



به نام خدا



دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

یادگیری عمیق با کاربردها

تمرین شماره یک

یاسمون حق‌بین	نام و نام خانوادگی
۸۱۰۱۰۱۳۲۶	شماره دانشجویی
۱۴۰۱/۱۲/۲۶	تاریخ ارسال گزارش

## فهرست

۶	مقدمه
۷	بخش اول – داده‌های جدولی
۷	نمایش داده‌ها
۷	سوال (۱.۱)
۹	سوال (۱.۲) آموزش مدل با پارامترهای خواسته شده
۱۱	سوال (۱.۳)
۱۳	سوال (۱.۴)
۱۵	سوال (۱.۵)
۲۰	سوال (۱.۶)
۲۶	سوال (۱.۷)
۳۳	سوال (۱.۸)
۳۸	سوال (۱.۹)
۴۲	سوال (۱.۱۰)
۴۵	سوال (۱.۱۱)
۴۸	سوال (۱.۱۲)
۴۹	سوال (۱.۱۳)
۵۲	بخش دوم – رگرسن
۵۲	سوال (۲.۱)
۵۲	سوال (۲.۲) پیاده‌سازی رگرسن
۵۳	سوال (۲.۳) نمودار هزینه
۵۶	مراجع

## فهرست شکل ها

۷	شکل ۱- نمایش تصاویر.....
۹	شکل ۲- نمودار دقت و هزینه با پارامترهای پیشفرض.....
۱۰	شکل ۳- ماتریس آشفتگی برای مدل اولیه.....
۱۱	شکل ۴- نمودار دقت و خطای Gaussian RBF.....
۱۲	شکل ۵- ماتریس آشفتگی Gaussian RBF.....
۱۳	شکل ۶- نمودار دقت و هزینه با مقدار دهی اولیه صفر برای وزن.....
۱۴	شکل ۷ - ماتریس آشفتگی مربوط به مدل با وزن اولیه صفر.....
۱۶	شکل ۸- نمودار دقت و هزینه بر روی داده پیش پردازش نشده.....
۱۷	شکل ۹- ماتریس آشفتگی برای داده پیش پردازش نشده.....
۱۸	شکل ۱۰ - نمودار دقت و هزینه در حالت نرمال بودن دادهها.....
۱۹	شکل ۱۱- نمودار هزینه و دقت مدل بر روی دادههای استاندارد.....
۲۰	شکل ۱۲- ماتریس آشفتگی در حالت نرمال و استاندارد بودن دادهها.....
۲۲	شکل ۱۳- نمودار دقت و هزینه مربوط به SGD.....
۲۲	شکل ۱۴- ماتریس آشفتگی SGD روی داده تست.....
۲۳	شکل ۱۵- نمودار دقت و هزینه با اندازه بج ۵۰۰.....
۲۳	شکل ۱۶- ماتریس آشفتگی با اندازه بج ۵۰۰.....
۲۴	شکل ۱۷- نمودار با اندازه بج ۶۴.....
۲۴	شکل ۱۸- ماتریس آشفتگی برای اندازه بج ۶۴.....
۲۵	شکل ۱۹- نمودار دقت و هزینه با اندازه بج ۱۶.....
۲۵	شکل ۲۰- ماتریس آشفتگی با اندازه بج ۱۶.....
۲۷	شکل ۲۱- نمودار دقت و زیان با نرخ یادگیری ۱۰۰۰۰۰۱.....
۲۷	شکل ۲۲- ماتریس آشفتگی برای نرخ یادگیری ۱۰۰۰۰۰۱.....
۲۸	شکل ۲۳- نمودار دقت و هزینه نرخ یادگیری ۱۰۰۰۰۱.....
۲۸	شکل ۲۴- ماتریس آشفتگی با نرخ یادگیری ۱۰۰۰۰۱.....
۲۹	شکل ۲۵- نمودار دقت و هزینه با نرخ یادگیری ۱۰۰۰۰۱.....
۲۹	شکل ۲۶- ماتریس آشفتگی برای نرخ یادگیری ۱۰۰۰۰۱.....

..... ۳۰	..... شکل ۲۷- نمودار دقت و هزینه برای نرخ یادگیری ۱
..... ۳۰	..... شکل ۲۸- ماتریس آشفتگی برای نرخ یادگیری ۱
..... ۳۱	..... شکل ۲۹- نمودار دقت و هزینه برای نرخ یادگیری ۱
..... ۳۱	..... شکل ۳۰- ماتریس آشفتگی برای نرخ یادگیری ۱
..... ۳۲	..... شکل ۳۱- توابع فعال‌ساز
..... ۳۴	..... شکل ۳۲- نمودار دقت و هزینه با تابع سیگموئید
..... ۳۴	..... شکل ۳۳- نمودار دقت و هزینه با تابع Tanh
..... ۳۵	..... شکل ۳۴- نمودار دقت و هزینه با تابع ReLu
..... ۳۵	..... شکل ۳۵- نمودار دقت و هزینه با تابع LeakyReLu
..... ۳۶	..... شکل ۳۶- ماتریس آشفتگی برای تابع Sigmoid
..... ۳۷	..... شکل ۳۷- ماتریس آشفتگی برای تابع Tanh
..... ۳۷	..... شکل ۳۸- ماتریس آشفتگی برای تابع ReLu
..... ۳۸	..... شکل ۳۹- ماتریس آشفتگی برای تابع LeakyReLu
..... ۳۹	..... شکل ۴۰- نمودار دقت و هزینه یک لایه پنهان با ۱۶ نورون
..... ۳۹	..... شکل ۴۱- نمودار دقت و هزینه دو لایه پنهان با ۱۶ نورون
..... ۴۰	..... شکل ۴۲- نمودار دقت و هزینه دو لایه پنهان با ۶۴ و ۱۶ نورون
..... ۴۱	..... شکل ۴۳- ماتریس آشفتگی مدل دو لایه ۱۶ تا ۱۶
..... ۴۱	..... شکل ۴۴- ماتریس آشفتگی مدل سه لایه ۱۶ تا ۱۶
..... ۴۲	..... شکل ۴۵- ماتریس آشفتگی مدل سه لایه ۶۴ و ۱۶ تا ۱۶
..... ۴۴	..... شکل ۴۶- نمودار دقت و هزینه مدل با مومنتوم ۰.۹
..... ۴۴	..... شکل ۴۷- ماتریس آشفتگی برای مومنتوم ۰.۹
..... ۴۷	..... شکل ۴۸- نمودار دقت و هزینه با regularization
..... ۴۸	..... شکل ۴۹- ماتریس آشفتگی برای regularization
..... ۵۰	..... شکل ۵۰- نمودار دقت و هزینه برای مدل بهینه
..... ۵۲	..... شکل ۵۱- ستون‌های دیتاست
..... ۵۴	..... شکل ۵۲- نمودار هزینه رگرسن با نرمال‌سازی ستون هدف
..... ۵۴	..... شکل ۵۳- نمودار هزینه رگرسن بدون نرمال‌سازی ستون هدف

## فهرست جدول ها

جدول ۱ - نتایج ارزیابی روی داده تست برای پارامترهای پیش فرض	۱۰
جدول ۲ - پارامترهای ارزیابی روی داده تست برای Gaussian RBF	۱۲
جدول ۳ - نتایج ارزیابی بر روی داده تست در مدل با وزن اولیه صفر	۱۴
جدول ۴ - نتایج ارزیابی بر روی مدل با داده پیش پردازش نشده	۱۷
جدول ۵ - مقایسه پارامترهای ارزیابی برای دادههای استاندارد و نرمال	۱۹
جدول ۶ - مقایسه نتایج اندازه بج روی داده تست	۲۶
جدول ۷ - مقایسه نتایج ارزیابی روی داده تست با نرخ یادگیری متفاوت	۳۲
جدول ۸ - مقایسه ارزیابی دادگان تست با توابع فعال ساز متفاوت	۳۶
جدول ۹ - مقایسه دقت سه مدل	۴۰
جدول ۱۰ - مقایسه هزینه سه مدل	۴۰
جدول ۱۱ - پارامترهای بهینه	۴۹
جدول ۱۲ - نتایج ارزیابی بر روی داده تست در مدل بهینه	۵۰
جدول ۱۳ - پارامترهای بهینه برای رگرسن	۵۳
جدول ۱۴ - مقایسه مقدار هزینه دادگان در رگرسن	۵۴
جدول ۱۵ - مقایسه مقدار هزینه دادگان در رگرسن بدون نرمال سازی	۵۵

