به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

مینی پروژه ۱

نام و نام خانوادگی	سجاد پاکدامن ساوجی فاطمه حقیقی
شماره دانشجویی	۸۱۰۱۹۵۵۱۷ ۸۱۰۱۹۵۳۸۵
تاریخ ارسال گزارش	۳ اردیبهشت

فهرست گزارش سوالات

سوال 1 – سوالات تشریحی

سوال ۱.

Mean Square Error و Mean Absolute Error دو تابع هزینه ی مناسب برای مسائل Regression problem می باشند.

تابع هزینه ی Mean Square Error یا MSE یکی از رایج ترین توابع هزینه ی مورد استفاده در مسائل regression است. MSE، مجموع مربع فواصل بین مقدار target و مقدار پیش بینی شده می باشد. فرمول محاسبه ی آن به صورت زیر می باشد:

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

تابع Mean Absolute Error یا MAE نوع دیگری از توابع هزینه ای است که در مسائل مربوط به regression مورد استفاده قرار می گیرد. MAE، مجموع تفاضل مطلق بین مقدار target و مقدار پیش بینی شده می باشد. بنابراین می توان گفت این تابع میانگین بزرگی خطاها را در مجموعه ای از پیش بینی ها بدون در نظر گرفتن جهت خطا ها، اندازه گیری می کند. فرمول محاسبه ی MAE به صورت زیر می باشد:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$

توابع Cross entropy loss و Hinge loss دو تابع هزینه ی مناسب برای مسائل Classification problem می باشند.

تابع هزینه ی cross entropy یک تابع هزینه ی دیفالت برای classification است. به عبارت دیگر می توان گفت cross entropy ، اندازه گیری تفاوت بین دو توزیع احتمال برای یک متغیر تصادفی یا مجموعه ای از وقایع است.

این تابع عملکرد یک مدل طبقه بندی که خروجی آن یک مقدار احتمال بین 0 تا 1 است را اندازه گیری می کند. هر چه احتمال پیش بینی شده از مقدار واقعی (target value) انحراف بیشتری پیدا کند، cross entropy loss افزایش می یابد. بنابراین پیش بینی احتمال 0.012 در صورتی که برچسب مشاهده واقعی 1 باشد بد خواهد بود و به مقدار ضرر بالا منجر می شود.

در یک binary classification که تعداد کلاس ها برابر با ۲ است، cross entropy را به صورت زیر می توان محاسبه کرد:

$$-(y \log(p) + (1-y) \log(1-p))$$

تابع Hinge loss برای آموزش classifier ها مورد استفاده قرار می گیرد. هنگامی که مقدار محاسبه شده توسط classifier درست باشد ولی این مقدار به صفر نزدیک باشد، مقدار کوچکی از Hinge loss را خواهیم داشت. اما در صورتی که مقدار محاسبه شده توسط classifier نادرست باشد،همیشه لزوما یک Hinge loss خواهیم داشت.

سوال۲.

در بیشتر الگوریتم های بهینه سازی ، قانون بروز رسانی از بسط تیلور تابع هزینه بدست‌می‌آید.

الگوریتم های بهینه سازی مختلف از تعداد جملات متفاوتی از بسط تیلور تابع استفاده می‌کنند. الگوریتم هایی که تنها از اختلاف استفاده می کنند، zero-order derivative method نامیده می‌شود. برای مثال الگوریتم دلتا. الگوریتم هایی که از جملات تا درجه ۱ استفاده می‌کنند(یعنی از بردار گرادیان استفاده می‌کنند) first-order derivative method نامیده می‌شوند. برای مثال الگوریتم GD یک الگوریتم مرتبه اول است. الگوریتم هایی که از جملات تا درجه ۲ استفاده می‌کنند، second-order derivative method نامیده می‌شوند(این الگوریتم ها از ماتریس Hessian استفاده می کنند). برای مثال: newton method , conjugate gradient, quasi-newton و جز آن. منبع

سوال۳.

مشکل overfitting به این معنی است که پیچیدگی شبکه برای داده ها بالا است و در فرایند یادگیری تنها کاهش هزینه ای که متناسب با دقت مدل بوده است ، معیار قرار گرفته است. این مشکل از مصداق‌های bias-variance tradeoff است زیرا که مدل به داده ها چسبیده است و اگر توزیع داده های ورودی کمی تغییر کند باعث خطا می‌شود. ۳ روش برای حل این مشکل آورده شده است:

روش۱. اضافه کردن regularization term: در این روش یک تابع به عنوان regularization به تابع هزینه اضافه می‌شود. این تابع باید به گونه ای باشد که درجه آزادی مدل را کم کند. برای مثال اضافه کردن اندازه مطلق تمامی وزن های مدل باعث می‌شود وزن ها نتوانند از حد بالایی بیشتر شوند و در نتیجه با زیاد شده یک وزن در شبکه نمی‌تواند تاثیر feature خاصی را زیاد کند.

روش۲. اضافه کردن dropout : روش drop out، روشی است که نورون های منتخب به طور تصادفی در طول آموزش نادیده گرفته می شوند. این بدان معنی است که سهم آنها در فعال شدن نورون های پایین دست به صورت موقت در forward pass برداشته می شود و هیچ گونه به روزرسانی وزنی بر روی نورون در backward pass اعمال نمی شود.

