به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

مینی پروژه سری اول

نام و نام خانوادگی	سینا شریفی	صدف صادقیان
شماره دانشجویی	۸۱۰۱۹۵۴۱۲	۸۱۰۱۹۵۴۱۹
تاریخ ارسال گزارش	۱۳۹۹/۲/۳	

فهرست گزارش سوالات

سوال ۱ – بخش تشریحی

۱) برای هریک از مسائل زیر دو نمونه تابع هزینه مناسب نام برده و عملکرد آنها را مختصرا توضیح دهید.

Regression Problems : یکی از توابع هزینه مناسب MSE (Mean Squared Error) است که به صورت زیر محاسبه می‌شود:

$$MSE(y.y^p) = \frac{\sum_{i=1}^{n}(y_i - {y_i}^p)^2}{n}$$

در این تابع زمانی که مقدار پیش بینی شده با مقدار واقعی برابر باشد کمترین مقدار را داریم و به نسبت هر چقدر دو شویم خطا اضافه می‌شود. همچنین از آن جایی که میانگین گرفته شده است پس برای مقایسه بهتر است و به علت نرمال شدن دیگر وابسته به تعداد داده‌هایمان نیست. همچنین گرادیان این تابع نیز مناسب است چرا که هر چقدر از مینیمم(جایی که مقدار پیش‌بینی شده با مقدار واقعی برابر است) دور باشیم گرادیان بیشتر بوده و با نزدیک شدن به مینیمم گرادیان به مرور کاهش پیدا می‌کند تا در مینیمم به صفر می‌رسد. پس MSE به ما راه‌حلی پایدار تر و با فرم بسته(مشتق=۰) به ما می‌دهد.

Classification Problems : می‌توان از تابع هزینه Categorical Crossentropy زمانی که هر داده تنها یک لیبل درست دارد استفاده کرد که به شکل زیر محاسبه می‌شود:

$$CC(y.y^p) = -\sum_{j=1}^{m}\sum_{i=1}^{n} y_{ij} * \log({y_{ij}}^p)$$

در این تابع کلاس درست به شکل یک بردار one hot دیده می‌شود و هر چه خروجی به این بردار نزدیک تر باشد، loss کمتری خواهیم داشت.

۲) منظور از توابع بهینه‌سازی مرتبه اول و مرتبه دوم چیست؟ تفاوت آنها را بیان کرده و از هر کدام یک مثال بزنید.

به توابعی که برای پیدا کردن نقطه‌ی بهینه فقط از مشتق اول توابع هزینه استفاده میکنند، درجه‌اول و به هر تابعی که حدقل از یکی از مشتقات درجه‌دوم هم استفاده کند، تابع بهینه‌سازی مرتبه دوم میگویند.

این یعنی توابع مرتبه اول، فضا را با یک تابع مرتبه یک تخمین زده و تلاش در کاهش آن دارند ولی توابع مرتبه دو، فضا را با یک تابع مرتبه دو تخمین میزنند یعنی عموما هم از مقدار گرادیان و هم از مقدار

هسین فضا در آن نقطه برای تخمین فضای اطراف خود استفاده میکنند. به همین دلیل توابع مرتبه دو عموما روش های قدرتمند تری برای بهینه‌سازی هستند ولی هنوز معمولا در شبکه های عصبی از روش کاهش گرادیان یا سایر روش های مبتنی بر آن استفاده میشود.
معروف ترین مثال برای تابع بهینه‌سازی مرتبه اول روش‌های مبتنی بر گرادیان هستند و یکی از معروف‌ترین روش های بهینه سازی مرتبه‌ی دوم، روش نیوتن می‌باشد.

۳) مشکل Overfitting را توضیح دهید .همچنین سه روش برای جلوگیری از Overfitting بیان کرده و نحوه عملکرد آنها را توضیح دهید.

هدف ما در machine learning تولید مدلی است که تقریبا با همان دقتی که روی داده‌های آموزش عمل می‌کند، روی داده‌هایی که آن‌ها را در فرآیند آموزش ندیده است هم دقت داشته باشد .
overfittingیک مشکل رایج در مسائل machine learning است به این صورت که شبکه بیش از حد جزئیات داده‌های آموزش را یاد می‌گیرد به همین دلیل دقت خیلی خوبی روی داده‌های ترین پیدا میکند ولی دقت آن روی داده‌های تست بسیار پایین می‌باشد به این علت که توانایی تعمیم ندارد و زیاد از حد رویnoise ها و جزئیات داده train مان fit شده است که روی پیش‌بینی اش برای داده‌های بیرون از آن مجموعه تاثیر منفی می‌گذارد.

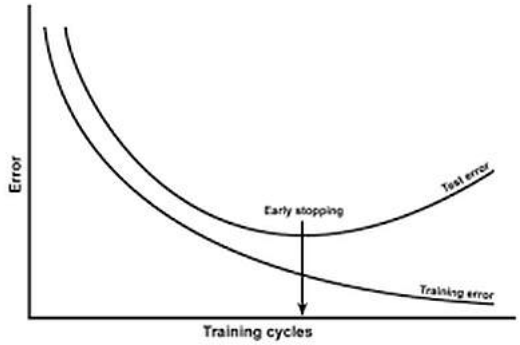
روش اول: regularization term

یکی از دلایل overfitting میتواند بزرگ شدن بعضی از وزن‌های شبکه باشد به این معنی که بعضی از ویژگی‌ها تاثیر خیلی زیادی در خروجی داشته باشند در حالتی که بعضی از وزن‌ها بی‌تاثیر باشند.
ایده‌ی روش regularization term این است که یک جمله به cost function اضافه کنیم، که در نتیجه در هر مرحله‌ی optimization علاوه بر کاهش مقدار خطای تابع، اندازه‌ی وزن‌های شبکه را نیز کاهش دهیم .
این جمله میتواند به فرم $\sum \omega_i^2$ به cost function اضافه شود.