## مقدمه

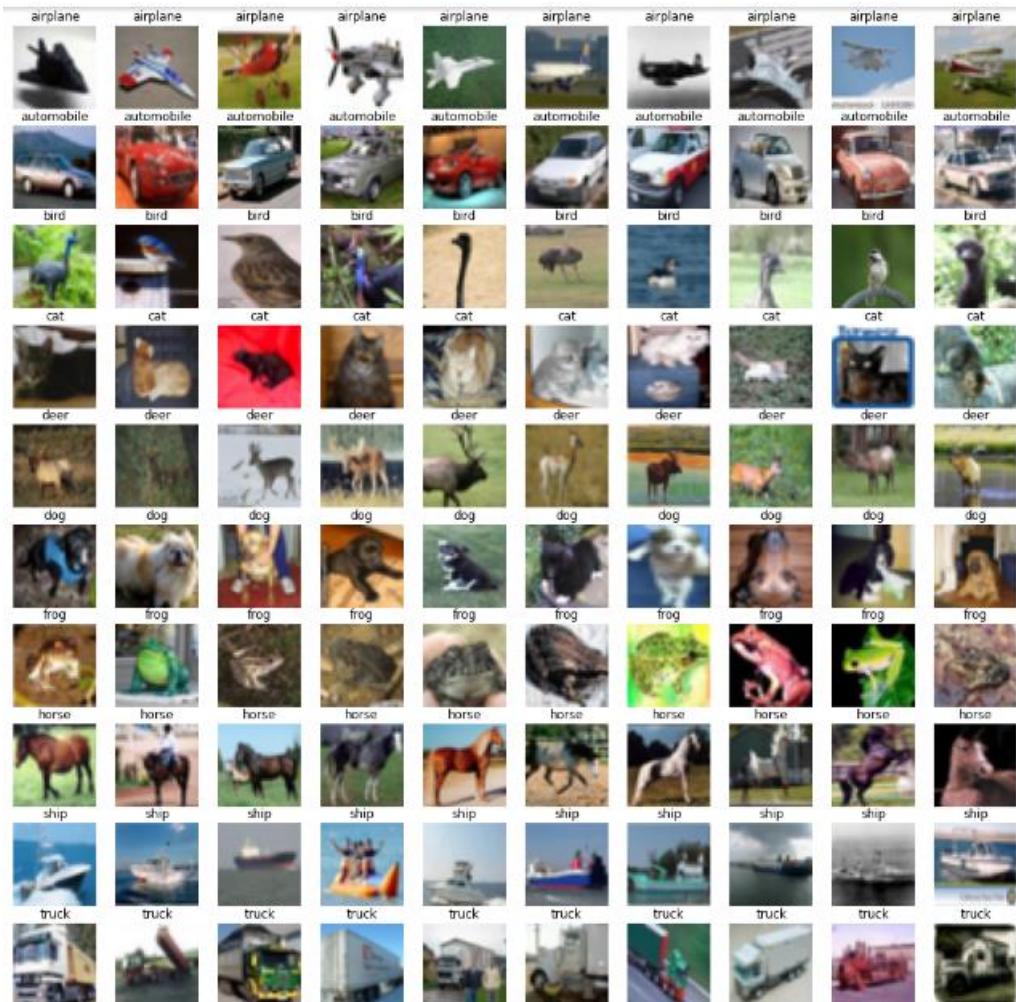
در ابتداء مقدماتی درباره این تمرین مطرح می‌کنیم. این تمرین با زبان برنامه نویسی پایتون نوشته شده است و طبق مقررات سوال از هیچگونه کتابخانه آماده یادگیری ماشین برای پیاده سازی مدل شبکه عصبی استفاده نشده و تنها از کتابخانه‌های ریاضی که پرکاربردترین آنها `numpy` است، استفاده شده است. در این تمرین طبق خواسته‌های سوال سعی شده است شبکه عصبی مورد نظر پیاده سازی شود، همچنانی پاسخ سوالات مطرح شده به صورت تشریحی و توصیفی در گزارش بیان شده است.

در هر مرحله طبق خواسته سوال، یکی از پارامترهای مسئله تغییر داده شده و تاثیرات آن بر روی آموزش شبکه عصبی را بیان کردیم.

## بخش اول – داده‌های جدولی

### نمایش داده‌ها

از هر کلاس، ۱۰ تصویر انتخاب شده و در شکل زیر نمایش داده‌ایم.



شکل ۱ - نمایش تصاویر

### سوال (۱.۱)

روابط پیاده‌سازی شده در مرحله انتشار رو به جلو و انتشار رو به عقب را به صورت مختصر شرح دهید و این دو فرآیند را پیاده‌سازی کنید.

در فرآیند انتشار به جلو در گام اول، وزن‌ها را به صورت رندوم مقدار دهی کرده و بایاس را نیز یک بردار تمام صفر در نظر می‌گیریم. ماتریس وزن با ابعاد ورودی در خروجی ساخته می‌شود. همچنان بایاس یک بردار به اندازه تعداد نورون‌های خروجی می‌باشد. برای هر لایه‌ای که در شبکه عصبی وجود دارد، به کمک فرمول زیر خروجی لایه را محاسبه می‌کنیم.

$$Z^{[L]} = (W^{[L]} \cdot A^{[L-1]}) + Bias^{[L]} \quad (\text{فرمول ۱})$$

سپس روی خروجی به دست آمده، از تابع فعال ساز استفاده می‌کنیم. تابع فعال ساز به ما کمک می‌کند تا خروجی غیرخطی شود و این غیرخطی شدن منجر به تشکیل مرزهای تصمیم با اشکال متفاوت و حل مسئله می‌گردد. تابع فعال سازی در لایه آخر با توجه به نوع مسئله انتخاب می‌شود. در این مسئله به صورت پیش‌فرض از softmax استفاده شده است.

$$A^{[L]} = g^{[L]}(Z^{[L]}) \quad (\text{فرمول ۲})$$

به طور کلی در مرحله انتشار رو به جلو به تعداد لایه‌هایی که داریم به کمک فرمول ۱ خروجی لایه را محاسبه کرده و پس از اعمال تابع فعال سازی روی آن، نتیجه را به عنوان ورودی لایه بعد می‌فرستیم. اولین گام برای انشار رو به عقب، محاسبه تابع هزینه است. ما مدل را در جهت کاهش این تابع آموزش خواهیم داد. تابع هزینه متناسب با نوع مسئله انتخاب می‌شود. در این قسمت به طور پیش‌فرض از Cross-entropy استفاده شده است. پیاده‌سازی این تابع به همراه مشتق آن در پوشه losses و فایل crossEntropy.py قرار گرفته است.

در هر لایه محاسباتی مشابه با انتشار رو به جلو پیش می‌رویم، با این تفاوت که به جای محاسبه  $Z$  و  $A$  باید مشتق تابع هزینه و با خروجی لایه بعدی را نسبت به  $W$  و  $Z$  و  $b$  به کمک فرمول‌های زیر محاسبه کنیم.

$$\frac{\partial L}{\partial Z^{[L-1]}} = (W^{[L]}).T \cdot \frac{\partial L}{\partial Z^{[L]}} * g^{[l-1]'}(Z^{l-1}) \quad (\text{فرمول ۳})$$

$$\frac{\partial L}{\partial W^{[L]}} = \cdot \frac{\partial L}{\partial Z^{[l]}} \cdot [A^{[l-1]}].T \quad (\text{فرمول ۴})$$

$$\frac{\partial L}{\partial b^{[L]}} = \sum_i^m Z^{[L]} \quad (\text{فرمول ۵})$$

سپس به کمک  $\frac{\partial L}{\partial b^{[L]}}$  و  $\frac{\partial L}{\partial W^{[L]}}$  مقادیر اولیه را آپدیت می‌کنیم. برای بروز رسانی مشتقان را در نرخ یادگیری ضرب کرده و از وزن یا بایاس کم می‌کنیم.