اگر نورون ها هنگام آموزش به طور تصادفی از شبکه خارج شوند ، دیگر سلول های عصبی باید در جایگاه های مورد نیاز برای پیش بینی نورون های مفقود شده گام بردارند و شرایط را کنترل کنند. در این شرایط می توان گفت چندین representation داخلی مستقل توسط شبکه آموزش داده می شود. در نتیجه ی این اقدام، شبکه نسبت به وزن مخصوص نورون ها حساسیت کمتری پیدا می کند. این به نوبه خود منجر به شبکه ای می شود که بتواند تعمیم بهتری داشته باشد و نیز احتمال آنکه بر روی داده های آموزش overfit داشته باشیم، کمتر می شود.

روش۳. انجام cross validation: با انجام تکنیک های cross validation مانند k fold در حال آموزش شبکه قابلیت تعمیم دهی آن مرتبا بررسی می‌شود و می توان از مشکل overfitting جلوگیری کرد.

روشهای دیگر: ensembling, early stopping, feature removal

سوال۴.

در صورتی که از توابع خطی بجای توابع غیر خطی در شبکه عصبی استفاده کنیم، درحقیقت تعداد لایه های شبکه افزایش پیدا نمی کند زیرا که تبدیل های خطی متوالی را میتوان به صورت یک تبدیل خطی نوشت. با این توضیح اگر برای مثال شبکه عصبی با ۱۰۰ لایه خطی تولید کنیم و آن را آموزش دهیم در حقیقت تنها یک لایه خطی آموزش داده ایم.

جدای از این اگر از توابع غیر خطی استفاده کنیم، شبکه قابلیت تفکیک های غیر خطی را پیدا می کند و می‌تواند داده هایی با پترن های غیر خطی را نیز جدا کند.

توابع sigmoid، tanh، relu سه نمونه از انواع توابع فعال ساز محبوب و رایج در شبکه های عصبی می باشند.

تابع فعال ساز sigmoid:

در صورتی که از تابع فعال سازی sigmoid استفاده کنیم، convergence یا همگرا شدن شبکه عصبی به یک جواب درست به کندی و آرام صورت خواهد گرفت و نیز با gradient vanishing روبرو هستیم. همچنین خروجی این تابع، صفر محور نیست زیرا خروجی آن بین ۰ و ۱ است به عبارت دیگر میانگین آن صفر نمی باشد، این امر باعث می شود به روز رسانی های شیب در جهت های مختلف خیلی زیاد پیش برود.

مزایا:

استفاده و فهم راحت آن

معایب:

مشکل vanishing gradient

کندی convergence در آن

میانگین در این تابع غیر صفر است، به عبارتی خروجی آن بین صفر و یک بوده، این امر باعث می شود به روز رسانی های شیب در جهت های مختلف خیلی زیاد پیش برود.

تابع فعال ساز tanh:

در تابع فعال سازی tanh با شیب صفر روبرو نیستیم و میانگین شیب در آن صفر است، به عبارت دیگر به دلیل آنکه خروجی این تابع بین -۱ و ۱ است، می توان گفت که خروجی آن صفر محور است در نتیجه optimization به وسیله ی آن راحت تر است اما همچنان در این تابع نیز با vanishing gradient روبرو هستیم. می توان گفت استفاده از این تابع نسبت به تابع فعال سازی sigmoid بهتر است و نتیجه ی بهتر و درست تری را به ما می دهد.

مزایا:

میانگین سیب در آن صفر است و با شیب صفر روبرو نیستیم

بهینه سازی در آن راحت تر است(به دلیل تقرن میان شیب های آن و عدم وجود شیب صفر در آن)

معایب:

وجود مشکل vanishing gradient

تابع فعال ساز relu:

تابع فعال سازی relu بسیار ساده و کارآمد می باشد . در این تابع، به ازای مقادیر کوچکتر از صفر، خروجی تابع برابر با صفر است. بنابراین در این بخش هم خروجی تابع و هم شیب آن صفر می باشد. به عبارتی می توان گفت در این تابع از بازه ی اعداد حقیقی به درستی استفاده نمی کنیم و به دلیل وجود شیب صفر در قسمتی از آن، در

optimization هایی که با شیب کار می کنند سرعت این تابع نسبت به tanh کمتر خواهد بود. به طور کلی در صورتی که تعداد لایه های شبکه عصبی زیاد باشد، سرعت همگرای آن از تابع tanh به مراتب بیشتر است همچنین در این تابع برخلاف دو تابع قبلی با gradient vanishing روبرو نیستیم.

اما محدودیت این تابع آن است که، از relu تنها می توان در لایه های مخفی یک شبکه ی عصبی استفاده کرد.

مزایا:

ساده و کارآمد

در صورتی که تعداد لایه های شبکه ی عصبی زیاد باشد، سرعت همگرایی آن از tanh بیشتر است.

