

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

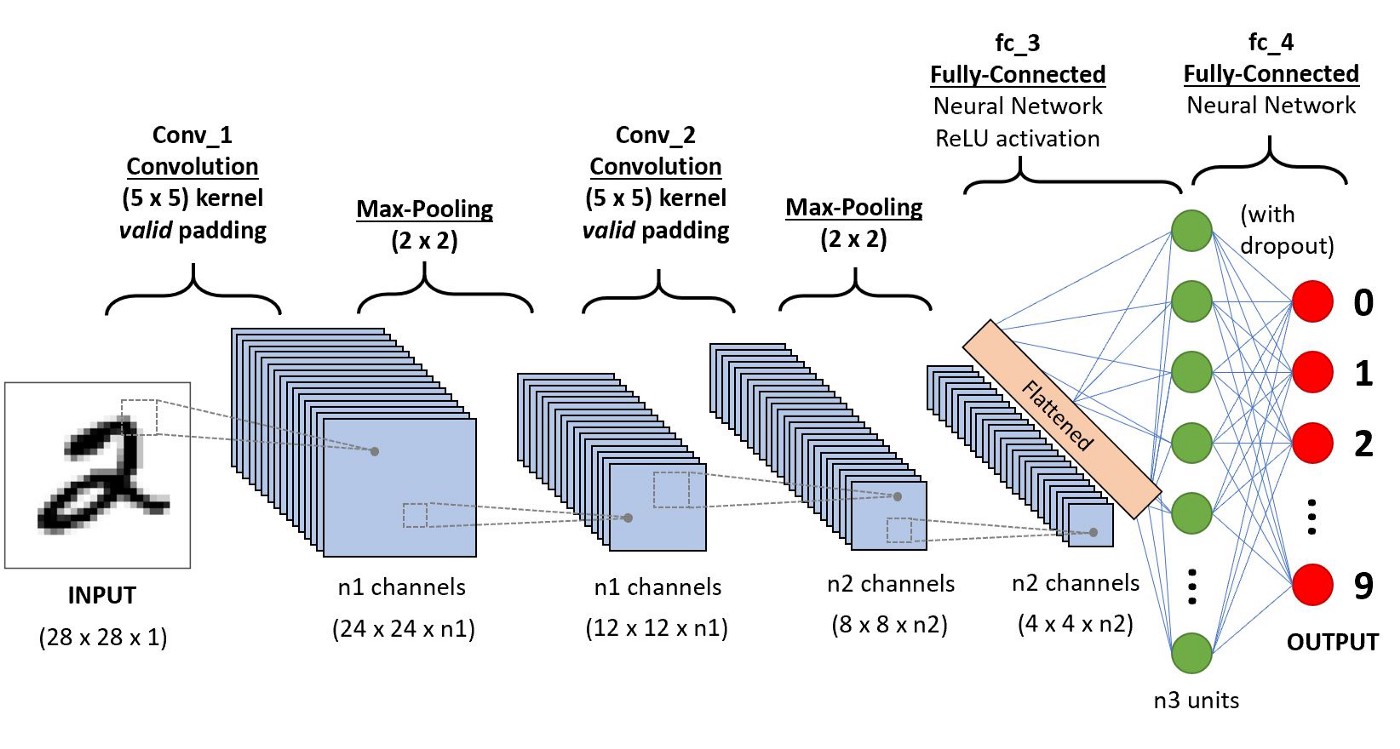
**تمارین........................................................................................................................................11**

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؛

**2-1-1 دیتاست Hotel**

توضیحات مربوط به این دیتاست در دستورکار گفته شده است پس از توضیح دادن آن پرهیز می­کنیم.

* **Dataset Preparation**
* Data Preprocessing

بعد از اضافه کردن ایمپورت کردن کتابخانه ها مورد نیاز در ابتدای کد، به قسمت خواندن دیتاست و انجام تعییرات به روی آن می­پردازیم. ابتدا دیتاست های آموزش و تست را با read\_csv به یک دیتافریم تبدیل میکنیم و دیتاهایی با مقدار NaN را با dropna() حذف می­کنیم. سپس

* Shuffling The Dataset

سپس با دستور df.sample() و با نسبت ۱ دیتا را شافل می­کنیم.

* Categorical And Numerical Features

در ابتدا نام تمامی فیچر ها و سپس فیچر های عددی و فیچر های کتگوریکال را چاپ می­کنیم تا متوجه شویم کدام فیچر هارا باید به عدد تبدیل کنیم.

برای تبدیل کردن فیچر هایی که دیتاتایپ آنها object است از ۲ روش بهره بردیم،‌ روش اول که بهینه نمی­باشد بدین ترتیب است که تمام مقادیر غیریکسان فیچر را بصورت دستی مقدار دهیم که اینکار را فقط برای فیچر ArrivalDateMonth انجام دادیم اما همانطور که گفتیم برای تعداد فیچر های زیاد و تعداد سمپل های زیاد کار وقت گیری است و به صرفه نیست پس برای همین علت از روش دوم استفاده کردیم که در آن ابتدا فیچرهایی که مقادیر آبجکت دارند را به دیتا categorical تبدیل می­کنیم تا از متد cat.codes که برای دیتاهای categorical تعریف شده استفاده می­کنیم. این متد یک سری از کدها را برای دیتا های متفاوت در نظر می­گیرد و به آنها نسبت می­دهد که خروجی در زیر سلول نشان داده شده است.