روش دوم: Early stopping

یک روش regularization برای مدل‌هایی که روش یادگیری iterativeای دارند، است. در مدل‌هایی که از gradient descent استفاده می کنند روش به این شکل است که مدل را به روز رسانی می‌کند تا در هر iteration مدل به دیتا آموزش بهتر fit شود. تا نقطه ای این اتفاق باعث بهتر شدن عملکرد

مدل روی داده تست هم می‌شود اما پس از این نقطه بیشتر fit شدن مدل به دیتا train باعث افزایش خطای مدل روی داده تست و کم شدن generalization مدل می‌شود.



شکل ۱- نمودارoverfitting و روش early stopping

همان طور که در شکل بالا قابل ملاحظه است بعد از نقطه ای با اینکه خطا روی داده train در حال کاهش است، خطا روی داده تست افزایش میابد که این نشان دهنده این است که مدل در حال overfit شدن به داده آموزش است. ما با متوقف کردن فرآیند آموزش در این نقطه می‌توانیم از overfit شدن مدل مان جلوگیری کنیم.

روش سوم dropout :
همانطور که قبلا هم اشاره شد یکی از دلایل overfitting میتواند تاثیر زیاد بعضی از ویژگی‌ها و بی‌تاثیر بودن بعضی دیگر از آنها اشاره کرد.
ایده‌ی اصلی این روش این است که با حذف تصادفی بعضی از نورون‌ها از شبکه، این شرایط را از بین ببرد.
برای مثال اگر یک نورون خیلی مهم حذف شود، چون مقدار loss خیلی بالا میرود، به همان نسبت نورون‌هایی که تا به حال تاثیر زیادی روی خروجی نداشته‌اند، train میشوند.
پس در مجموع در این روش همه‌ی وزن‌ها تاثیر نسبتا مشابهی روی ساخته شدن خروجی دارند و در نتیجه از overfitting جلوگیری میشود.

۴) مشکل استفاده از توابع خطی در لایه‌های مخفی شبکه‌های عصبی عمیق چیست؟ همچنین سه نمونه از توابعی که در شبکه‌های عصبی عمیق استفاده میشود را نام برده و هرکدام را به همراه مزایا و معایب توضیح دهید.

در مورد استفاده از توابع خطی به عنوان activation function میتوان به دو مورد اشاره کرد، اول اینکه این توابع به علت داشتن مشتق مساوی در همه‌ی نقاط، کار یادگیری را در مرحله back propagation دشوار میکنند و علاوه بر این در این شرایط، خروجی هر لایه فقط برابر ترکیب خطی ورودی‌های آن لایه است پس در نهایت لایه‌ی آخر نیز حاصل ترکیب خطی بردار ویژگی میشود در نتیجه فارغ از تعداد لایه‌ها، شبکه ما در اصل معادل یک شبکه تک لایه است.

به همین علت در شبکه‌های عمیق از توابع غیرخطی استفاده میشود.

تابع اول Tanh:

از مزایای این تابع میتوان به مشتق‌پذیر بودن در تمامی نقاط و zero-centered بودن اشاره کرد به این معنا که تابع در نواحی مثبت و منفی متقارن است.

از معایب این تابع میتوان به سنگینی محاسبات و اشباع تابع در نواحی خیلی کوچک یا خیلی بزرگ اشاره کرد.

تابع دوم Relu :

از مزایای این تابع میتوان به سادگی در محاسبه و اشباع نشدن (برعکس سیگموید و tanh) اشاره کرد.

از معایب این تابع میتوان به مشتق نداشتن در صفر و صفر بودن مشتق در سمت چپ محور اشاره کرد.

تابع سوم Leaky Relu:

این تابع شامل تمامی مزایای Relu میباشد با این تفاوت که برخلاف Relu مشتق آن سمت چپ محور برابر صفر نمیباشد و این موضوع به آموزش تابع کمک میکند.

۵) مفهوم Data Augmentation در شبکه‌های عصبی عمیق به چه معناست؟

در حالت کلی عملکرد شبکه‌ها، به‌خصوص شبکه‌های cnn خیلی وابسته به کیفیت دیتاست آن‌ها است. یکی از روش‌ها برای بالا بردن کیفیت دیتاست، روش data augmentation می‌باشد. این روش با

تولید دیتای جدید از روی دیتاهای قبلی، باعث بزرگ شدن دیتاست می‌شود.
علاوه بر این موضوع با ایجاد نویز روی ورودی شبکه، از overfit شدن هم جلوگیری میکند و باعث generalizationبیشتر شبکه می‌شود.

به وسیله data augmentation در هر ایپاک روی هر تصویر یکسری تغییر بصورت رندم اعمال می‌شوند. در نتیجه تصاویری که شبکه در هر ایپاک با آن‌ها train می‌شود تغییراتی جزئی با ایپاک‌های دیگر دارد که به شکل regularization کار می‌کند و باعث می‌شود که شبکه روی دیتا آموزش مان overfit نشود و باعث generalization بیشتر شبکه می‌شود.
روش‌های مختلفی برای بزرگ‌تر کردن دیتاست وجود دارد، چندمورد از روش‌های معروف شامل zoom کردن، flipکردن، cropکردن و غیره می‌باشند. هدف از هریک از این موارد، ساخت دیتای جدیدی است که با داشتن تغییرات جزئی نسبت به دیتای اولیه، مانع از اتفاق افتادن overfit روی داده‌های trainاولیه شود.