با مشکل vanishing gradient روبرو نیستیم

معایب:

وجود شیب صفر در آن

به دلیل وجود شیب صفر و عدم استفاده ی درست از بازه ی اعداد حقیقی در آن ، سرعت آن پر optimizer های ,وابسته به شیب نسبت به tanh کمتر است.

تنها می توان از آن در لایه های مخفی استفاده کرد.

سوال۵.

تکنیک data augmentation روشی است که با استفاده از آن می‌توان به صورت مصنوعی حجم مجموعه داده را افزایش داد. در این روش تعدادی نگاشت های خطی و غیر خطی به صورت تصادفی به تصویر اعمال می‌شود و خروجی این تبدیل ها به مجموعه داده اضافه می‌شود. برای مثال تبدیل های مانند دوران، انتقال، تغییر روشنایی، تغییر رنگ، قرینه‌سازی عمودی و افقی و جز آن روی تصاویر اعمال می‌شود.

سوال۶.

آموزش شبکه های عصبی با تعداد لایه زیاد می‌تواند چالش انگیز باشد. یکی از دلایل این است که توزیع احتمال در لایه ورودی برای لایه های عمیق تر، پس از هر بروزرسانی وزن ها توسط minibatch تغییر می‌کند. بنابر این الگوریتم آموزش در حقیقت هدفی را دنبال می کند که دائما تغییر می‌کند. این پدیده تغییر توزیع ورودی لایه ها با نام علمی *internal covariate shift* شناخته می‌شود.

تکنیک batch normalization ورودی هر لایه را برای هر mini batch استاندارد می‌کند. تاثیر این این فرایند ایستان شدن فرایند آموزش و کاهش چشمگیر تعداد epoch های لازم برای آموزش می باشد.

سوال ۲ – سوالات عملی (الف)

مشخصات شبکه طراحی شده به صورت زیر است:

stride size = 1 for all layers

Input: 30x30x3

Conv2d(32, 3, 3) → 28x28x32

Mxpoll(2, 2) → 14x14x32

Conv2d(64, 3, 3) → 12x12x64

Mxpoll(2, 2) → 6x6x64

Conv2d(128, 3, 3) → 4x4x128

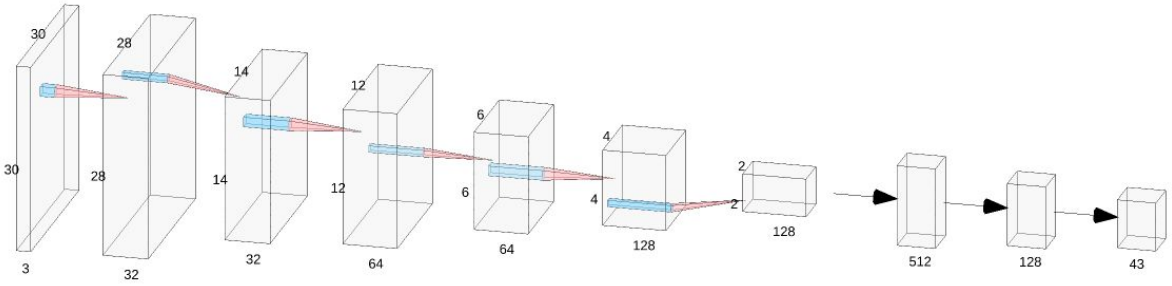
Mxpoll(2, 2) → 2x2x128

Flatten() → 512

Dense(512, 128) → 128

Dense(128, 43) → 43

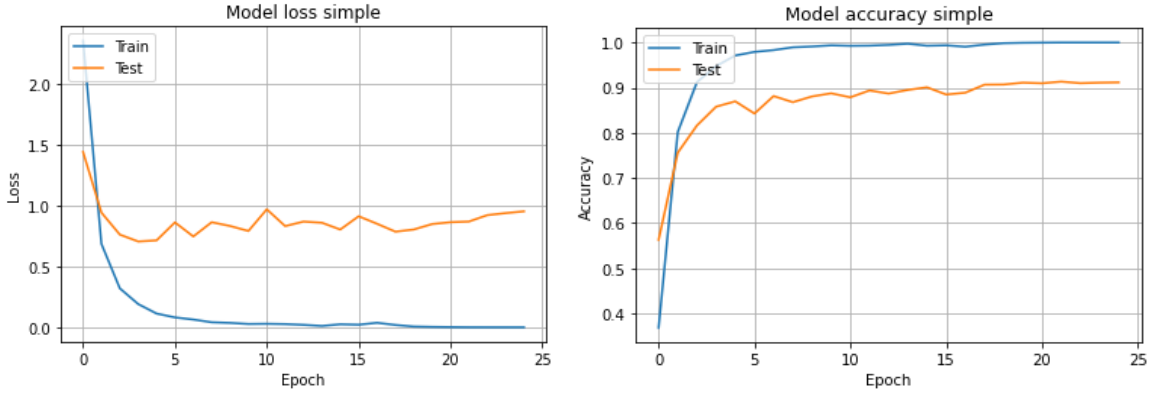
برای آموزش از optimizer = adam و batch_size = 128 استفاده شده است. تابع هزینه نیز cross entropy انتخاب شده است. آموزش را برای ۲۵ epoch انجام دادیم. در شکل ۱ معماری شبکه مورد نظر آورده شده است.



