

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

ﮔﺰﺍﺭﺵکار ﺁﺯﻣﺎﻳﺶ ششم

ﺁﺯﻣﺎﻳﺸﮕﺎﻩ مقدمه ای بر ﻫﻮﺵ ﻣﺤﺎﺳﺒﺎﺗﻲ

پیاده سازی CNN

نگارش

ارشیا اسمعیل طهرانی

علی بابالو

پویا ابراهیمی

استاد راهنما

سرکار خانم موسوی

آذر ماه 1401

# چکیده

در این آزمایش هدف پیاده سازی یک شبکه کانولوشنی روی(سی فار 10) است. شبكه عصبي كانولوشني از پركاربردترين شبكه هاي عصبي برای پردازش تصوير مي باشد. عملكرد بسياري از حملات نيز بر روي اين شبكه عصبي بررسي مي شوند .

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

**پیش گزارش....................................................................................................................................1**

ساختار یک شبکه عصبی CNN.................................................................................................................1

لایه کانوولوشنی...............................................................................................................................2

لایه ادغام...........................................................................................................................................2

لایه کاملا متصل.............................................................................................................................3

**شرح آزمایش..................................................................................................................................3**

معماری VGG.............................................................................................................................................4

Dropout Regularization..................................................................................................................4

تنوع Dropout Regularization.......................................................................................................5

Batch Normalization.........................................................................................................................5

Adam Learning Rate... ....................................................................................................................5

محیط Python..............................................................................................................................................6

دیتاست mnist...............................................................................................................................6

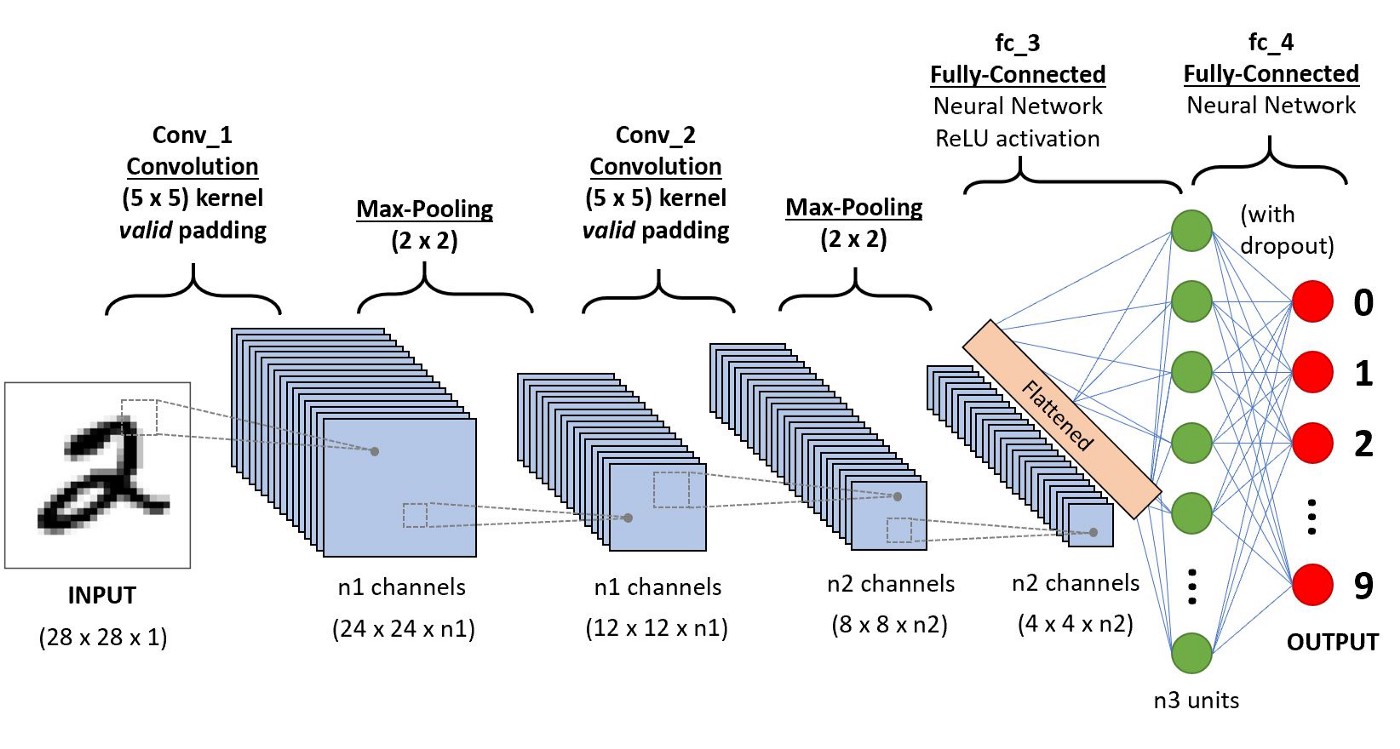
**تمارین........................................................................................................................................11**

دیتاست CIFAR-10 **.......................................................................................................**11

تمرین۲..........................................................................................................................................................21

# پیش گزارش

## ساختار یک شبکه عصبی CNN.

شبكه كانولوشني از لايه هاي اصلي زير تشكيل شده است:

شکل 1) نمونه ای شبکه عصبی کانوولوشنی

• لایه کانولوشنی

• لایه ادغام

• لایه کاملا متصل

### لايه كانولوشني:

شکل 2) عملگر کانوولوشن

در اين لايه عمل كانولوشن بر روي داده ورودي و با استفاده از تعدادي فيلتر انجام مي شود. براي محاسبه هر درايه خروجي، ماتريس فيلتر بر روي ماتريس ورودي لغزانده مي شود. عمل كانولوشن به اين صورت تعريف مي شود كه ابتدا اولين عنصر فيلتر بر روي اولين عنصر ماتريس ورودي قرار مي گيرد. سپس مجموع ضرب درايه هاي متناظر فيلتر با درايه هاي متناظز ماتريس ورودي محاسبه مي شود. در نهايت فيلتر بر روي تصوير ورودي به اندازه پارامتر از پيش تعيين شده s يه جلو برده مي شود. با تكرار اين مراحل ماتريس خروجي محاسبه مي شود.

### لايه ادغام:

يكي ديگر از لايه هاي شبكه كانولوشني لايه ادغام است. اين لايه هيچ پارامتر آموزشي ندارد. هدف اين لايه كاهش ابعاد ماتريس ورودي ورودي و همزمان حفظ اطلاعات ارزشمند ورودي است. در اين لايه ابتدا ماتريس با ابعاد از پيش تعيين شده k در نظر گرفته مي شود. اين پارامتر معمولا برابر با ۲ در نظر گرفته مي شود. سپس با لغزاندن فيلتر بر روي ورودي اندازه ماتريس ورودي كاهش مي يابد. يك نمونه از عملگرهايي كه براي نمونه برداري در اين لايه استفاده مي شود، عملگر بيشينه است. در اين حالت ماتريس از پيش تعيين شده بر روي داده ورودي لغزانده مي شود و تنها بيشينه عناصري كه در هر بخش قرار مي گيرند را به عنوان خروجي در نظر مي گيرد.

### لايه كاملا متصل:

در اين لايه يك شبكه عصبي كاملا متصل قرار گرفته است. در اين لايه هدف مرتبط كردن ماتريس نهايي با خروجي هاي نهايي شبكه است. وزن هاي شبكه كاملا متصل از طريق پس انتشار خطا بدست مي آيد.

# شرح آزمایش

CIFAR مخفف عبارت Canadian Institute for Advanced Research است و مجموعه داده CIFAR-10 همراه با مجموعه داده CIFAR-100 توسط محققان موسسه CIFAR توسعه یافته است.