به طور کلی برای پیاده‌سازی پروژه، برای هر لایه نورون یک کلاس به نام Layer تعریف شده است که دو متدهای Forward و backward داشته و این دو عملیات برای آن لایه توسط این دو متدهای انجام می‌شود. این کلاس در پوشه nets و فایل ANN.py قرار دارد.

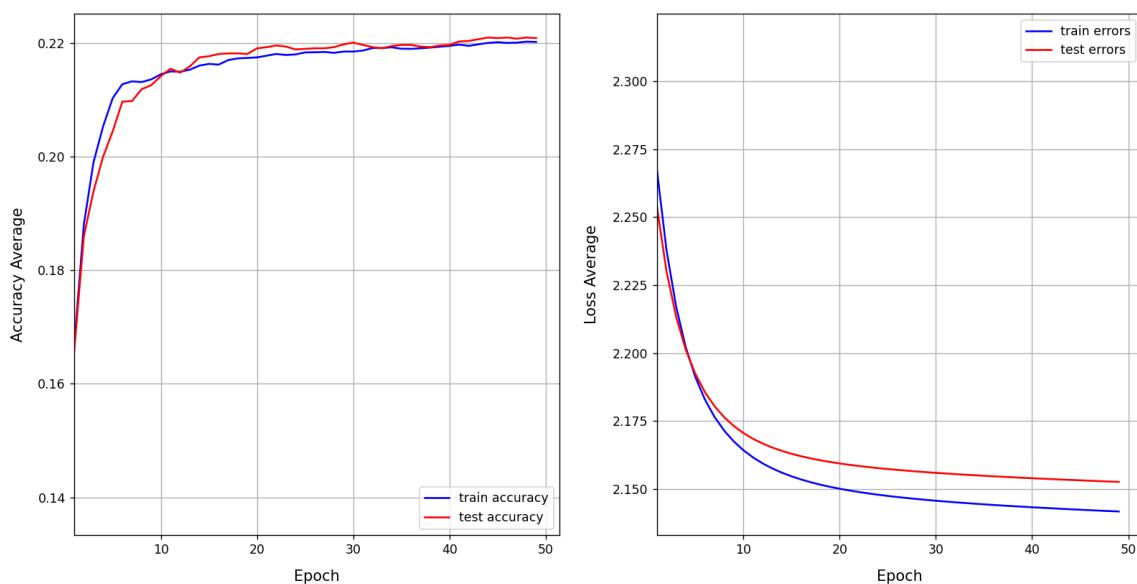
علاوه بر این برای توابع فعال ساز نیز کلاس ساخته شده است که هر کدام از آن‌ها نیز متدهای backward و Forward دارند. کدهای این توابع نیز در پوشه nets و فایل ANN.py قرار دارد.

حال در ابتداء برنامه براساس پارامترهای داده و به کمک کلاس‌های بالا، مدل را می‌سازیم. برای ساخت مدل یک ابتداء یک کلاس **Layer** با ماتریس وزن با ابعاد تصویر ورودی و نورون لایه اول می‌سازیم. سپس تابع فعال‌سازی را به آن اضافه می‌کنیم. در نهایت نیز یک **Layer** دیگر با ماتریس با ابعاد نورون لایه مخفی و نورون خروجی می‌سازیم. (این مدل برای مقادیر دیفالت داده شده یعنی یک لایه مخفی ۱۶ تایی است. در صورتی که تعداد لایه‌ها افزایش پیدا کند، مشکلی وجود ندارد). متده `get_model` در `deeplearning/learning.py` این وظیفه را برعهده دارد. پس از ساخت مدل، تابع `fit` صدا زده می‌شود. در این تابع به تعداد ایپاک‌ها یک فور می‌زنیم و در آن داده‌ها به `batch` تقسیم می‌کنیم. داده‌های هر `batch` را به تابع `train` می‌فرستیم. در این تابع، عملیات `forward` انجام می‌شود. برای عملیات `forward` بر روی مدلی که داریم `iterate` کرده و متده `forward` مربوط به هر لایه فراخوانی می‌شود. ورودی متده `forward` هر لایه، خروجی `forward` لایه قبلی است. پس از `forward`، مقدار تابع هزینه محاسبه شده و سپس `backward` مربوط به هر لایه را فراخوانی کرده و وزن‌ها آپدیت می‌شوند. برای اجرای کد، دیتاست `cifar` را باید در پوشه `dataset/cifar` قرار دهید. زیرا به دلیل حجم زیاد این دیتاست، آپلود نشده است.

## سوال ۱.۲) آموزش مدل با پارامترهای خواسته شده

۰.۰ از داده‌ها برای داده اعتبارسنجی انتخاب شده‌اند. برای اجرای این قسمت فایل `main.py` را اجرا کنید. این فایل ابتداء `train.py` و سپس `test.py` را اجرا می‌کند. مدل با پارامترهای خواسته شده آموزش دیده شد و نمودار دقت و خطا برای داده تست و اعتبار سنجی به صورت زیر است.

Accuracy and Loss plot for train and validation data



شکل ۲- نمودار دقت و هزینه با پارامترهای پیش‌فرض

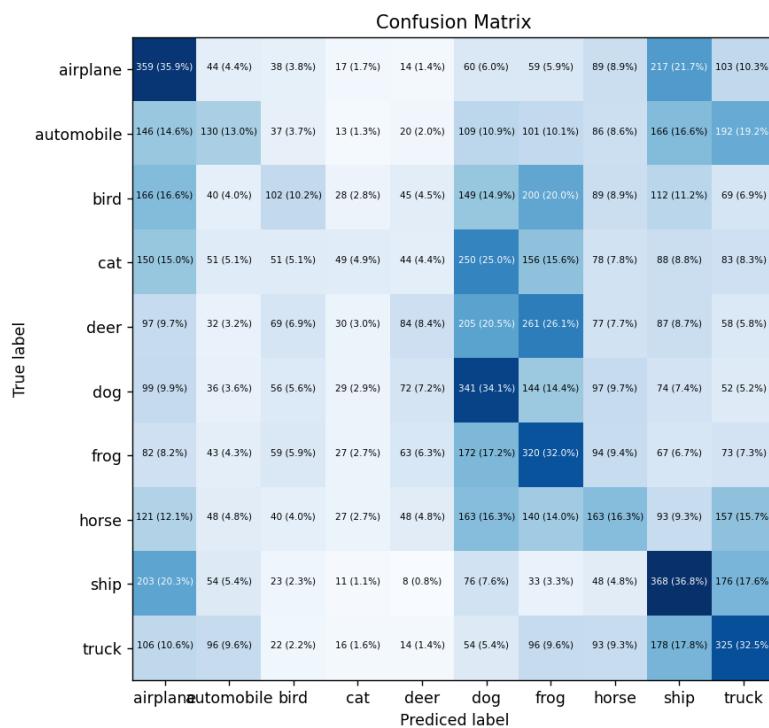
مجموعه داده‌های cifar از مسائل سخت می‌باشد و همانطور که از مقدار دقت و loss در داده‌های ارزیابی مشاهده می‌کنیم، مقادیر هایپرپارامترها به گونه‌ای نمی‌باشد که بتواند به دقت خوبی برسد. هر چند مدل آموزش دیده و مقدار دقت از رندوم فاصله گرفته است، اما با دقت موردنظر مناسب فاصله زیادی دارد.

پس از آموزش مدل، بر روی داده‌های تست مدل را آزمایش می‌کنیم. جدول زیر پارامترهای ارزیابی به دست آمده را نشان می‌دهد. همانطور که مشاهده می‌کنید، دقت به دست آمده از دقت داده‌های اعتبارسنجی و آموزش کمتر است و این مسئله منطقی می‌باشد.

جدول ۱- نتایج ارزیابی روی داده تست برای پارامترهای پیش فرض

Accuracy	۲۲.۴۱۰
precision	۲۲.۴۱۰
recall	۲۲.۴۱۰
F1 score	۲۲.۴۱۰
Loss	۲.۱۴۲

در شکل زیر ماتریس آشفتگی برای داده‌های تست را مشاهده می‌کنید.



شکل ۳- ماتریس آشفتگی برای مدل اولیه

۳۶ درصد از داده‌های کشتی به درستی طبقه‌بندی شده‌اند. پس از آن در کلاس هواپیما نیز ۳۵ درصد از داده‌ها به درستی طبقه‌بندی شده‌اند. مدل در تشخیص قروباگه، کامیون و سگ نیز به درصدهای قابل قبولی رسیده است. اما عمکلر آن در سایر کلاس‌ها ضعیف بوده است. بنابراین تشخیص ۵ کلاس کشتی، سگ، قورباگه، کامیون و هواپیما برای مدل ساده‌تر است.