شکل ۱. معماری شبکه پیش‌نهادی

سوال ۳ - سوالات عملی (ب)

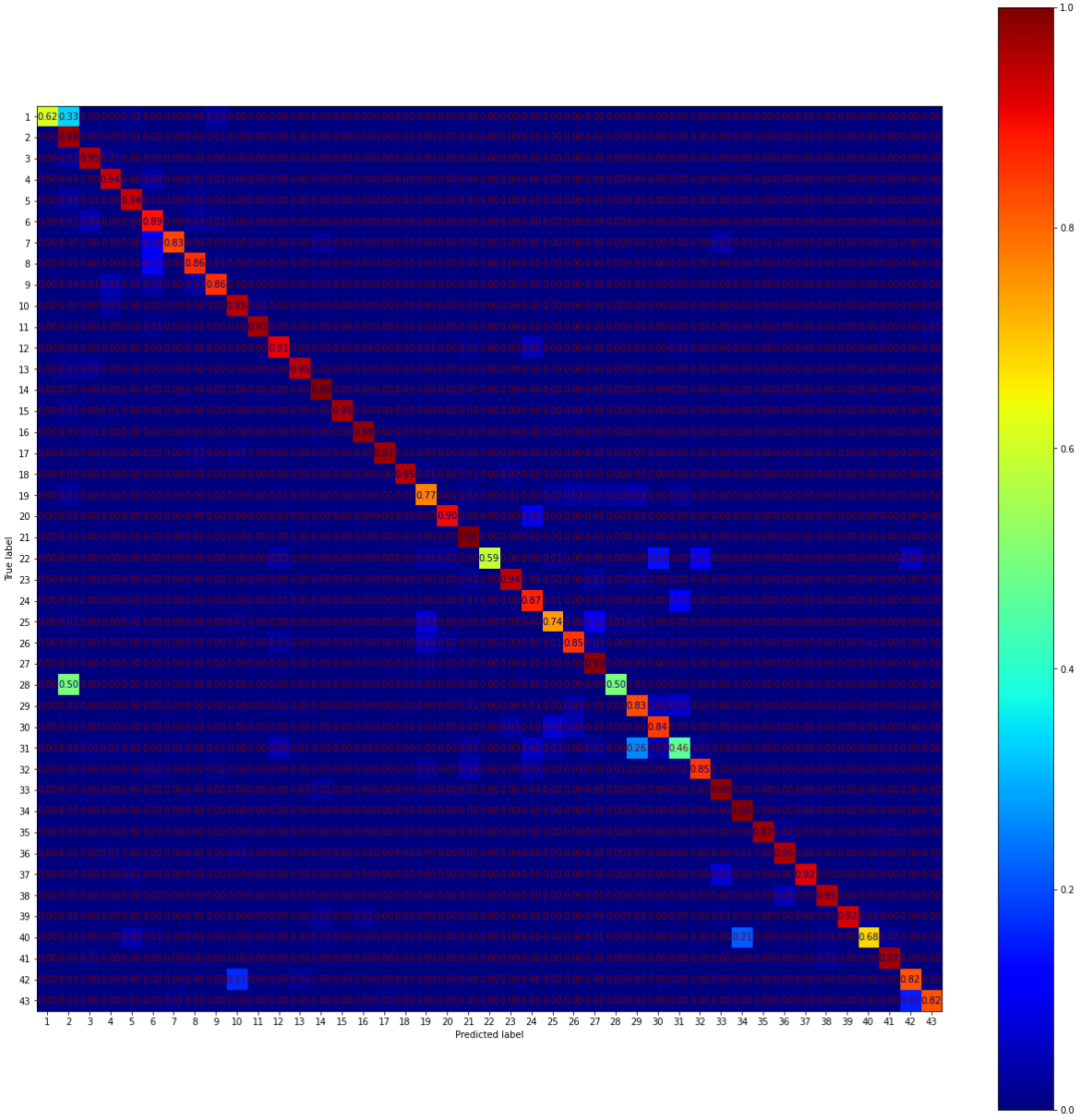
شبکه شکل ۱ را در ۲۵ epoch آموزش دادیم. دقت نهایی روی داده های آموزش ۱۰۰ شد و دقت نهایی روی داده های آزمایش برابر ۹۱ درصد شد. نمودار دقت مدل و هزینه مدل در نمودار های شکل ۲ آمده است.