مجموعه داده شامل 60000 عکس رنگی 32×32 پیکسل از اشیاء از 10 کلاس، مانند قورباغه، پرندگان، گربه‌ها، کشتی‌ها و غیره است. برچسب‌های کلاس و مقادیر عدد صحیح مرتبط با آنها در زیر فهرست شده‌اند.

1. هواپیما
2. خودرو
3. پرنده
4. گربه
5. آهو
6. سگ
7. قورباغه
8. اسب
9. کشتی
10. کامیون

این تصاویر بسیار کوچک هستند، بسیار کوچکتر از یک عکس معمولی، و مجموعه داده برای تحقیقات بینایی کامپیوتری در نظر گرفته شده است.

CIFAR-10 یک مجموعه داده کاملاً درک شده است که به طور گسترده برای محک زدن الگوریتم‌های بینایی رایانه در یادگیری ماشین استفاده می‌شود. مشکل حل شده است." دستیابی به دقت طبقه بندی 80 درصد نسبتاً ساده است. عملکرد برتر در مورد مشکل با یادگیری عمیق شبکه های عصبی کانولوشن با دقت طبقه بندی بالای 90٪ در مجموعه داده آزمایشی به دست می آید.

اکنون می‌توانیم یک مدل پایه برای مجموعه داده CIFAR-10 بررسی کنیم.

## معماری VGG

یک مدل پایه حداقل عملکرد مدل را ایجاد می کند که همه مدل های دیگر ما را می توان با آن مقایسه کرد و یک معماری مدلی که می توانیم به عنوان مبنای مطالعه و بهبود استفاده کنیم.

یک نقطه شروع خوب، اصول کلی معماری مدل های VGG است. اینها نقاط شروع خوبی هستند زیرا در رقابت ILSVRC 2014 به عملکرد برتر دست یافتند و به دلیل اینکه ساختار مدولار معماری قابل درک و پیاده سازی آسان است.

این معماری شامل انباشته شدن لایه‌های کانولوشن با فیلترهای کوچک 3×3 و به دنبال آن یک لایه ترکیبی حداکثر است. این لایه‌ها با هم یک بلوک را تشکیل می‌دهند و این بلوک‌ها می‌توانند در جایی که تعداد فیلترها در هر بلوک با عمق شبکه افزایش می‌یابد، تکرار شوند، مانند 32، 64، 128 و 256 برای چهار بلوک اول مدل. بالشتک روی لایه های کانولوشن برای اطمینان از مطابقت ارتفاع و عرض نقشه های ویژگی خروجی با ورودی ها استفاده می شود.

ما این معماری را در مسئله CIFAR-10 بررسی می کنیم و مدلی را با این معماری با بلوک های 1، 2 و 3 مقایسه می کنیم.

## Dropout regularization از جمله راه های بهبود مدل

Dropout یک تکنیک ساده است که به طور تصادفی گره ها را از شبکه خارج می کند. این یک اثر منظم کننده دارد زیرا گره های باقی مانده باید برای برداشتن سستی گره های حذف شده سازگار شوند.

Dropout را می توان با افزودن لایه های Dropout جدید به مدل اضافه کرد، جایی که مقدار گره های حذف شده به عنوان یک پارامتر مشخص می شود. الگوهای زیادی برای افزودن Dropout به یک مدل وجود دارد، از نظر اینکه در کجای مدل باید لایه‌ها را اضافه کرد و از چه تعداد حذفی استفاده کرد.

در این حالت، لایه‌های Dropout را بعد از هر لایه جمع‌آوری حداکثر و بعد از لایه کاملاً متصل اضافه می‌کنیم و از نرخ خروج ثابت 20% استفاده می‌کنیم (به عنوان مثال، 80٪ گره‌ها را حفظ می‌کنیم).

## تنوع dropout regularization

یک تغییر این است که میزان dropout از 20% به 25% یا 30% افزایش پیدا کند. یکی دیگر از تغییراتی که ممکن است جالب باشد، استفاده از الگوی افزایش dropout از 20% برای بلوک اول، 30% برای بلوک دوم و به همین ترتیب به 50% در لایه کاملا متصل در بخش طبقه‌بندی‌کننده مدل است.

این نوع افزایش dropout با عمق مدل یک الگوی معمولی است. موثر است زیرا لایه‌های عمیق مدل را مجبور می‌کند تا بیش از لایه‌های نزدیک به ورودی را منظم کنند.

## استفاده از Batch Normalization

نرمال سازی دسته ای در تلاش برای تثبیت یادگیری و شاید تسریع روند یادگیری اضافه شده است. برای جبران این شتاب، از الگوی dropout فزاینده استفاده می شود.

در ادامه به دو راهکار دیگر که برای تقویت این مدل استفاده شده است نیز پرداخته شده است؛

## استفاده از Adam learning rate

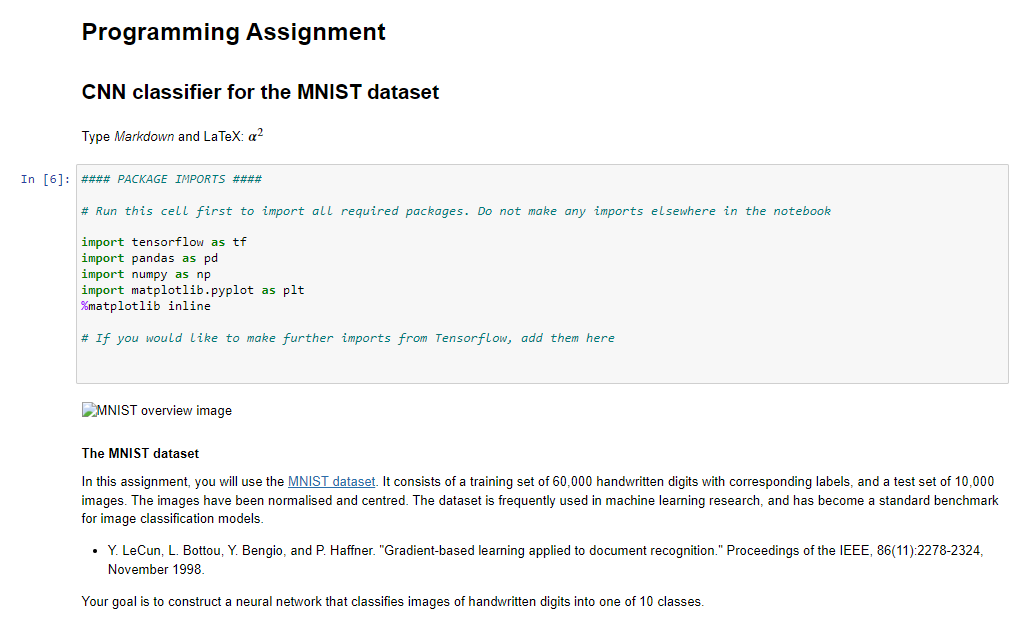
بهینه‌سازی آدام یک روش stochastic gradient descent است که مبتنی بر تخمین تطبیقی ممان‌های مرتبه اول و مرتبه دوم است. به خاطر ذات تطبیقی یا adaptive بسیار مورد استفاده قرار می‌گیرد به گونه ای که به گفته کینگما و همکاران، 2014، این روش "از نظر محاسباتی کارآمد است، نیاز به حافظه کمی دارد، نسبت به مقیاس مجدد مورب گرادیان ها ثابت است، و برای مسائلی که از نظر داده/پارامترها بزرگ هستند، مناسب است."

## محیط Python

به طور کلی مراحل زیر در پیاده سازی این پروژه طی شده است؛

## 2-1-2 دیتاست mnist

در این قسمت ازمایش با استفاده از کتابخانه tensorflow و با استفاده از دیتاست mnist که شامل ۷۰۰۰۰ تصویر از اعداد۰-۹ به صورت دست نوشت است قرار است اعداد دست نوشته شده را با استفاده از یک شبکه CNN پیش­بینی کنیم. برای اینکار ابتدا کتابخانه ها و دیتاست را import می­کنیم.

