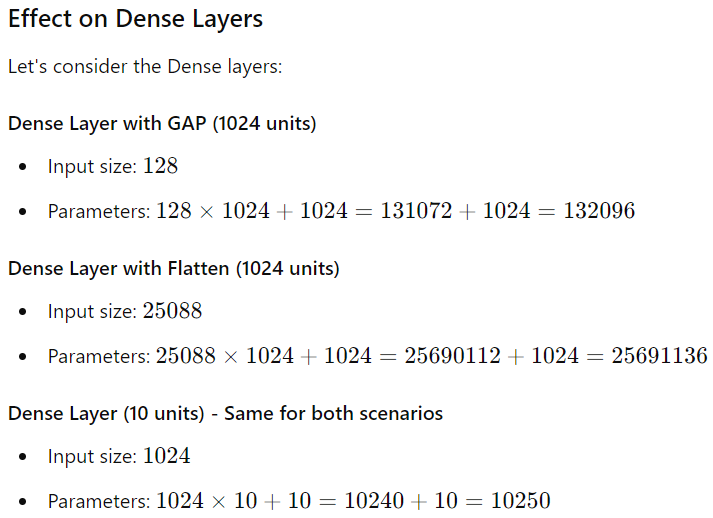


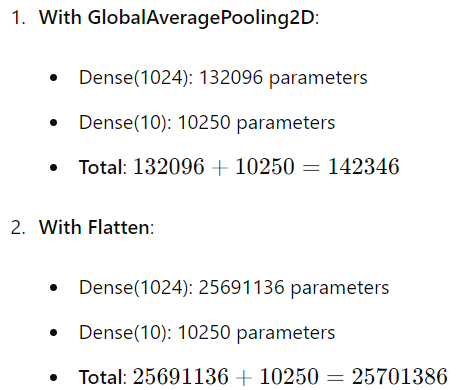
ج)

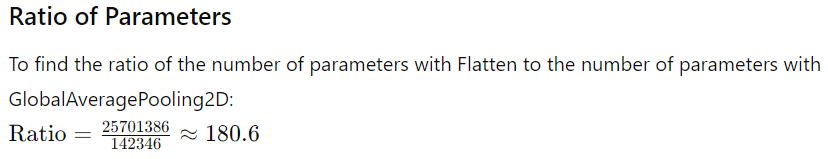
اگر به جای لایه GAP از flatten استفاده شود، تعداد پارامترهای شبکه چند برابر می شود؟



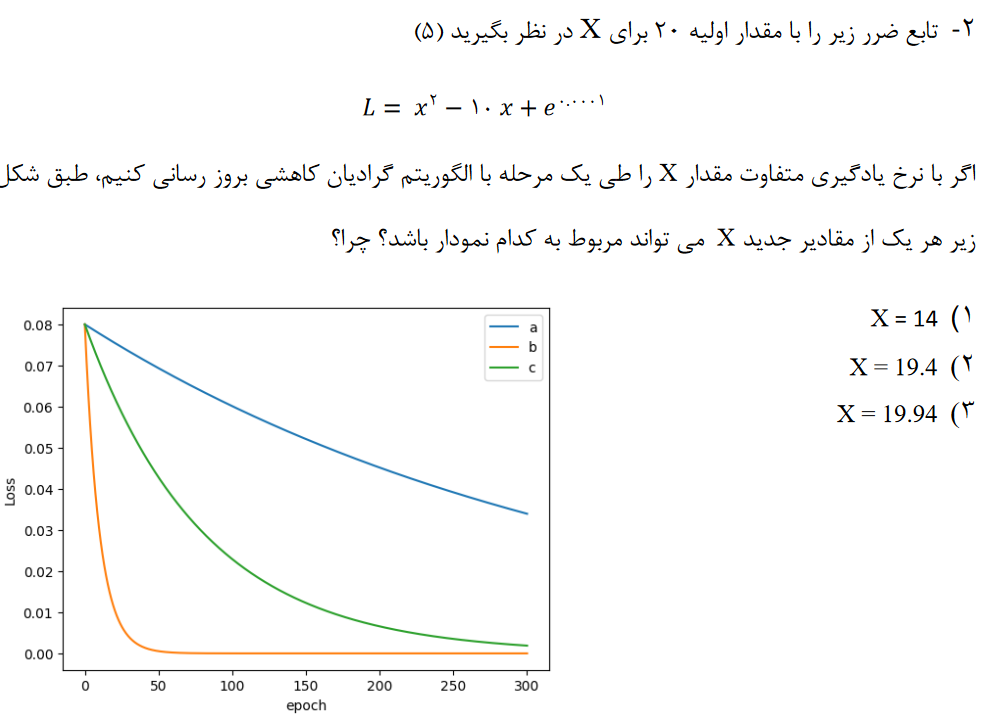








2)



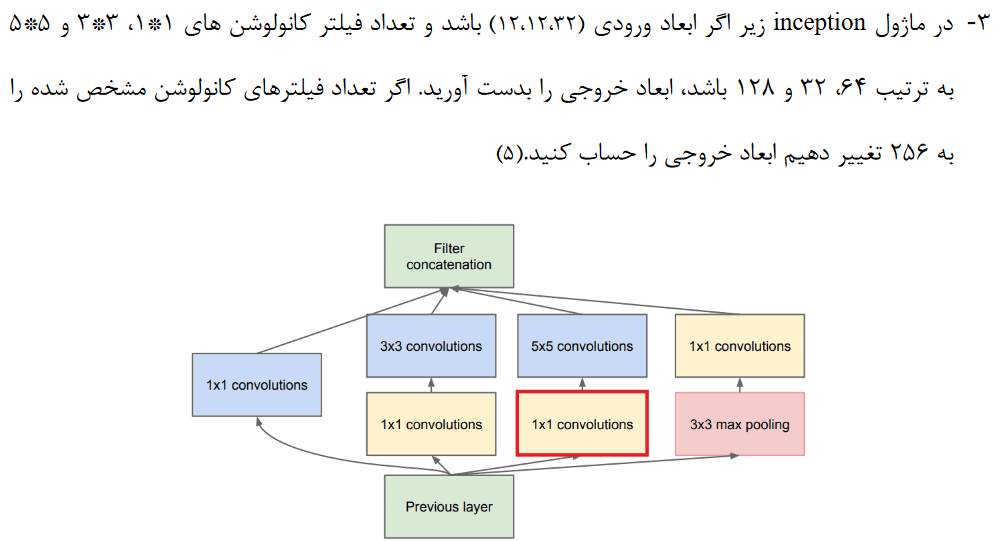
پاسخ:

از آنجایی که باید مقدار تابع ضرر را مینیمم کنیم، لذا به سراغ مشتق می رویم، و آن را برابر با صفر قرار می دهیم تا مقدار مینیمم پیدا شود لذا داریم:

2x - 10 = 0 → x = 5

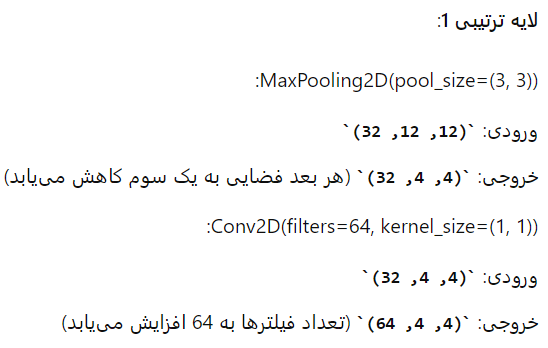
چون از مقدار 20 شروع کرده ایم، لذا باید با مقدار a مناسب آن را به کمترین مقدارش یعنی 5 برسانیم، بنابراین از آنجایی که خط آبی با سرعت بسیار کمی به سمت کم شدن قدم بر میدارد لذا مقدار 19.94 را به آن assign می کنیم چون فقط 0.06 از آن کم شده است که حاکی از سرعت کم آن است و از آنجایی که خط سبز با سرعت مناسبی در حال کم شدن است لذا مقدار X = 19.4 را برای آن assign می کنیم و از آنجایی که خط نارنجی با سرعت بسیار زیادی به سمت کم شدن می رود مقدار 14 برای آن مناسب است که از مقدار قبلی خود 6 واحد فاصله دارد.