شکل ۲. نمودار هزینه و دقت مدل شبکه عصبی ساده

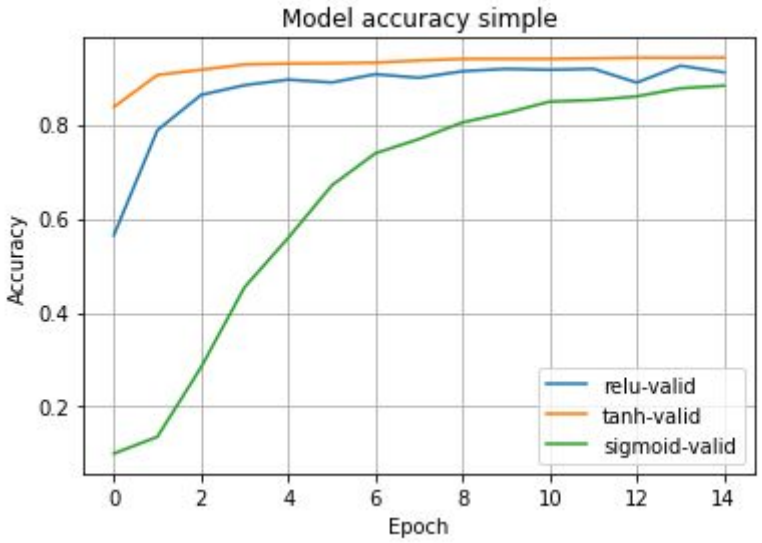
سوال ۴ - سوالات عملی (پ)

از آن جایی که در مدل ابتدایی روی داده های آموزش به دقت ۱۰۰ رسیدیم، رسم ماتریس آشفتگی برای داده های آموزش اهمیتی نخواهد داشت. به همین جهت ماتریس آشفتگی را برای داده‌های آزمایش رسم نموده‌ایم.



شکل ۳. تصویر normalized confusion matrix برای شبکه اولیه روی داده های آزمایش

سوال ۵ - سوالات عملی (ت)



شکل۴. نمودارهای دقت داده ارزیابی در شبکه عصبی توابع فعال سازی مجزا

به طور کلی می توان گفت، در صورتی که از تابع فعال سازی sigmoid استفاده کنیم، convergence یا همگرا شدن شبکه عصبی به یک جواب درست به کندی و آرام صورت خواهد گرفت و نیز با gradient vanishing روبرو هستیم. همچنین خروجی این تابع، صفر محور نیست زیرا خروجی آن بین ۰ و ۱ است به عبارت دیگر میانگین آن صفر نمی باشد، این امر باعث می شود به روز رسانی های شیب در جهت های مختلف خیلی زیاد پیش برود.

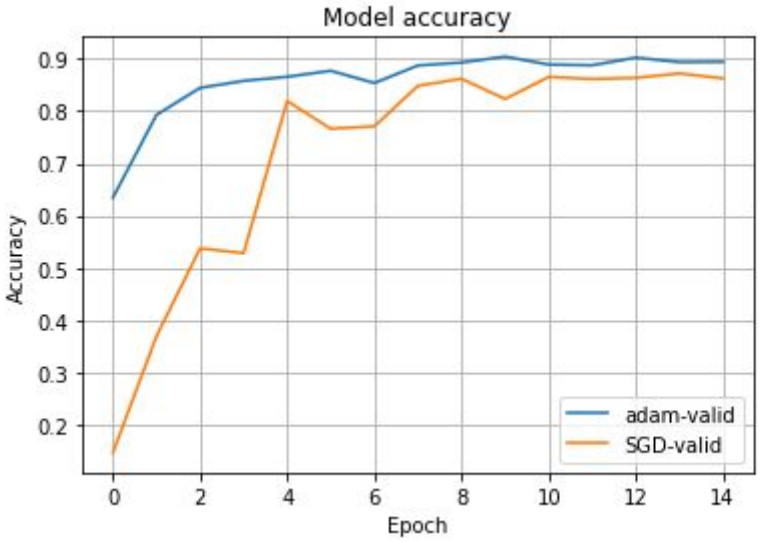
در تابع فعال سازی tanh با شیب صفر روبرو نیستیم و میانگین شیب در آن صفر است، به عبارت دیگر به دلیل آنکه خروجی این تابع بین -۱ و ۱ است، می توان گفت که خروجی آن صفر محور است در نتیجه optimization به وسیله ی آن راحت تر است اما همچنان در این تابع نیز با gradient vanishing روبرو هستیم. می توان گفت استفاده از این تابع نسبت به تابع فعال سازی sigmoid بهتر است و نتیجه ی بهتر و درست تری را به ما می دهد.

تابع فعال سازی relu بسیار ساده و کارآمد می باشد . در این تابع، به ازای مقادیر کوچکتر از صفر، خروجی تابع برابر با صفر است. بنابراین در این بخش هم خروجی تابع و هم شیب آن صفر می باشد. به عبارتی می توان گفت در این تابع از بازه ی اعداد حقیقی به درستی استفاده نمی کنیم و به دلیل وجود شیب صفر در قسمتی از آن، در optimization هایی که با شیب کار می کنند سرعت این تابع نسبت به tanh کمتر خواهد بود. به طور کلی در صورتی که تعداد لایه های شبکه عصبی زیاد باشد، سرعت همگرای آن از تابع tanh به مراتب بیشتر است همچنین در این تابع برخلاف دو تابع قبلی با gradient vanishing روبرو نیستیم.

اما محدودیت این تابع آن است که، از relu تنها می توان در لایه های مخفی یک شبکه ی عصبی استفاده کرد.