برای دیتاست تست هم مانند دیتا آموزش رفتار می­کنیم و بجز فیچر ArrivalDateMonth باقی فیچر های دارای آبجکت را با cat.codes به عدد تبدیل می­کنیم. اما تفاوت اصلی دیتاست تست و آموزش در مقادیر آنها در فیچر های ­مختلف است به عنوان مثال تعداد کشور های ذکر شده در دیتای تست با تعداد کشور های دیتا آموزش متفاوت است و از آن بیشتر است. برای حل آن از یک تابع لامبدا استفاده می­کنیم به این صورت که اگر دیتای فیچر در دیتاست ترین قرار داشت که مقدار همان را بهش می­دهیم اگر وجود نداشت مقدار -۱ را برای آن قرار می­دهیم و این تابع را با متد apply به دیتاست وارد می­کنیم.

* Normalizing Dataset

برای نرمالایز کردن دیتای آموزش از تابع Preprocessing در کتابخانه Sklearn استفاده می­کنیم که دیتاست را نرمالایز می­کند

* Split Dataset to Train and Validation

برای جدا کردن دیتاست به بخش آموزش و ولیدیشن نیز از متد sample استفاده می­کنیم با تسبت .۰.۸ به ۰.۲ و سپس به جداسازی لیبل خروجی از باقی دیتاست می­پردازیم بدین صورت که فیجر ADR را به لیبل خروجی و باقی فیچر هارا به ورودی شبکه می­دهیم.

* **Building The Multilayer Perceptron**

در این بخش به آموزش شبکه می­پردازیم:

* Model Building

برای بیلد کردن مدل از Sequential\_API استفاده می­کنیم که در آن از ۲ لایه مخفی با تعداد ۲۰ نورون استفاده می­کنیم که تالع فعال ساز های آنها نیز RELU می­باشد.

* Compile The Model

برای کامپایل کردن مدل شبکه نیز آپتیمایزر آدام و متریکس های MAE, MSE استفاده می­کنیم.

* Train The Model

در این قسمت شبکه را با دیتای ترین، آموزش می­دهیم و برای اینکار از تعداد Epoch های ۳۰ و ۵۰ و ۱۰۰ با batch size = 64 استفاده می­کنیم که اوردر خطای هر کدام بترتیب ۰.۰۰۲ و ۰.۰۰۱۸ و ۰.۰۰۱۵ می­باشد.

* Plotting

در قسمت پلات هم مقادیر MAE, MSE را بر حسب اپوک ترسیم می­کنیم.

* Model Evaluation

در این قسمت با متد Evaluation خطا دیتای ولیدیشن را بر روی شبکه محاسبه می­کنیم که برابر با ۰.۰۰۱۴ می­باشد.

* **Model Prediction**

در این قسمت با استفاده از تابع predict مقدار ADR هر دیتا را برحسب فیچرهایش بر روی دیتای تست محاسبه می کنیم. که مقدار پیش بینی شده و مقدار واقعی در این مدل شبکه ما تمامی فیچر ها در فایل result.csv ذخیره شده است.

* **Feature Selection**

در قسمت د تمرین اول گفته شده کدام فیچر ها ببشترین تاثیر در یادگیری شبکه را دارند و چگونه آنها را جداسازی کنیم. برای اینکار از ماتریس Correlation استفاده کردم. برای این منظور تمام فیچر هایی که مقدار همبستگی آن با لیبل خروجی یعنی ADR بیشتر از مقدار دلخواه بود را به عنوان فیچر با تاثیر زیادتر تعیین می­کنم. در اینجا مقدار همبستگی بیشتر از ۰.۱ را انتخاب کردم که فیچر های انتخاب شده را مشاهده میکنید.

* Train With Selected Features

سپس شبکه را با دیتاست جدید که فقط شامل فیچر های انتخابی است آموزش می­دهیم که خروجی آن را نیز مانند قسمت قبلی پیشبینی می­کنیم و آن را در فایل result.csv ذخیره می­کنیم.

**2-1-2 دیتاست CIFAR-10**

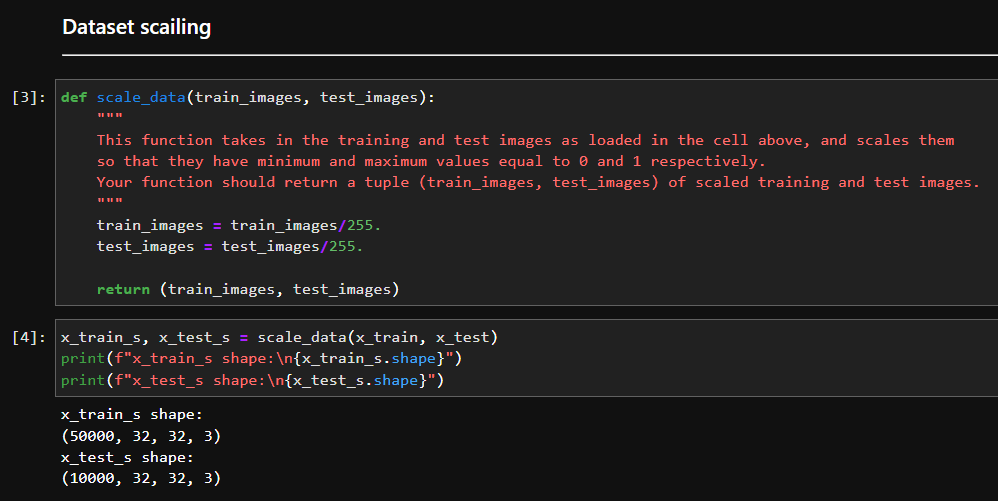
از این دیتاست در گذشته در درس هوش محاسباتی خانم دکتر عبداللهی استفاده کرده بودیم که مدل شبکه طراحی شده را و گزارشش را در پایین به صورت کامل مشاهده می­کنید. برای تمرین دوم نیز برای مشاهده کد وارد قسمت ML Second Exercise بشوید. در این قسمت صرفا کد را یک برسی کوتاه می­کنیم و برای برسی کامل می­توانید گزارش پایین را بخوانید.

