دانشگاه شهید بهشتی

دانشکده علوم ریاضی

شبکه های کانولوشنی

تمرین دوم درس شبکه عصبی

نگارش

زهرا موسی خانی

استاد راهنما

دکتر سعیدرضا خردپیشه

بهار ۱۴۰۰

فهرست مطالب

فهرست شکل‌ها

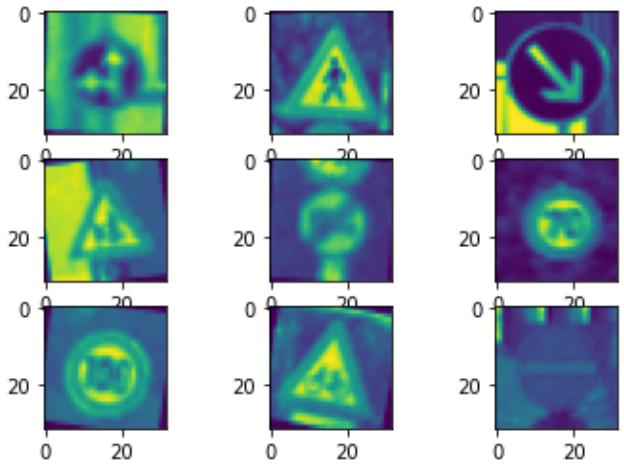
فصل ۱

تمرین اول

دیتاست این تمرین شامل عکس های تابلوهای راهنمایی و رانندگی از ۴۳ نوع تابلو مختلف می باشد و هدف طراحی شکبه عصبی ای برای طبقه بندی این عکس ها میباشد. عکس های موجود در این دیتاست دارای ابعاد ۳۲ در ۳۲ هستند. نمونه ای از عکس های موجود در دیتاست در تصویر شماره ۱ قابل مشاهده می باشد. هم چنین این مجموعه داده شامل دیتاست هایی از شماره ۰ تا ۸ میباشد که دیتاست شماره صفر مجموعه داده های خام میباشد و بقیه دیتاست ها پروسس هایی بر روی دیتا اعمال شده است. در این گزارش تاثیر مدل منتخب ما را بر روی هر ۹ مجموعه داده بررسی میکنیم و هم چنین پروسس های از جمله چرخش ۴۵ درجه ای و اضافه کردن نویز به عکس ها را انجام میدهیم تا تاثیر آنها را بر روی آموزش شبکه بررسی کنیم که در بخش پاسخ به سوال $data\ augmentation$ مفصل توضیح داده خواهد شد.

سوال : آیا سایز کلاس های دیتاست از فراوانی بالانسی برخوردار است؟ اگر نیست راه حل ارائه دهید.

با بررسی ای که بر روی فراوانی داده های هر کلاس انجام شد متوجه شدیم که کلاس های این دیتاست از فراوانی بالانسی برخوردار هستند و تمام کلاس ها شامل ۲۰۲۳ عکس می باشند. این بررسی را بر روی $y\ train$ انجام دادیم و فراوانی هر یک از لیبل ها را در این مجموعه بررسی کردیم. حال در این بخش معماری شبکه منتخب خود را ارائه میکنیم. معماری برتری که برای آموزش شبکه عصبی انتحاب کردیم شامل سه لایه کانولوشنی که بعد از لایه کانولوشنی دوم و سوم لایه $drop\ out$ و $batch\ normalization$ استفاده کردیم و دو لایه $fully$