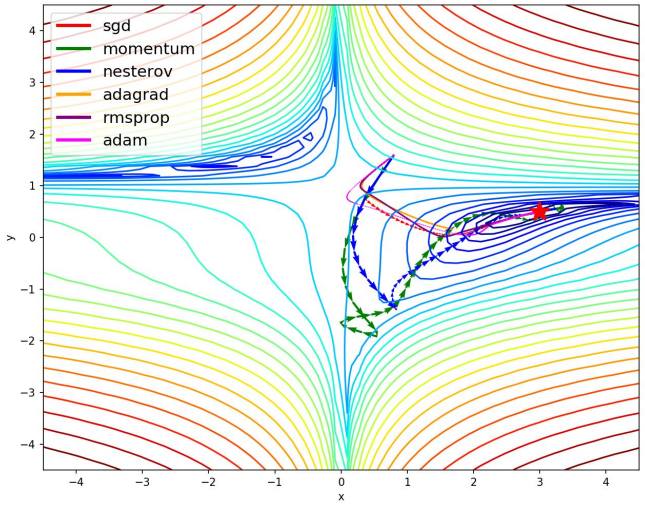
بنابراین با توجه به نمودار حاصل از دقت این سه تابع بر روی داده ی ارزیابی و معایب استفاده از هر تابع، استفاده از تابع فعال سازی tanh برای این تعداد لایه ی مخفی مناسب تر است. صرف نظر از این مسئله که در relu با gradient vanishing روبرو نیستیم، اما به دلیل عدم تقارن در شیب و وجود شیب صفر در آن سرعت آن نسبت به tanh کمتر بوده پس استفاده از tanh در این مسئله بهتر است.

سوال ۶ - سوالات عملی (ث)



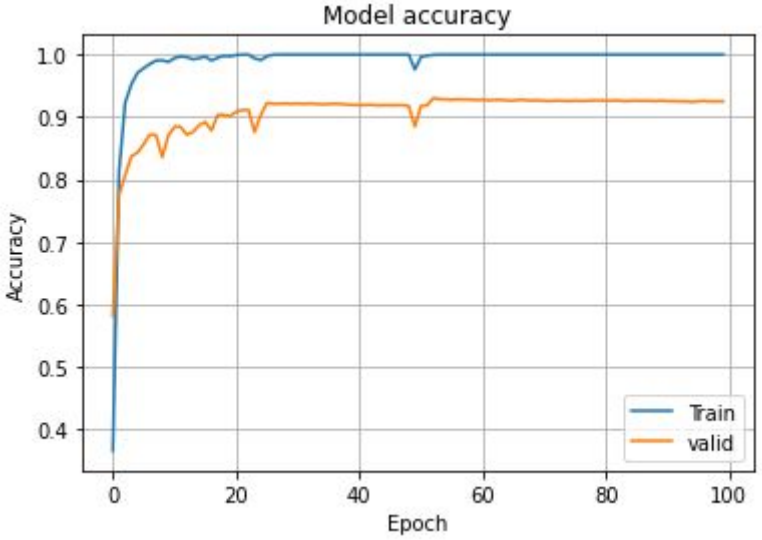
شکل۵. نمودارهای دقت داده ارزیابی در شبکه عصبی برای بهینه سازهای مختلف

سرعت همگرایی الگوریتم های بهینه سازی متفاوت است..... در شکل ۵.۱ چندی از الگوریتم های بهینه سازی مقایسه شده اند

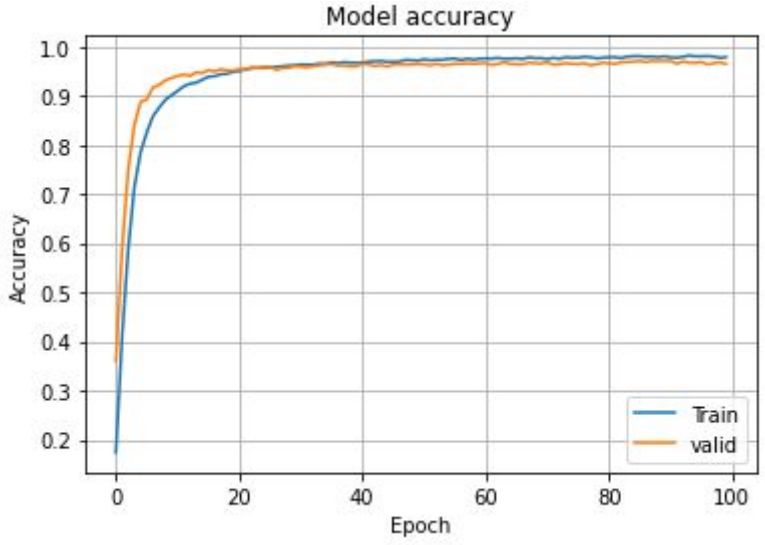


شکل ۵.۱ . مقایسه همگرایی الگوریتم های مختلف بهینه سازی

سوال۷ - سوالات عملی (ج)



شکل۶ .نمودارهای دقت داده ارزیابی و آموزشی در شبکه عصبی بدون لایه ی dropout



شکل۷ .نمودارهای دقت داده ارزیابی و آموزشی در شبکه عصبی با لایه ی dropout

drop out روشی است که نورون های منتخب به طور تصادفی در طول آموزش نادیده گرفته می شوند. به عبارت دیگر این نورون ها بطور تصادفی کنار گذاشته می شوند. این بدان معنی است که سهم آنها در فعال شدن نورون های پایین دست به صورت موقت در گذرگاه جلو برداشته می شود و هیچ گونه به روزرسانی وزنی بر روی نورون در forward pass اعمال نمی شود.