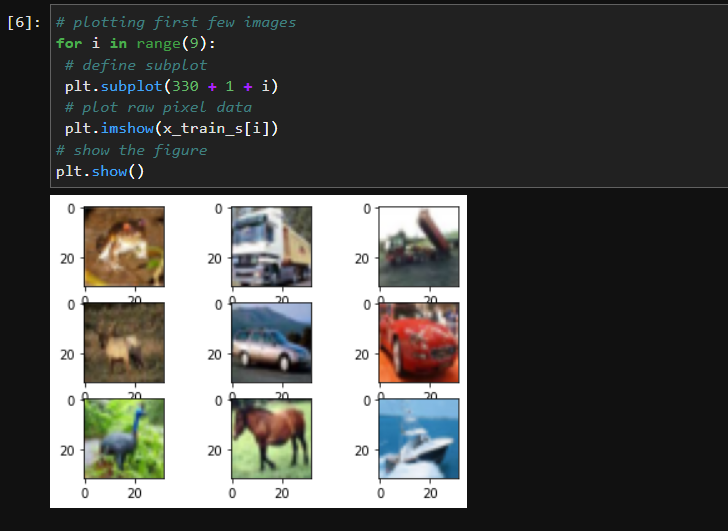
بعد از خواندن دیتاتست و اسکیل کردن آن شبکه را طراحی، کامپایل می­کنیم و برای آموزش شبکه از batch size های ۳۲و۶۴و۱۲۸ تایی با تابع فعال ساز رلو استفاده کردیم و در مرحله بعد نیز همان شبکه را با تابع فعال ساز های tanh, sigmoid تست کردیم که نتایح مشخص هستند سپس مدل را evaluate کردیم و آن را تست کردیم که درنهایت آنرا مشاهده می­کنید

**2-1-2 دیتاست CIFAR-10**

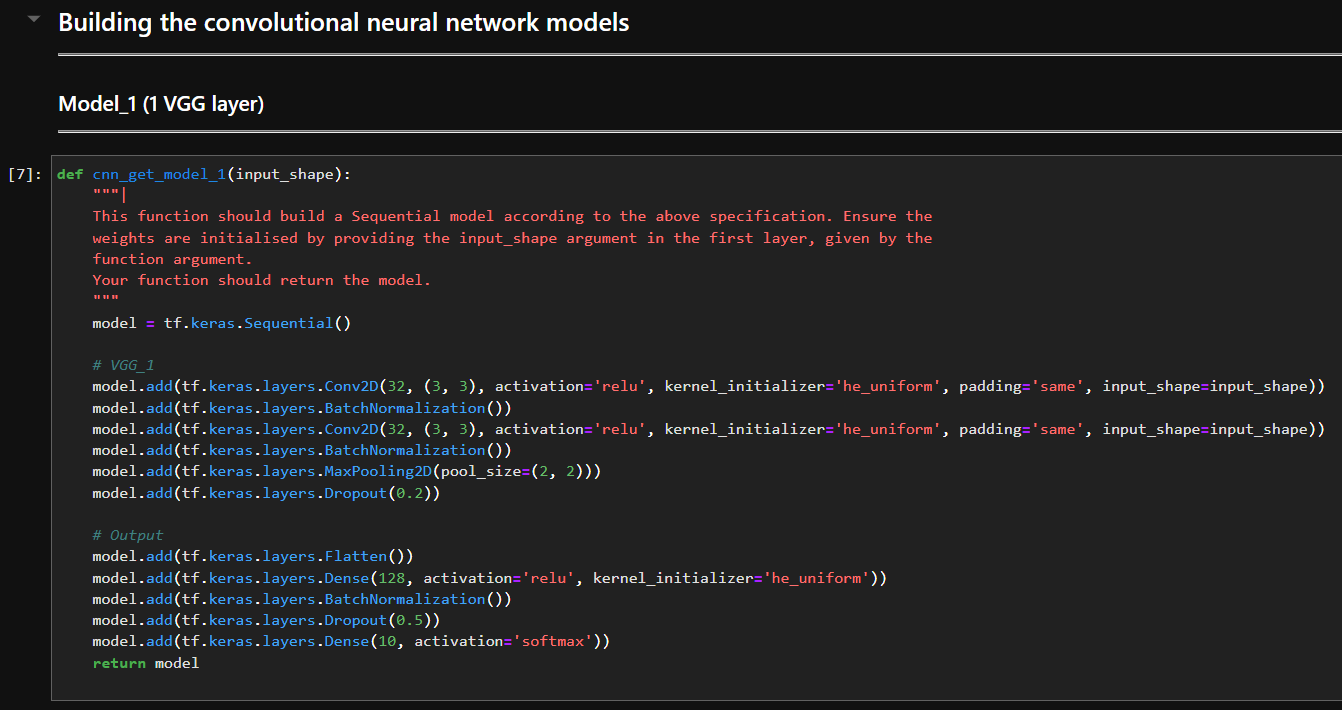
توضیحات مربوط به دیتاست cifar-10 در پیش گزارش داده شده است پس مستقیم به توضیح کد می­پردازیم.