شکل۳) ایمپورت کردن کتابخانه ها

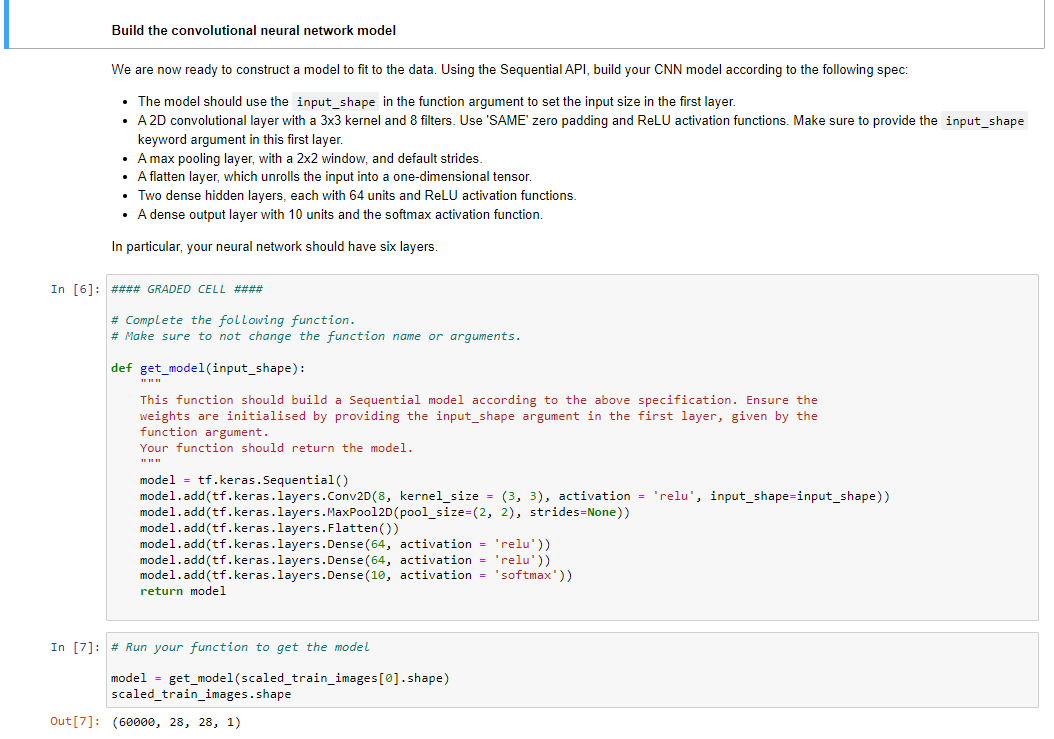
شکل۴) لود کردن دیتاست و اسکیل کردن آن بین۰-۱

بعد از جداکردن دیتا ست به دوبخش test و train دیتای ترین را بین دوعدد ۰-۱ اسکیل می­کنیم، برای اینکار نیاز است که کل دیتاست را تقسیم بر ۲۵۵ بکنیم به این علت که رنگ ها به ۰-۲۵۵ تقسیم شده اند و برای اسکیل کردن آن باید بر ۲۵۵ تقسیم کنیم.

سپس به ساختن مدل می­پردازیم. برای بیلد کردن مدلمان با توجه به دستور کار از sequential API استفاده می­کنیم که این شامل ۶ لایه (۱لایه conv2D با ۸ فیلتر و سایز ۳\*۳ با تابع فعال ساز relu ، یک لایه MaxPooling2D، یک لایه flatten برای یک بعدی کردن داده ها و ۲ لایه Dense با ۶۴ نورون و تابع فعال ساز relu و لایه آخر با فعال ساز softmax )می­شود.

بعد از تعریف مدلمان باید ان را compile کنیم که برای اینکار از تابع compile خود کتابخانه tensorflow استفاده می­کنیم که در آن از اپتیمایز adam ، metrics = accuracy

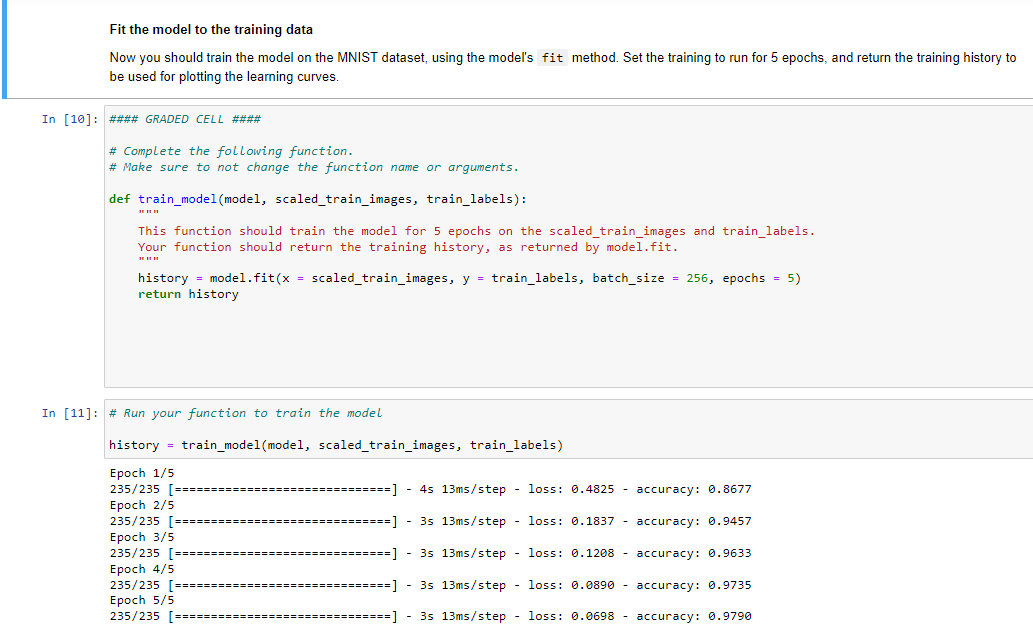
loss = sparse\_categorical\_crossentropyاستفاده کردیم.

شکل۵) بیلد کردن مدل شبکه

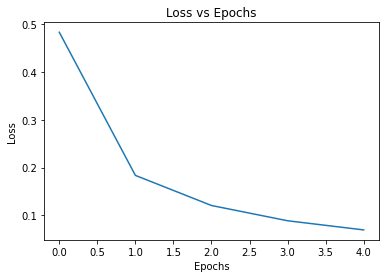
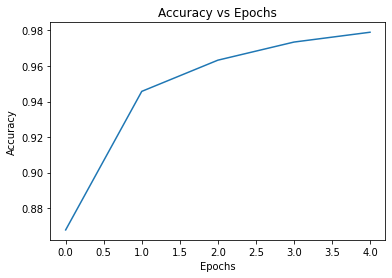
**Graphical user interface, text, application, email

Description automatically generated**شکل۶) کامپایل کردن شبکه

پس از آنکه شبکه با موفقیت ساخته شد نوبت به آموزش آن با استفاده از داده های آموزش است که برای اینکار از متد fit در sequential API استفاده می­کنیم که ارگیومنت های ورودی آن علاوه بر داده های آموزش شامل ۵ epoch و batch\_size برابر با ۲۵۶ می­باشد. پس از آموزش شبکه مشاهده می­کنیم که به دقت ۹۸درصدی می­رسیم

