گزارش تمرین دوم شبکه عصبی

علیرضا آزادبخت ۹۹۴۲۲۰۱۹

تمرین ۱:

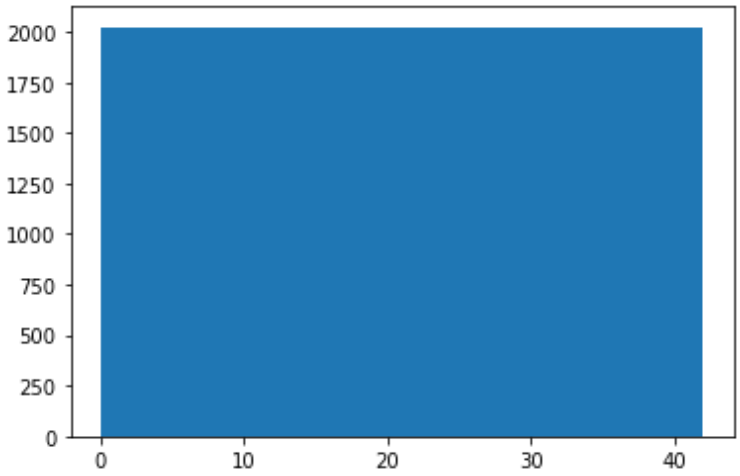
مسئله تابلوهای راهنمایی:

این مسئله شامل تقریبا ۱۰۰۰۰۰ عکس تابلو های راهنمایی از ۴۳ نوع تابلوی مختلف در ابعاد ۳۲ در ۳۲ پیکسل می‌باشد و وظیفه ما طراحی و آموزش شبکه عصبی جهت طبقه بندی این ۴۳ نوع تابلو میباشد.

نمونه ای از مجموعه داده های مسئله:

سوال اول: آیا کلاس ها تعداد یکسانی دارند یا خیر؟

ما از دیتا ست پیش پردازش شده data0 و data4 استفاده کردیم که بعد از بررسی ها متوجه شدیم که در این دو دیتا ست کلاس ها از فراوانی بالانسی برخوردارند و برای اینکه این مشکل از دیتا ست اولیه و خام حذف شود از روش های آگمنتیشن استفاده شده مثلا کلاس هایی که جمعیت کمتری داشتند به صورت تصادفی مانند مثال سمت راست بالا چرخش های مختلف آن ها به دیتا ست اضافه شده است. فراوانی کلاس ها در مجموعه داده آموزش:



مدل سازی:

معماری شبکه ای برتری که انتخاب کردیم شامل سه لایه کانولوژنال و دو لایه فولی کانکت بود که با اپتیمایزر adam و لرنینگ ریت ۰.۰۰۰۵ و تابع خطای کراس انتروپی و به اندازه ۳ ایپاک به صورت تیکه های بچ کوچیکتر ۲۵۶ تایی روی مجموعه داده data0 آموزش داده و تست شد.

معماری دقیق تر را میتوانید در زیر مشاهده کنید:

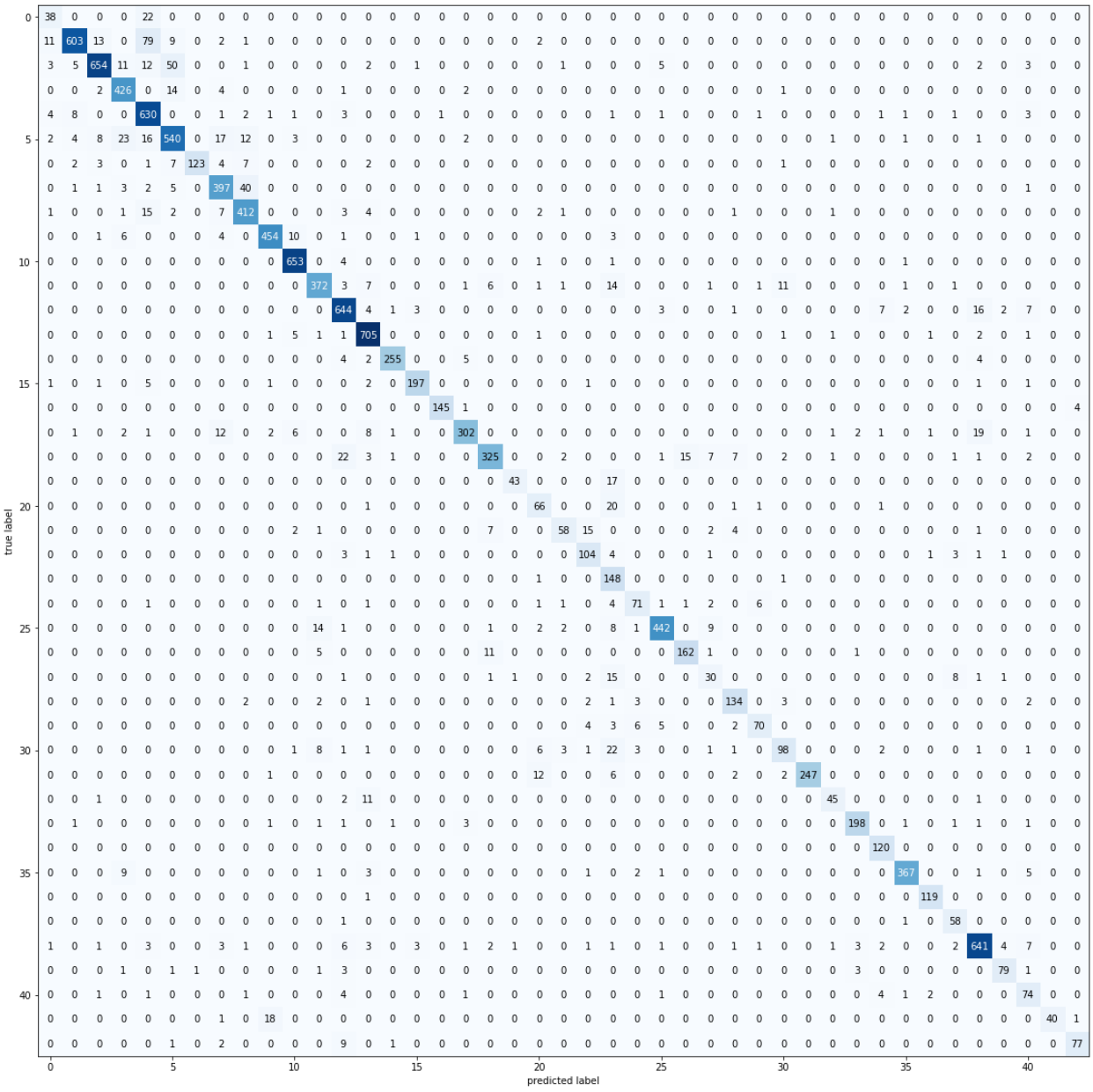
```
Model(
  (conv): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
    (4): ReLU()
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
    (7): ReLU()
    (8): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (flatten): Sequential(
    (0): AdaptiveMaxPool2d(output_size=1)
    (1): Flatten(start_dim=1, end_dim=-1)
  )
  (fc): Sequential(
    (0): Linear(in_features=64, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=43, bias=True)
  )
)
```