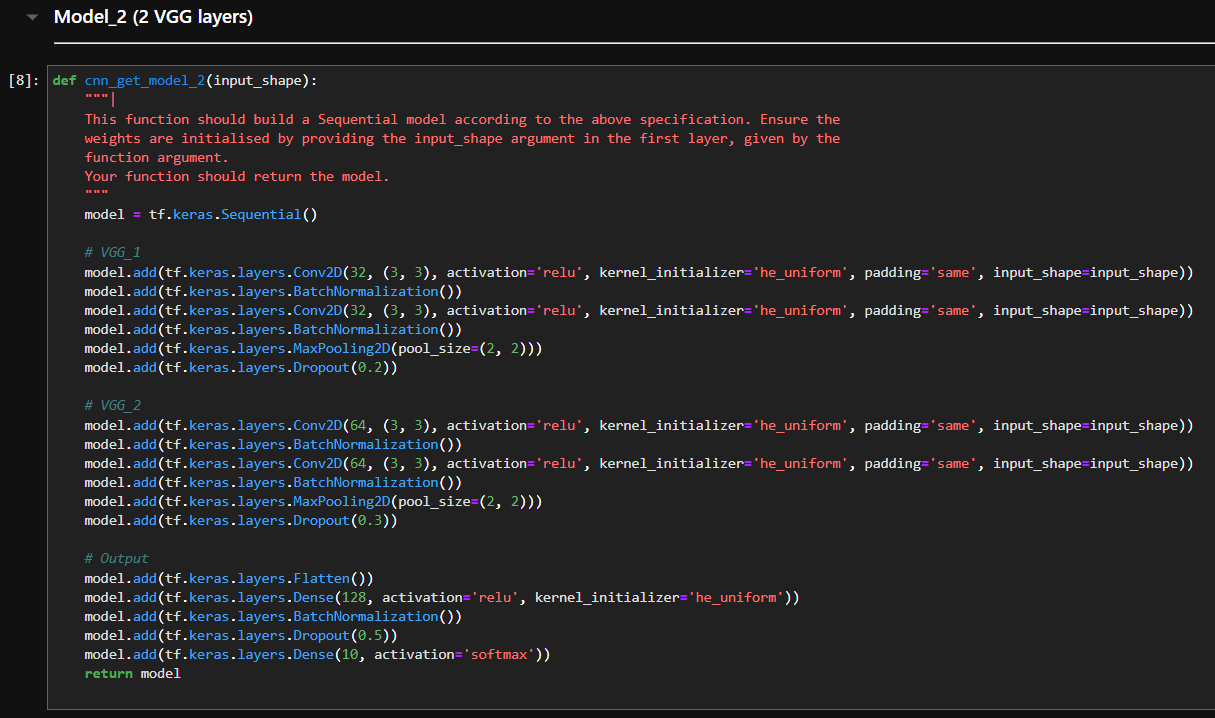
در ابتدا مانند دیتاست mnist کتابخانه هارا ایمپورت می­کنیم و دیتاست را ادد می­کنیم.

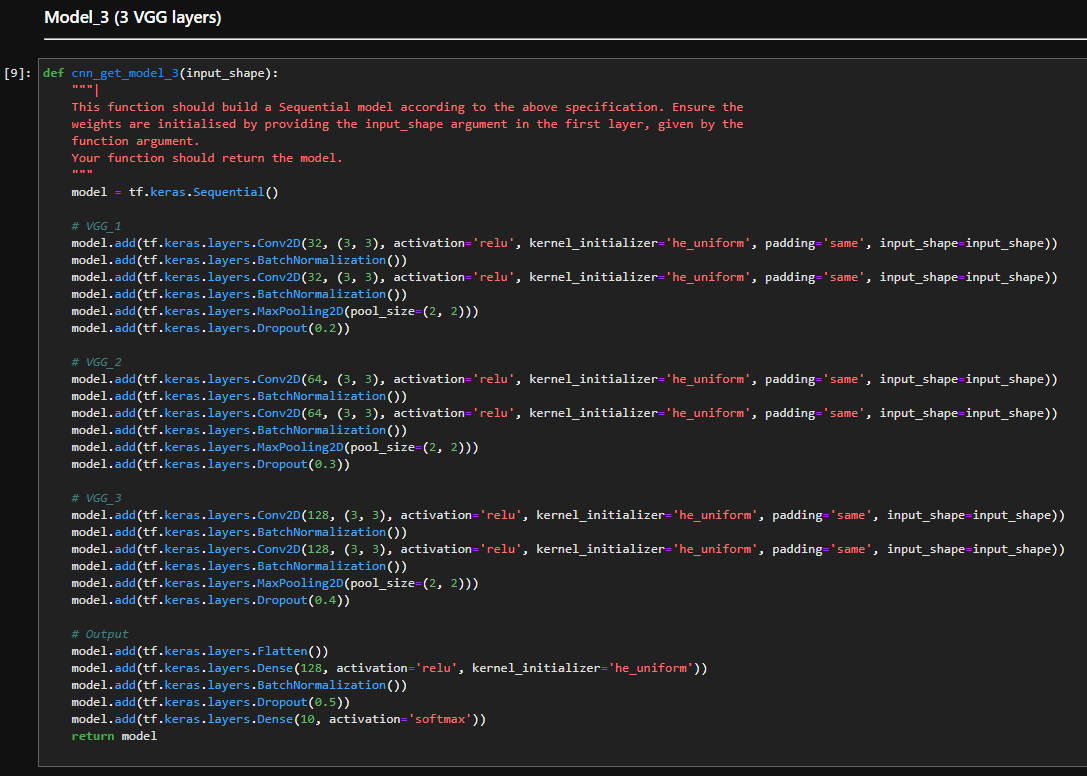
شکل۱۱) ایمپورت کردن دیتاست و کتابخانه ها

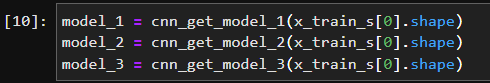
شکل۱۲) مشاهده چند نمونه از دیتاهای cifar-10

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.