۶) Batch Normalization چیست و نحوه کارکرد و اهمیت آن در شبکه‌های عصبی را توضیح دهید.

batch normalization عمل normalization را روی ورودی hidden layerها با مقادیر میانگین و واریانس همان batch انجام می‌دهد.

مشکلی که در شبکه‌های عصبی وجود دارد internal covariate shift است. این مشکل به این علت وجود دارد که در مرحله آموزش هر لایه سعی می‌کند خود را بر اساس خطای به وجود آمده به روز رسانی کند. حال به علت اینکه این به روز رسانی‌ها مستقل از هم انجام می‌شوند پس به علت به روزرسانی وزن‌های لایه‌های پیشین توزیع ورودی‌های لایه عوض می‌شود و لایه باید فرآیند آموزش را با ورودی جدید تکرار کند که این می‌تواند باعث طولانی شدن فرآیند آموزش شود.

برای حل این مشکل لایه‌های normalization را بین لایه‌ها اضافه می کنیم و به علت لایه‌های normalization محدوده توزیع ورودی هر لایه مستقل از تغییر وزن‌های لایه‌های قبلی ثابت می‌ماند در نتیجه تغییر زیادی در ورودی لایه‌ها رخ نخواهد داد. در نتیجه لایه‌های شبکه می‌توانند همزمان آموزش داده‌شوند و لازم نیست منتظر آموزش دیدن لایه‌های پیش از خود باشند که باعث افزایش سرعت یادگیری شبکه خواهد شد.

سوال ۲ – بخش عملی

در ابتدای کار پس از دانلود داده های German Traffic Sign Recognition با استفاده از دستور resize از کتابخانه‌ی cv2 اندازه‌ی همه آنها را به ۳۰*۳۰ پیکسل RGB که معادل ۳۰*۳۰*۳ است، تغییر میدهیم.

پس از طراحی شبکه ۳لایه‌ی CNN و یک شبکه ۳ لایه‌ی Fully connected در ادامه‌ی آن چندین دسته پارامتر مختلف را بررسی می کنیم تا به دقت خوبی برسیم.

در طراحی شبکه برای مقدار دهی اولیه از تابع normal با seed = 5 استفاده شد تا همه‌ی نتایج مشابه هم باشند. و برای اسکیل کردن از max pooling با اندازه پنجره ۲ در ۲ استفاده شد.

الف) مشخصات شبکه خود را در گزارش بنویسید .

مشخصات شبکه به صورت زیر میباشد:

تعداد فیلتر در لایه اول ۳۲ و در لایه دوم و سوم ۶۴

اندازه پنجره کانولوشن ۳ در ۳ و stride برابر (1, 1)

تابع فعالسازی برای همه‌ی لایه‌ها به جز لایه آخر relu و برای لایه ی آخر softmax

در شبکه ۳ لایه‌ی Fully connected استفاده شد، به ترتیب با تعداد نورون های ۵۱۲، ۲۵۶ و43

برای تابع هزینه از categorical cross entropy و برای روش بهینه سازی از adam استفاده شد

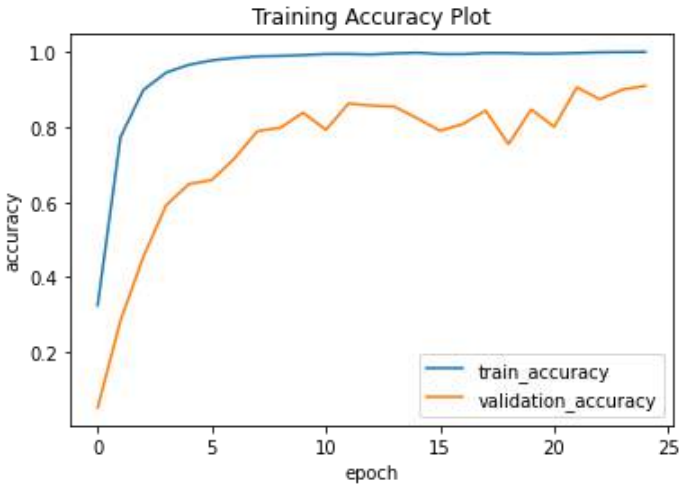
اندازه‌ی هر mini batch ۲۵۶

تعداد Epoch برابر با ۲۵

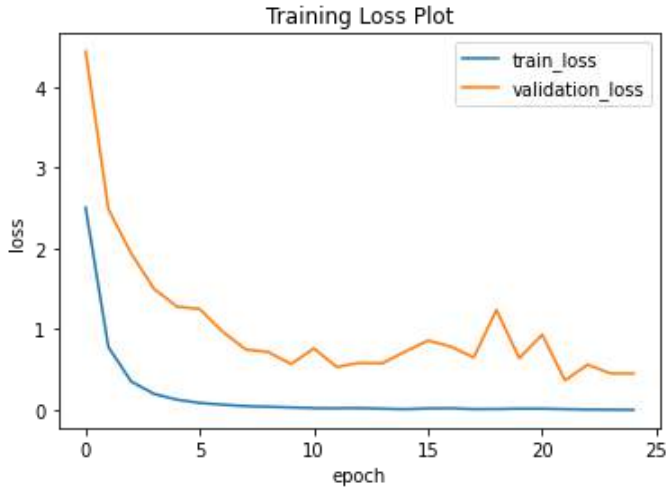
ب -شبکه عصبی که طراحی نموده اید را اجرا کرده و نمودار Accuracy و loss را برای داده‌های train و validation رسم کنید. همچنین مقدار Accuracy محاسبه شده برای داده‌های تست را گزارش کنید .