ابتدا این معماری را بر روی مجموعه داده های رنگی (data0) اجرا گرفتیم و برای پاسخ به سوال دوم همین معماری را بر روی مجموعه داده سیاه و سفید(data4) تنها با ایجاد تفاوت در تعداد چنل ورودی عکس اجرا کردیم

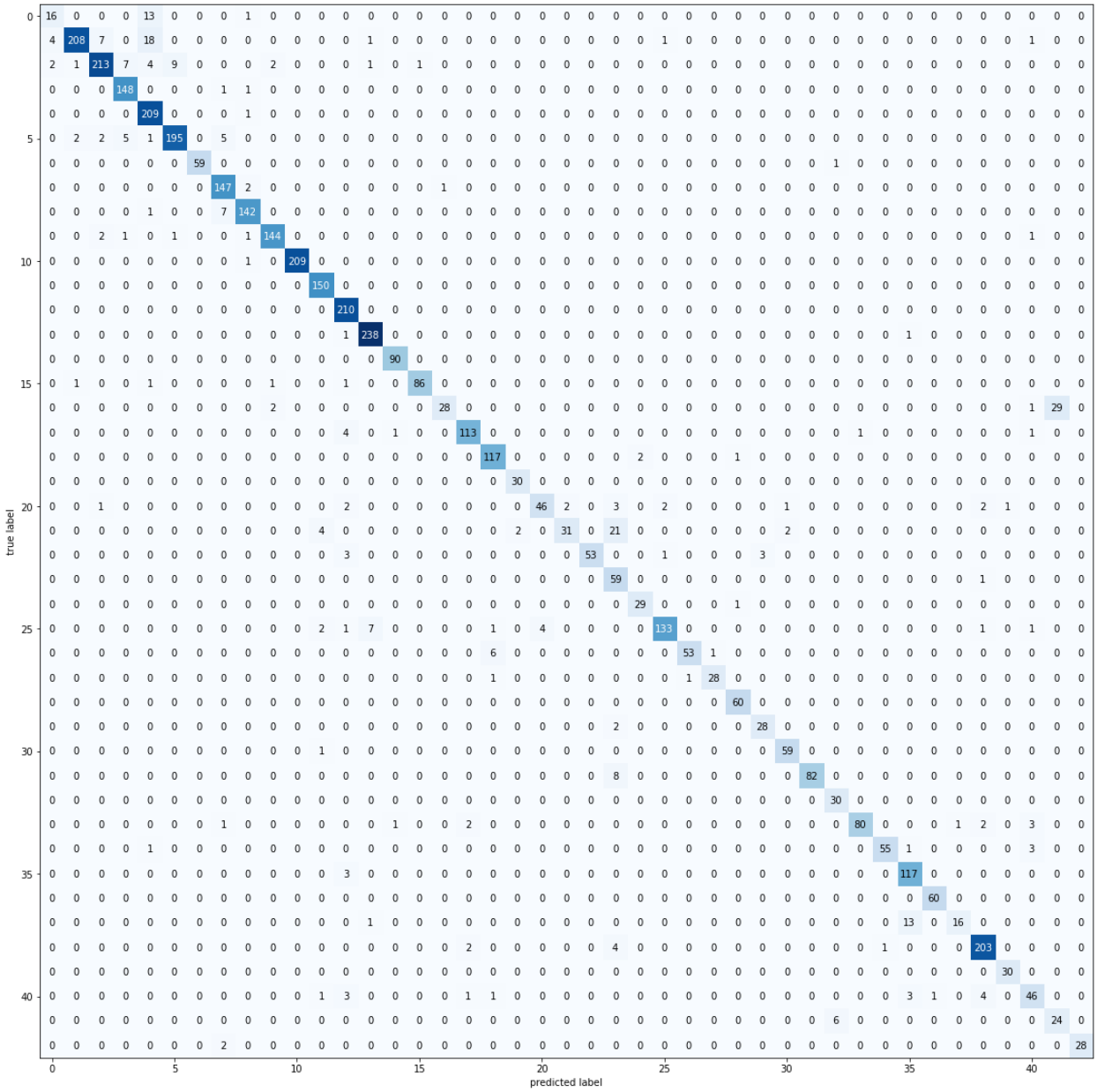
نتیجه بدست آمده برای مجموعه داده های رنگی بر روی مجموعه داده های تست به صورت زیر میباشد:

مجموعه داده	accuracy	F1- score macro avg	F1- score weighted avg	زمان آموزش
test	0.90	0.86	0.90	17min 22s
validation	0.93	0.90	0.93	-

ماتریس کانفیوژن مجموعه داده تست:



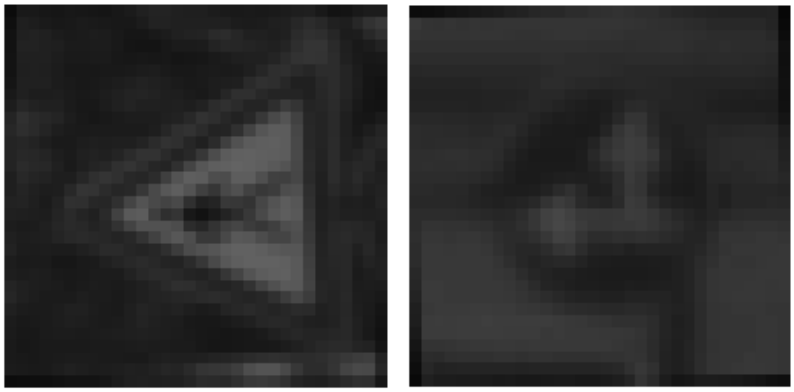
ماتریس کانفیوژن مجموعه داده ولیدیشن:



همانطور که مشاهده میشود و از دقت مدل قابل حدس است بیشتر عناصر بر روی قطر پررنگ بوده و نشانه تشخیص درست عکس ها توسط مدل میباشد، چند نکته در این قسمت قابل بیان است: فراوانی کلاس ها در مجموعه داده تست و ولیدیشن بالانس نشده و ممکن است رنگ های ماتریس کانفیوژن باعث برداشت اشتباه از

دقت مدل شود مثلا در اولین سطر مدل ما تقریبا با دقت خیلی پاینی تابلو های کلاس صفرم را تشخیص میدهد و این به دلیل تشابه زیاد این تابلو به تابلو شماره چهارم می‌باشد. و اینکه دقت بر روی داده های ولیدیشن بیشتر از تست است هم اطلاعات زیادی به ما نمیدهد و مقایسه دقت ولیدیشن و تست اشتباست و تنها برای مقایسه بین مدل ها میتوانیم از این دو مستقلا استفاده کنیم. و چون مدل ما در فرایند آموزشش از توقف زود هنگام به کمک خطای داده های تست استفاده نکرده ایم عملا مجموعه داده تست و ولیدیشن ما یک ماهیت دارند.

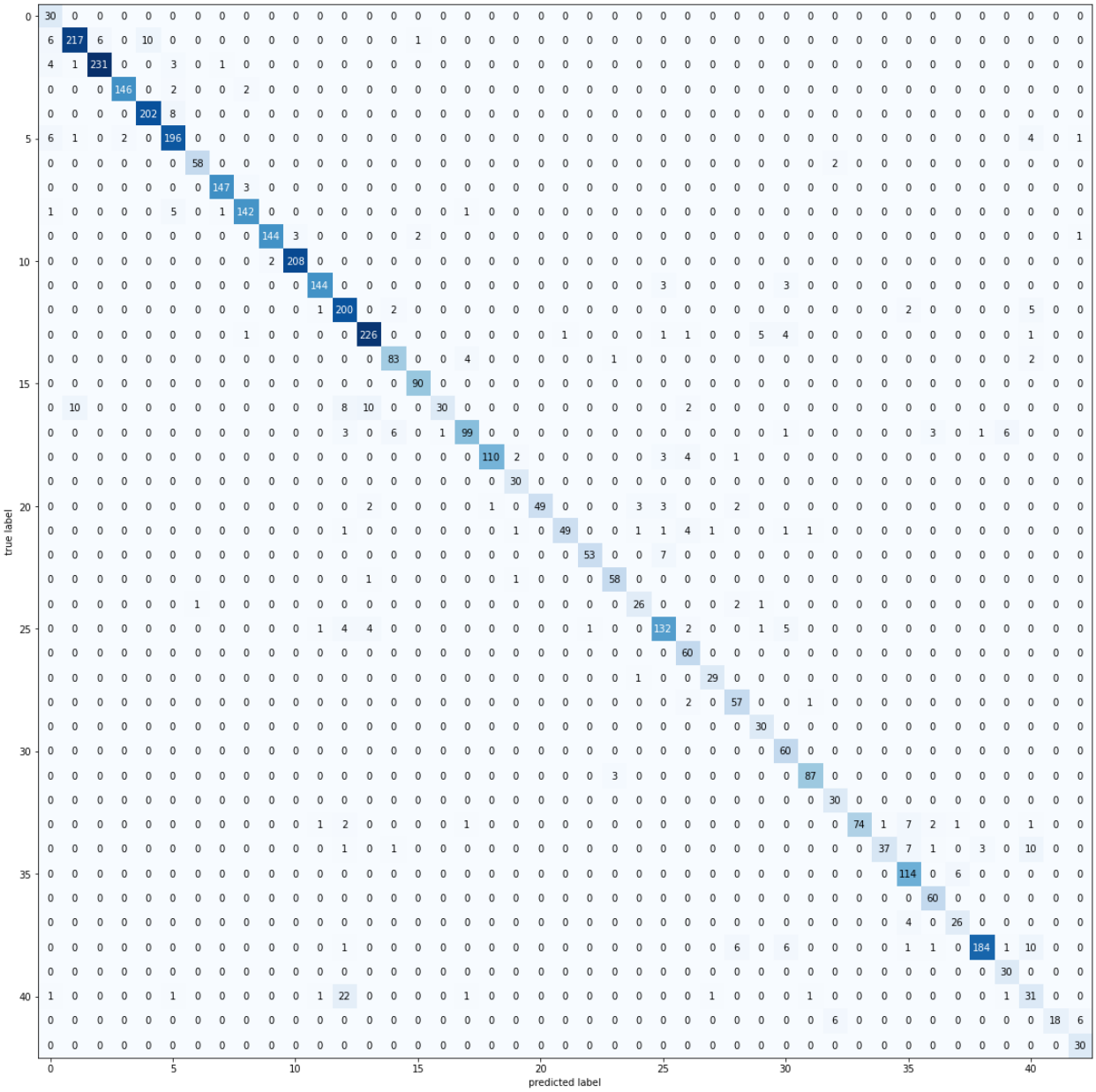
دقیقا مراحل قبلی را با همان معماری قبلی بر روی مجموعه داده های سیاه و سفید انجام دادیم:



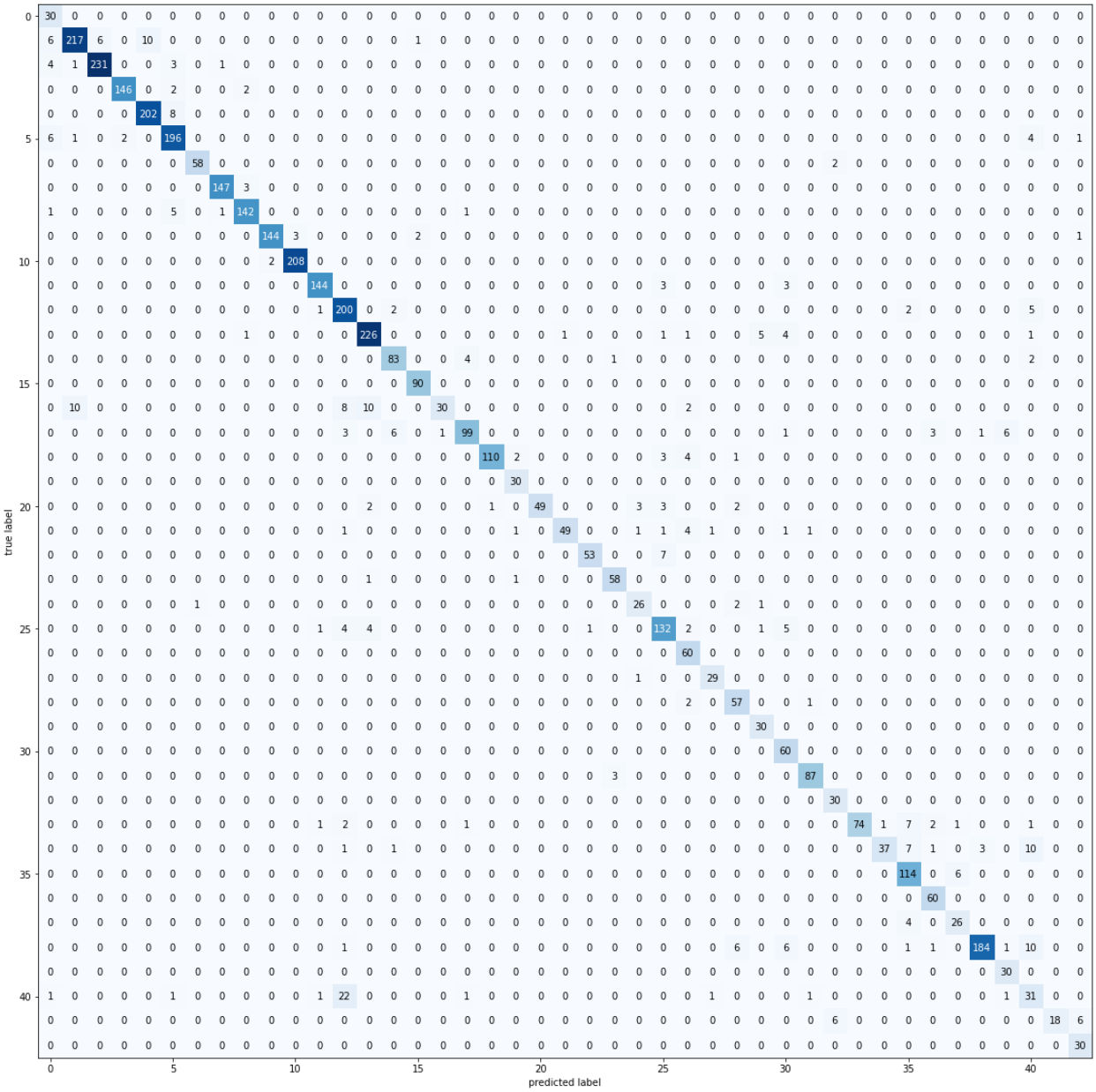
دقت مدل بر روی مجموعه داده تست و ولیدیشن سیاه و سفید به صورت زیر بود:

مجموعه داده	accuracy	F1- score macro avg	F1- score weighted avg	زمان آموزش
test	0.89	0.86	0.90	17min 42s
validation	0.92	0.90	0.92	-

ماتریس کانفیوژن مجموعه داده های تست:



ماتریس کانفیوژن مجموعه داده های ولیدیشن:



نکاتی که در قسمت قبل گفتیم اینجا هم برقرار است .

سوال دوم: آیا رنگی بودن عکس ها بر روی دقت مدل موثر است؟

بر خلاف چیزی که تصور میکردیم هر دو مدل مدت زمان برابری برای آموزش نیاز داشتند هر چند مدل رنگی تعداد پارامتر های بیشتری برای محاسبات داشت (بخاطر تعداد چنل های ورودی بیشتر در عکس رنگی ۳ چنل و در عکس سیاه و سفید ۱ چنل ورودی وجود دارد) باز هم از نظر زمانی خیلی به هم نزدیک بودند و از نظر دقت مدل رنگی با اختلاف خیلی کمی اندازه ۱ درصد از مدل سیاه و سفید بهتر بود و این هم قابل توجیه است که مدل رنگی اطلاعات بیشتری برای کار کردن داشت تقریبا ۳ برابر مدل رنگی ولی دقیت خیلی زیاد تری کسب نکرد، پس میتوان گفت با معماری مشابه و دو مجموعه داده رنگی و سیاه و سفید تفاوت چندانی وجود ندارد و با اختلاف کمی مدل رنگی برتر میباشد.

در این قسمت بر پایه تصوری که داشتیم و حقیقتی که مدل سیاه و سفید پارامتر های کمتری برای محاسبات دارد سعی کردیم معماری بهینه را پیدا کنیم و چند معماری مختلف را بررسی کردیم و عمق های متخلف و تعداد نورون های مختلف را بررسی کردیم:

مدل	معماری	accuracy
بررسی- ۱	Model((conv): Sequential((0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1)) (4): ReLU() (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)) (flatten): Sequential((0): AdaptiveMaxPool2d(output_size=1) (1): Flatten(start_dim=1, end_dim=-1)) (fc): Sequential((0): Linear(in_features=32, out_features=256, bias=True) (1): ReLU() (2): Dropout(p=0.3, inplace=False) (3): Linear(in_features=256, out_features=43, bias=True)))	**0.68**

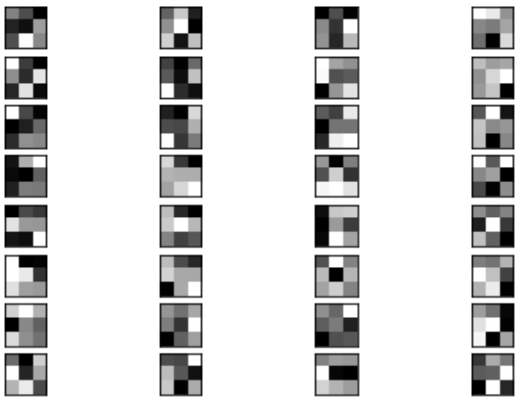
بررسی-۲	Model((conv): Sequential((0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1)) (1): ReLU() (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1)) (4): ReLU() (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)) (flatten): Sequential((0): AdaptiveMaxPool2d(output_size=1) (1): Flatten(start_dim=1, end_dim=-1)) (fc): Sequential((0): Linear(in_features=64, out_features=512, bias=True) (1): ReLU() (2): Linear(in_features=512, out_features=43, bias=True)))	**0.81**

و مشاهده کردیم که مدل منتخب ما بهترین نتیجه را بین معماری های مختلف میگیرد. (ابتدا این بخش را انجام دادیم و بعد از انتخاب معماری نسبتا خوب به ادامه بخش ها پرداختیم)