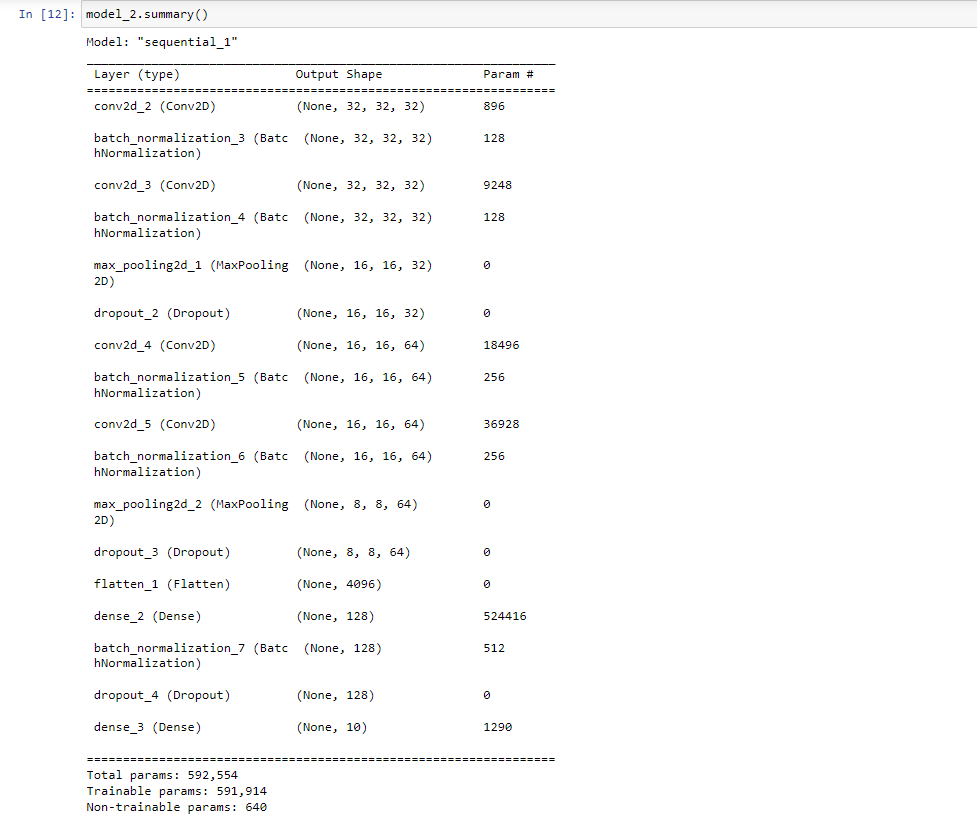
شکل۱۳) تابع برای مدل اول شبکه

شکل۱۴) تابع برای بیلد کردن شبکه دوم

شکل۱۵)تابع برای بیلد کردن شبکه سوم

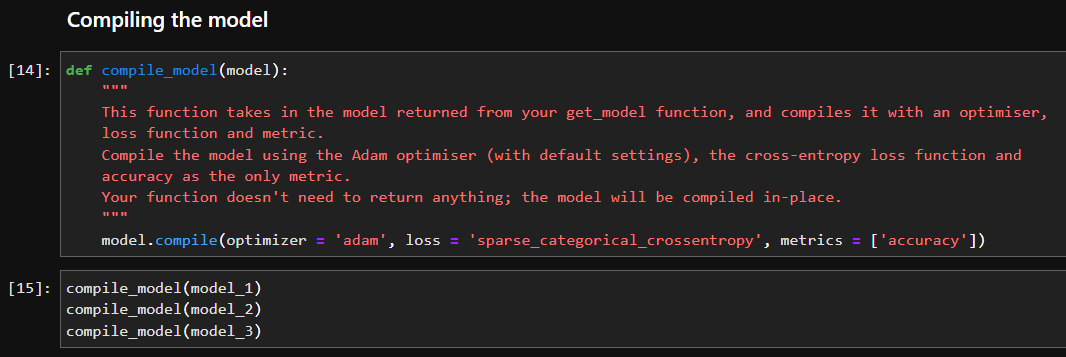
شکل۱۶) ۳ مدل را با استفاده از توابعشان تعریف می­کنیم.