### (۱.۳) سوال

تابع Gaussian-RBF را پیاده‌سازی کنید. آیا استفاده از تابع هزینه خطای میانگین مربعات با توجه به ماهیت داده‌ها منطقی به نظر می‌رسد؟

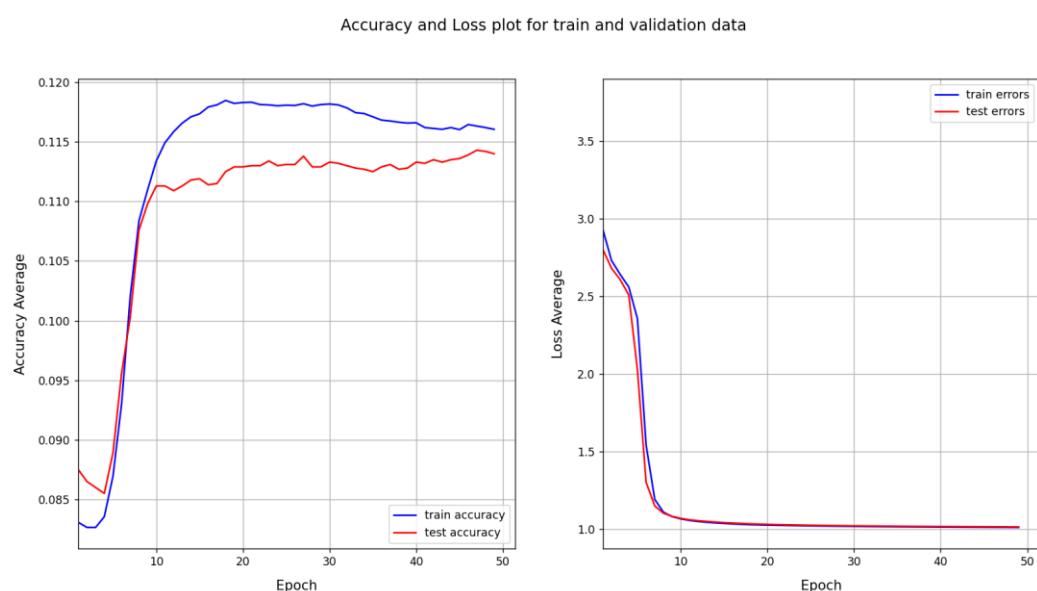
پیاده‌سازی این تابع زیان در پوشه losses و فایل MSE.py قرار گرفته است. برای پیاده‌سازی تابع هزینه و مشتق آن از فرمول زیر کمک گرفتیم.

$$p_i = \text{sigmoid}(X)$$

$$MSE_{loss} = \frac{1}{2} (y_i - p_i)^2$$

$$\text{gradian}_{MSE} = (y_i - p_i) * p_i * (1 - p_i)$$

پس از آموزش مدل نمودار دقت و هزینه به شکل زیر درآمد. همانطور که از نمودار دقت مشخص است، دقت قابل قبولی حاصل نشده است. اگرچه مدل آموزش دیده و از حالت رندوم خارج شده اما تفاوت زیادی با دقت در cross-entropy دارد.



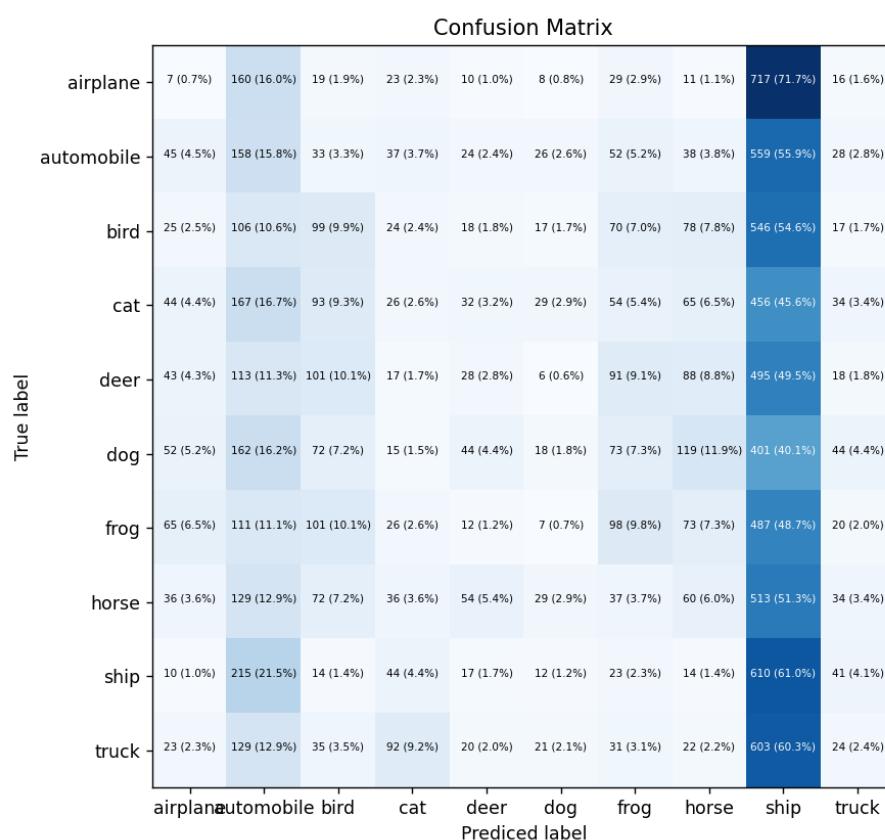
شکل ۴- نمودار دقت و خطا با Gaussian RBF

در جدول زیر نتایج ارزیابی بر روی داده تست، نوشته شده است. همانطور که می‌بینیم، مدل آموزش دیده با تابع فعالیت softmax و زیان categorical\_crossentropy بهتر از سیگموئید تابع هزینه عمل می‌کند.

جدول ۲- پارامترهای ارزیابی روی داده تست برای Gaussian RBF

Accuracy	۱۱.۲۸
precision	۱۱.۲۸
recall	۱۱.۲۸
F1 score	۱۱.۲۸
Loss	۲.۳۴

ماتریس آشفتگی در شکل زیر نیز، عملکرد نامطلوب MSE را نشان می‌دهد.



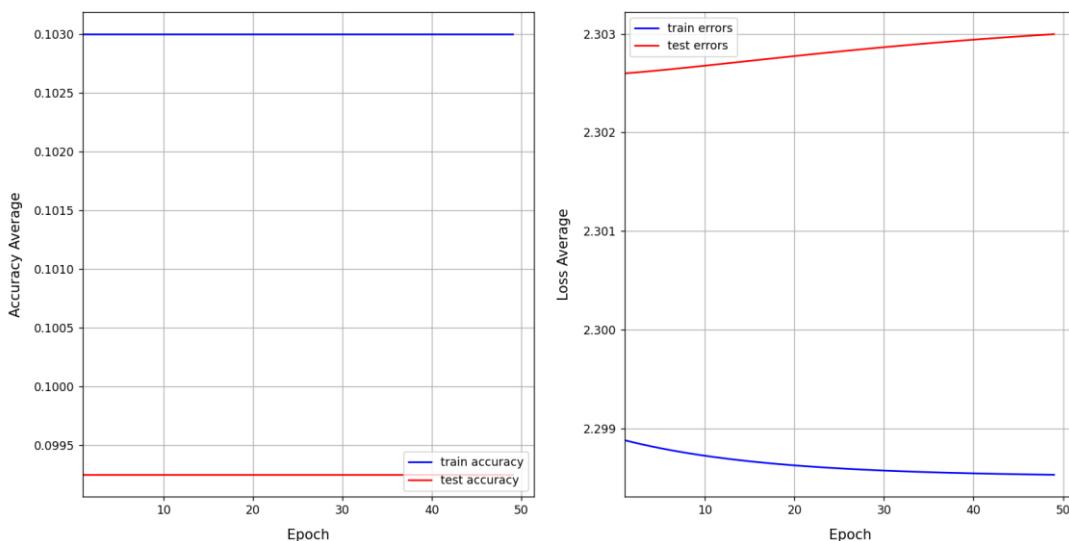
شکل ۵- ماتریس آشفتگی Gaussian RBF

میانگین مربعات خطای متریک فاصله است بنابراین برای این مسئله انتخاب مناسبی نیست. زیرا، بر روی تخمین‌های نقطه‌ای خوب کار می‌کند و نه توزیع‌ها. خروجی مدل رگرسیون تخمین نقطه‌ای است و بنابراین MSE برای آن کاربرد مناسب است. حال، Cross entropy برای محاسبه زیان برای طبقه‌بندی مناسب‌تر است. در مجموع Cross entropy در کارهای طبقه‌بندی بر MSE ترجیح داده می‌شود. زیرا، احتمالات هر کلاس را در نظر می‌گیرد و پیش‌بینی‌های نادرست را به شدت جریمه می‌کند. در حالی که MSE خطاهای بزرگ را شدیدتر از خطاهای کوچک جریمه می‌کند. MSE برای کارهای رگرسیونی که در آن خروجی یک مقدار پیوسته است، مناسب‌تر است.