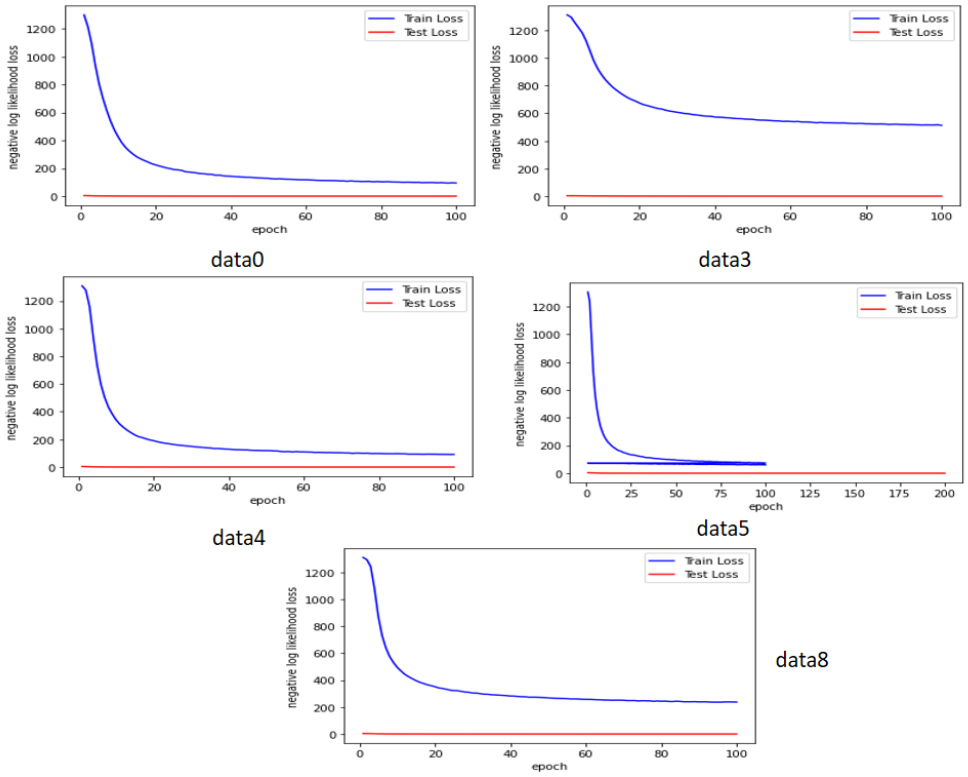


شکل ۱.۱: شکل اول : نمونه داده ها

```
CNN(
  (conv1): Conv2d(3, 12, kernel_size=(3, 3), stride=(1, 1))
  (conv2): Conv2d(12, 32, kernel_size=(3, 3), stride=(1, 1))
  (conv2_bn): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (conv3_bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3_drop): Dropout2d(p=0.5, inplace=False)
  (fc1): Linear(in_features=256, out_features=150, bias=True)
  (fc2): Linear(in_features=150, out_features=43, bias=True)
)
True
```

شکل ۲.۱: شکل دوم : معماری شبکه

$connected$ در انتها داریم. از آپتیمایزر SGD با $momentum = 0,5$, $learning\ rate = 0.01$ استفاده کردیم . هم چنین بچ سایز ۲۵۰ می باشد. معماری دقیق تر شبکه در شکل شماره ۲ قابل مشاهده می باشد. این مدل را برروی داده های شماره ۰ تا ۸ اجرا کردیم. لازم به ذکر است که پس از آزمایش همین ساختار شبکه با آپتیمایزر های متخلف و هایپرپارامتر های مختلف به این نتیجه رسیدیم که به طور میانگین برروی این ۹ سری داده پارامتر های گفته شده بهتر عمل میکنند. به تعداد ۱۰۰ ایپاک این ساختار شبکه را بر روی مجموعه داده های $data0$ تا $data8$ اجرا کردیم و روی مجموعه داده های $data1$ و $data2$ و $data6$ و $data7$ به نتایج خوبی نرسیدیم و نهایتا بعد از تعداد ۱۰۰ ایپاک به $accuracy$ ۱۵ درصد رسیدیم که خیلی مطلوب نبود. اما برروی بقیه مجموعه داده ها به نتایج نسبتا خوبی رسیدیم و مقدار $accuracy$ برای این مجموعه داده ها به مقدار ۹۴ و ۹۵ درصد رسید به جز دیتاست شماره ۳ و ۸ که به ترتیب به مقادیر ۷۰ و ۸۰ درصد رسیدند. نمودار کاهش تابع خطا را بر روی این



شکل ۳.۱: شکل سوم : مقایسه برروی دیتاهای مختلف

۵ مجموعه داده در شکل ۳ مشاهده میکنیم.