شکل۷) آموزش شبکه با متد fit

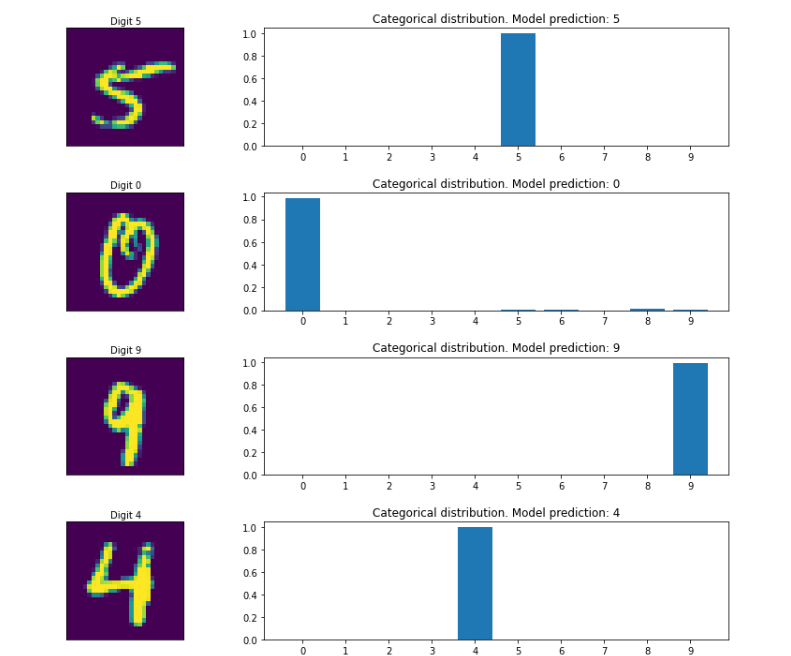
در شکل زیر نیز میزان دقت و خطا بر حسب epoch را مشاهده می­کنیم که شبکه به دقت ۹۸ درصدی بر روی داده های تست رسیده است.



شکل۸) دقت و خطا بر حسب epoch ها

در انتها نیز با استفاده از متد evaluate میزان دقت شبکه اموزش داده شده بر روی داده های تست را امتحان می­کنیم.

شکل۹) تست شبکه آموزش داده شده با دیتا تست که به دقت ۹۷.۸ درصدی منجر شد.

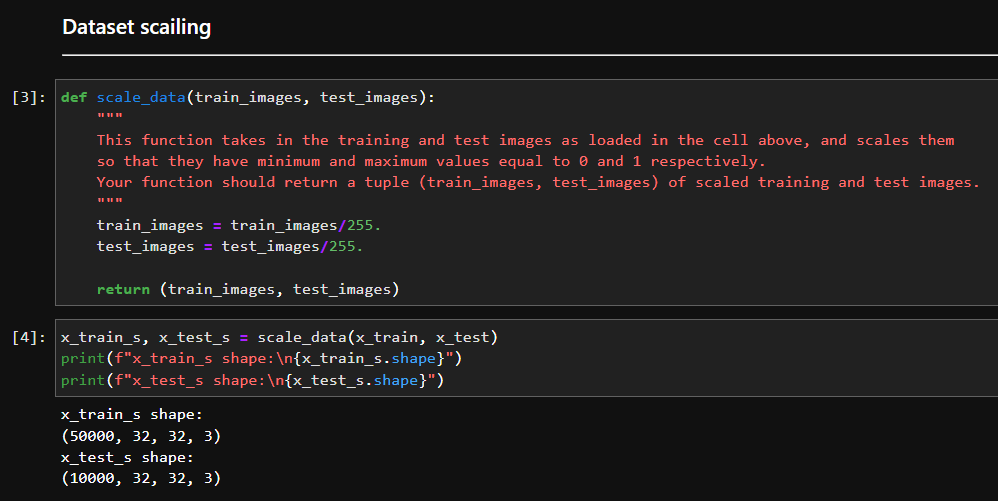
شکل ۱۰) مشاهده مقدار خروجی داده شده.

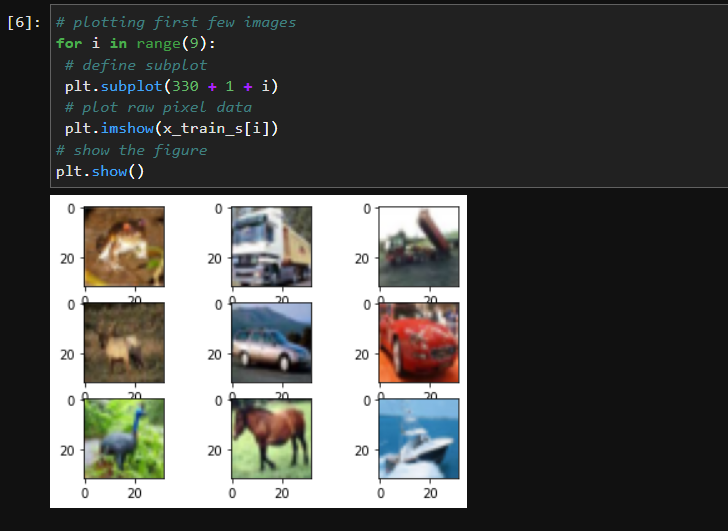
**3-1 تمرین**

**3-1-1 دیتاست CIFAR-10**

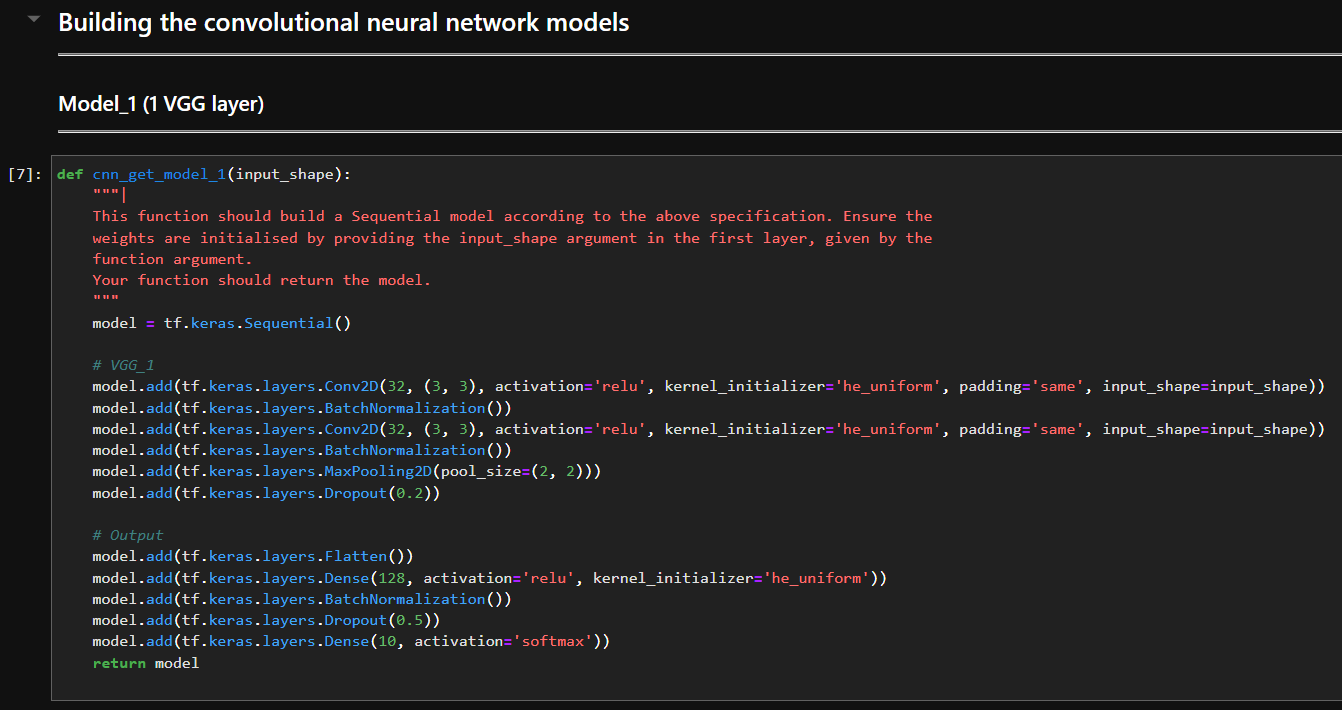
توضیحات مربوط به دیتاست cifar-10 در پیش گزارش داده شده است پس مستقیم به توضیح کد می­پردازیم.