نتایج حاصل از train کردن با validation split = 0.2 :

نمودار دقت و loss در شکل ۱ و ۲ مشخص شده است.



شکل ۲- نمودار دقت داده های برای ۲۵ epoch

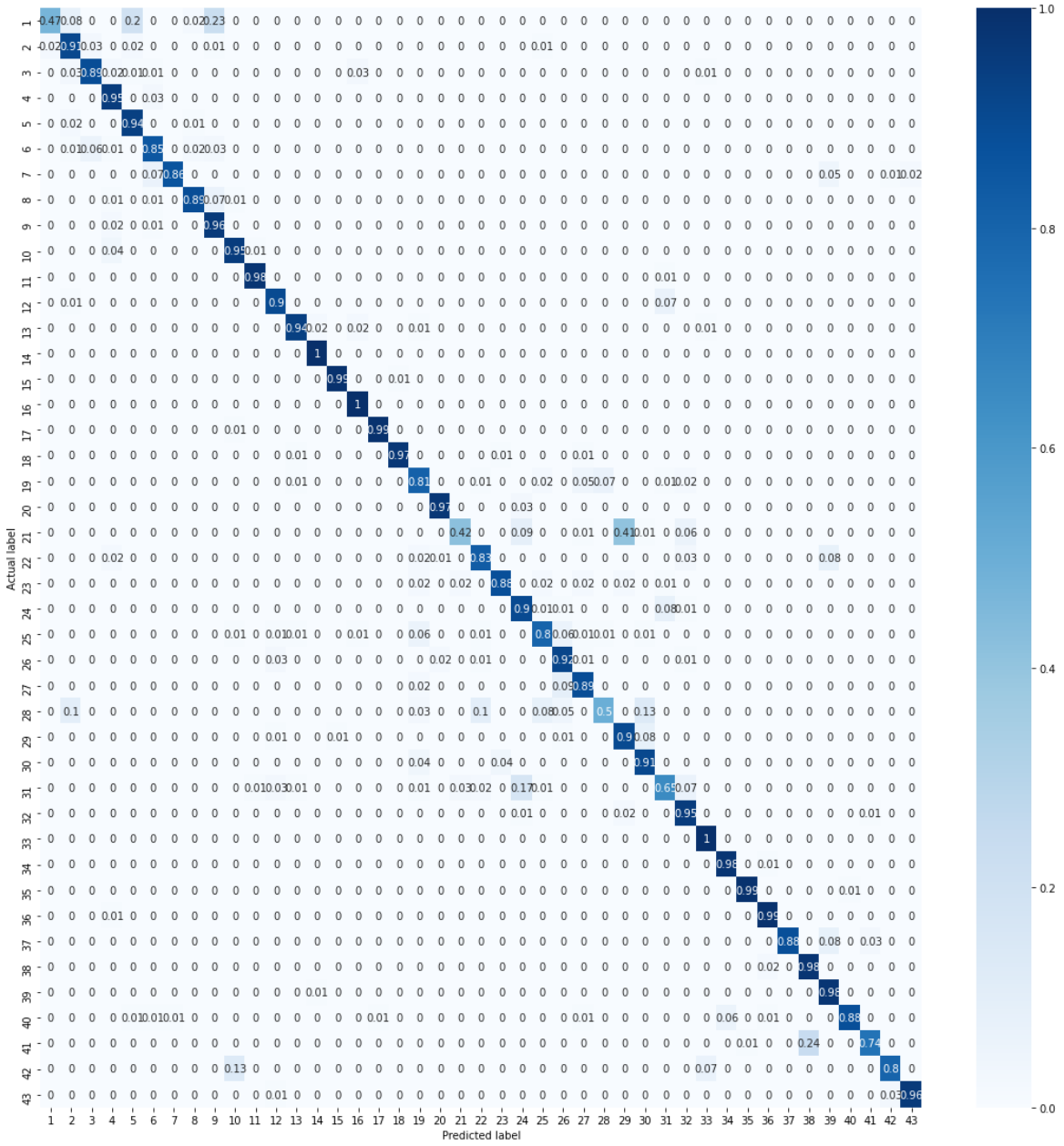


شکل ۳- نمودار loss داده های برای ۲۵ epoch

در انتها دقت روی داده های تست بررسی شد که مقدار آن ۹۱,۹۹٪ بدست آمد.

پ) Confusion matrix را محاسبه کرده و در گزارش بنویسید .

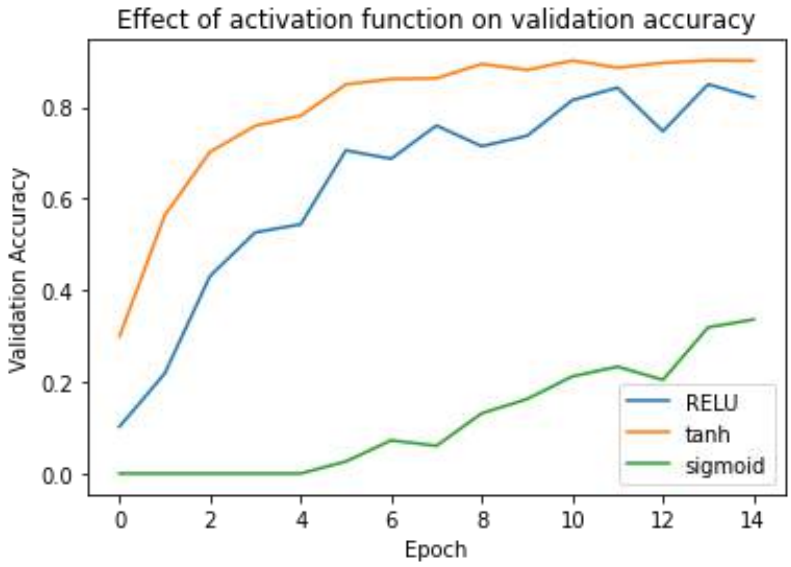
در شکل ۳ ماتریس آشفتگی حاصل مشخص شده است.