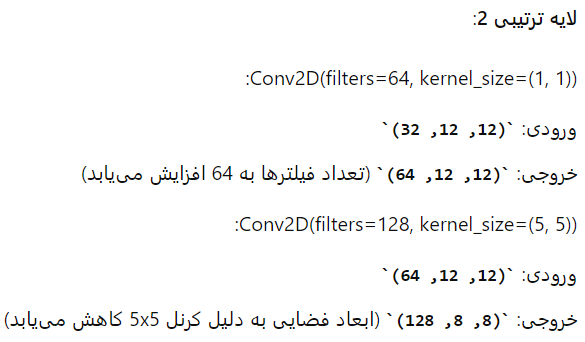
3)

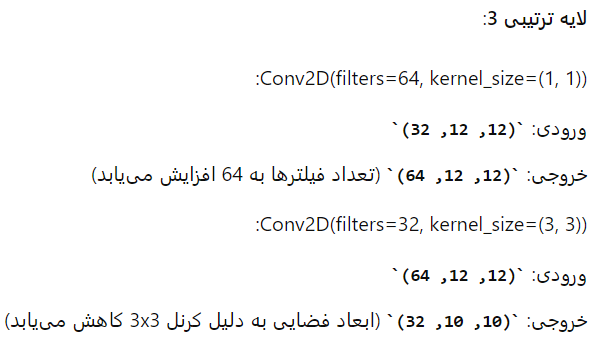


پاسخ:

در ابتدا در شکل می بینیم که سه لایه ی متوالی داریم و در آخر 4 لایه با هم convatenate می شوند. کد مربوط به آن در فایل آپلود شده موجود می باشد که می بینیم قابلیت concatenate کردن ندارد و به ارور خواهیم خورد که برای رفع این ارور باید از لایه هایی مثل globalMaxPooling یا globalAveragePooling استفاده کرد تا بتوان آنها را concatenate کرد، لذا داریم:



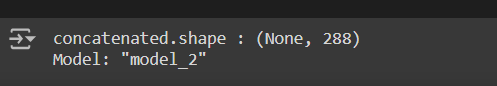




برای ترکیب خروجی‌ها با استفاده از Concatenate، باید خروجی‌ها ابعاد فضایی یکسانی داشته باشند. که در اینجا خروجی‌ها دارای ابعاد متفاوتی هستند و نمی‌توان آنها را مستقیماً ترکیب کرد. لذا از روشی که در ابتدا گفته شد، استفاده می کنیم و در صورتی که بخواهیم ابعاد خروجی‌ها را به درستی برای هر یک از لایه‌های ترتیبی و موازی محاسبه کنیم، این ابعاد به شکل زیر خواهند بود:

* لایه ترتیبی 1: (4, 4, 64)
* لایه ترتیبی 2: (8, 8, 128)
* لایه ترتیبی 3: (10, 10, 32)
* لایه موازی دیگر: (12, 12, 64)

در نهایت، این خروجی ها را با استفاده از GlobalMaxPooling خروجی های 64، 128، 32، 64 و … را می گیریم و سپس خروجی‌های یک‌بعدی را با استفاده از Concatenate ترکیب می‌کنیم.

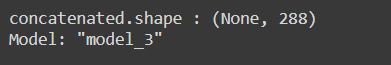


که کد کامل آن در در قسمت دوم فایل q3\_hw6\_cv.ipynb موجود می باشد. که پس از concatenate پاسخ آن (None, 288) نمایش داده شده است، لذا در لایه concatenate ابعاد خروجی از جمع لایه های بالا به دست می آید که از طرفی None را می توانیم 12 \* 12 نیز در نظر بگیریم. اما در کد زده شده همان None به ما برگردانده شد چون برای رفع ارور نیز از Dense استفاده می کنیم که در فایل q3\_hw6\_cv.ipynb نیز موجود می باشد و نتیجه ی آن مشخص است

حال در صورت تغییر تعداد فیلترها خواهیم داشت:

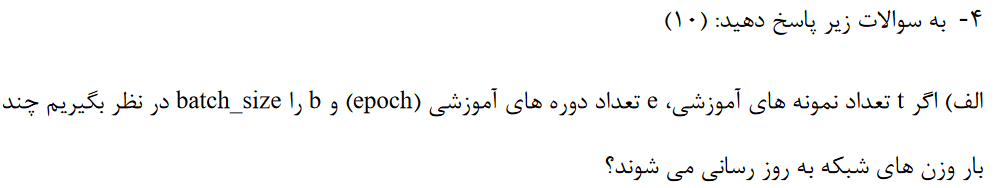
* لایه ترتیبی 1: (4, 4, 64)
* لایه ترتیبی 2: (8, 8, 128)
* لایه ترتیبی 3: (10, 10, 32)
* لایه موازی دیگر: (12, 12, 64)

همانطور که در بخش سوم فایل بالا دیده می شود ابعاد خروجی آن با اعمال GlobalMaxPooling یا GlobalAveragePooling هر کدام به صورت (None, 288) می شوند. زیرا پس از این کانولوشن، ما کانولوشن دیگری داریم که تعداد فیلترها را به سایز اولیه ی خود برمی گرداند.



لذا تغییری ایجاد نمی شود.

4)

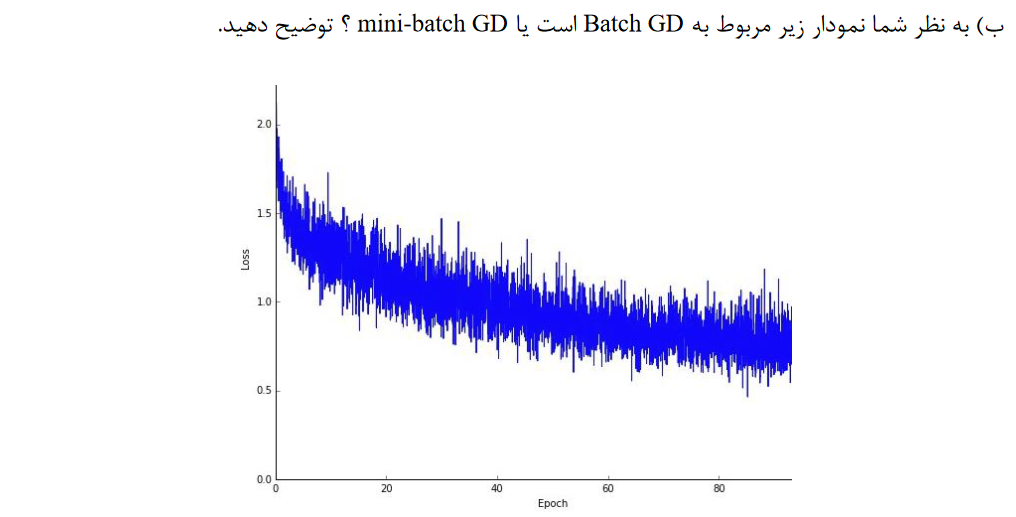


پاسخ:

در این صورت از فرمول های زیر استفاده می کنیم:







پاسخ:

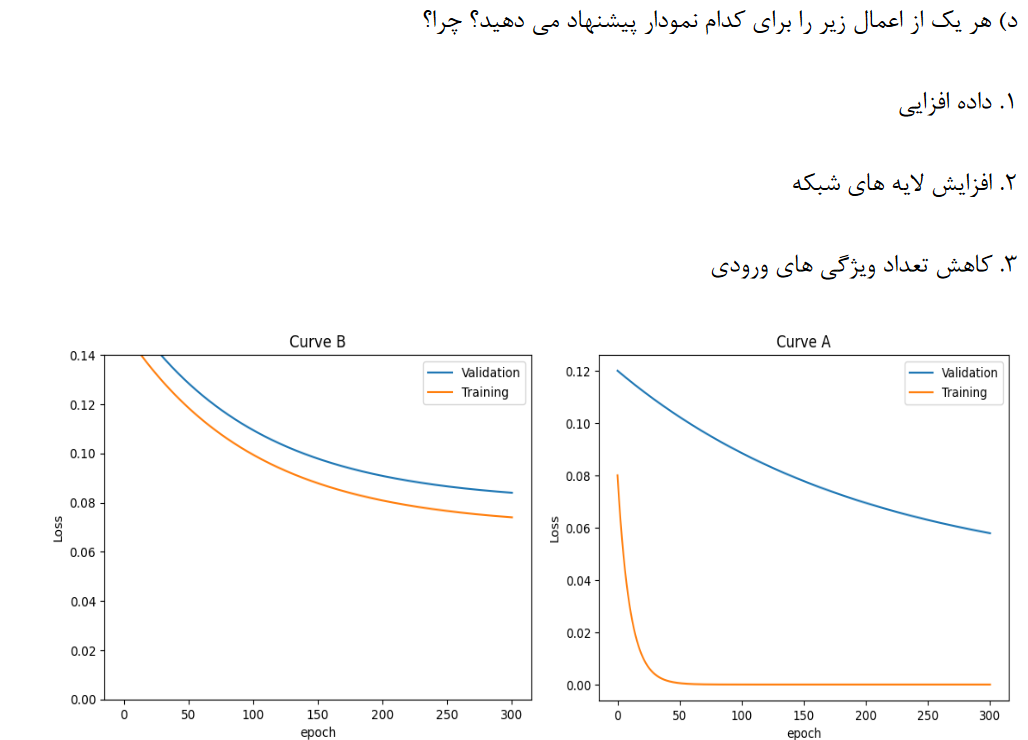
در Batch Gradient Descent، کل مجموعه داده‌های آموزشی در یک دسته (batch) قرار داده شده و استفاده می‌شود. در هر epoch، یک به‌روزرسانی برای وزن‌های شبکه انجام می‌شود. این به‌روزرسانی با استفاده از تمام داده‌های آموزشی انجام می‌شود.

در Mini-Batch Gradient Descent، مجموعه داده‌های آموزشی به دسته‌های کوچک‌تر (mini-batches) تقسیم می‌شوند. هر دسته شامل تعداد کمی از نمونه‌های آموزشی است. در هر epoch، وزن‌های شبکه با استفاده از هر mini-batch به‌روزرسانی می‌شوند حال به مقایسه ی نمودارها می پردازیم.

### **مقایسه نمودارها:**

1. Batch Gradient Descent:
   * نمودار خطا در طول زمان بسیار هموار است.
   * تغییرات هموار و پایدار است، زیرا به‌روزرسانی‌ها با استفاده از تمام داده‌ها انجام می‌شود.
   * همگرایی کندتری دارد زیرا به‌روزرسانی‌ها کمتر انجام می‌شود اما دقیق‌تر است.
2. Mini-Batch Gradient Descent:
   * نمودار خطا دارای نوسانات کوچکی است که به دلیل به‌روزرسانی‌های مکرر با mini-batch‌ها است.
   * همگرایی سریع‌تری نسبت به Batch GD دارد، زیرا به‌روزرسانی‌های بیشتری در هر epoch انجام می‌شود.
   * در نمودار آن تغییرات نسبتاً صاف اما با نوسانات کوچک است، زیرا به‌روزرسانی‌ها بر اساس mini-batch‌ها انجام می‌شود.

لذا با توجه ویژگی های بالا که نمودار هموار است اما نوسانات کوچک بسیار زیادی دارد، لذا این نمودار مربوط به mini-batch می باشد.



پاسخ:

Curve A → data augmentation, decrease input features

زیرا train با validation تفاوت بسیار زیادی دارد، لذا داده ها را زیاد می کنیم تا با تنوع بیشتری روبر باشیم و مدل ما نمونه ها را حفظ نکند و انقدر تفاوت فاحش بین این دو نمونه نداشته باشیم.

احتمالاً این مدل شما دچار بیش‌برازش (overfitting) شده است. این وضعیت به این معناست که مدل به خوبی داده‌های آموزش را یاد گرفته اما نمی‌تواند به خوبی روی داده‌های جدید (اعتبارسنجی یا تست) عمل کند.

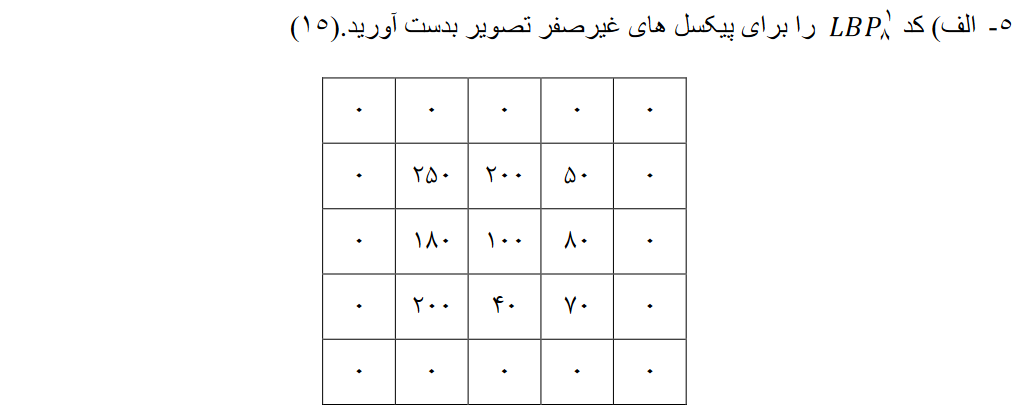
از کاهش ورودی ها نیز می توانیم برای جلوگیری از حفظ کردن داده ی train توسط مدل استفاده کنیم.

Curve B → Increasing Network Depth, data augmentation

افزایش تعداد لایه‌های شبکه به مدل اجازه می‌دهد تا ویژگی‌های پیچیده‌تر و سطح بالاتری را از داده‌ها استخراج کند. این تکنیک معمولاً برای مسائل پیچیده و بزرگتر مورد استفاده قرار می‌گیرد. که با این کار می توانیم loss این دو را به هم نزدیک کنیم.