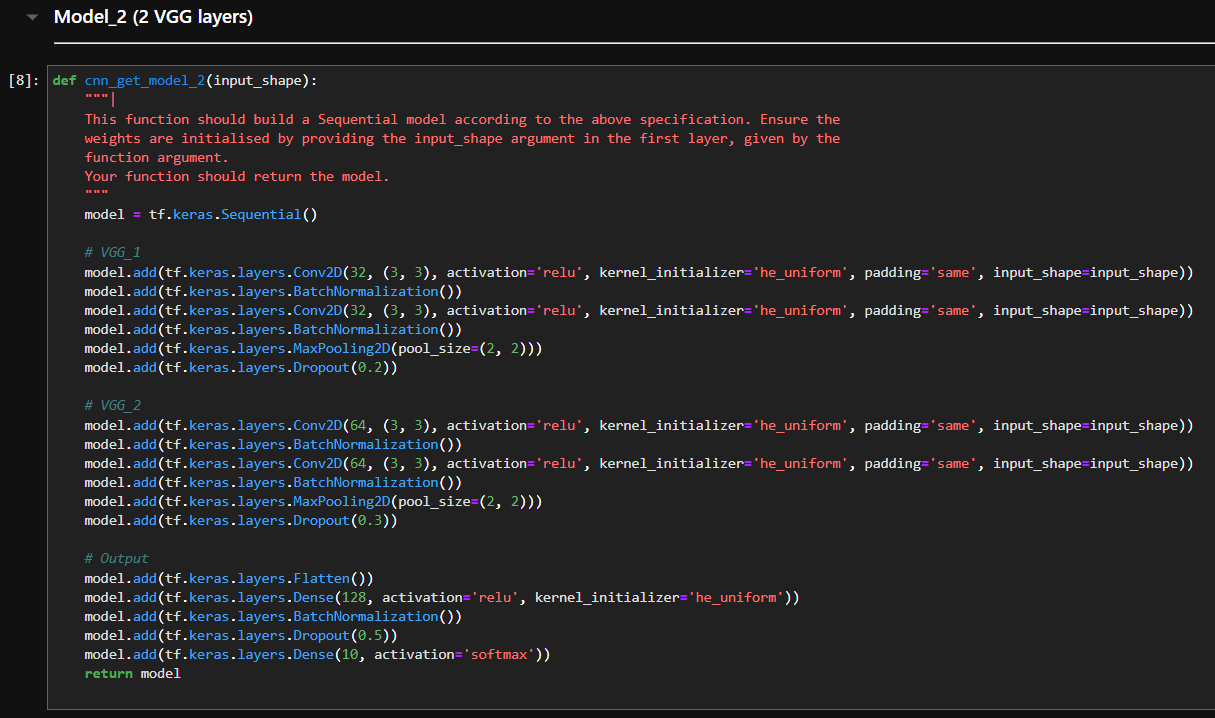
در ابتدا مانند دیتاست mnist کتابخانه هارا ایمپورت می­کنیم و دیتاست را ادد می­کنیم.

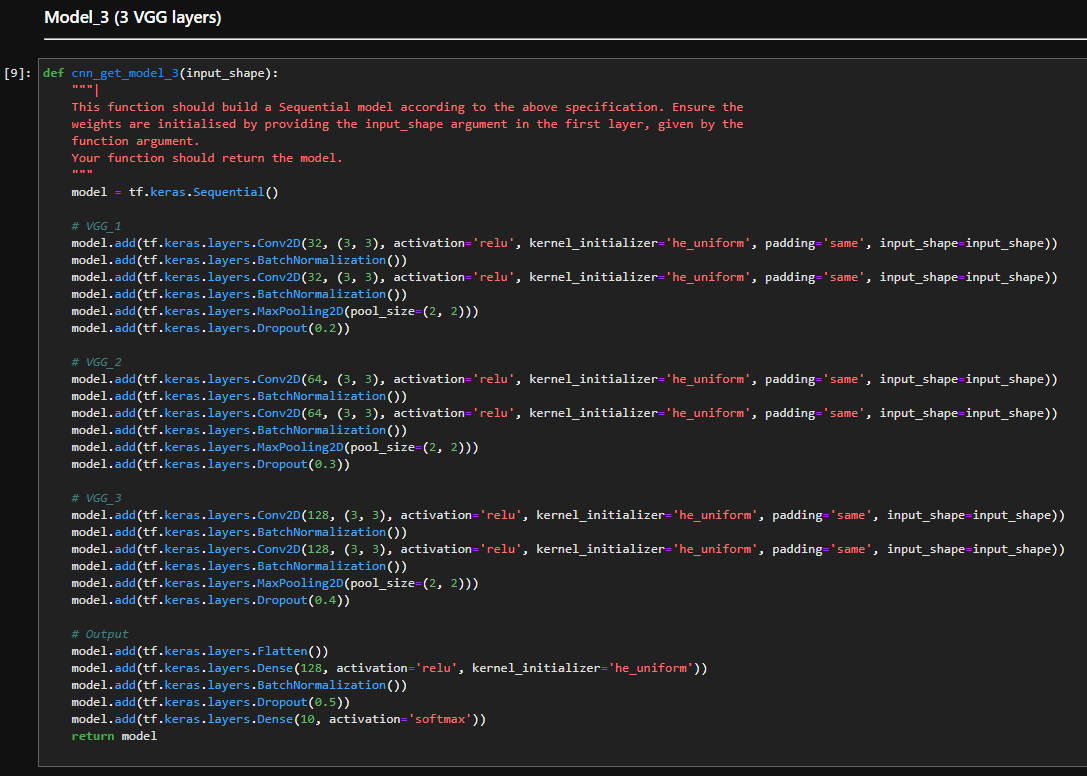
شکل۱۱) ایمپورت کردن دیتاست و کتابخانه ها

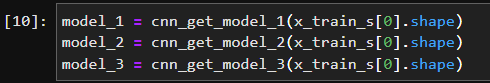
شکل۱۲) مشاهده چند نمونه از دیتاهای cifar-10

در مرحله بعدی به بیلد کردن شبکه مشغول می­شویم. در اینجا نیز از API sequential استفاده می­کنیم و لایه های مختلف را به شبکه اضافه می­کنیم. برای اینکه تاثیر افزایش لایه ها را متوجه بشویم ۳ مدل استفاده می­کنیم که هر کدام به اندازه شمارشان VGG دارند. در مدل۱ علاوه بر نرمالایزکردن بچ ها و لایه maxpooling از ۲ لایه conv2D استفاده کردیم که به تعداد ۳۲ فیلتر دارد و سایز آن ۳\*۳ می­باشد. در لایه مخفی اول نیز از ۱۲۸ نورون با تابع فعال ساز relu و لایه اخر نیز از تابع فعال سازsoftmax استفاده کردیم. در مدل ۲ علاوه بر VGG استفاده شد مدل ۱ از ۲لایه conv2D با ۶۴ فیلتر و تابع فعال ساز relu استفاده شده است و در مدل سوم نیز علاوه بر استفاده از مدل اول از ۲ لایه conv2D با ۱۲۸ فیلتر و با همان تابع فعال ساز relu استفاده کردیم که در ادامه تاثیر این افزایش لایه هارا مشاهده می­کنیم.