سوال : آیا رنگی بودن عکس ها تاثیری بر دقت مدل میگذارد؟ توضیح دهید.

همانطور که در شکل شماره ۳ مشاهده میشود روند کاهش تابع خطا در $data0$ که شامل عکس های خام و رنگی میباشند و هم چنین $data4$ که شامل عکس های با یک کانال رنگی می باشد تقریبا نزدیک به هم می باشد و اصلا می توان گفت عین هم عمل میکند. و هم چنین در نهایت نیست مقدار $accuracy$ حاصل بر روی مجموعه داده تست توسط این دو دیتا ست در ۳ درصد اختلاف متفاوت بودند و $data0$ مقدار بالاتر را به خود اختصاص داد. از نظر زمان آموزش هم مشابه بودند. در نتیجه برخلاف تصور ما تاثیر چندانی بر روی دقت مدل نداشت.

سوال: خروجی لایه های مختلف شبکه و هم چنین فیلتر ها را رسم کنید.

برای بررسی خروجی لایه های شبکه و هم چنین فیلتر ها خروجی حاصل از لایه های شبکه بر روی دیتاست $data0$ که دیتاست خام بود بررسی میکنیم. همان طور که در شکل چهارم مشاهده می شود با توجه به کوچک بودن فیلتر ها خیلی نمیتوان راجع به اینکه هر فیلتر چه ویژگی ای از عکس را مشخص میکند صحبت کرد اما با

توجه به رنگ های پر رنگ تر که در شکل هستند و همان طور که در شکل اول مشاهده می شد که معمولا با رنگ آبی پر رنگ جدایی اشیا از پس زمینه یا برعکس مشخص می شود به نظر می آید فیلتر لایه اول به دنبال خط های افقی و عمودی، لبه های اشیا و به خصوص لبه های تابلوها از پس زمینه باشد. پس از تشخیص این موارد گفته شده، فیلتر لایه های بعدی به دنبال پیدا کردن اشکال مختلف با جزئی تر شدن روی بخش های مختلف عکس میشوند (برای مثال بخش هایی که هم رنگ هم هستند). در شکل ۵ و ۶ به ترتیب فیلتر لایه دوم و سوم کانولوشنی را مشاهده میکنیم. حال به رسم خروجی لایه های مختلف شبکه کانولوشنی میپردازیم.

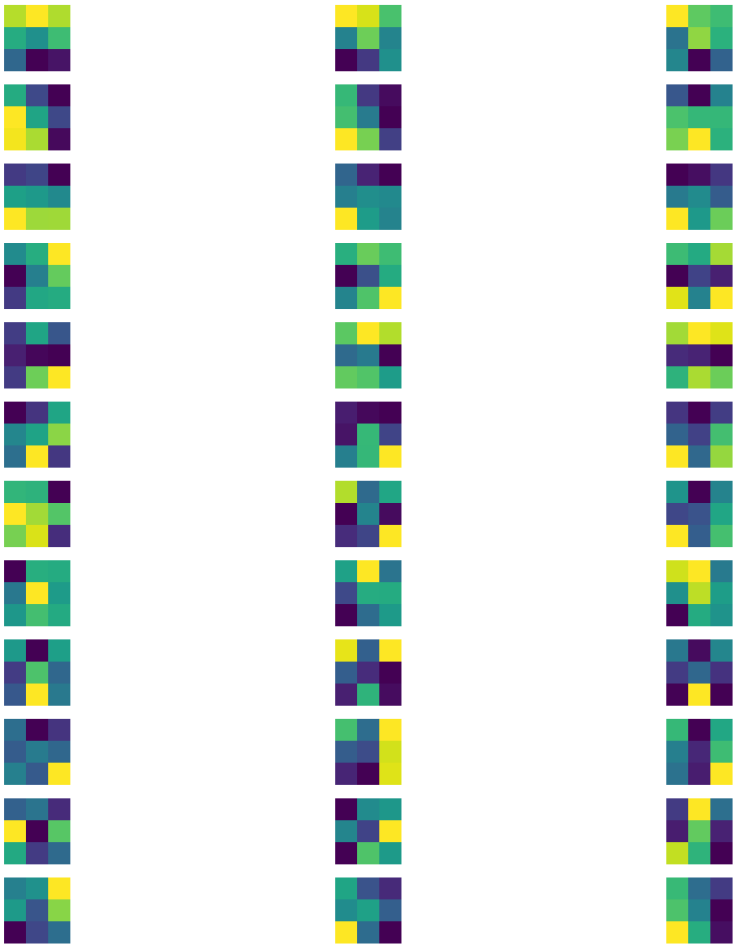
سوال: آیا $data\ augmentation$ میتواند برروی دقت مدل تاثیر بگذارد؟