این نمودار به این معناست که مدل شما به خوبی به داده‌های آموزش و داده‌های اعتبارسنجی آموزش داده شده است و مشکل بیش‌برازش یا کم‌برازش (underfitting) ندارد. با این حال، اگر هنوز به دقت مطلوب نرسیده‌اید، می‌توانید اقداماتی انجام دهید تا دقت مدل را بیشتر بهبود بخشید. در این شرایط، می‌توانید از تکنیک‌های زیر استفاده کنید:

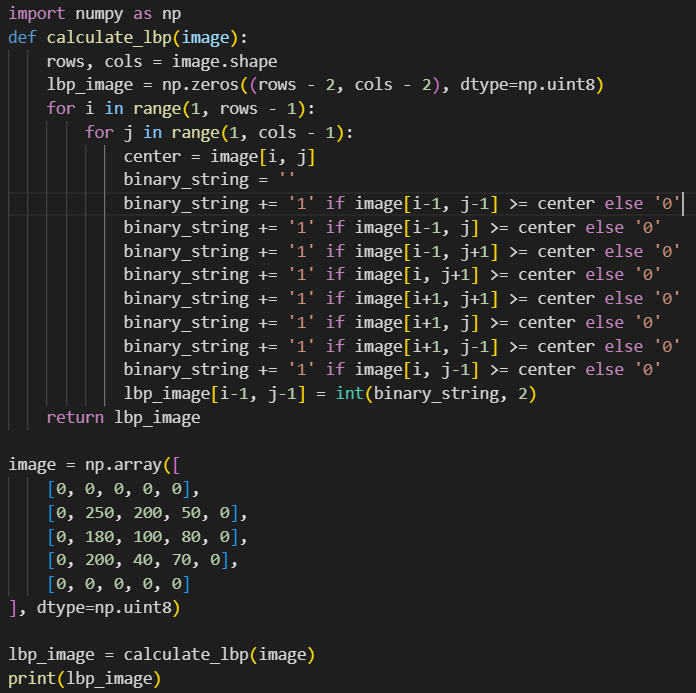
5)



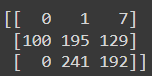
پاسخ:

در ابتدا مربع های سه در سه را در نظر می گیریم و برای هر پیکسل در صورتی که بزرگتر یا مساوی با مرکز باشد، یک و در غیر این صورت صفر می گذاریم:

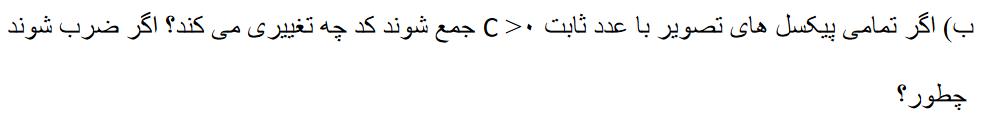
و در از پیکسل بالا سمت راست شروع می کنیم و به صورت ساعتگرد 0 و 1 ها را می نویسیم که یک عدد 8 بیتی می شود، و در آخر آن را به بیت تبدیل می کنیم که برای محاسبه ی آن می توانیم کد مربوط به LBP را می زنیم که به صورت زیر است:



که پاسخ آن به صورت زیر است:



که در کد ما همانطور که دیده می شود، بیت ها تبدیل به integer شده اند.

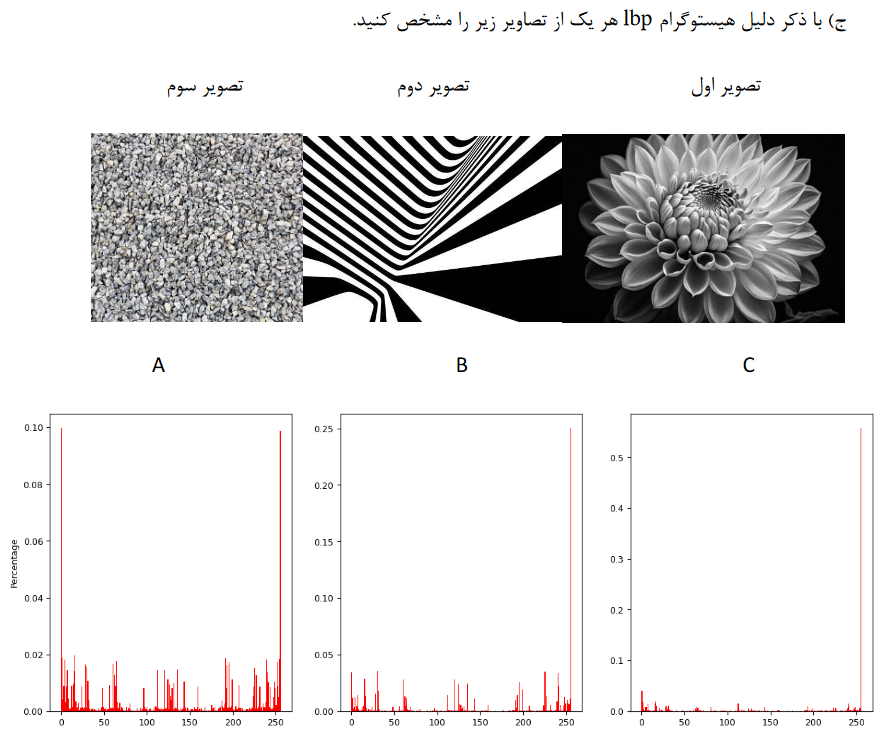


پاسخ:

از آنجا که کوچکی و بزرگی اعداد در صورت جمع شدن با یک عدد ثابت عوض نمی شود لذا بر روی صفر و یک ها نیز تاثیری نمی گذارد، بنابراین هیچ تغییری نمی کند و در صورت ضرب شدن نیز از آنجا که C > 0 می باشد، لذا باز هم جهت علامت عوض نمی شود و همین جواب بالا را خواهیم داشت.

A > D → A + b > D + b

A > D → (k > 0) → Ak > Dk



پاسخ:

می دانیم که LBP بافت یک تصویر را مورد ارزیابی قرار می دهد. در ابتدا می توانیم به سراغ تصویر دوم برویم که بیشتر تصویر را سفید یا مشکی یک دست تشکیل داده است. لذا بیشتر اطرافیان یک پیکسل مرکزی با آن برابر هستند، لذا مقادیر آنها همگی 1 می شود، پس مقدار 255 را خواهیم داشت، و از آنجایی که تنوع بافت در این عکس دیده نمی شود، بنابراین C نمودار مناسبی برای آن به شمار می رود.

در تصویر اول نیز از آنجا که بافت ها ساده هستند و بخش زیادی از تصویر را background مشکی گرفته است، لذا پراکندگی کمتری به نسبت تصویر سوم داریم و فراوانی مربوط به کد 255 در آن نیز نسبت به تصویر سوم بیشتر می باشد. لذا نمودار B مربوط به آن است.