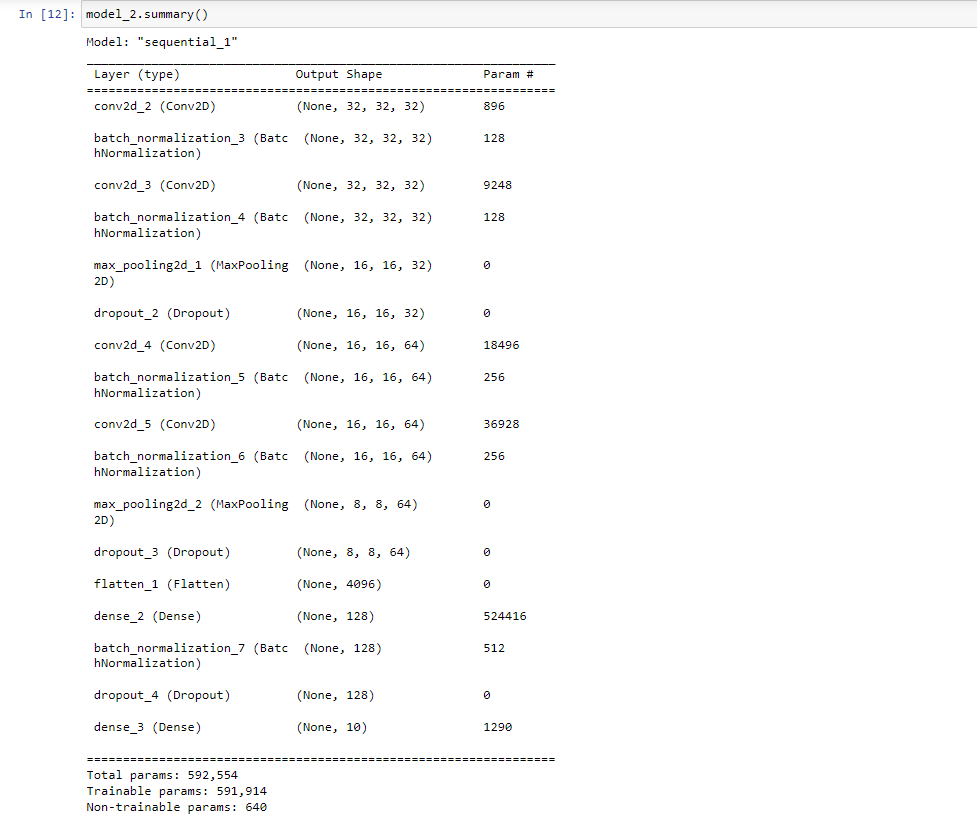
شکل۱۳) تابع برای مدل اول شبکه

شکل۱۴) تابع برای بیلد کردن شبکه دوم

شکل۱۵)تابع برای بیلد کردن شبکه سوم

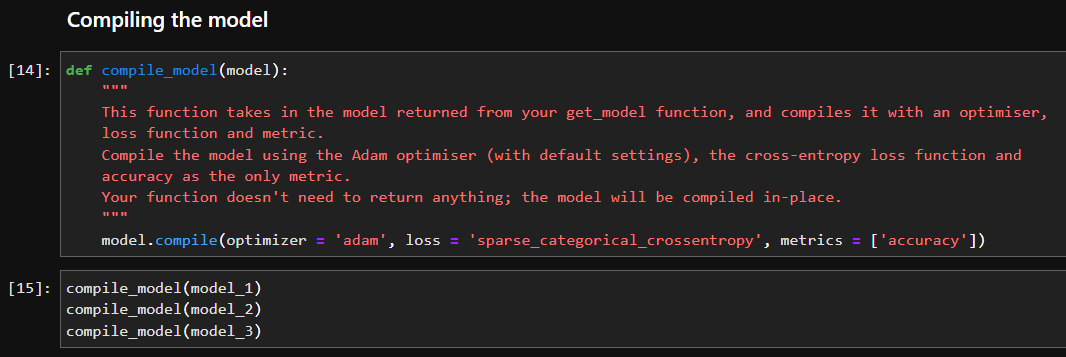
شکل۱۶) ۳ مدل را با استفاده از توابعشان تعریف می­کنیم.

در ادامه نیز می­توانید سامری شبکه هارا در کد مشاهده کنید که در شکل پایین لایه های شبکه مدل ۲ را مشاهده می­کنید.

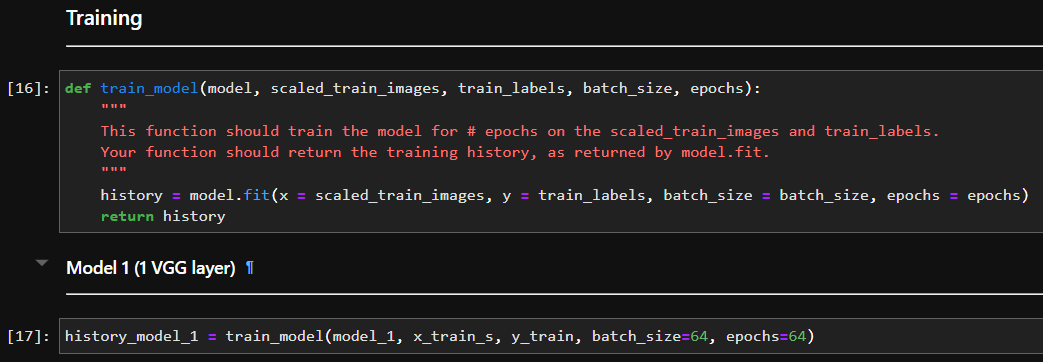


شکل۱۷) شبکه تشکیل شده برای مدل ۲

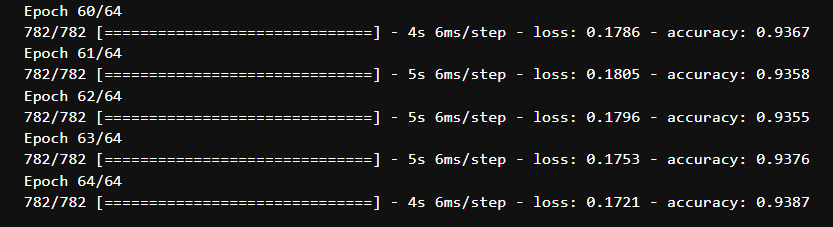
بعد از تشکیل دادن شبکه هایمان نیاز به کامپایل کردن آنها داریم که برای اینکار از متد compile با اپتیمایزر adam استفاده می­کنیم.

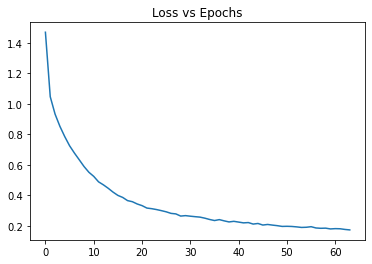
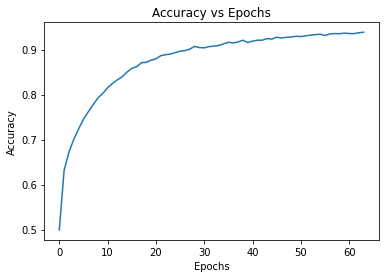
شکل۱۸) کامپایل کردن شبکه ها

بعد از کامپایل کردن شبکه به آموزش شبکه با استفاده از داده های اموزش می­پردازیم که برای اینکار از متد fit استفاده می­کنیم.