شکل ۴- ماتریس آشفتگی شبکه قسمت الف

ت) از توابع فعالساز مختلف استفاده کنید. (RELU, tanh, sigmoid) **نمودار را برای ۱۵ ایپاک رسم کنید و نتایج را تحلیل کنید.**

نمودار مقایسه‌ی دقت شبکه روی داده های validation به ازای activation function های مختلف در شکل ۴ مشخص شده است. در این قسمت سوال فقط توابع فعالسازی تمامی لایه‌ها به جز لایه آخر را تغییر دادیم.



شکل ۵- نمودار مقایسه‌ی دقت شبکه به ازای activation function های مختلف

در ادامه برای مقایسه بهتر، دقت روی داده های تست را هم گزارش میکنیم.

دقت برای شبکه با تابع فعالسازی Relu: ٪۸۸,۵۲

دقت برای شبکه با تابع فعالسازیTanh: ٪۹۲,۱۴

دقت برای شبکه با تابع فعالسازی Sigmoid: ٪۷۰,۴۷

از نمودار و نتایج بالا نتیجه میگیریم که tanh بهترین تابع فعالسازی و تابع sigmoid ضعیف ترین تابع بوده است.

ضعف sigmoid به دلیل این است که تابع برای مقادیر بزرگ مثبت و منفی، اشباع شده و در نتیجه گرادیان در این بخش‌ها تقریبا صفر است و در زمان backpropagation در گرادیان خروجی ضرب شده و مقدار نهایی بسیار کوچک می‌شود که باعث می‌شود شبکه چیزی یاد نگیرد. به این مشکل vanishing gradient گفته می‌شود که در Relu حل شده است. همچنین مشکل دیگر sigmoid این است که zero centered نمی‌باشد.

در مقایسه relu و tanh میتوان به موضوع صفر بودن مشتق relu در ناحیه منفی اشاره کرد که باعث مشکل dying Relu می‌شود که در آن تعدادی از نورون‌های شبکه inactive می‌شوند و در فرآیند یادگیری شرکت نمی‌کنند.

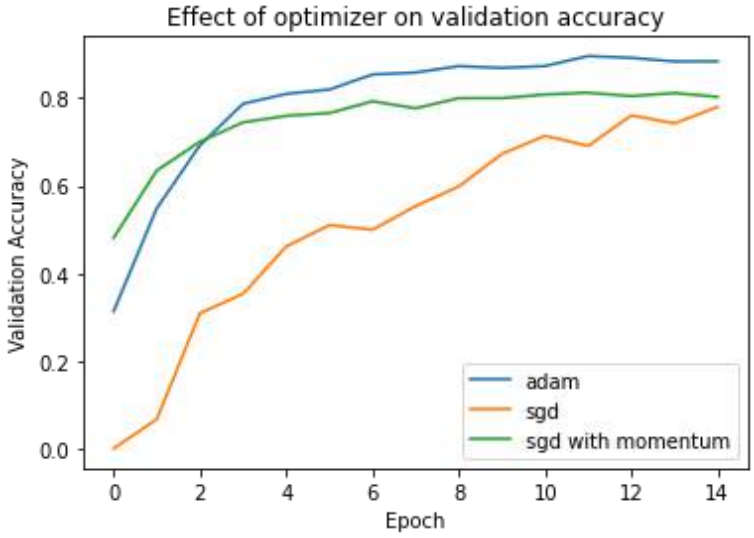
در هر حال tanh هم می‌تواند دچار مشکل vanishing problem بشود ولی روی این داده‌ها و با این شبکه نتیجه بهتری از دو تابع دیگر به ما داده است.

ث) از روش‌های بهینه سازی مختلف استفاده کنید. (Gradient descent, adam) **نمودار را برای ۱۵ ایپاک رسم کنید و نتایج را تحلیل کنید.**

نمودار مقایسه‌ی دقت شبکه روی داده های validation به ازای روش های بهینه سازی مختلف در شکل ۵ مشخص شده است.

علاوه بر adam که طبق انتظار کامل ترین و بهترین روش است، دو روش gradient based دیگر را نیز تست کردیم. اولی sgd ساده با learning rate = 0.1 و دومی sgd با momentum و decaying learning rate.

در مقایسه بین دو روش sgd، اهمیت کاهش learning rate در مرور زمان و استفاده از تکانه حاصل از مراحل قبلی کاملا مشخص میشود.



شکل ۶ - نمودار مقایسه‌ی دقت شبکه به ازای روش های بهینه سازی مختلف

در ادامه برای مقایسه بهتر، دقت روی داده های تست را هم گزارش میکنیم.