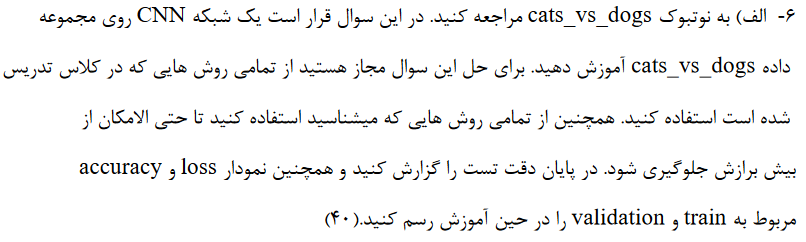
در تصویر سوم نیز چون در تصویر پراکندگی بافت بسیار زیادی دیده می شود و بافت مشخص و تکرارشونده ای در آن دیده نمی شود لذا نمودار A نیز مربوط به آن است.

A → تصویر سوم

B → تصویر اول

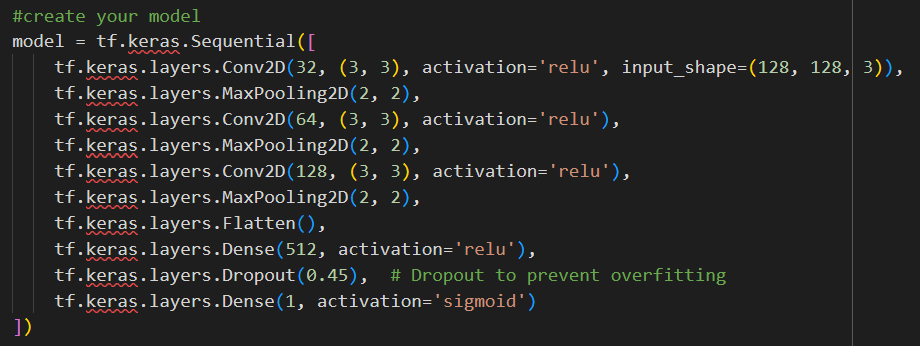
C → تصویر دوم

6)

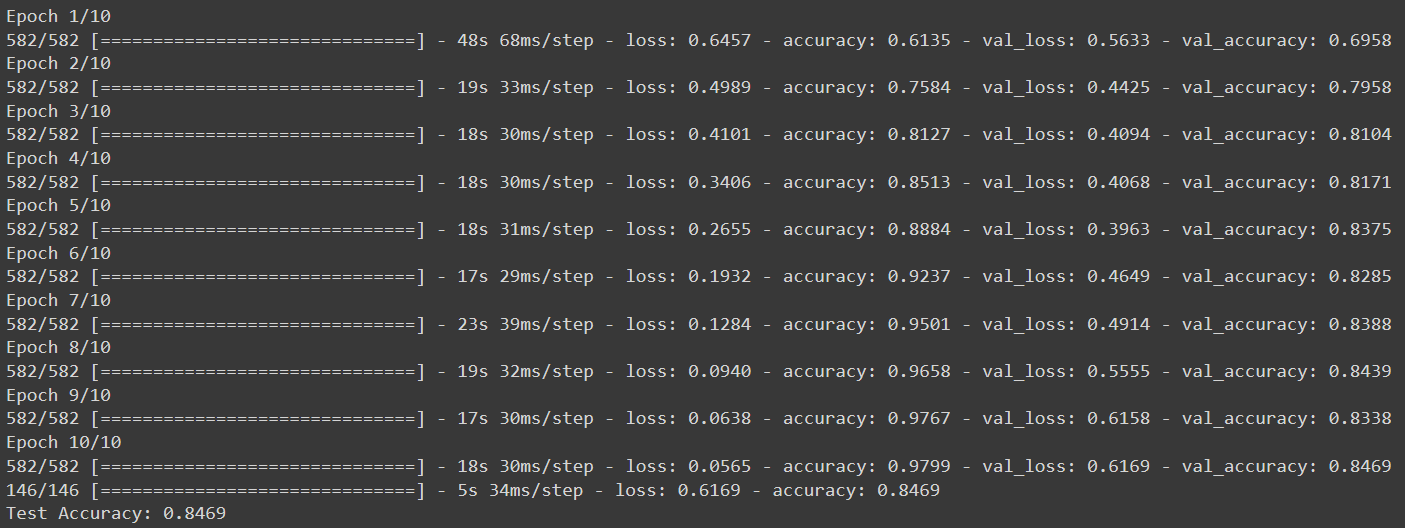


پاسخ:

شبکه ی CNN ای که برای این سوال در نظر گرفتم، به شرح زیر است و طرز عملکرد آنها دقیقا مثل همان اسلایدهای سر کلاس می باشد. شبکه ی ما به شرح زیر می باشد:



پس از استفاده از شبکه ی بالا و fit و predict کردن برای 10 epoch، دقت آن را به دست می آوریم و سپس plot آن را رسم می کنیم. از آنجا که در شکل اول train\_data، دقت خوبی دارند ولی test\_data از دقت خیلی کمتری برخوردار است لذا می توانیم بگوییم که مدل ما train\_data را حفظ کرده است و overfit رخ داده است که نتایج آن در زیر موجود است.





همانطور که دیده می شود دقت داده train 97 درصد ولی برای test\_data، هشتاد و چهار درصد می باشد.

لذا در قسمت دوم نوتبوک از روش های دیگری استفاده کردم که در زیر گفته ام:

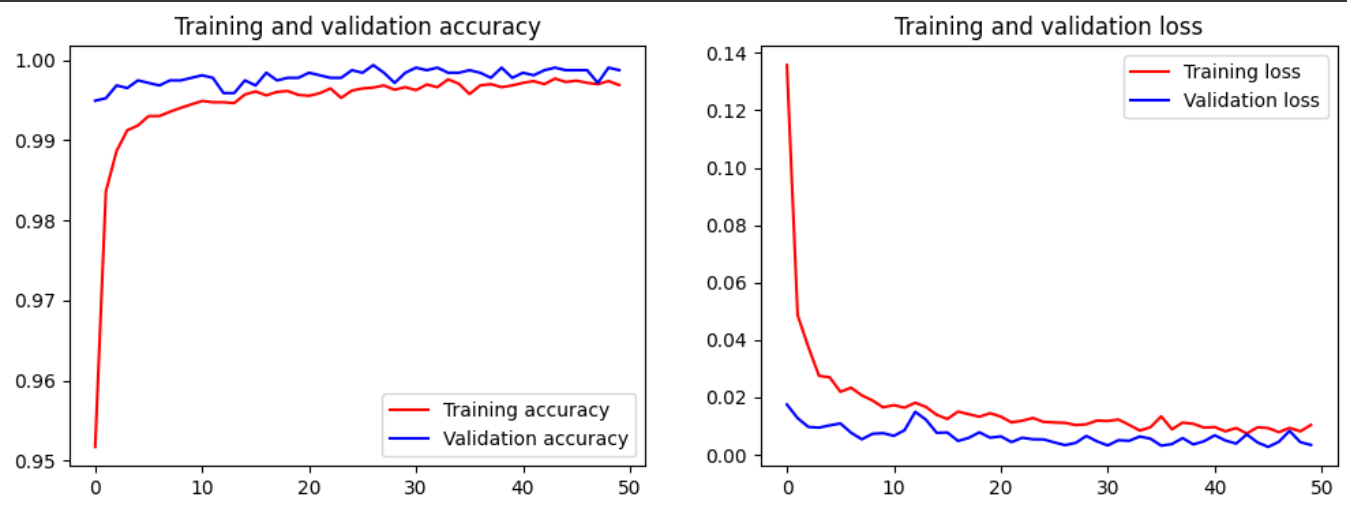
1. افزایش پیچیدگی مدل:
   * اضافه کردن لایه‌های بیشتر و نورون‌های بیشتر به مدل.
   * استفاده از لایه‌های کانولوشن (Convolutional Layers) که برای کار با تصاویر بسیار مناسب هستند.
2. داده‌افزایی:
   * استفاده از تکنیک‌های داده‌افزایی مانند چرخش، تغییر مقیاس، انتقال و وارونگی برای افزایش تنوع داده‌های آموزش.
3. منظم‌سازی:
   * استفاده از Dropout.
   * استفاده از L2 regularization.
4. تنظیم ابرپارامترها:
   * تنظیم تعداد اپوک‌ها (epochs) و اندازه دسته‌ها (batch size).