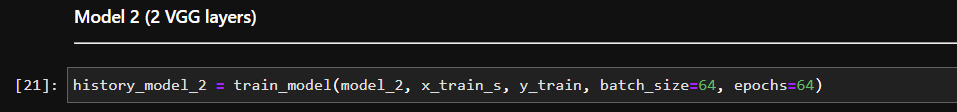
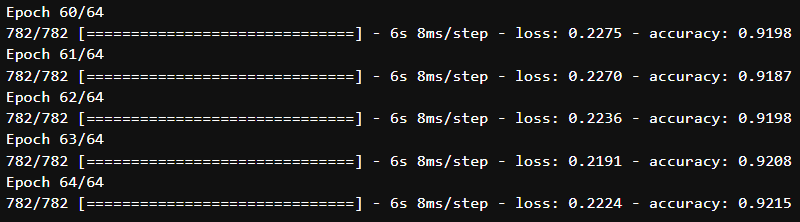
شکل۱۹) اموزش شبکه ها

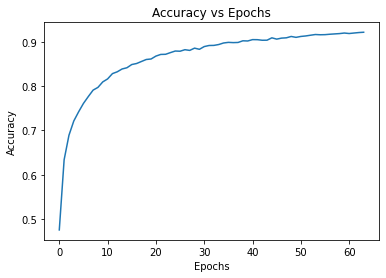
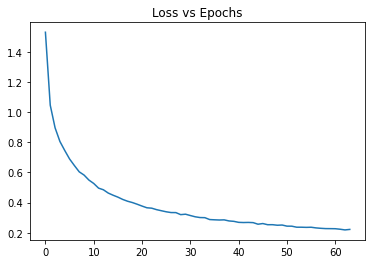
برای اموزش شبکه از بچ سایز های ۶۴تایی استفاده کردیم و به این علت که دوست عزیزمان! اقای طهرانی علاقه زیاده به دقت های بالا دارند از ۶۴ epoch استفاده کردند که ان­شالله لپتاپشان بعد از ران گرفتن این شبکه ها سالم مانده باشد اما ما برای مقایسه دقت این ۳ مدل به همان ۱۰ epoch اکتفا می­کنیم.

شکل۲۰) پایان آموزش شبکه ۱ با دقت ۹۳.۹ درصدی در ۶۴ epochs و دقت ۸۰.۲۵ درصدی در ۱۰ epochs

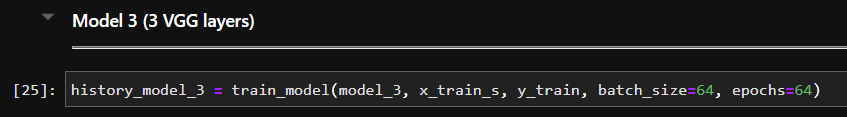
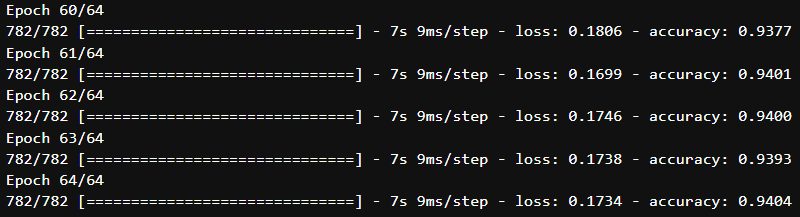


شکل۲۱) دقت و خطا شبکه ۱ بر حسب epochs

****شکل۲۲) أموزش شبکه ۲

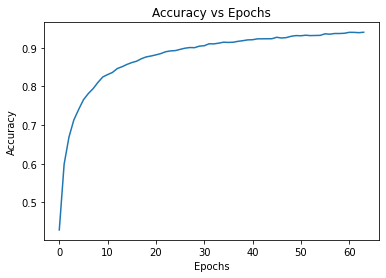
****شکل ۲۳) پایان آموزش شبکه ۲ با دقت نهایی ۹۲.۱۵ درصدی و دقت ۸۰.۹۶ درصدی در ۱۰epochs

شکل۲۴) دقت و خطا مدل ۲ بر حسب epochs

شکل ۲۵) آموزش شبکه ۳

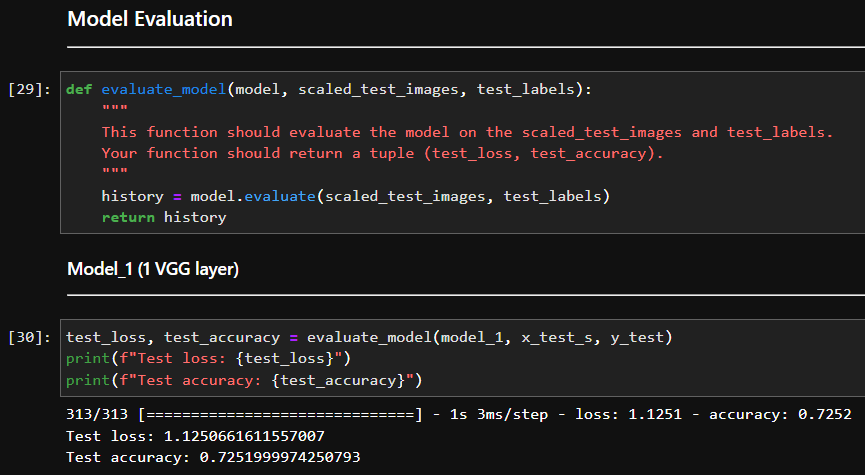
شکل ۲۶) پایان آموزش مدل۳ با دقت ۹۴ درصدی و دقت ۸۲.۴۵ درصدی در ۱۰ epochs