#### (۱.۴) سوال

در صورتی که وزن و بایاس‌ها را صفر بگذاریم، آیا شبکه آموزش می‌یابد؟ این مسئله را به صورت تئوری و عملی بررسی کنید.

خیر شبکه آموزش نمی‌بیند. برای بررسی این مسئله ابتدا به صورت عملی نتیجه را مشاهده می‌کنیم. با قرار دادن صفر به عنوان وزن‌های اولیه، نمودار دقت و هزینه برای داده آموزش و اعتبارسنجی به صورت زیر می‌گردد.



شکل ۶- نمودار دقت و هزینه با مقدار دهی اولیه صفر برای وزن

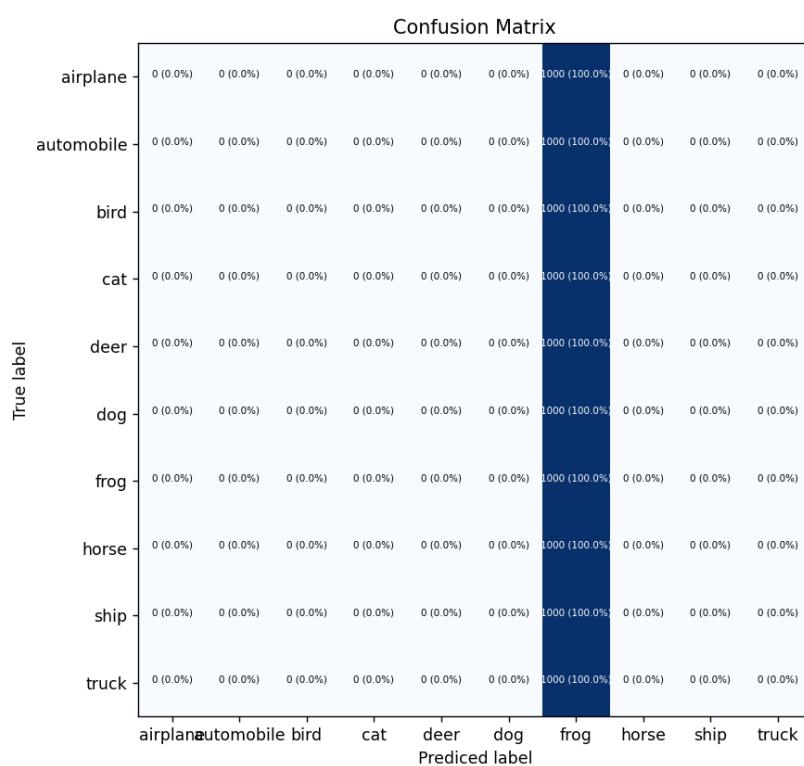
همانطور که در شکل بالا مشاهده می‌کنید، مقدار دقت کاملاً ثابت است و مقدار هزینه نیز در داده اعتبارسنجی رو به افزایش و در داده آموزش با شیب بسیار کمی رو به کاهش است. این مسئله نشان می‌دهد که مدل آموزش ندیده است.

برای بررسی بیشتر در جدول زیر نتایج روی داده آزمایش را مشاهده می‌کنید. تمامی پارامترها برابر ۱۰٪ است. با توجه به اینکه ۱۰ کلاس داریم، دقت ۱۰ درصد نشان دهنده عملکرد تصادفی مدل است. بنابراین مدل با وزن‌های اولیه صفر نمی‌تواند آموزش ببیند.

جدول ۳ - نتایج ارزیابی بر روی داده تست در مدل با وزن اولیه صفر

Accuracy	۱۰۰
precision	۱۰۰
recall	۱۰۰
F1 score	۱۰۰
Loss	۲۰۵

شکل زیر ماتریس آشفتگی مربوط به داده‌های تست را نشان می‌دهد. تمام تصاویر به عنوان قورباغه برچسب‌گذاری شده‌اند و دقت ۱۰ درصدی که به دست آمده است به خاطر همین مسئله است. زیرا، تصاویری که واقعاً قورباغه بوده‌اند درست پیش‌بینی شده‌اند.



شکل ۷ - ماتریس آشفتگی مربوط به مدل با وزن اولیه صفر

برای امتحان این مسئله در کد، در config.yml مقدار W\_init را برابر Zero قرار دهید.

حال به بررسی مسئله از منظر تئوری می پردازیم.

در انتشار رو به جلو، ورودی با مقدار وزن که صفر است ضرب می شود و مقدار ورودی صفر را می سازد. همچنین تمام خروجی ها مقدار ثابتی خواهند بود(۰.۵). برای سیگموئید و صفر برای Relu. در مرحله انتشار رو به عقب، رابطه زیر را خواهیم داشت.

$$\frac{\partial l}{\partial W} = \frac{\partial l}{\partial A} \frac{\partial A}{\partial Z} \frac{\partial Z}{\partial W}$$

$\frac{\partial l}{\partial A}$  مشتق تابع هزینه و  $\frac{\partial Z}{\partial W}$  مشتق تابع فعال سازی می باشد. برابر با مقدار خروجی لایه قبل می شود( $A^{l-1}$ ). بنابراین مقدار  $dW$  یک بردار خواهد بود که تمام المان های آن یکسان است. پس،  $W_L = W_L - \alpha dW_L$  همیشه در یک جهت حرکت می کند.

حال اگر تابع فعال سازی به گونه ای باشد که  $g(0) = 0$  مانند Relu، در مرحله انتشار رو به جلو، چون وزن ها صفر است، بنابراین تمام خروجی ها صفر خواهد شد. بنابراین ورودی لایه بعدی نیز صفر خواهد شد. در انتشار رو به عقب نیز گرادیان وزن ها نیز صفر می شود و در عمل وزن ها به روز رسانی نمی شوند.

## سوال (۱.۵)

آیا نرمال / استاندارد کردن داده های ورودی تاثیری در دقت مدل دارد؟ این موضوع را به صورت تئوری و عملی بررسی کنید.