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

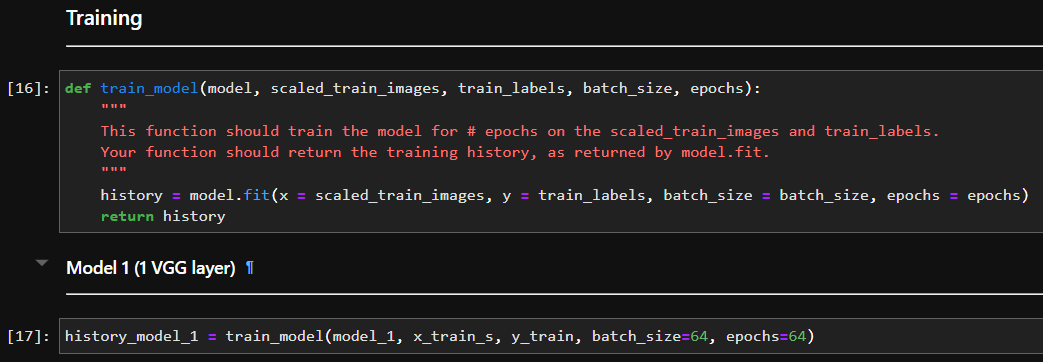


شکل۱۷) شبکه تشکیل شده برای مدل ۲

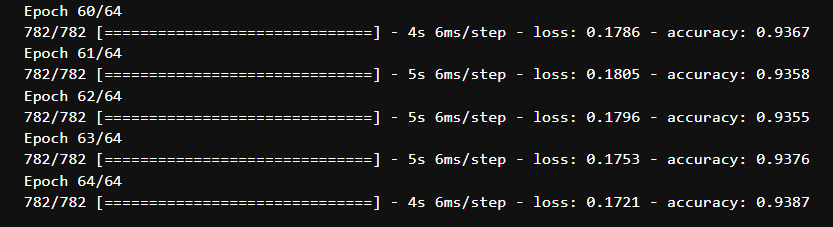
بعد از تشکیل دادن شبکه هایمان نیاز به کامپایل کردن آنها داریم که برای اینکار از متد compile با اپتیمایزر adam استفاده می­کنیم.

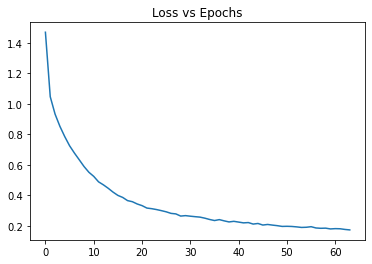
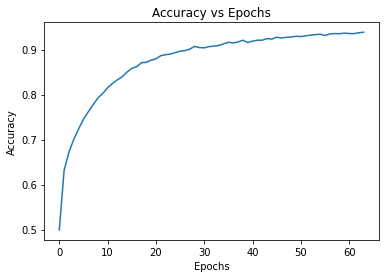
شکل۱۸) کامپایل کردن شبکه ها

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

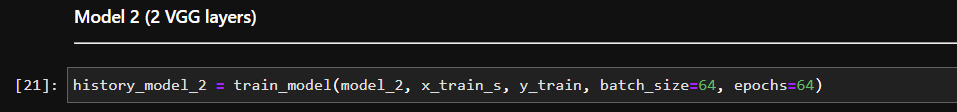
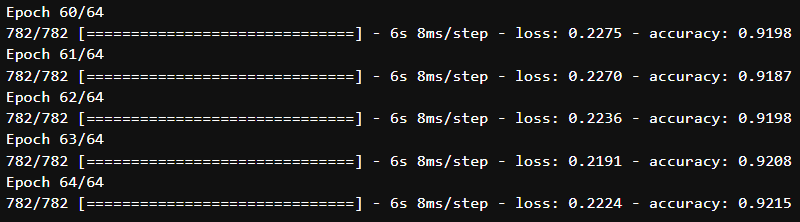
شکل۱۹) اموزش شبکه ها

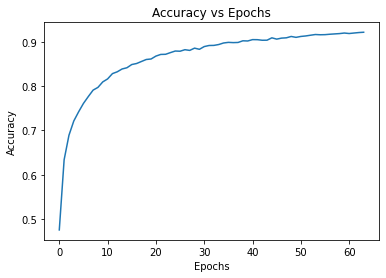
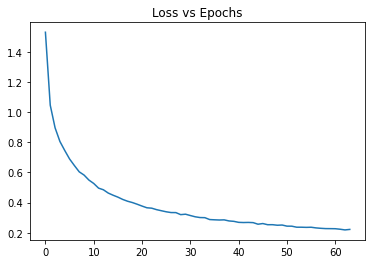
برای اموزش شبکه از بچ سایز های ۶۴تایی استفاده کردیم و به این علت که دوست عزیزمان! اقای طهرانی علاقه زیاده به دقت های بالا دارند از ۶۴ epoch استفاده کردند که ان­شالله لپتاپشان بعد از ران گرفتن این شبکه ها سالم مانده باشد اما ما برای مقایسه دقت این ۳ مدل به همان ۱۰ epoch اکتفا می­کنیم.

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

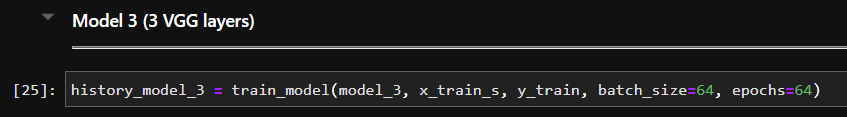
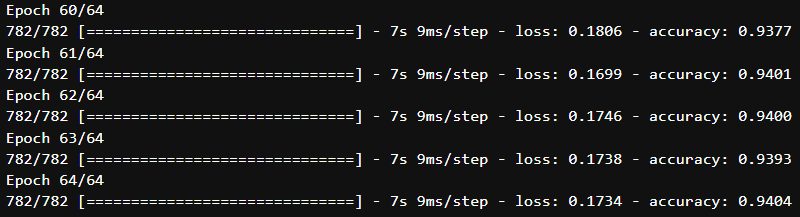