همانطور که یک شبکه عصبی می آموزد ، وزن نورون درون بافت خود در شبکه قرار می گیرد. وزن سلولهای عصبی برای ویژگی های خاصی تنظیم شده است که تخصص خاصی را ارائه می دهند. نورون های همسایه به این

تخصص تکیه می کنند ، که اگر خیلی دور برداشته شود، می تواند منجر به یک مدل شکننده بیش از حد خاص به داده های آموزش شود.

اگر نورون ها هنگام آموزش به طور تصادفی از شبکه خارج شوند ، دیگر سلول های عصبی باید در جایگاه های مورد نیاز برای پیش بینی نورون های مفقود شده گام بردارند و شرایط را کنترل کنند. در این شرایط می توان گفت چندین representation داخلی مستقل توسط شبکه آموزش داده می شود.

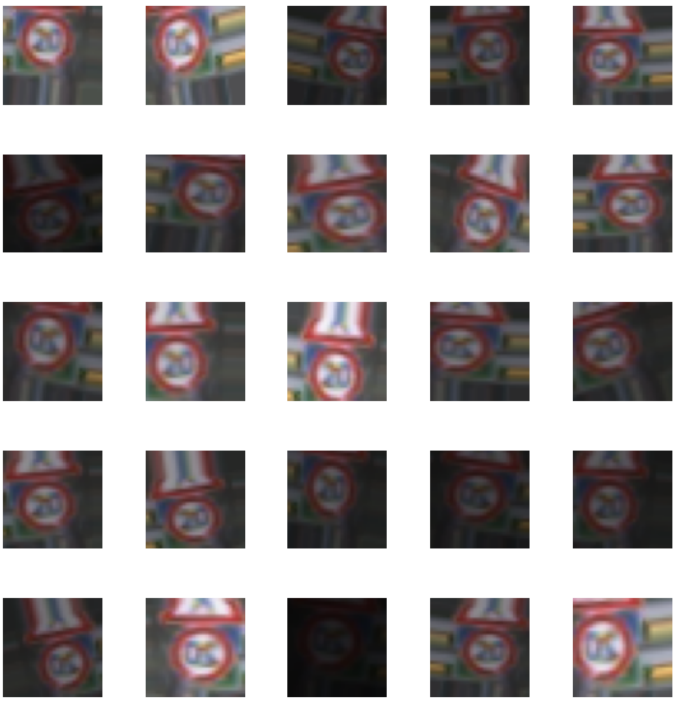
در نتیجه ی این اقدام، شبکه نسبت به وزن مخصوص نورون ها حساسیت کمتری پیدا می کند. این به نوبه خود منجر به شبکه ای می شود که بتواند تعمیم بهتری داشته باشد و نیز احتمال آنکه بر روی داده های آموزش overfit داشته باشیم، کمتر می شود.

به طور کلی می توان گفت Dropout یک شبکه عصبی را مجبور به یادگیری ویژگی های قوی تر می کند که در رابطه با بسیاری از زیر مجموعه های مختلف تصادفی نورون های دیگر مفید هستند. همچنین این روش تقریباً تعداد تکرارهای مورد نیاز برای همگرایی را دو برابر می کند.

از مقایسه ی دو نمودار شکل۶ و شکل ۷ می توان دید که در شکل ۶، میزان دقت در داده های آموزشی در هر دور بیشتر از دقت در داده های ارزیابی است و در واقع نوعی overfitting رخ داده. اما در نمودار شکل ۷ (نمودار دقت برای شبکه با لایه ی Dropout) تا دور ۴۰ام، مقدار دقت داده های ارزیابی بیشتر یا برابر با دقت داده های آموزشی در شبکه می باشد. به عبارت دیگر در شبکه با لایه ی dropout میزان تعمیم پذیری شبکه بیشتر و شبکه دقت بیشتری برای داده های جدید و ویژگی های ناشناخته دارد.