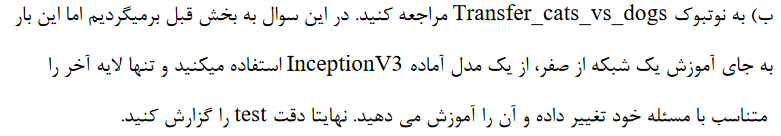
داده‌افزایی (ImageDataGenerator): با استفاده از تکنیک‌های داده‌افزایی، تنوع داده‌های آموزش افزایش یافته و به شبکه کمک می‌کند تا بهتر آموزش ببیند و از بیش‌برازش جلوگیری شود و تعداد epoch ها به 50 افزایش یافت که نتیجه ی آن را در زیر می بینیم:



که می بینیم برای train دقت 1 و برای test دقت بالای 99 درصد را داریم:

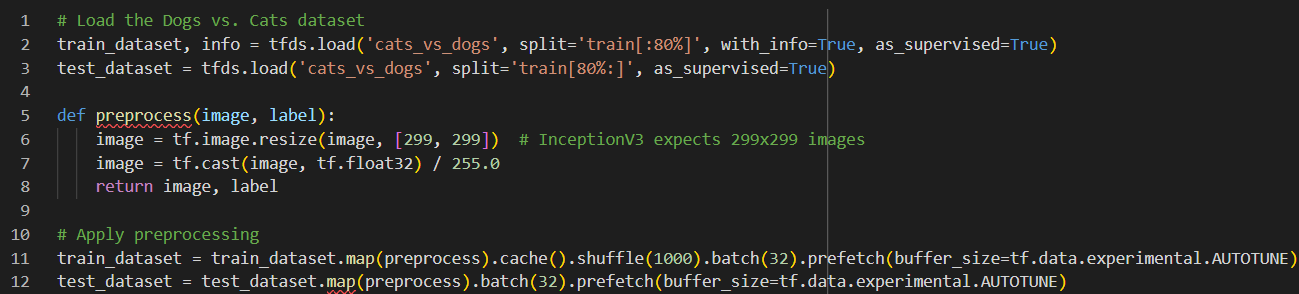
و نتیجه ی آن به این صورت شد که دقت بسیار خوبی دارد:





پاسخ:

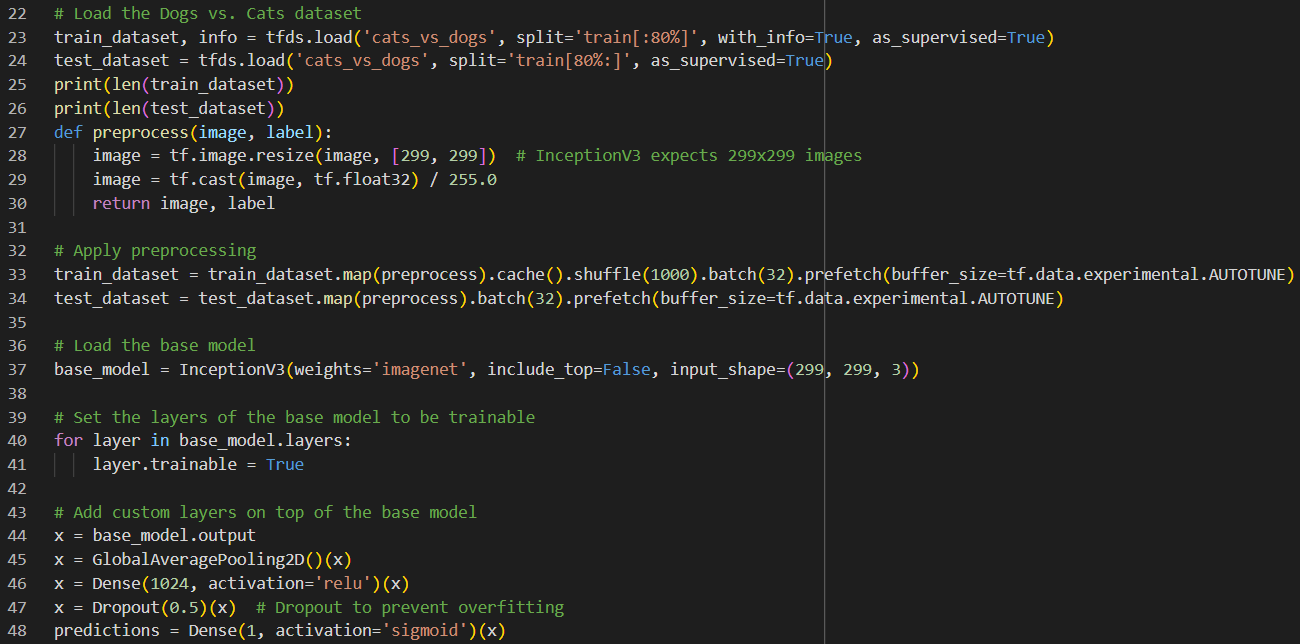
در ابتدا به preprocess داده هایی که از tdfs خوانده ایم، می پردازیم که به صورت زیر است:



که همانطور که دیده می شود در ابتدا همه ی تصاویر را resize می کنیم و سپس همه ی آنها را بین صفر و یک cast می کنیم و سپس train\_dataset و test\_dataset را با preprocess به دست می آوریم. بخشی از نمونه ها هم به شرح زیر است که در پایین دیده می شود.



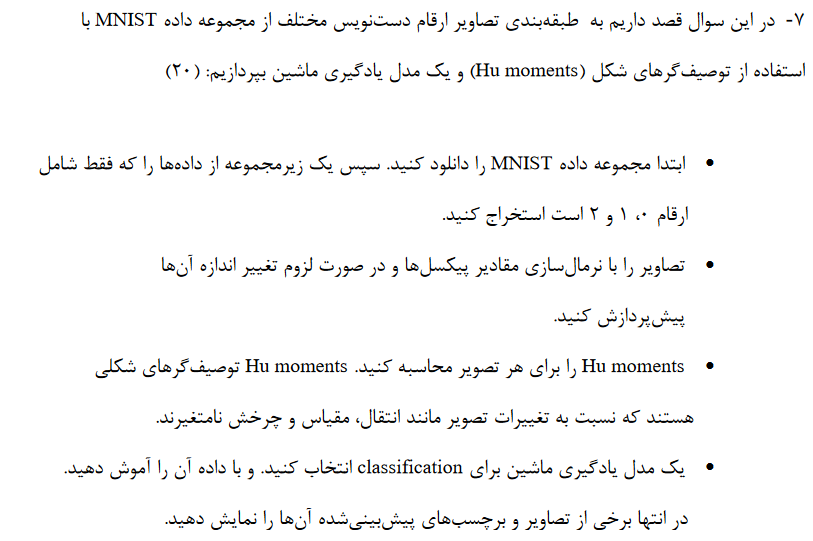
حال با استفاده از کد زیر اسکوپ های مختلف کد را میزنیم و با استفاده از imagenet، می توانیم InceptionV3 را پر کنیم وهمینطور که در کد زیر دیده می شود، هر layer را trainable می کنیم



در آخر نیز همانطور که در کد بالا دیده می شود با استفاده از لایه های مختلف GlobalAveragePooling2D، Dense, Droupout لایه های اخر را ویرایش می کنیم و با استفاده از activation های مختلف خروجی ها را به دست می آوریم.