شکل۲۱) دقت و خطا شبکه ۱ بر حسب epochs

****شکل۲۲) أموزش شبکه ۲

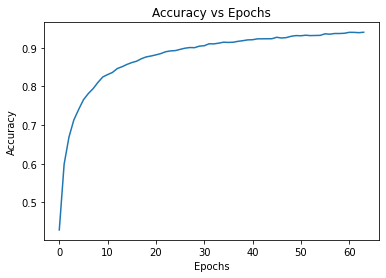
****شکل ۲۳) پایان آموزش شبکه ۲ با دقت نهایی ۹۲.۱۵ درصدی و دقت ۸۰.۹۶ درصدی در ۱۰epochs

شکل۲۴) دقت و خطا مدل ۲ بر حسب epochs

شکل ۲۵) آموزش شبکه ۳

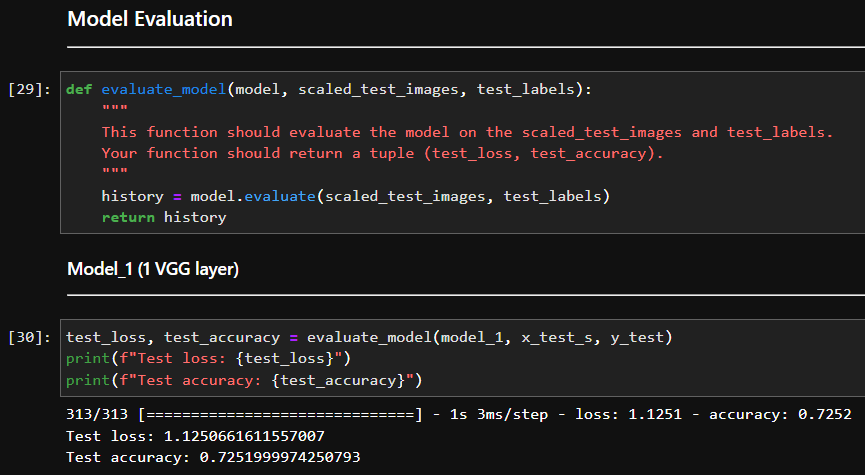
شکل ۲۶) پایان آموزش مدل۳ با دقت ۹۴ درصدی و دقت ۸۲.۴۵ درصدی در ۱۰ epochs