در این بخش به توضیح $data\ augmentation$ برروی دقت مدل میپردازیم. همان طور که بیان کردیم، برروی $data0$ که دیتای خام بود و دیتاهای دیگر که هریک پروسسی را بر روی دیتا های ما انجام داده بودند شبکه را آموزش دادیم و دقت مدل را بررسی کردیم و دیتا هایی که بر روی آنها مدل به دقت خوبی رسید را عنوان کردیم. حال میخواهیم پروسس هایی رو بر روی دیتای خام خودمان اعمال کنیم و تاثیر آنها را برروی مدل ببینیم. پروسس هایی که ما انتخاب کردیم شامل $rotation\ ,\ flip\ left\ to\ right\ ,\ flip\ up\ down\ and\ adding\ random\ noise$ می باشند. در شکل شماره ۱۰ نمونه ای از این پروسس های انجام شده برروی داده های مسئله مشاهده می شود. ابتدا قصد داشتیم تمام این پروسس ها را برروی تک تک عکس ها ایجاد کرده و به دیتاست اضافه کرده تا به دقت خیلی خوبی برسیم. اما به علت برخوردن با محدودیت حافظه در کولب قادر به انجام این کار نشدیم. حتی اضافه کردن یک پروسس بر روی تمام عکسای دیتاست نیز به همین دلیل مشابه قابل انجام نبود. بنابراین تصمیم گرفتیم تک به تک پروسس ها را برروی دیتاست اجرا کنیم و همچنین برروی همه داده ها این کار را انجام ندهیم و برای یک دهم داده ها این کار را انجام دادیم تا نتیجه را مشاهده کنیم. یک نتیجه جالب که از اضافه کردن پروسس اول، اصل شد این بود که با اضافه کردن این پروسس به مجموعه داده های $data1$ که مدل ما برروی آن به نتایج خوبی نرسیده بود، باعث شد تا اضافه کردن این پروسس پیشرفت چشمگیری در کاهش تابع خطا و افزایش دقت (تا ۸۰ درصد) داشته باشد که میتوانیم نمودار آن را در شکل ۱۱ مشاهده کنیم. اضافه کردن هر یک از این پروسس ها، باعث شد تا سرعت بالارفتن دقت مدل به شکل چشمگیری افزایش یابد. برای مثال برای مجموعه داده های $data0$ در حالتی که دیتای خام بودند و هیچ پروسسی برروی آنها انجام نشده بود، تعداد ۵۰ ایپاک طول کشید تا