همانطور که در نوتبوک دیده می شود نیز در اسکوپ های مختلف قسمت های خواسته شده ی سوال پر شده اند و در آخر plot مورد نظر کشیده شده است و دقت 86 درصد در آخر گزارش شده است که در نوتبوک نیز مشاهده می شود.

7)



پاسخ:

برای انجام این پروژه، باید مراحل زیر را طی کنیم:

دانلود مجموعه داده MNIST و استخراج ارقام 0، 1 و 2.

1. پیش‌پردازش تصاویر (نرمال‌سازی مقادیر پیکسل‌ها و در صورت نیاز تغییر اندازه آنها).
2. محاسبه Hu moments برای هر تصویر.
3. انتخاب و آموزش مدل یادگیری ماشین برای طبقه‌بندی.
4. نمایش برخی از تصاویر و برچسب‌های پیش‌بینی شده.

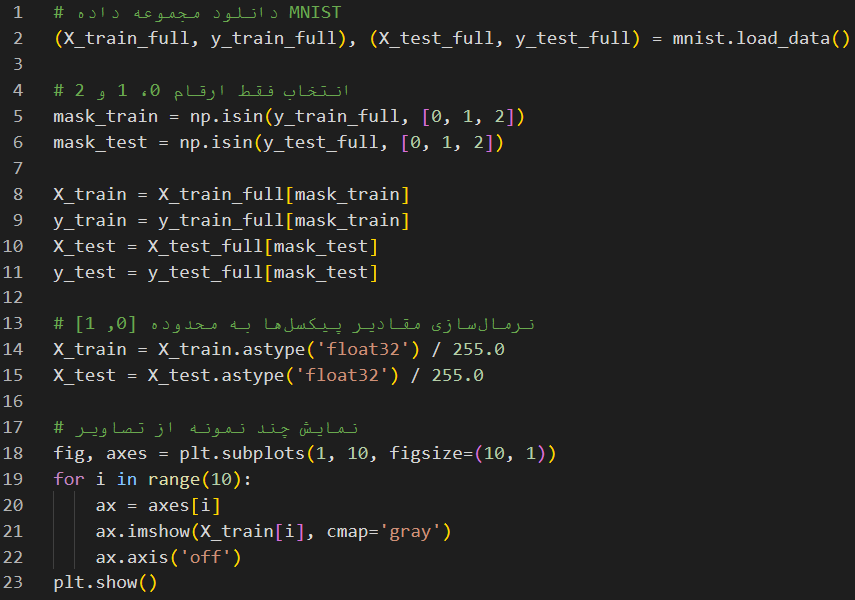
بیایید این مراحل را به ترتیب انجام دهیم:

### **مرحله 1: دانلود و استخراج داده‌ها**

ابتدا مجموعه داده MNIST را دانلود می‌کنیم و فقط ارقام 0، 1 و 2 را استخراج می‌کنیم که چند نمونه ی اول آن نیز در نوتبوک با استفاده از plt نمایش داده شده اند.

### **مرحله 2: پیش‌پردازش تصاویر**

نرمال‌سازی مقادیر پیکسل‌ها به مقادیر بین 0 و 1 که برای این کار می دانیم که 0 تا 255 را به 0 تا 1 تبدیل کنیم و سایز همه ی تصاویر را یکی کنیم تا برای process کردن به چالش برنخوریم.



کد مربوط به پارت اول و دوم اینجا موجود است

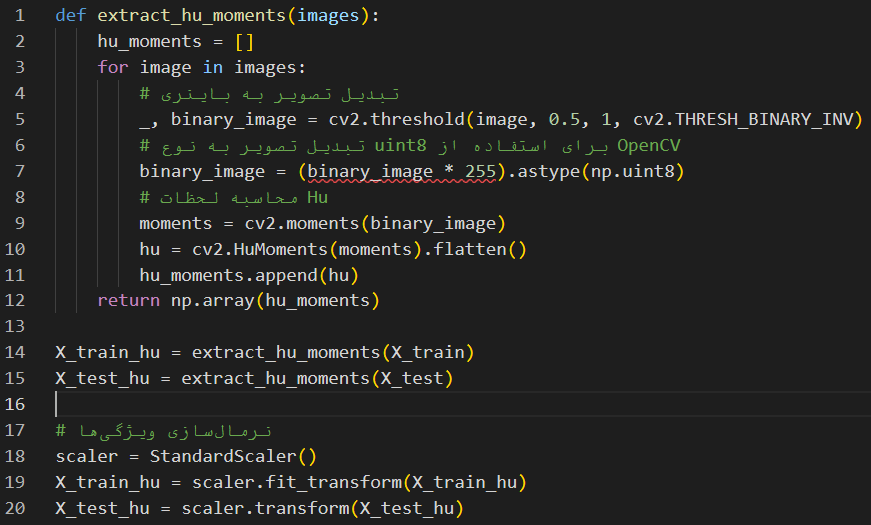


در اینجا نیز چند نمونه فقط از دیتاهای 0 و 1 و 2 می بینیم.

### **مرحله 3: محاسبه Hu moments**

برای محاسبه Hu moments، ابتدا باید تصاویر را به فرمت مناسب تبدیل کنیم و برای محاسبه hu نیز از cv2 استفاده می کنیم که این الگوریتم را در خود دارد.

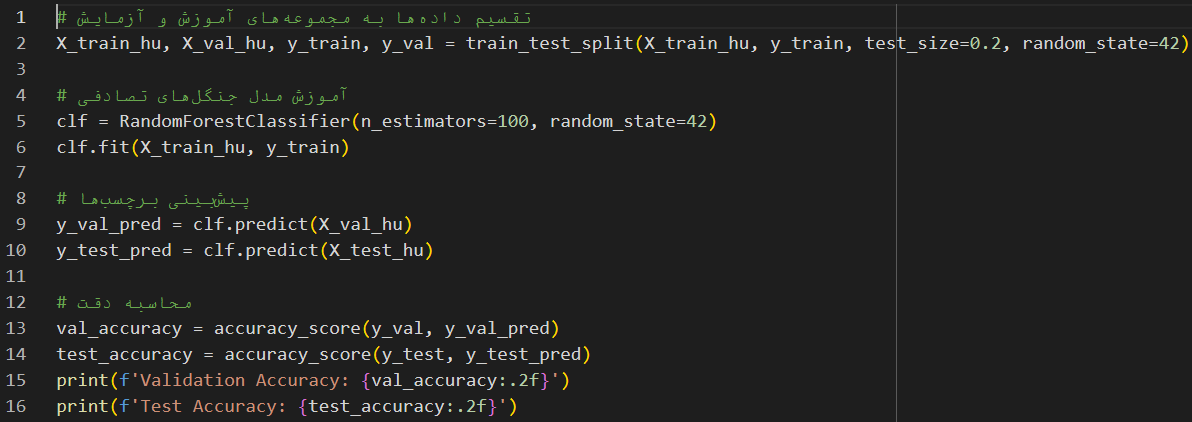
کد آن در زیر موجود است:



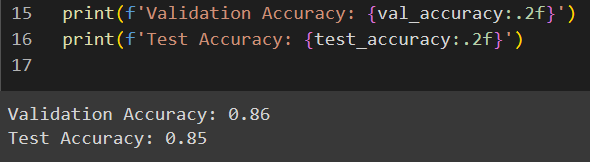
حال به مرحله ی بعدی می رسیم:

### **مرحله 4: انتخاب و آموزش مدل یادگیری ماشین**

ما از یک مدل طبقه‌بندی ساده مانند RandomForestClassifier استفاده می‌کنیم که کتابخانه ی آماده ی آن نیز در پایتون وجود دارد و دیتاهای خود را به train و test تقسیم می کنیم و سپس آن را fit و predict می کنیم و نتیجه ی accuracy آن نیز پایینتر موجود است.