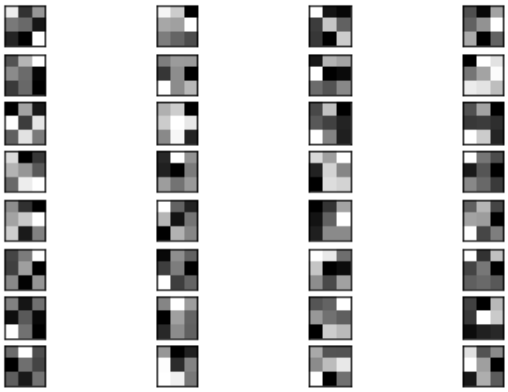
سوال ششم: خروجی هر لایه کانولوژنی و فیلتر ها را پلات کنید؟

نمونه ای از هر فیلتر هر لایه مدل منتخب را پلات کریدم:

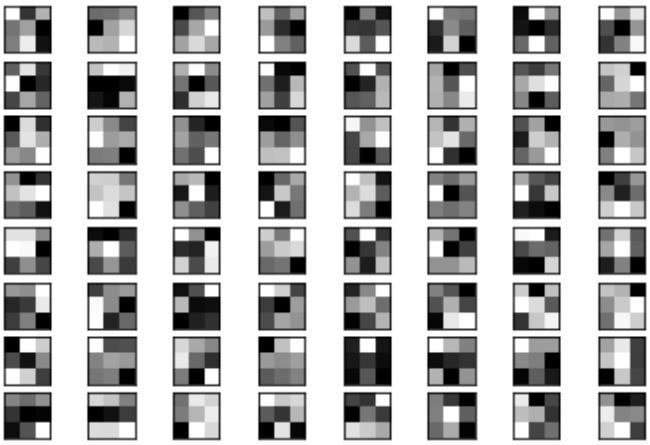
لایه اول:



لایه دوم:



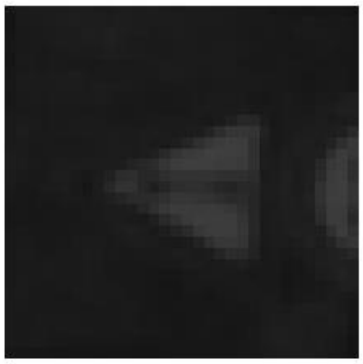
لایه سوم:



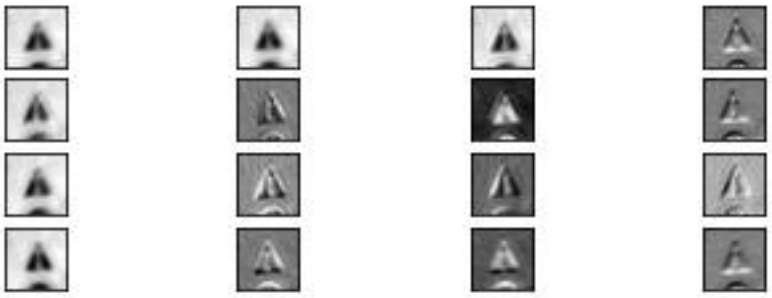
از آنجایی که فیلتر های ما سه در سه هستند خیلی قابل تشخیص نیست که هر یک از آن ها چه مفهومی را مدل را استخراج میکنند ولی چند فیلتر لایه اول بیشتر دنبال خط های صاف و تیره عمودی و افقی میگردند.

خروجی لایه های مختلف برای یک عکس ورودی:

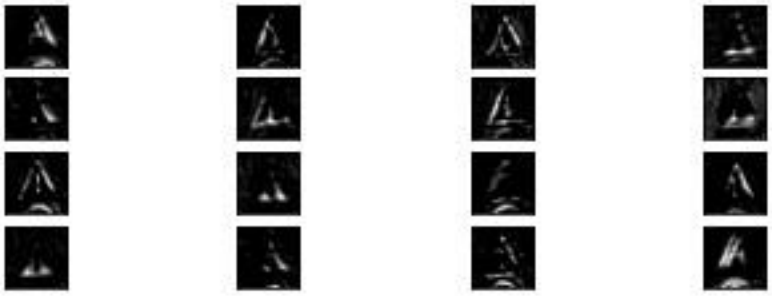
عکس ورودی:



Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))



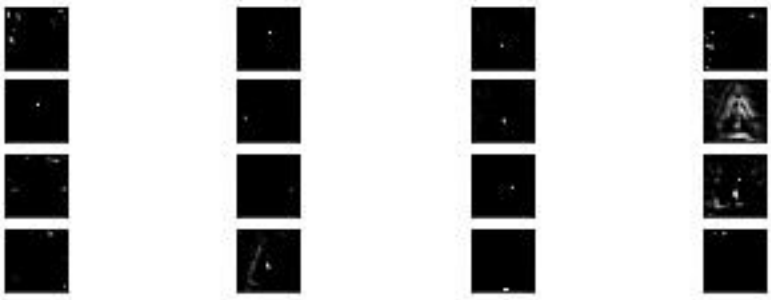
BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)



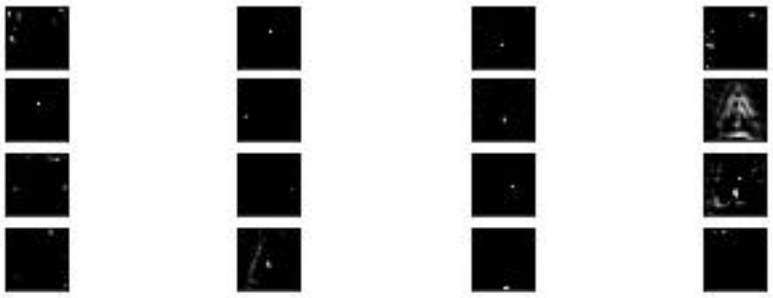
Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))

Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))

ReLU()



BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)



در نگاه اول متوجه میشویم که لایه اول وظیفه اینورینس کردن نسبت به چرخش را بر عهده دارد و در حالتی که عکس ورودی چرخش ۹۰ درجه داشته اما خروجی لایه اول عکس کاملا صاف می‌باشد.