دقت برای روش adam: ٪۹۲,۲۸

دقت برای روش gd عادی: ٪۸۹,۶۱

دقت برای روش gd با momentum: ٪۹۰,۰۶

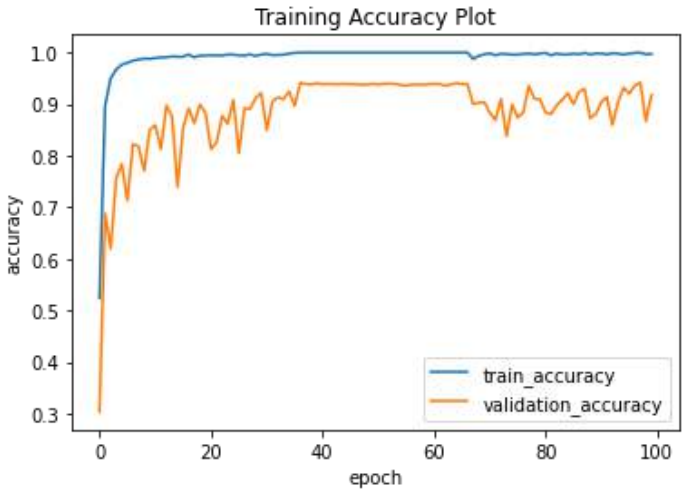
ج) کد خود را تغییر دهید تا شاملdrop out **باشد . نمودار ها را برای ۱۰۰ ایپاک رسم کنید و مقایسه کنید.**

در حالتی که از dropout استفاده نکنیم، احتمال overfit شدن شبکه روی داده‌های train وجود دارد و علت آن نبودن ترم برای regularization می‌باشد.

در صورت وجود dropout در شبکه، به طور رندوم تعدادی از node ها حذف میشوند و نتیجه‌ی این امر این است که هیچ نورون یا نورون های خاصی در تعیین خروجی خیلی تاثیرگذار نخواند بود و این همان عمل regularization است.

استفاده از dropout ممکن است زمان یادگیری را طولانی تر کند اما در نهایت باعث نزدیک شدن نمودارهای train و validation میشود. که این موضوع به معنای بیشتر شدن generalization شبکه و کاهش احتمال overfit شدن آن می‌باشد.

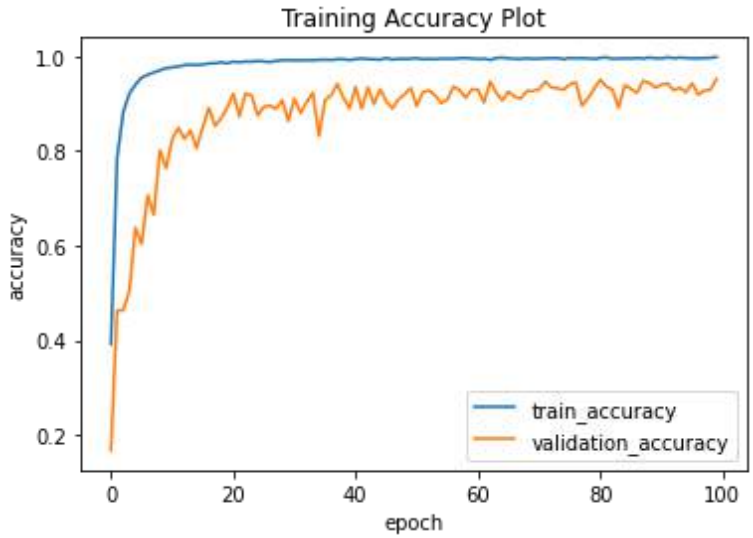
در ادامه شبکه خود را با تغییرات جزیی برای صد ایپاک ترین میکنیم و نتایج را در ادامه نشان میدهیم:



شکل ۷- نمودار دقت داده ها در صد ایپاک

و دقت روی داده های تست برابر ٪۹۳,۷ شد.

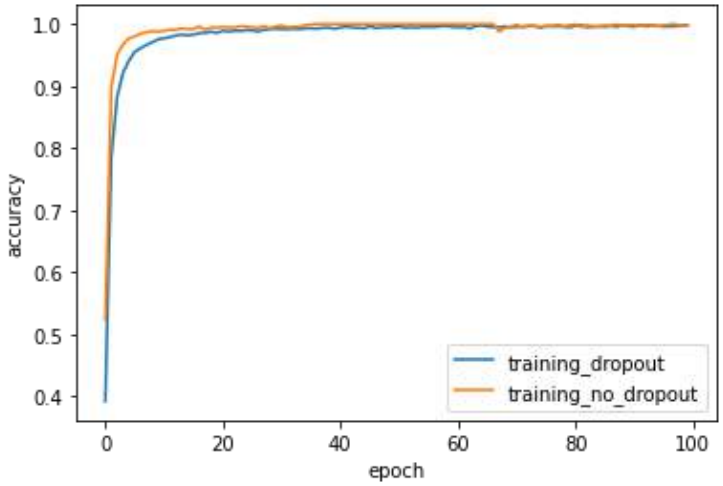
سپس با ۲ لایه‌ی dropout با احتمال ۰,۱ و ۰,۲ شبکه را آموزش میدهیم.



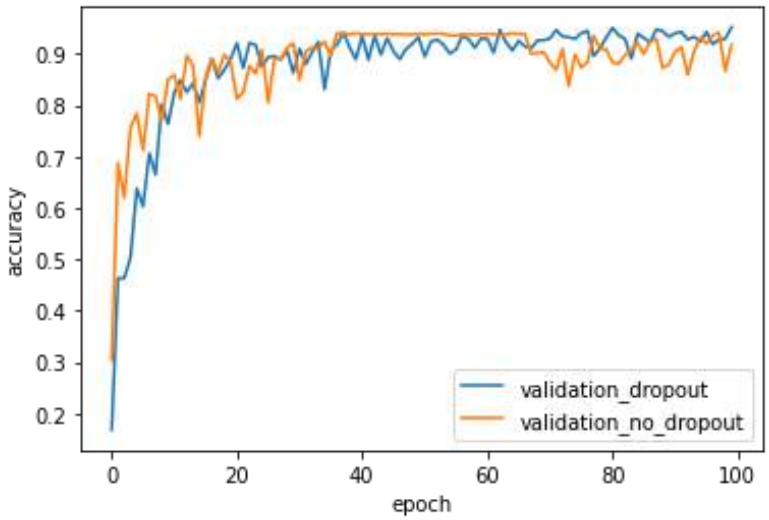
شکل ۸- نمودار دقت در شبکه با dropout

و دقت روی داده های تست برابر. ۹۵٪ شد.