از نظر تئوری و نگاه کلی به مسئله نباید تاثیری روی دقت داشته باشد. در هر مرحله خروجی از رابطه  $Y = X * W + B$  به دست می آید. حال اگر  $X_{new} = (X - M) * T$  باشد، خواهیم داشت:

$$Y = (X - M) * T * W + B .$$

این عبارت برای با  $Y = X_1 * W_1 + B_1$  می باشد، به طوریکه:

$$W_1 = T^{-1} W , \quad B_1 = B + M * W$$

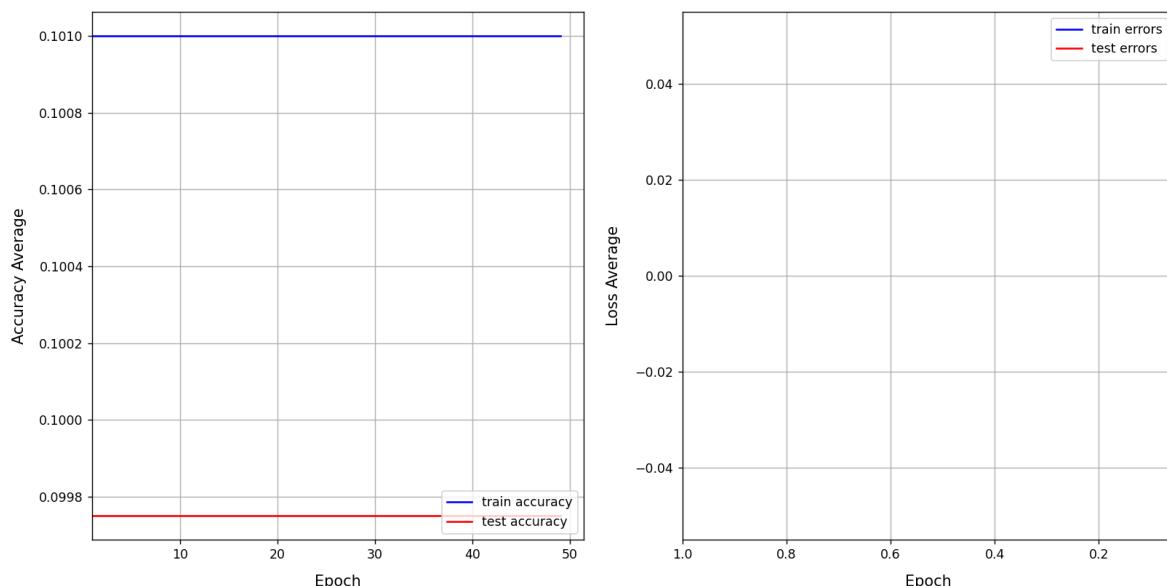
بنابراین، استانداردسازی داده های ورودی نباید بر خروجی یا دقت تأثیر بگذارد. اما در عمل تاثیر آن را مشاهده می کنیم. علت این تفاوت در مرحله انتشار رو به عقب و مقادیر گرادیان است. اگر یک داده تک بعدی ساده X داشته باشیم و از MSE به عنوان تابع ضرر استفاده کنیم، به روز رسانی گرادیان با استفاده از گرادیان نزولی به صورت زیر است:

$$\frac{\partial l}{\partial W} = \frac{\partial l}{\partial Y'} \cdot \frac{\partial Y'}{\partial W} = \frac{2(Y - Y').T}{N} * X$$

مقدار ورودی در فرمول gradient descent قرار می‌گیرد و در میزان به روز رسانی تاثیر می‌گذارد. یعنی زمانی که مقدار ورودی بزرگ است، جهش بیشتری در گرادیان خواهیم داشت و مقدار  $W$  کوچکتر می‌شود. در واقع زمانی که وزن به طور رندوم انتهاب می‌شود، فاصله این وزن رندوم و global min کم است. اما با بزرگ بودن مقدار  $X$  جهش‌های بزرگی به سمت هدف خواهیم داشت. در صورتی که به جهش‌های کوچک نیاز داریم، این مسئله منجر می‌شود که نوسان‌های بسیار زیادی داشته باشیم و فرآیند آموزش مختل شود.

در صورتی که هیچ پردازشی روی داده‌ها صورت نگیرد، در تابع softmax برای محاسبه خروجی overflow رخ می‌دهد. این overflow منجر به nan شدن مقدار loss می‌گردد و نمی‌توان مشتق آن را نسبت به وزن و بایاس محاسبه کرد. به همین خاطر، وزن‌ها به روز رسانی نمی‌شوند و یادگیری صورت نمی‌گیرد. در شکل زیر نمودار دقت و هزینه را بر حسب تعداد ایپاک مشاهده می‌کنید. به دلیل اینکه تابع هزینه nan شده است، نموداری برای آن نداریم. همچنین به دلیل عدم یادگیری مدل، دقت نیز برای داده آموزش و اعتبارسنجی ثابت بوده و تغییری نداشته است.

Accuracy and Loss plot for train and validation data



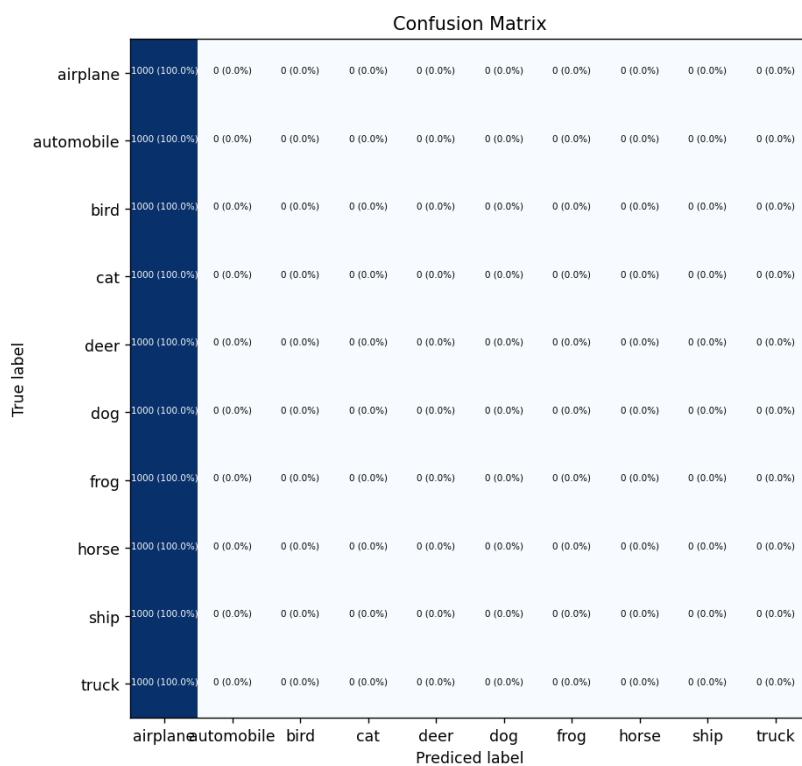
شکل ۸- نمودار دقت و هزینه بر روی داده پیش پردازش نشده

در جدول زیر دقت روی داده‌های تست را مشاهده می‌کنید. مقدار به دست آمده ۱۰٪ است که نشان می‌هد طبقه‌بندی به صورت رندوم انجام گرفته است.

جدول ۴- نتایج ارزیابی بر روی مدل با داده پیش پردازش نشده

Accuracy	۱۰۰
precision	۱۰۰
recall	۱۰۰
F1 score	۱۰۰

ماتریس آشفتگی نیز صحت مطالب بالا را تائید می کند. تمام تصاویر به عنوان کلاس هواپیما طبقه بندی شده اند.



شکل ۹- ماتریس آشفتگی برای داده پیش پردازش نشده

کاهش مقیاس داده ورودی اندازه گرادیان مورد استفاده برای به روز رسانی وزن ها را کاهش می دهد و منجر به مدل و فرآیند آموزشی پایدارتر می شود. در صورتی که مقادیر ورودی بین ۰ و یک باشند، نیازی به نرمال یا استاندارد کردن نیست. اما در این مسئله که مقادیر داده بین ۰ تا ۲۵۵ می باشند، باید روى داده ها پردازش انجام شود. با نرمال کردن داده ها، داده ها بین صفر و یک قرار می گیرند. بنابراین برای انجام این پردازش باید از فرمول زیر کمک بگیریم.

$$X_{new} = \frac{X - \min}{(\max - \min)} \quad (\text{فرمول ۶})$$

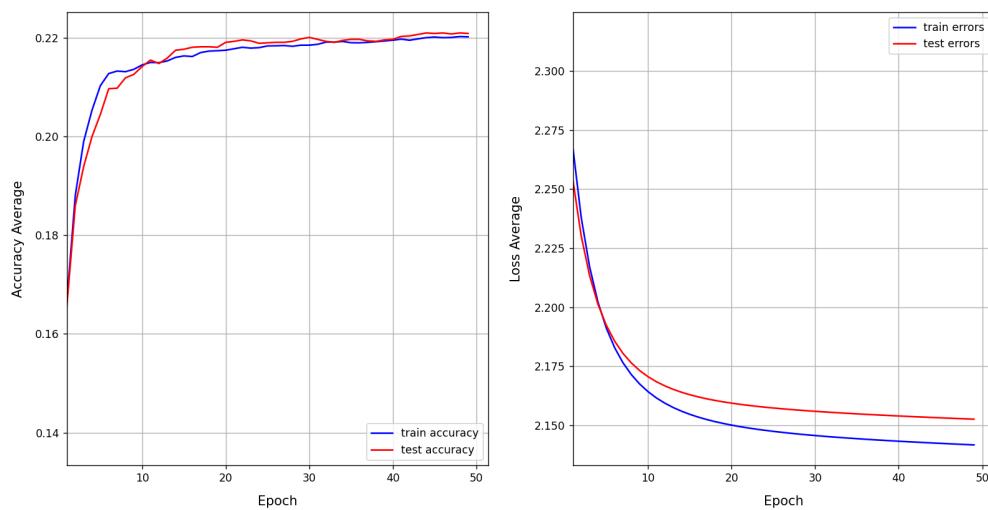
با توجه به اینکه داده‌های ما پیکسل‌های رنگی می‌باشند، کمترین مقدار صفر و بیشترین مقدار ۲۵۵ است. بنابراین برای نرمال کردن داده‌ها، هر داده بر ۲۵۵ تقسیم می‌شود.