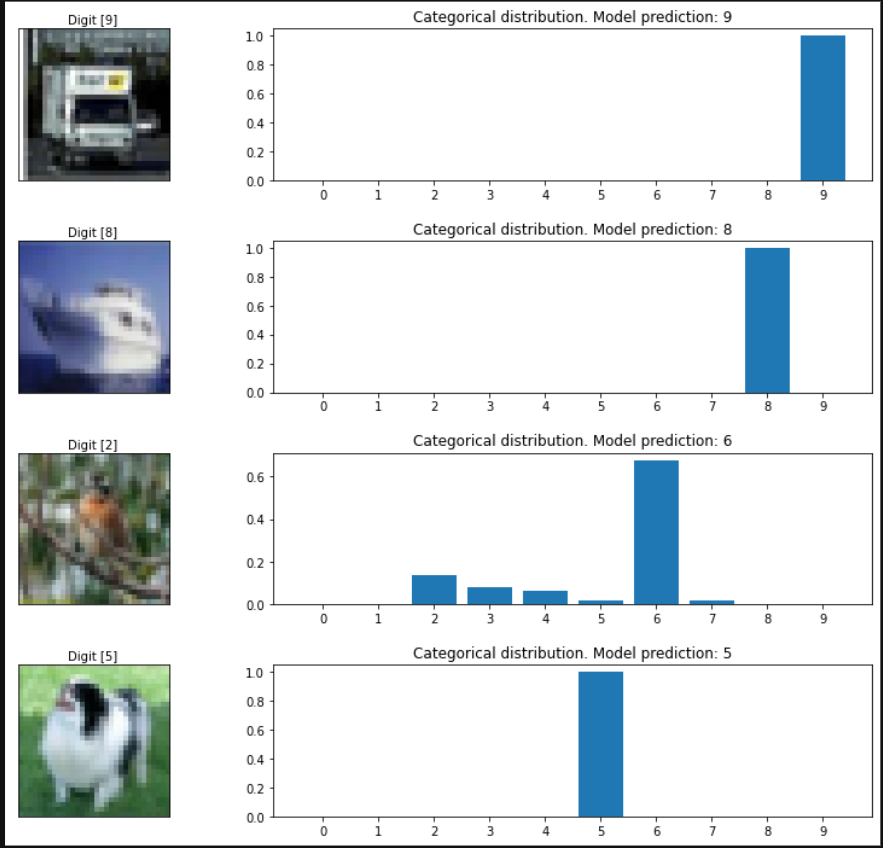
Shape, square

Description automatically generated

شکل۲۷) دقت و خطا مدل ۳ برحسب epochs

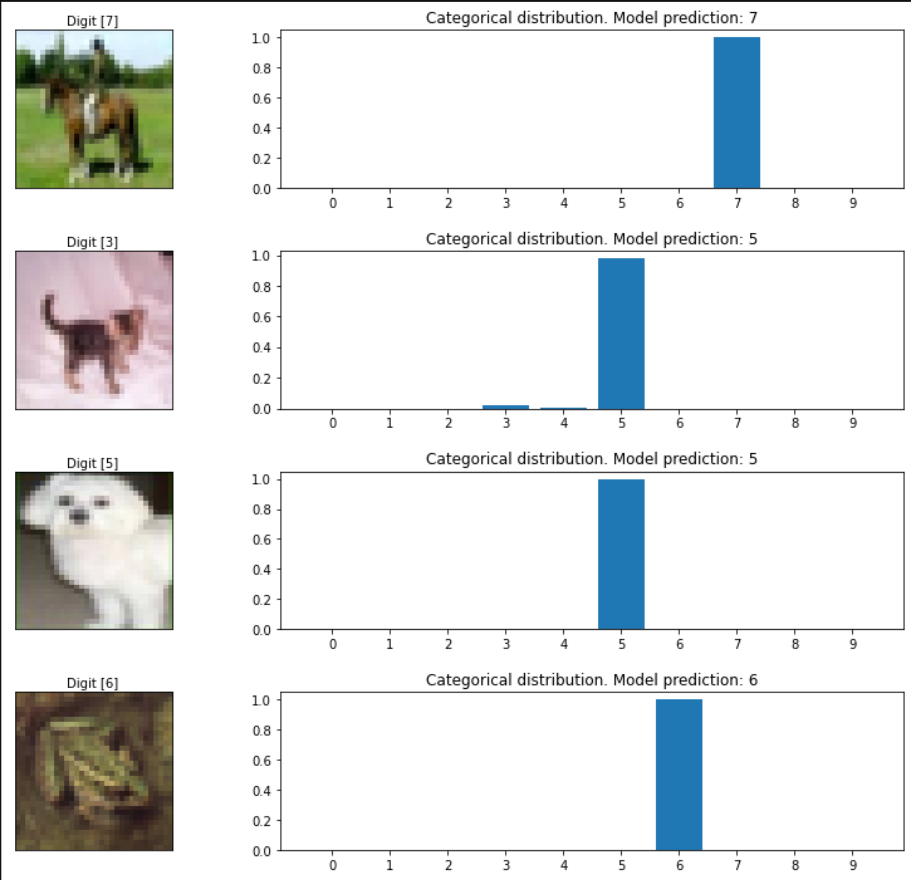
بعد از آموزش شبکه نوبت به تست دیتای تست بر روی هر ۳ مدل است که برای اینکار از متد evaluate استفاده می­کنیم که در پایین نتیجه آن برای هر ۳ مدل را مشاهده می­کنید.

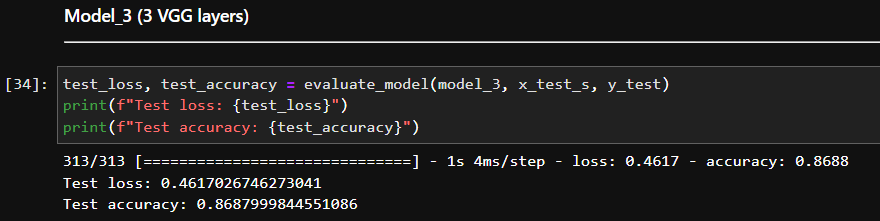
شکل۲۸) تست شبکه که به دقت ۷۲.۵ درصدی ختم شد

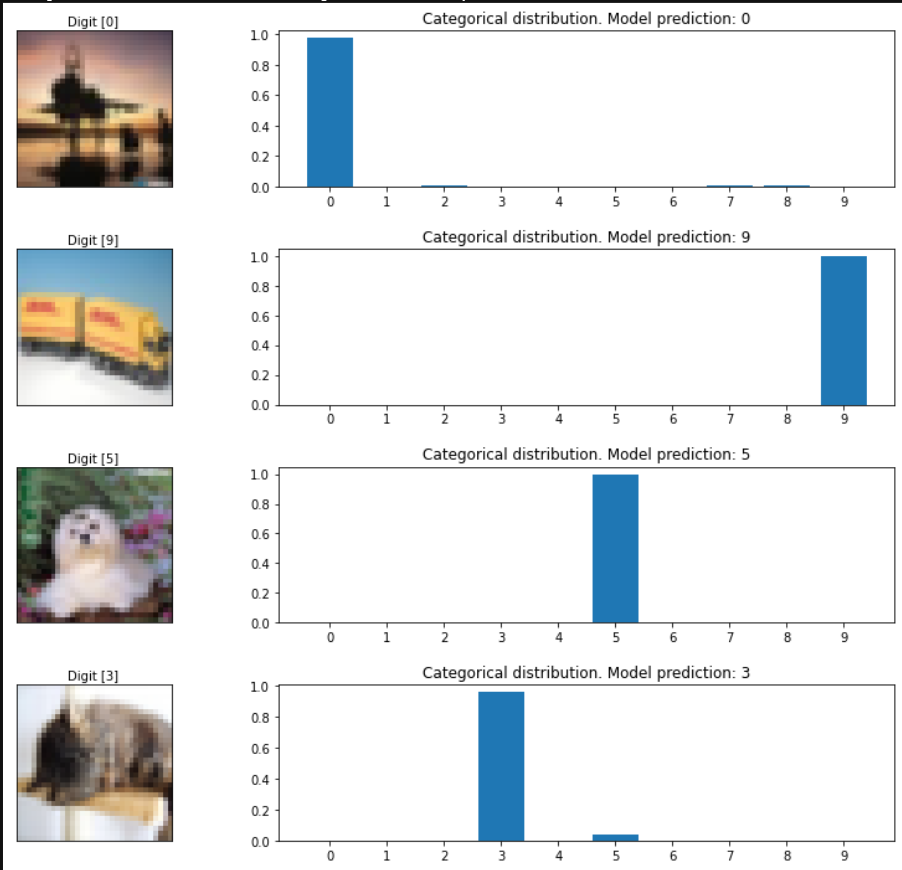
شکل۲۹) تست خروجی شبکه ۱

Text

Description automatically generatedشکل۳۰) تست شبکه ۲ بر روی دیتا تست که دقت ۸۲.۳ درصدی منجر شد

شکل۳۱) خروجی شبکه ۲

شکل۳۲) تست شبکه ۳ که دقت ۸۶.۸ درصدی منجر شد

شکل۳۳) خروجی شبکه ۳

همانطور که مشاهده شد با افزایش تعداد لایه ها دقت شبکه ها بیشتر شد که این مقدار تفاوت در شبکه اول و سوم تقریبا برابر با ۱۵ درصد است که مقدار قابل توجهی است و با توجه به اینکه در این ۳ مدل اختلاف میان دقت در داده های اموزش و تست قابل توجه نیست پس داده اموزش بر روی شبکه overfit نکرده است پس بهتر است از همان شبکه سوم استفاده کنیم.

**3-1-2 تمرین دوم**

در صورتی که نیاز به چند خروجی در شبکه داشته باشیم یا قرار باشد دو شبکه عصبی با یکدیگر ارتباط داشته باشند یا لایه بعدی تعداد ورودی بیشتری از لایه قبلی داشته باشد بهتر است از functional-API به جای sequential-API استفاده بکنیم.

**پایان**