سوال پنجم: توضیح نتایج بدست آمده بهترین مدل؟

همانطور که در بخش های قبل توضیح داده شد مدل منتخب ما با اختلاف کمی مدل رنگی بود که اشاره کردیم و نتایج آن را گزارش دادیم و تا حدی توضیح دادیم در این بخش به نتیجه کامل تر آن بر روی مجموعه داده تست میپردازیم:

Classification Report:

Class	precision	recall	f1-score	support
0	0.42	1.00	0.59	60
1	0.93	0.88	0.90	720
2	0.96	0.95	0.95	750
3	0.96	0.92	0.94	450
4	0.93	0.93	0.93	660
5	0.90	0.90	0.90	630

6	0.99	0.80	0.89	150
7	0.92	0.92	0.92	450
8	0.90	0.91	0.90	450
9	0.95	0.94	0.94	480
10	0.97	0.91	0.94	660
11	0.98	0.85	0.91	420
12	0.84	0.94	0.89	690
13	0.94	0.92	0.93	720
14	0.93	0.94	0.93	270
15	0.95	0.95	0.95	210
16	0.99	0.99	0.99	150
17	0.94	0.79	0.86	360
18	0.92	0.79	0.85	390
19	0.75	1.00	0.86	60
20	0.98	0.96	0.97	90
21	0.76	0.63	0.69	90
22	0.93	0.79	0.86	120
23	0.76	0.95	0.85	150
24	0.80	0.87	0.83	90
25	0.85	0.93	0.89	480
26	0.79	0.92	0.85	180
27	0.90	0.45	0.60	60
28	0.81	0.91	0.86	150
29	0.82	0.70	0.75	90

30	0.75	0.85	0.79	150
31	0.88	0.96	0.92	270
32	0.73	0.87	0.79	60
33	0.95	0.83	0.89	210
34	0.93	0.89	0.91	120
35	0.89	0.91	0.90	390
36	0.82	0.88	0.85	120
37	0.82	0.90	0.86	60
38	0.97	0.84	0.90	690
39	0.61	0.77	0.68	90
40	0.44	0.86	0.58	90
41	0.90	0.92	0.91	60
42	0.76	0.67	0.71	90

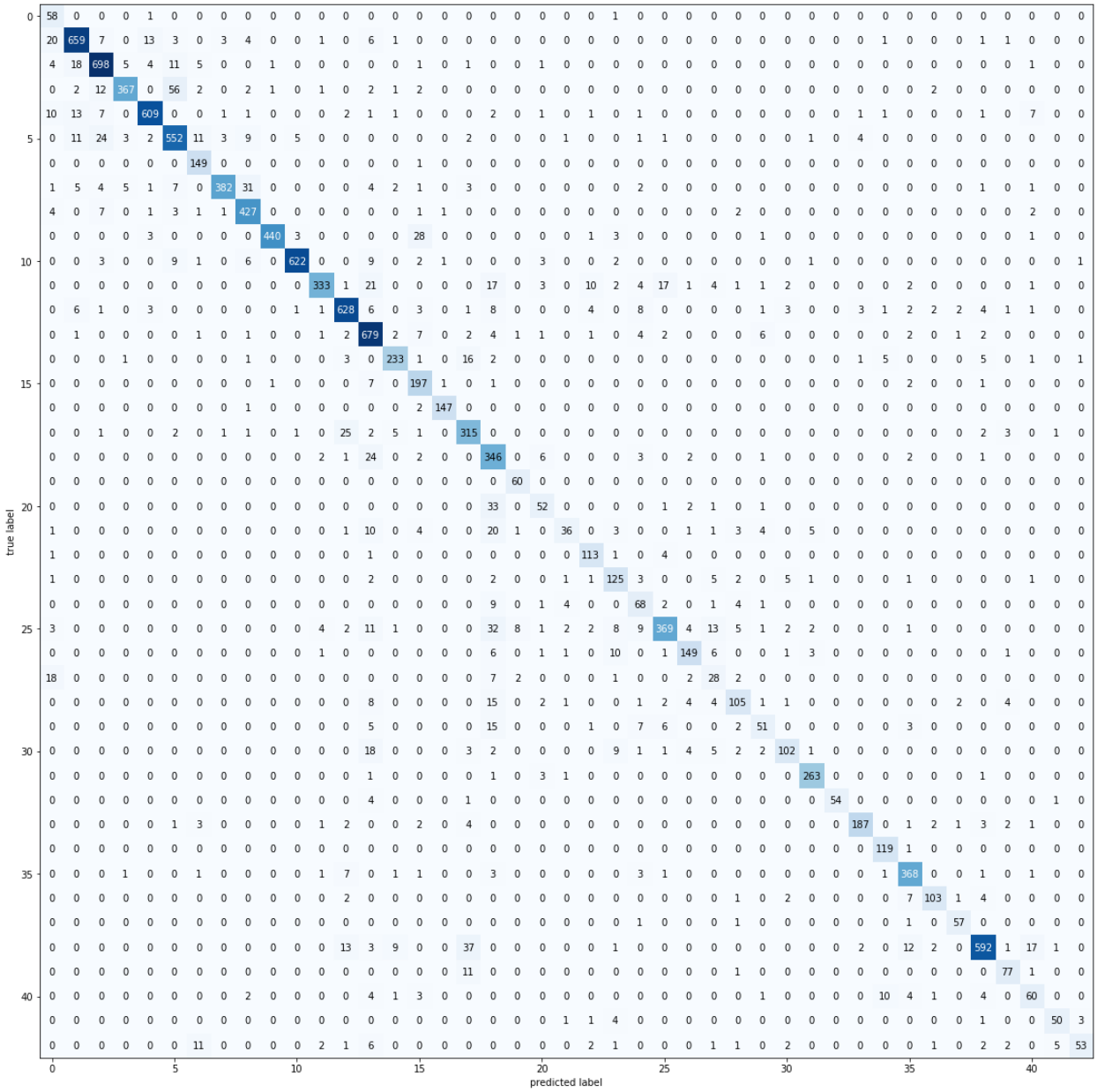
ملاک f1-score میانگین هارمونیک بین دو مقدار precisionو recall میباشد و مقدار آن هر چقدر بیشتر باید مدل کار بهتری در در طبقه بندی انجام داده است در آن کلاس خاص و همانطور که دیده میشود در کلاس های ۰ و ۲۱و۲۷و۴۰ مدل وظیفه طبقه بندی را به خوبی انجام نداده و این کلاس ها از فراوانی نسبتا کمتری نسبتا به باقی کلاس ها دارند و با توجه به اینکه دیتای مورد استواده آگمنت شده بود پس میتوان نتیجه گرفت که برای بهتر شدن نتیجه این کلاس ها یا باید به طور خاص مدلی برای این کلاس ها جدا طراحی کرد که اثر نویز آن ها از باقی کلاس ها حذف شود و یا داده های واقعی بیشتری به شبکه برای این کلاس ها اضافه کنیم و یا معماری بهتر و عمیق تری در قسمت کانوولوژنی مورد استفاده قرار بگیرد که بتوانند این کلاس ها را بهتر تمیز کنند و نتیجه بهتری بگیریم ولی به دلیل سنگینی محاسبات و مشکلات کرنل های کگل معماری عمیق تری را نتوانستیم پیاده سازی کنیم.