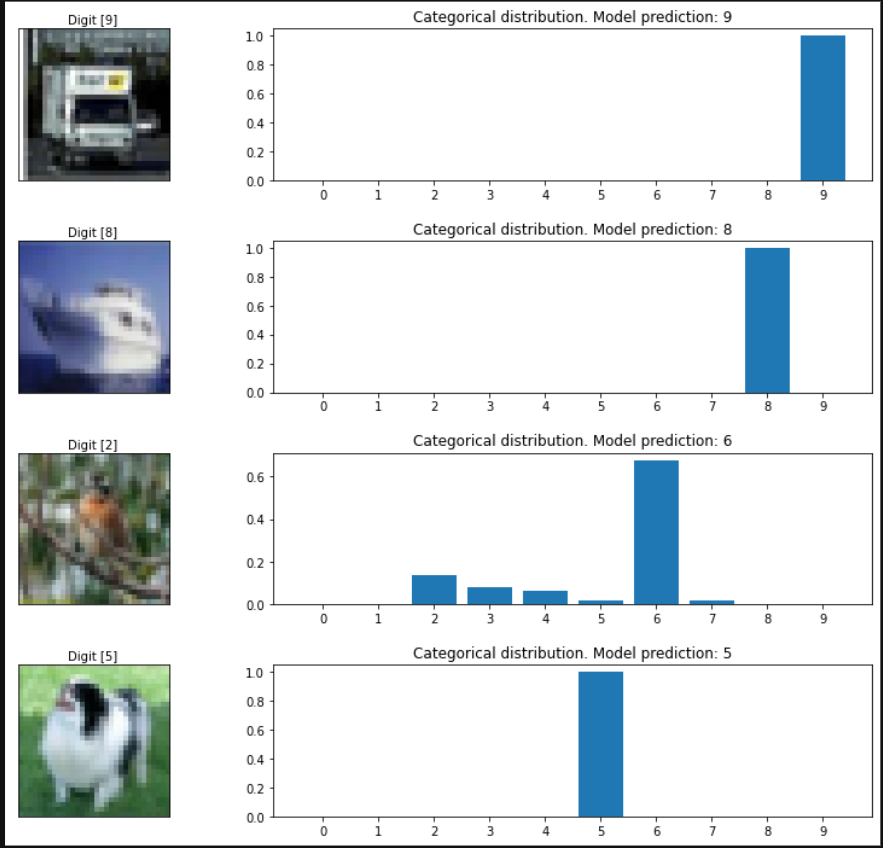
Shape, square

Description automatically generated

شکل۲۷) دقت و خطا مدل ۳ برحسب epochs

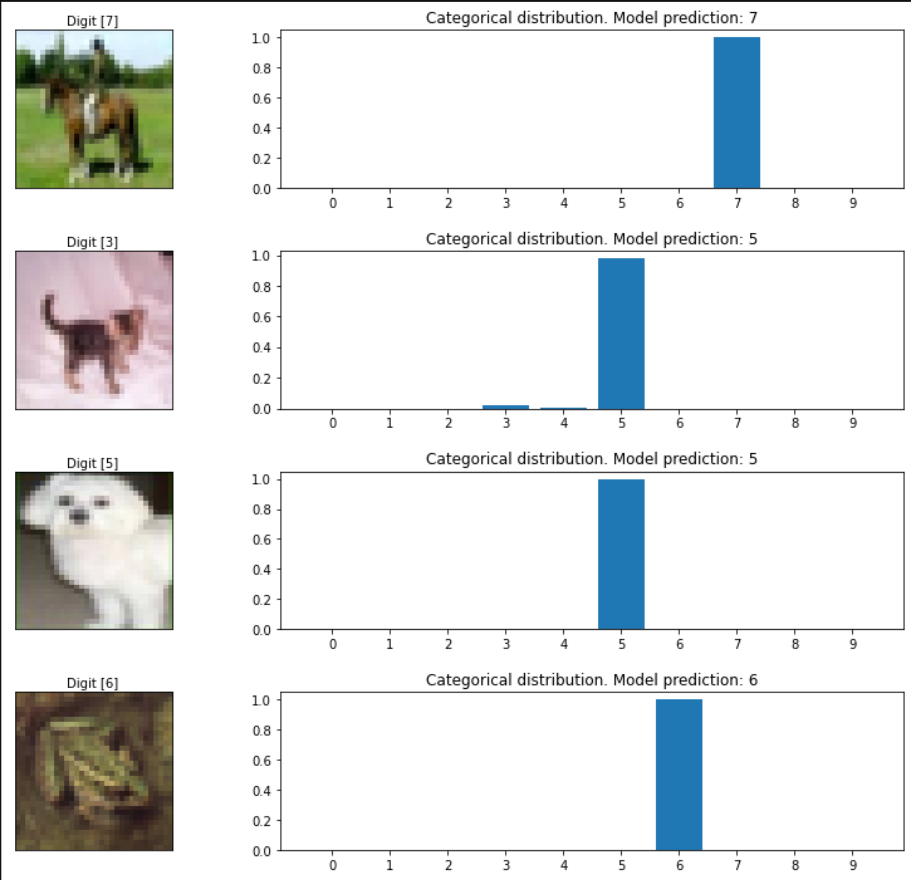
بعد از آموزش شبکه نوبت به تست دیتای تست بر روی هر ۳ مدل است که برای اینکار از متد evaluate استفاده می­کنیم که در پایین نتیجه آن برای هر ۳ مدل را مشاهده می­کنید.

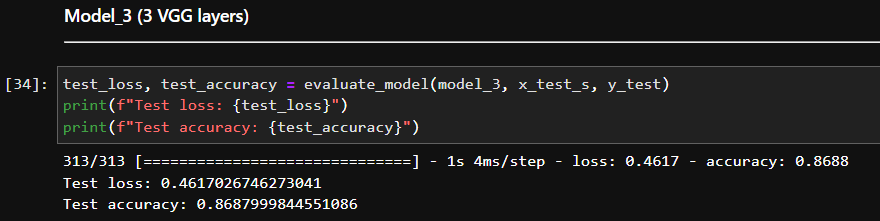
شکل۲۸) تست شبکه که به دقت ۷۲.۵ درصدی ختم شد

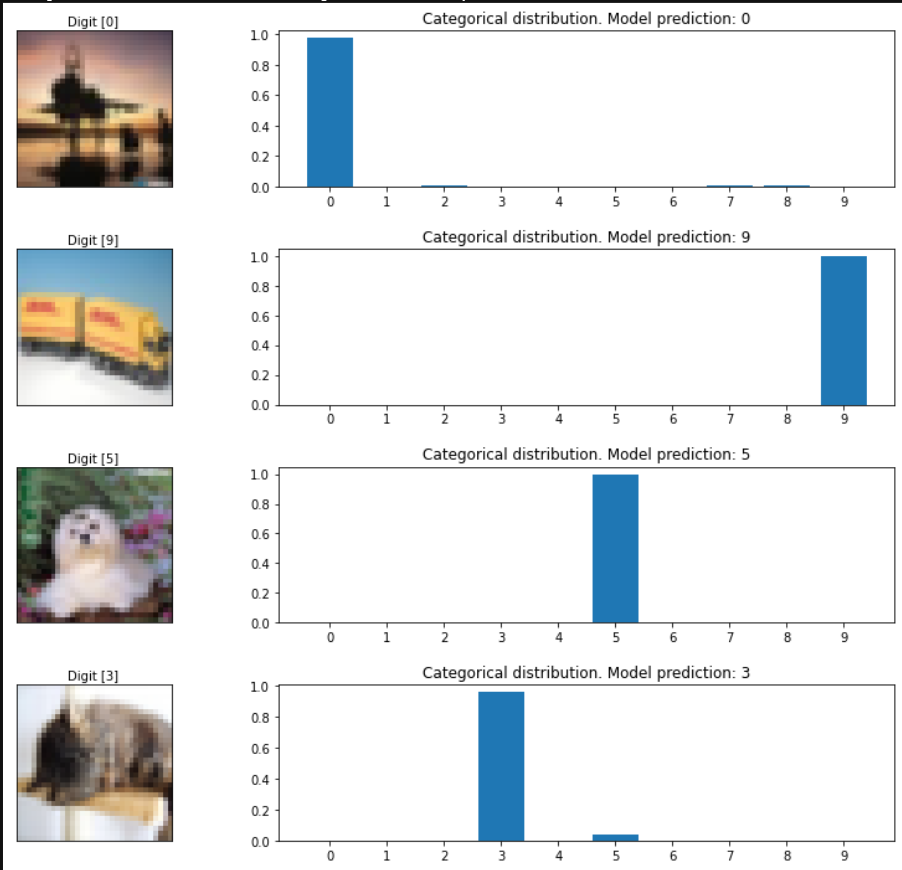
شکل۲۹) تست خروجی شبکه ۱

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.

**پایان**