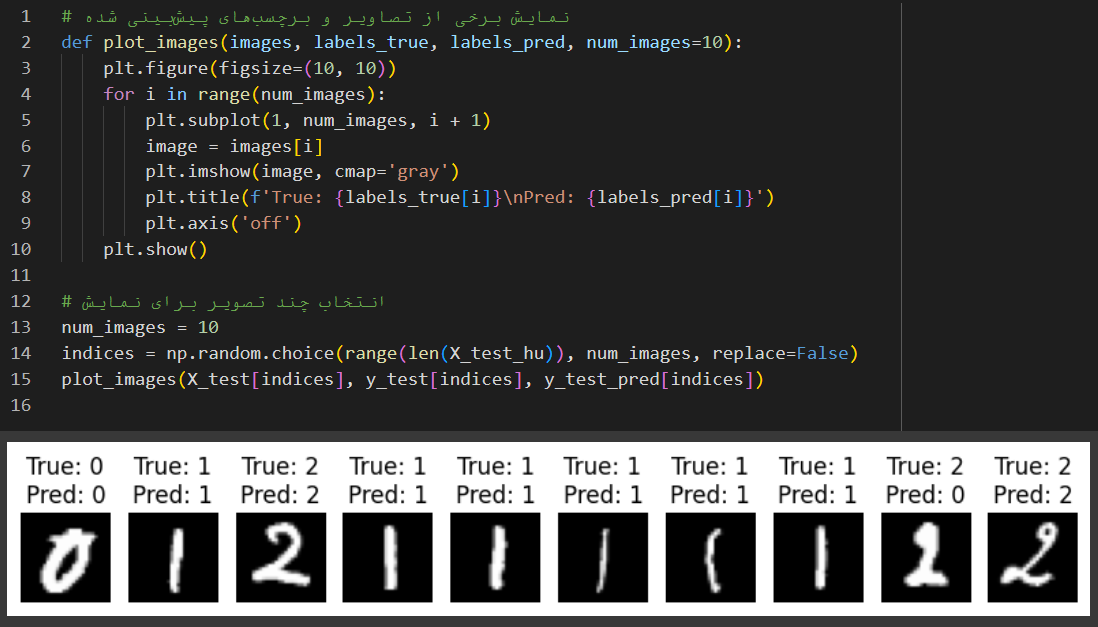
نتیجه آن نیز به صورت زیر است:



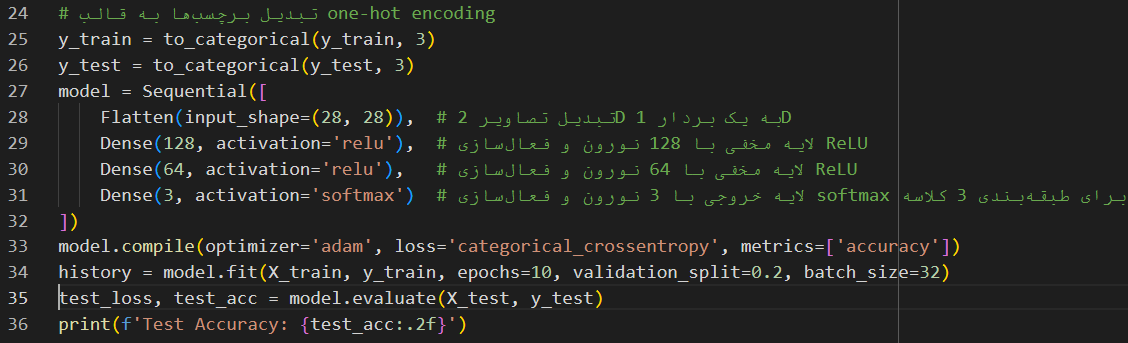
### **مرحله 5: نمایش تصاویر و برچسب‌های پیش‌بینی شده**

در این قسمت آخر نیز باید برخی از تصاویر و برچسب‌های پیش‌بینی شده آنها را نمایش می‌دهیم که به صورت زیر می باشد:

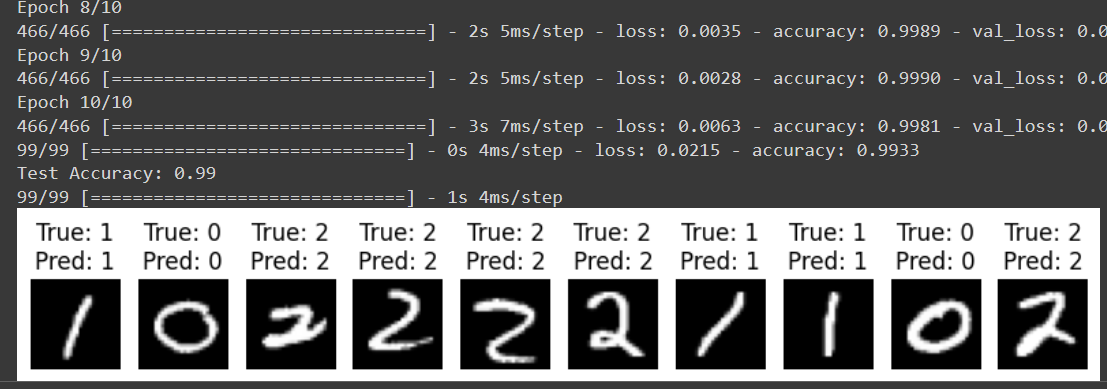
و نتیجه ی آن نیز به صورت زیر است:



که برای این ده نمونه تنها یک نمونه اشتباه می باشد و accuracy ما بالاتر از 85 درصد است. در ابتدا از Svm استفاده کردم و برای حالت های مختلف از gridCV استفاده کردم اما بسیار طول کشید تا نتیجه ی حالت های مختلف را ببینم و برای حالت های مختلف زیر Accuracy 65 تا 80 را به دست آوردم که بهترین آن مربوط به kernel = 'rbf' بود. اما به دلیل accuracy کم تصمیم گرفتم از random forest استفاده کنم. اما متوجه شدم با شبکه های عصبی بسیار راحت تر می توان accuracy و دقت را بالا برد که داریم:



و با توجه به شبکه ی عصبی بالا، دقت به شرح زیر است که بالای 99 درصد می باشد:



که این قسمت در بخش آخر نوتبوک مربوط به mnist نشان داده شده است.