سوال سوم: آیا میتوان نتایج بهتری به کمک دیتا اگمنتیشن بدست آورد؟

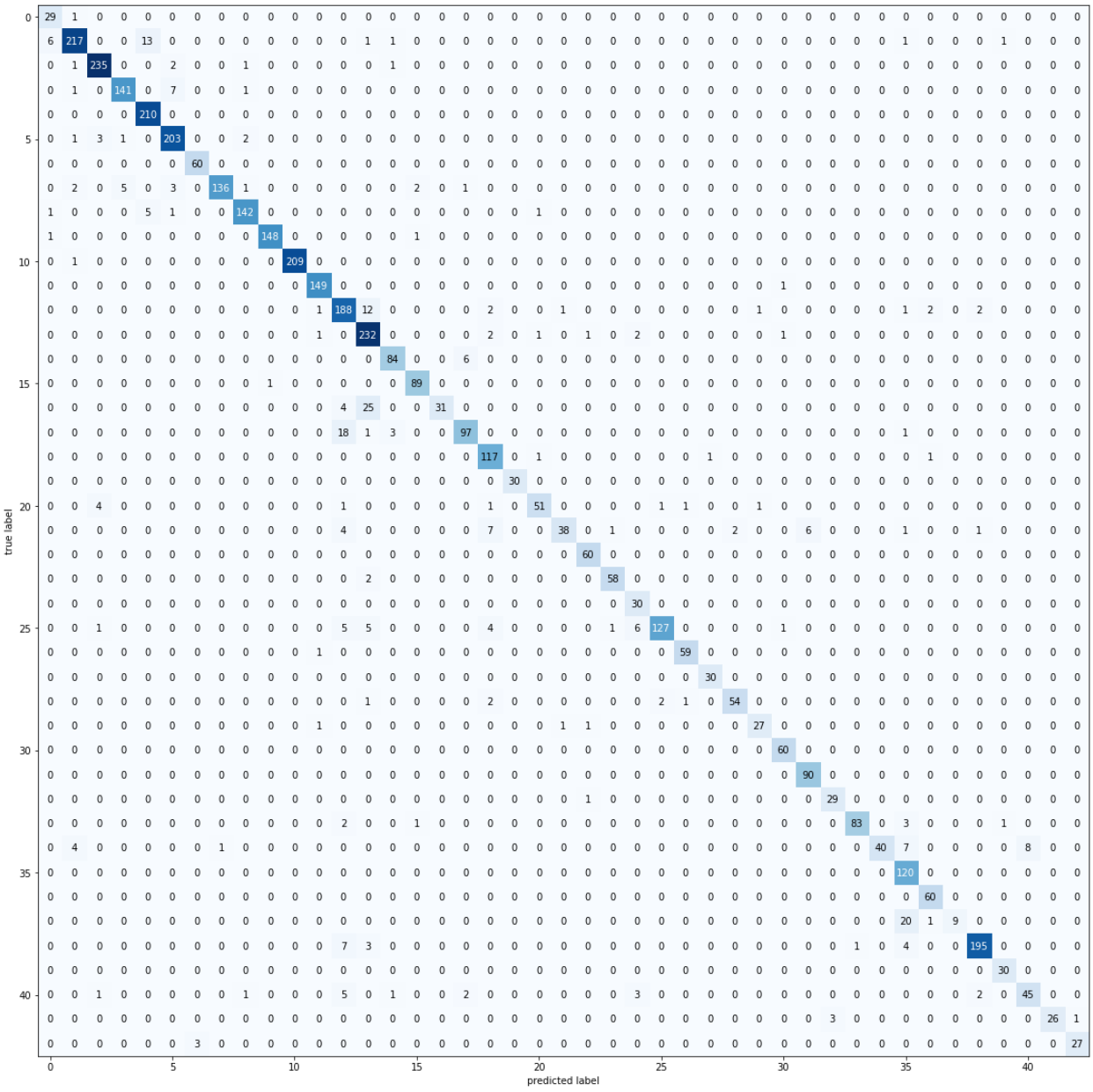
برای بررسی این سوال به کمک توابع در دسترس در پایتورچ و به طور خواص تابع blur و ShiftScaleRotate با محدودیت چرخش ۴۵ درجه ای بر روی داده ها اعمال کردیم و در پایپلاین مورد استفاده مدل در مراحل قبل این توابع را قرار دادیم و آزمایش مشابه حالت های قبل بر روی داده های سیاه و سفید با معماری منتخب انجام دادیم:

مجموعه داده	accuracy	F1- score macro avg	F1- score weighted avg	زمان آموزش
test	0.88	0.83	0.88	17min 46s
validation	0.93	0.91	0.93	-

ماتریس کانفیوژن مجموعه داده تست:



ماتریس کانفیوژن مجموعه داده ولیدیشن:



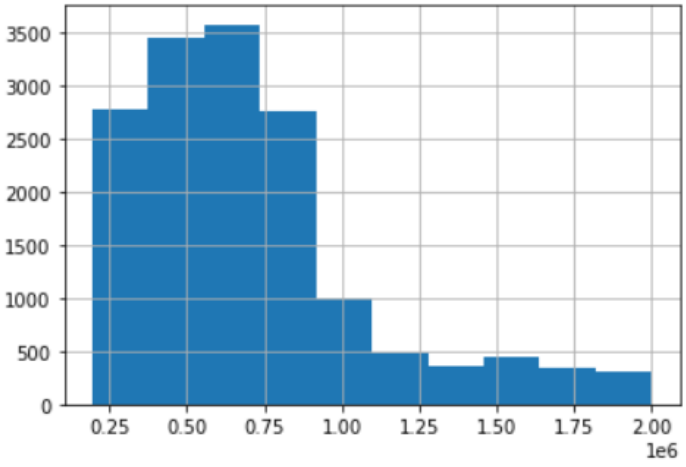
دقت بر روی مجموعه داده تست کاهش یافت و با اختلاف کمی بر روی ولیدیشن افزایش داشتیم و اگر معماری عمیق تری انتخاب کنیم میتوانیم نتیجه بهتری به کمک اگمنتیشن های جدید بگیریم و نتیجه فعلی کسب شده نسبتا بهتر از حالت قبلی است.