در ادامه دو نمودار خواسته شده در صورت سوال را نمایش میدهیم.



شکل ۹ – نمودار مقایسه دقت روی داده های train در دو شبکه



شکل ۱۰ - نمودار مقایسه دقت روی داده های validation در دو شبکه

در حالت کلی میدانستیم که استفاده از dropout در ابتدا باعث کاهش دقت روی داده‌های آموزش میشود که با کم کردن overfitting در نهایت باعث افزایش دقت روی داده‌های تست و validation میشود.

افزایش دقت در داده های تست کاملا مشخص است، افزودن لایه‌ی دراپ اوت حدودا ۱,۷٪ دقت روی داده‌های تست را افزایش داده.

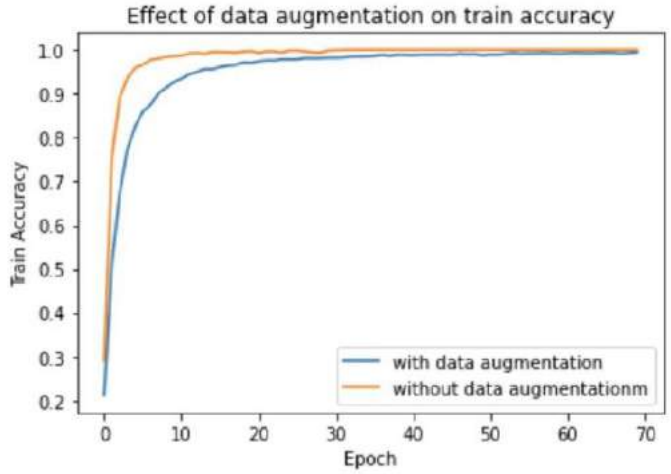
در مورد شکل ۸ هم انتظارمان براورده شده، به این معنا که نمودار دقت برای شبکه با dropout در حدود ۲۰ ایپاک اول تماما زیر نمودار دوم قرار دارد و با گذشت زمان و از حدود ایپاک ۴۰ام به آن میرسد.

و در مورد دقت روی داده‌های validation نیز طبق انتظار، در بلند مدت، دقت شبکه با دراپ‌اوت از شبکه بدون دراپ‌اوت بیشتر شده، البته این نمودار از نمودار قبلی نویزی تر است و لزوما در تمامی ایپاک‌ها دقت روی داده های validation افزوده نشده است ولی در ابتدا پایین تر و در سپس به آن رسیده و از آن میگذرد.

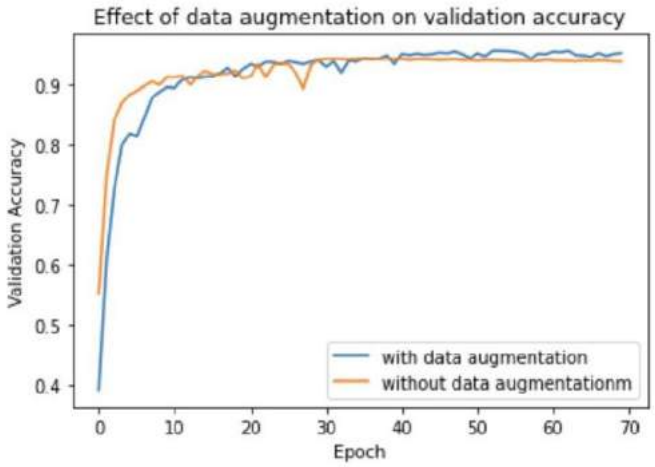
چ) کد خود را تغییر دهید تا شامل data augmentation **باشد . نمودار ها را برای ۱۰۰ ایپاک رسم کنید و مقایسه کنید.**

در حالت کلی عملکرد شبکه‌ها، به‌خصوص شبکه‌های cnn خیلی وابسته به کیفیت دیتاست آن‌ها است. یکی از روش‌ها برای بالا بردن کیفیت دیتاست، روش data augmentation می‌باشد. این روش با تولید دیتای جدید از روی دیتاهای قبلی، باعث بزرگ شدن دیتاست می‌شود.
علاوه بر این موضوع با ایجاد نویز روی ورودی شبکه، از overfit شدن هم جلوگیری میکند و باعث generalizationبیشتر شبکه می‌شود.

به وسیله data augmentation در هر ایپاک روی هر تصویر یکسری تغییر بصورت رندم اعمال می‌شوند. در نتیجه تصاویری که شبکه در هر ایپاک با آن‌ها train می‌شود تغییراتی جزئی با ایپاک‌های دیگر دارد که به شکل regularization کار می‌کند و باعث می‌شود که شبکه روی دیتا آموزش مان overfitنشود و باعث generalization بیشتر شبکه می‌شود .
این موضوع در نتایج مان نیز به وضوح دیده می‌شود ، در حالتی که data augmentation داریم دقت روی داده train در نهایت به ۹۹/۰ می‌رسد و دقت validation به ۹۵/۰ این در حالتی است که شبکه بدون data augmentation روی دادهtrain به ۱ می‌رسد و دقت validation به ۹۴/۰ که این نشان می‌دهد شبکه در این حالت بیش از حد به داده‌های trainحساس می‌شود و به همین دلیل با اینکه به دقت ۱ روی داده‌های train می‌رسد دقت آن روی داده های validation از حالتی که data augmentationدارد کمتر است که این نشان دهنده ی کمتر بودنgeneralization مدل است.