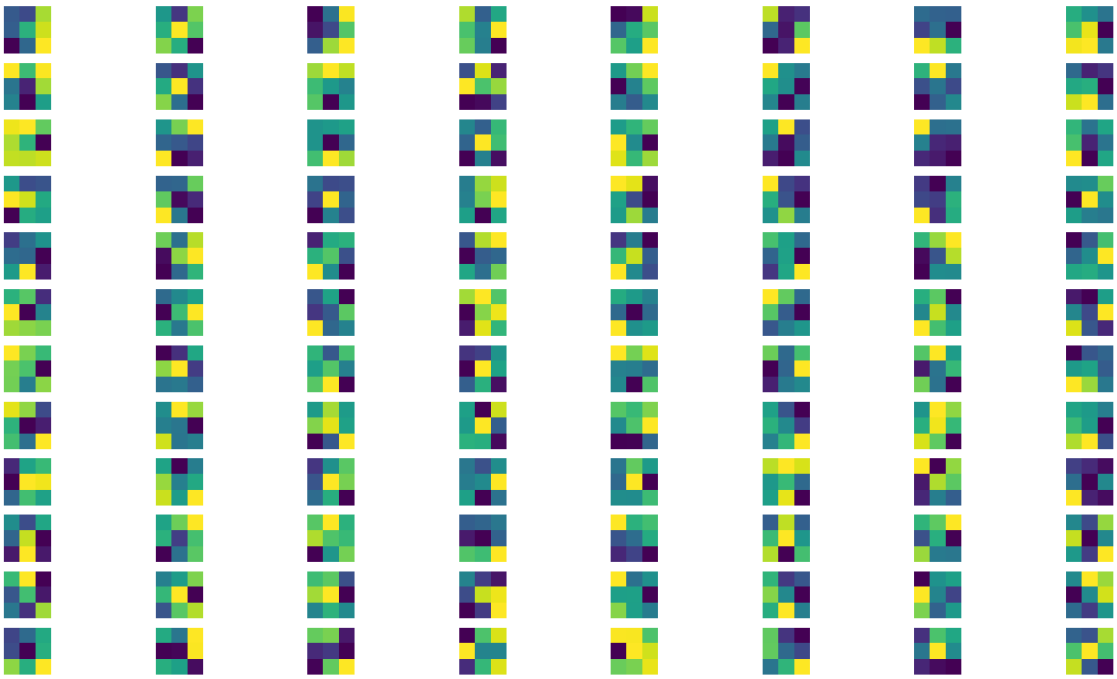
مدل به دقت نسبتا خوبی (برابر ۷۰ درصد) برسد. این درحالی است که با اضافه کردن پروسس $left\ to\ right$ $flip$ در ایپاک ۶ دقت مدل به ۵۴ درصد و در ایپاک ۲۱ دقت مدل به ۹۰ درصد رسید که بسیار قابل توجه است. برای پروسس های دیگر نیز داستان به همین روال ادامه داشت. برای مثال دیگر برای پروسس $up\ down\ flip$ در ایپاک ۵ دقت مدل به ۴۹ درصد و در ایپاک ۲۵ به ۹۰ درصد رسید. این روند برای پروسس های دیگر برروی $data0$ نیز به همین روال می باشد و نمودار کاهش تابع خطا تقریبا برای همه به یک شکل می باشد و به سرعت همگرا میشود که برای نمونه یکی از این نمودار ها در شکل شماره ۱۲ آورده شده است که در ایپاک ۲۰ مشاهده میشود که به سرعت به سمت همگرایی میرود. (دقیقا همان ایپاکی که دقت مدل به ۹۰ میرسد) حال در این مرحله به رسم کردن خروجی لایه های کانولوشنی پس از اضافه کردن پروسس $rotation$ میپردازیم و مشاهده میکنیم که در خروجی لایه اول چرخشی مشاهده نمی شود.

سوال : مدل خود را برروی داده تست ران کنید.