تمرین ۲:

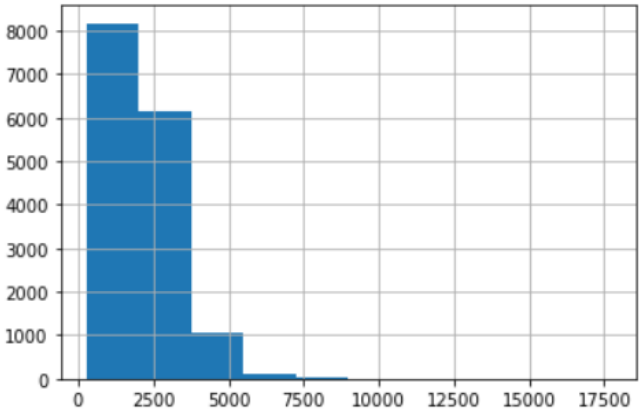
مسئله تشخیص قیمت خانه های همراه تصویر آنان؟

در این مسئله ۱۵۴۷۳ رکورد از اطلاعات خانه ها شامل عکس آن ها در ابعاد (311, 415, 3), (350, 350, 3), (350, 525, 4) و ویژگی های عددی و کتگوریکال مثل ادرس خیابان، اسم شهر، کد شهر، تعداد تخت، نوع حمام و مساحت خانه میباشد و متغیر هدف قیمت خانه میباشد در مرحله اول توزیع قیمت و مساحت های خانه هارا برای پیدا کردن داده های پرت بررسی کردیم:

قیمت:



مساحت:



و همان طور ک مشاهده میشود داده های پرتی در بخش قیمت دیده نمیشود اما از نظر مساحتی مقداری از داده ها خیلی دور تر از میانگین هستند ولی چون نمیدانیم که این املاک از چه نوعی هستند که به تمیز سازی داده ها بپردازیم داده ها را به حال خود رها میکنیم.
در قسمت تصاوری خانه ها تنها تصاویری که شامل ابعاد(311, 415, 3) میباشند را انتخاب میکنیم و باقی ابعاد که شامل ۱۷۷ داده میباشند یا عکس آن ها لود نشده یا دیفالت عکس سیستم میباشد و از داده ها دراپ میشوند.

مجموعه داده ها را با نسبت ۷۰ به ۳۰ به دو مجموعه داده تست و آموزش تقسیم میکنیم.
در مراحل پروسس عکس ها آن ها را برای اینکه بتوانیم محاسبات را بهتر انجام دهیم به ابعاد ۶۴ در ۶۴ تبدیل میکنیم.
سپس شبکه ای شامل دو شبکه کوچک تر کانولوژنال و فولی کانکت میشاسزیم که به طور همراه هم آموزش داده میشوند عکس های هر خانه به شبکه کانولوژنی داده میشود و باقی فیچر های عددی به شبکه فولی کانکت داده میشود و بر روی خروجی لایه اخر هر دو شبکه یک شبکه فولی کانکت دیگر سوار میشود و کل شبکه به کمک اپتیمایزر adam با تابع خطای mseloss و لرنینگ ریت ۰.۰۰۰۵ در روی مجموعه داده آموزش آموزش داده میشود.

معماری شبکه کانولوژنی:

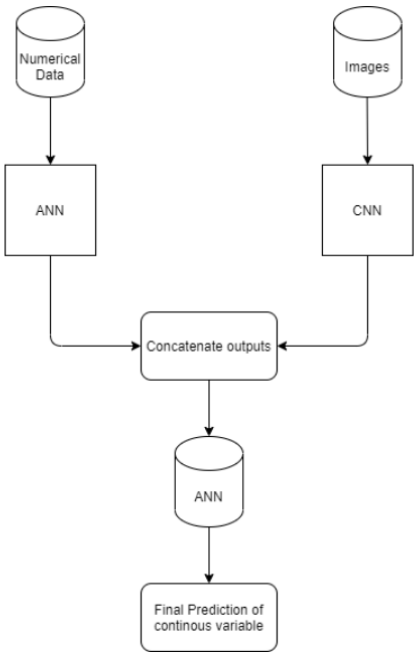
```
(conv): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
    (3): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
    (4): ReLU()
    (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
```

معماری شبکه فولی کانکت:

```
(fc): Sequential(
    (0): Linear(in_features=4, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): ReLU()
  )
```

معماری کلی شبکه:

```
Model(
  (conv): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
    (3): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
    (4): ReLU()
    (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
  )
  (flatten): Sequential(
    (0): AdaptiveMaxPool2d(output_size=1)
    (1): Flatten(start_dim=1, end_dim=-1)
  )
  (fc): Sequential(
    (0): Linear(in_features=4, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): ReLU()
  )
  (final_fc): Sequential(
    (0): Linear(in_features=192, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=1, bias=True)
  )
)
```



نتایج بدست آمده :

دیتاست	RSE	R^2	MSE	Accuracy with 10% margin
Train	309515.80	0.338	95782134943.09	0.188
Test	300817.07	0.327	90451466081.50	0.181

نتیجه گیری:

هدف این مسئله بیشتر پیاده سازی یک مدل شبکه عصبی با ورودی های متنوع و ترکیب ان ها با هم و آموزش end to end آن ها بودو مراحل لازم را با کتابخانه پایتورچ انجام دادیم و نتیجه ای که کسب کردیم خیلی دل چسب نیست و تنها ۱۸ درصد دقت با فاصله ۱۰ درصدی از جواب داریم، چند مورد را میتوان در نظر گرفت ری‌سایز کردن عکس ها مخصوصا از ابعاد خیلی زیاد به ابعاد کم مقدار خیلی زیادی از اطلاعات را از بین میبرد و برای اینکه بتوان در رم محدود با بچ سایز های نسبتا کوتاه مدل را اموزش دهیم عکس هارا کوچک کردیم و میتوانستیم یک مرحله فیچر انجنیرینگ بر روی مجموعه داده های عددی انجام دهیم مانند بین بندی مساحت خانه ها و... اما هدف مسئله رسیدن به جواب بهتر نبود و بیشتر همان پیاده سازی مد نظر بود.