با استاندارد کردن داده‌ها، مقادیر بین ۱ و ۰ قرار می‌گیرند. در واقع مقادیر به گونه‌ای تغییر می‌کنند که توزیع داده‌ها میانگین صفر و انحراف معیار یک داشته باشد. پروسه استاندارد کردن در نظر می‌گیرد که داده‌ها توزیع گوسی دارند. بنابراین بهترین نتیجه از استاندارد کردن زمانی به دست می‌آید که داده‌ها توزیع نرمال داشته باشند. برای استاندارد کردن از فرمول زیر کمک می‌گیریم.

$$X_{new} = \frac{x - \text{mean}}{\text{standard\_deviation}} \quad (\text{فرمول ۷})$$

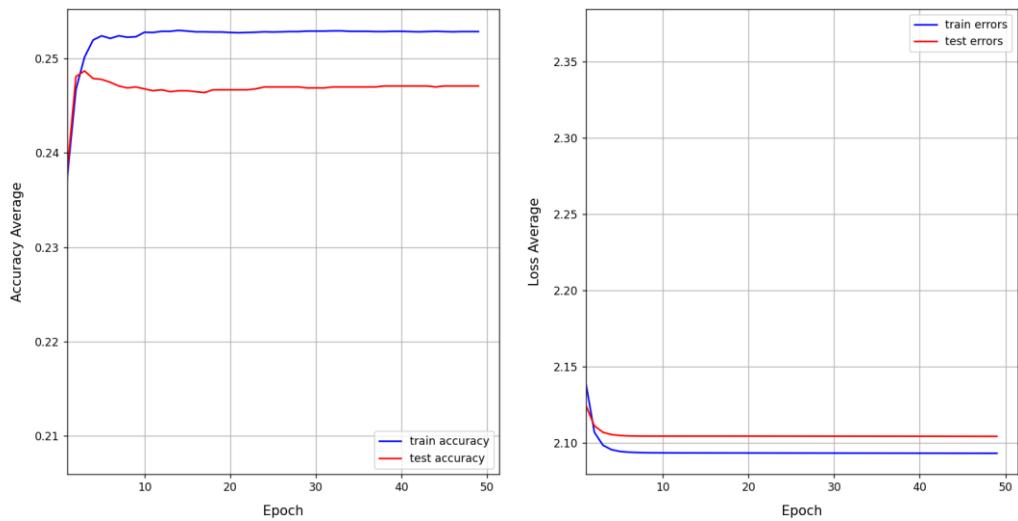
تصویر زیر نمودار دقت و هزینه برای داده آموزشی و اعتبارسنجی را در حالت نرمال بودن داده‌ها نشان می‌دهد. همانطور که می‌بینید، مدل آموزش دیده و مقادیر loss رو به کاهش هستند.

Accuracy and Loss plot for train and validation data



شکل ۱۰ - نمودار دقت و هزینه در حالت نرمال بودن داده‌ها

نمودار زیر مربوط به فرآیند آموزش در حالت استاندارد بودن داده‌ها می‌باشد. کاهش تابع زیان در این حالت کمتر است اما مقدار نهایی loss کمتر می‌باشد. دقت مدل در داده‌های اعتبارسنجی نیز اندکی بیشتر از حالت نرمال بودن داده‌ها می‌باشد. در حالت نرمال، نمودار loss و دقت تا آخرین ایپاک در حال کاهش بودند، اما در داده‌های استاندارد، نمودار پس از حدود ۲۰ ایپاک تقریباً ثابت می‌شود.



شکل ۱۱- نمودار هزینه و دقت مدل بر روی داده‌های استاندارد

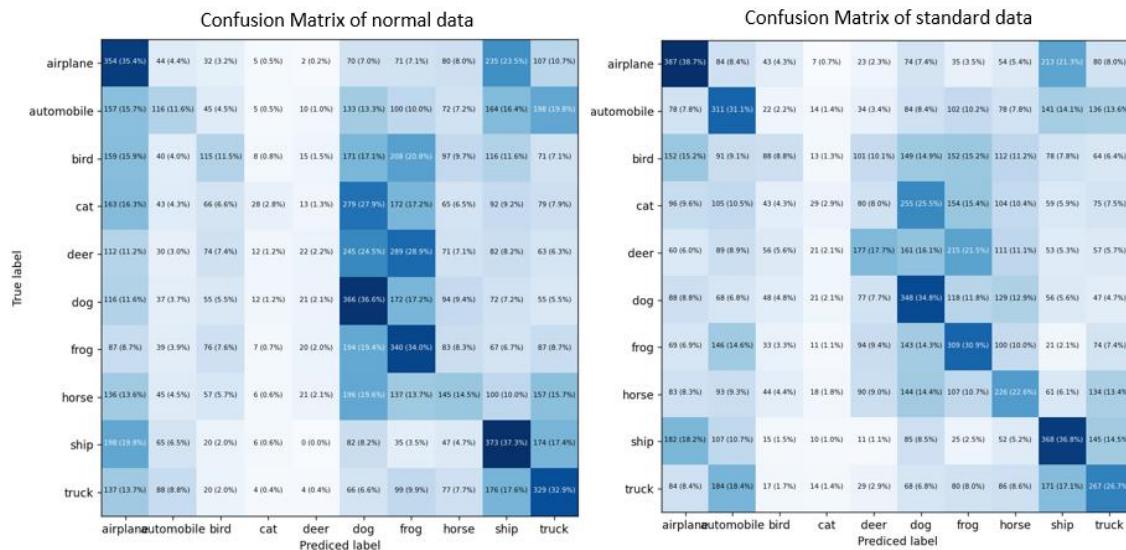
در جدول زیر پارامترهای ارزیابی مدل بر روی داده تست آمده است. همانطور که مشاهده می‌کنید، عملکرد روی داده استاندارد شده از داده نرمال شده بیشتر بهتر بوده است. این مسئله می‌تواند به خاطر ماهیت داده‌های ورودی باشد. همانطور که قبلاً گفته شد، اگر داده‌ها توزیع نرمال داشته باشند، استانداردسازی بهتر از نرمال‌سازی جواب می‌دهد.

جدول ۵- مقایسه پارامترهای ارزیابی برای داده‌های استاندارد و نرمال

	Normalized	Standardized
Accuracy	۲۲.۴۱۰	۲۵.۶۹۰
precision	۲۲.۴۱۰	۲۵.۶۹۰
recall	۲۲.۴۱۰	۲۵.۶۹۰
F1 score	۲۲.۴۱۰	۲۵.۶۹۰
Loss	۲.۱۴۲	۲.۰۹۳

ماتریس آشفتگی برای هر دو حالت نرمال و استاندارد در شکل زیر نشان داده شده است. در حالت نرمال ۵ کلاس بیشترین تشخیص را داشتند. اما در حالت استاندارد بودن داده‌ها، حدود ۷ کلاس قابل تشخیص

بوده است. هر چند تعداد تشخیص‌ها برای هر کلاس در استاندار از نرمال کمتر است، اما تنوع شناسایی کلاس‌ها در استاندارد بیشتر است.



شکل ۱۲- ماتریس آشفتگی در حالت نرمال و استاندارد بودن داده‌ها

پیاده سازی این قسمت در فایل datasets/Cifar preprocessing در پوشه انجام شده است.