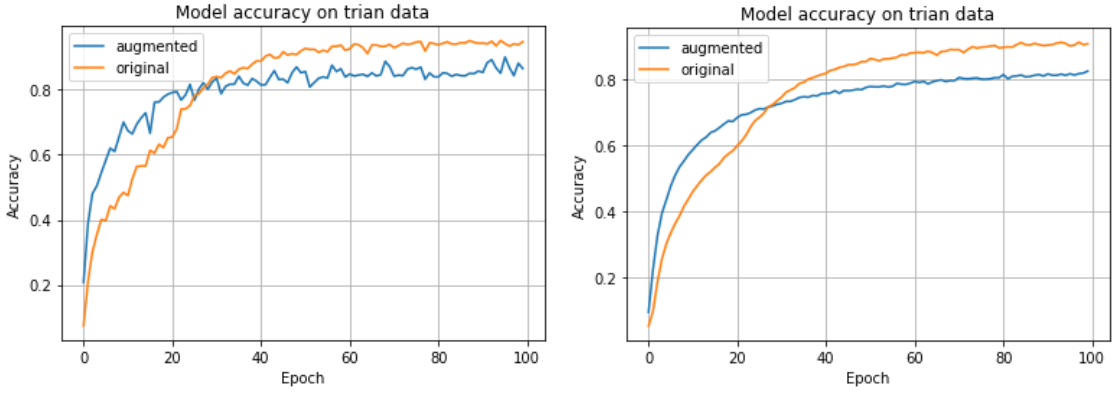
سوال۸ - سوالات عملی (چ)

تکنیک data augmentation روشی است که با استفاده از آن می‌توان به صورت مصنوعی حجم مجموعه داده را افزایش داد. در این روش تعدادی نگاشت های خطی و غیر خطی به صورت تصادفی به تصویر اعمال می‌شود و خروجی این تبدیل ها به مجموعه داده اضافه می‌شود. برای مثال تبدیل های مانند دوران، انتقال، تغییر روشنایی، تغییر رنگ، قرینه‌سازی عمودی و افقی و جز آن روی تصاویر اعمال می‌شود. در شکل ۸ نمونه ای از این روش نشان داده شده است.



شکل ۸. نمونه ای از عملکرد data augmentation

نمودار های دقت بر روی داده های ارزیابی و آموزش در شکل ۹ آمده اند. این نمودار ها برای ۱۰۰ epoch است. دقت شود که به دلیل اینکه data augmentation با استفاده از کتابخانه keras به طور چشم گیری حجم مجموعه داده را افزایش می‌دهد، در حالتی که از data augmentation استفاده کرده ایم دقت مدل بالا نرفته است ولی قابلیت تعمیم پذیری آن و robustness آن در برابر داده های واقعی بشدت افزایش می یابد. علاوه بر آن این روش در مواردی که حجم مجموعه داده کم است ولی مسئله به مدل پیچیده ای نیاز دارد، بسیار کاربردی است. چرا که با آموزش شبکه پیچیده بر روی تعداد کمی داده به پاسخ نامناسب میرسیم.

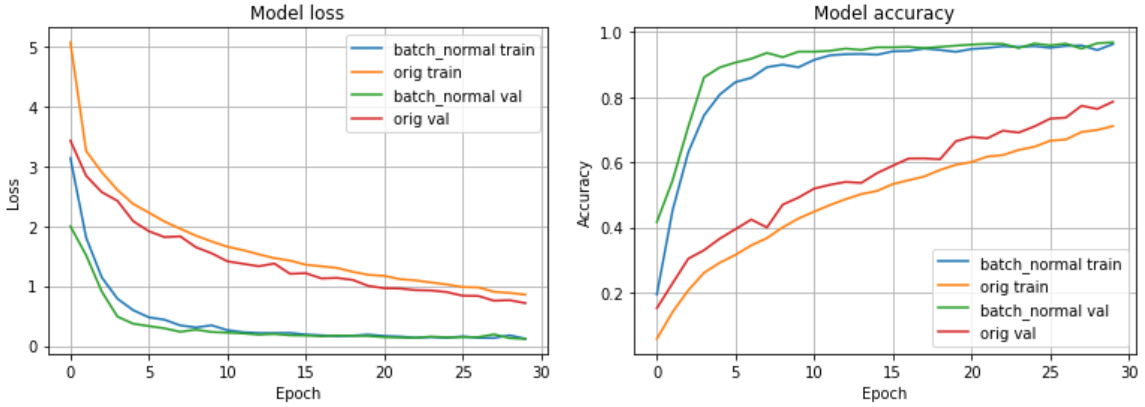


شکل ۹. تاثیر data augmentation بر روی شبکه عصبی

سوال ۹ - سوالات عملی (ح)

آموزش شبکه های عصبی با تعداد لایه زیاد می‌تواند چالش انگیز باشد. یکی از دلایل این است که توزیع احتمال در لایه ورودی برای لایه های عمیق تر، پس از هر بروزرسانی وزن ها توسط minibatch تغییر می‌کند. بنابر این الگوریتم آموزش در حقیقت هدفی را دنبال می کند که دائما تغییر می‌کند. این پدیده تغییر توزیع ورودی لایه ها با نام علمی *internal covariate shift* شناخته می‌شود.

تکنیک batch normalization ورودی هر لایه را برای هر mini batch استاندارد می‌کند. تاثیر این این فرایند ایستان شدن فرایند آموزش و کاهش چشمگیر تعداد epoch های لازم برای آموزش می باشد. همانطور که در شکل ۱۰ مشاهده می‌شود با استفاده از batch normalization با تعداد کمتری epoch به پاسخ مطلوب رسیده‌ایم.



شکل ۱۰. نتیجه اعمال batch normalization

نحوه اجرای کدها

تمامی کد های تمرین در فایل NNDL_proj1.inpy می‌باشد.