شکل ۱۱- تاثیر data augmentation روی یادگیری و دقت روی داده های train

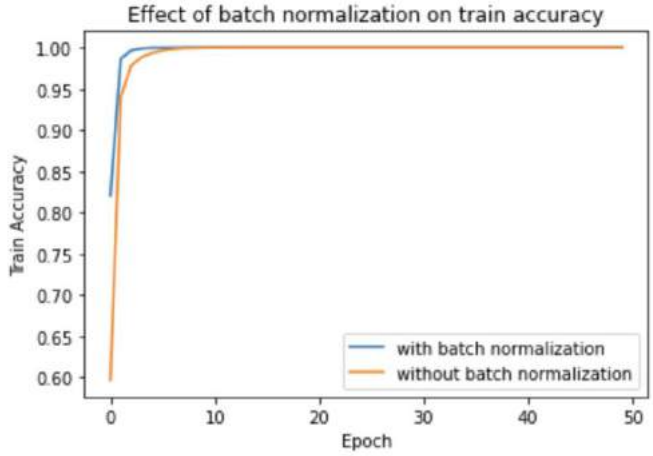


شکل ۱۲- تاثیر data augmentation روی یادگیری و دقت روی داده های validation

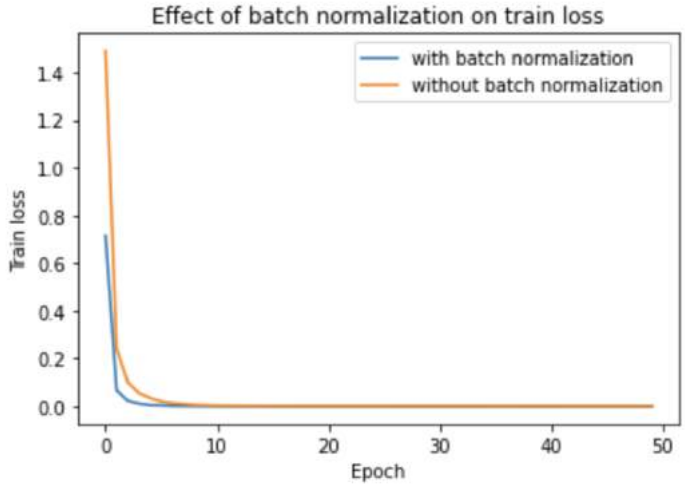
ح) تاثیر اضافه کردن لایه‌ی batch normalization را در شبکه خود مورد ارزیابی قرار دهید.

همان طور که قبلا توضیح داده شد مشکل internal covariate shift باعث کند شدن فرآیند یادگیری شبکه می‌شود batch normalization عمل normalization را روی ورودی hidden layerها با مقادیر میانگین و واریانس همان batch انجام می‌دهد و در نتیجه آن محدوده توزیع ورودی هر لایه مستقل از تغییر وزن‌های لایه‌های قبلی ثابت می‌ماند و تغییر زیادی در ورودی لایه‌ها رخ نخواهد داد که این مشکل internal covariate shift را حل می‌کند و فرآیند آموزش را سریع تر می‌کند.

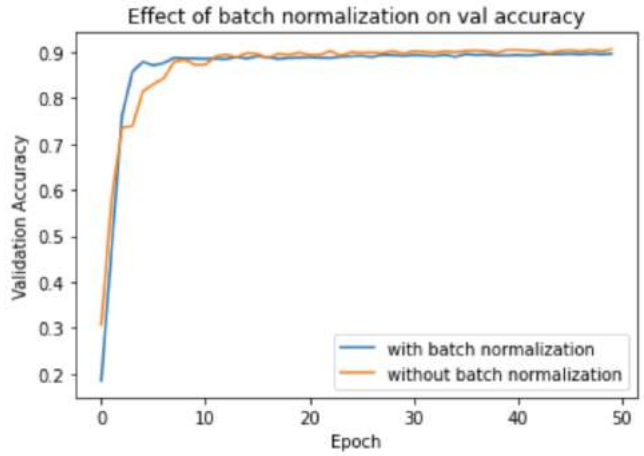
این موضوع که batch normalization آموزش شبکه را سریع تر کرده است را می توان در هم در سریع تر بالا رفتن دقت شبکه روی داده‌های train و validation دید و هم روی سریع تر کم شدن loss روی این دو دیتاست.



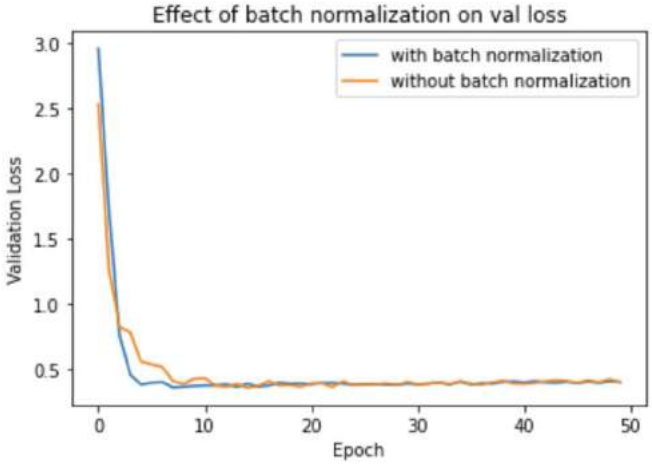
شکل ۱۳– تاثیر batch normalization روی دقت شبکه



شکل ۱۴ – تاثیر batch normalization روی loss شبکه



شکل ۱۵ – مقایسه تاثیر batch normalization روی داده های validation



شکل ۱۶ – مقایسه تاثیر batch normalization روی loss داده های validation