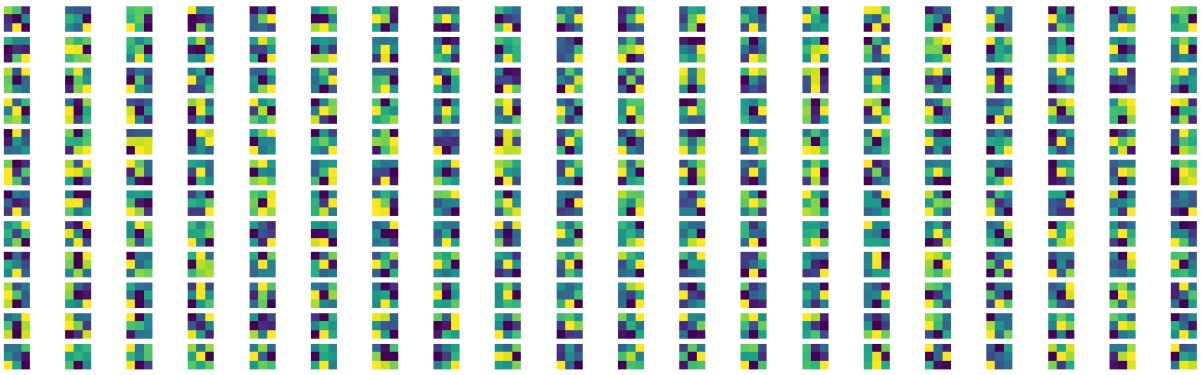
در بخش های قبل مفصل درباره ی دقت مدل برروی داده تست، سرعت افزایش دقت مدل و تاثیر پروسس های مختلف و حتی رنگی بودن یا نبودن عکس ها برروی این دقت صحبت کردیم.



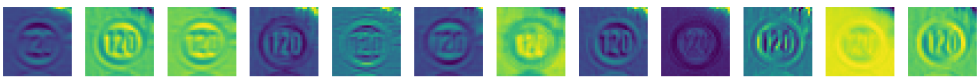
شکل ۴.۱: شکل چهارم : فیلتر لایه اول شبکه



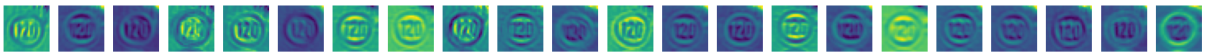
شکل ۵.۱: شکل پنجم : فیلتر لایه دوم شبکه



شکل ۶.۱: شکل ششم : فیلتر لایه سوم شبکه

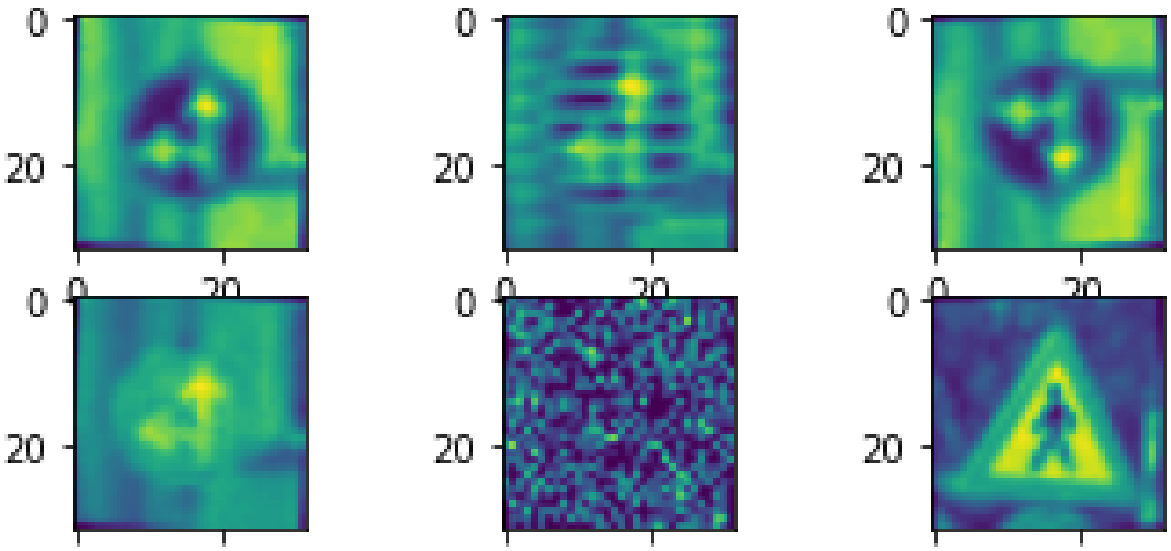


شکل ۷.۱: شکل هفتم : خروجی لایه اول شبکه

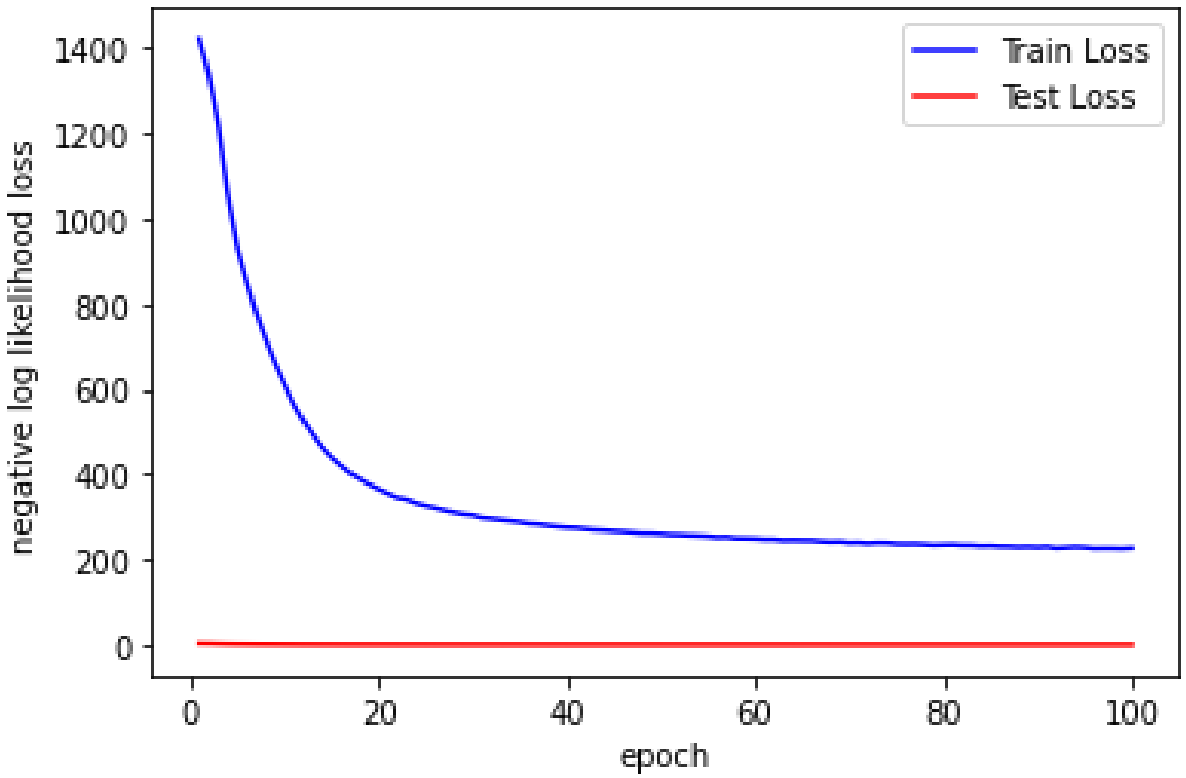


شکل ۸.۱: شکل هشتم : خروجی لایه دوم شبکه

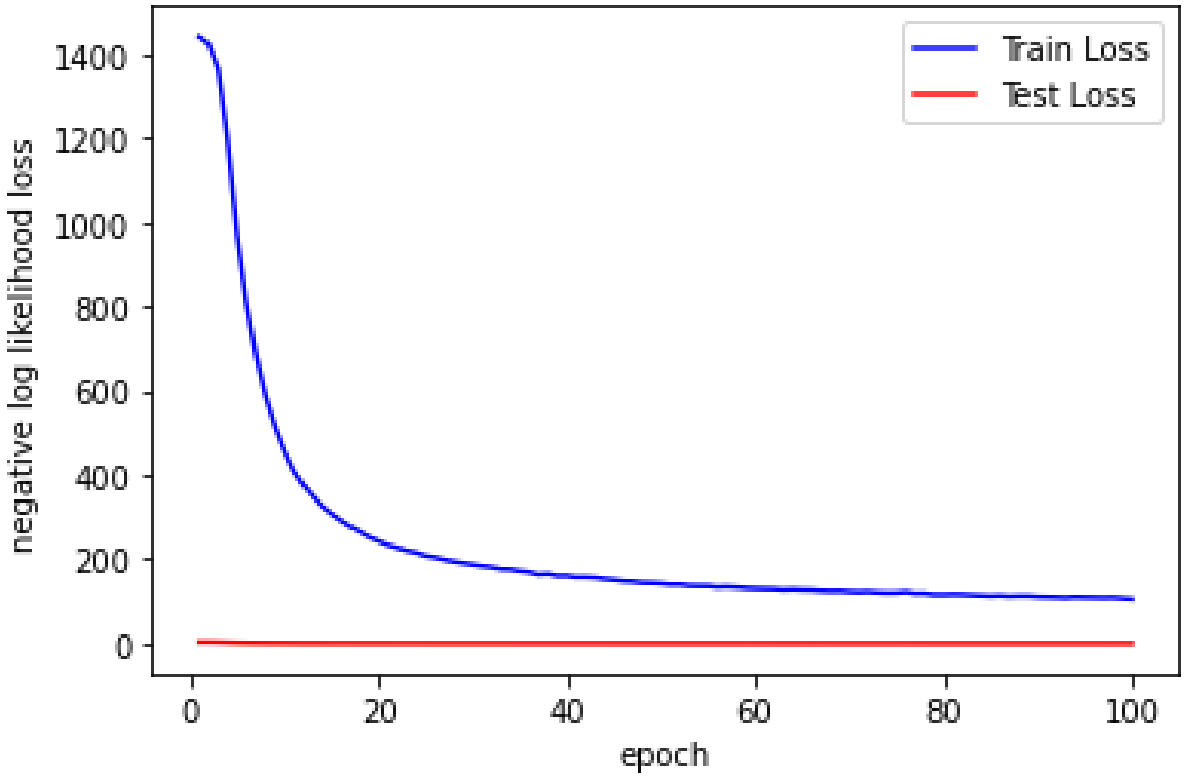
شکل ۹.۱: شکل نهم : خروجی لایه سوم شبکه



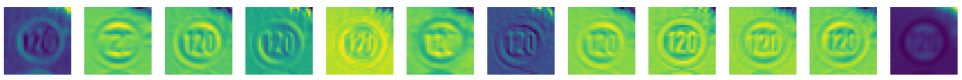
شکل ۱۰.۱: شکل دهم : نمونه پروسس برروی داده اصلی مسئله



شکل ۱۱.۱: شکل یازدهم : تاثیر اضافه کردن پروسس اول بر دیتای ۱

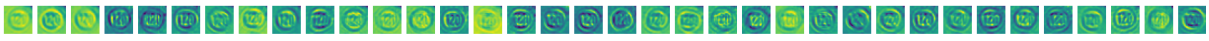


شکل ۱۲.۱: شکل دوازدهم : تاثیر اضافه کردن پروسس بر روی دیتای خام



شکل ۱۳.۱: شکل سیزدهم : خروجی لایه اول بعد از اضافه کردن روتیشن

شکل ۱۴.۱: شکل چهاردهم : خروجی لایه دوم بعد از اضافه کردن روتیشن



شکل ۱۵.۱: شکل پانزدهم : خروجی لایه سوم بعد از اضافه کردن روتیشن