## سوال (۱.۶)

سه روش زیر را با یکدیگر مقایسه کنید. برای آموزش مدل با هر کدام از روش‌ها اندازه بسته را باید چه مقداری بگذاریم؟ تاثیر هر کدام از این سه روش را به صورت عملی بررسی و مقایسه کنید.

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini Batch Gradient Descent

در تمام داده‌های آموزشی در یک مرحله آموزش و محاسبه گرادیان در نظر گرفته می‌شوند. میانگین گرادیان تمام داده‌های آموزشی را می‌گیریم و سپس از آن گرادیان میانگین گرفته و برای بهروزرسانی پارامترهایمان استفاده می‌کنیم.

در تمام داده‌ها را برای هر مرحله از Gradient Descent Batch Gradient Descent در نظر می‌گیریم. اما در SGD (Stochastic Gradient Descent)، در هر زمان فقط یک داده را برای برداشتن یک گام در نظر می‌گیریم. مراحل زیر در یک دوره از SGD انجام می‌شود:

- یک داده را نظر گرفته و آن را به شبکه عصبی می‌دهیم.

- خروجی را به دست آورده و به کمک آن گرادیان را محاسبه می‌کنیم.
- از این گرادیان برای به روز رسانی وزن‌ها استفاده کنیم.
- مراحل بالا را برای تمام نمونه‌های مجموعه داده آموزشی تکرار می‌کنیم.

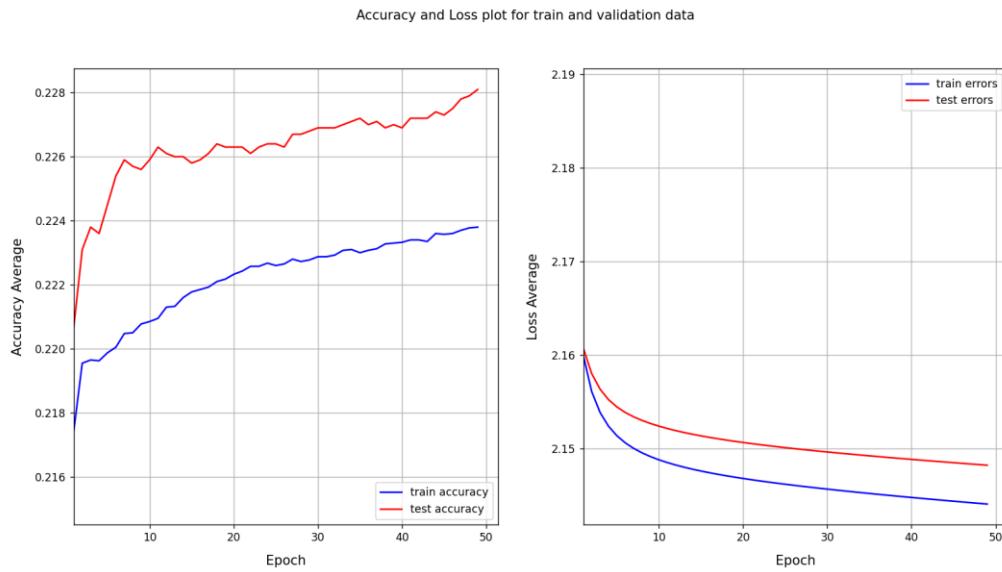
از آنجایی که ما در هر زمان فقط یک داده را در نظر می‌گیریم، هزینه بر روی نمونه‌های آموزشی نوسان دارد و لزوماً در هر گام کاهش نمی‌یابد. اما در مجموع شاهد کاهش هزینه خواهیم بود. همچنانی به دلیل نوسانی بودنتابع هزینه، دقیقاً به نقطه مینیمم نمیرسیم و در اطراف و نزدیکی آن نوسان‌های ریز داریم.

زمانی که مجموعه داده بزرگ باشد می‌توان از SGD استفاده کرد. Batch Gradient Descent حداقل همگرا می‌شود و SGD برای مجموعه داده‌های بزرگتر سریعتر همگرا می‌شود. اما، از آنجایی که در SGD ما در هر زمان فقط از یک داده استفاده می‌کنیم، نمی‌توانیم پیاده‌سازی برداری شده را روی آن پیاده‌سازی کنیم. این می‌تواند محاسبات را کند کند. برای مقابله با این مشکل، ترکیبی از Batch Gradient Descent و SGD استفاده می‌شود که به آن Mini Batch Gradient Descent گفته می‌شود. در هر ایپاک مراحل زیر انجام می‌شود:

- یک mini batch انتخاب می‌کنیم.
- آن را به شبکه عصبی می‌دهیم.
- میانگین گرادیان mini batch را محاسبه می‌کنیم.
- از گرادیان میانگینی که در مرحله ۳ محاسبه کردیم برای به روز رسانی وزن‌ها استفاده می‌کنیم.
- مراحل ۴-۱ را برای مینی بچهایی که ایجاد کردیم تکرار می‌کنیم.

ابتدا روش Batch Gradient Descent را به طور عملی بررسی می‌کنیم. به همین منظور مقدار config.yml در batch\_size کل داده‌ها قرار می‌دهیم. در این حالت، به دلیل زیاد بودن داده‌ها و پرشدن رم، سیستم از کار افتاد و بسیار بسیار کند شد. به دلیل اینکه حجم زیادی از داده‌ها برای انجام محاسبات وارد رم می‌شود، سیستم به شدت کند شده و هنگ می‌کند. به همین جهت امکان آموزش بر روی سیستم شخصی و با منابع محدود وجود ندارد.

در گام بعد Stochastic Gradient Descent را آزمایش می‌کنیم. به همین منظور مقدار batch\_size را برابر یک قرار می‌دهیم. در این حالت چون محاسبات به صورت برداری انجام نمی‌شوند و روی تک تک داده‌ها پردازش انجام می‌دهیم، فرآیند آموزش بسیار کند است.



شکل ۱۳ - نمودار دقت و هزینه مربوط به SGD

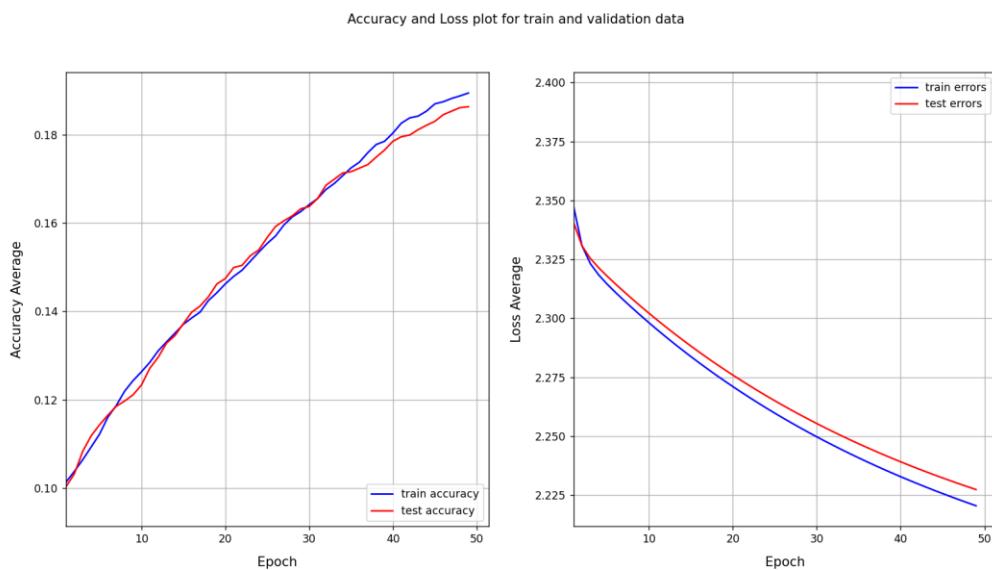
در این حالت به دقت  $22.630\%$  رسیدیم که بسیار بهتر از Batch Gradient Descent است. شکل زیر ماتریس آشتفتگی SGD برای داده تست را نشان می‌دهد.

Confusion Matrix											
True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
	airplane	305 (30.5%)	30 (3.0%)	21 (2.1%)	22 (2.2%)	0 (0.0%)	44 (4.4%)	148 (14.8%)	28 (2.8%)	342 (34.2%)	60 (6.0%)
	automobile	115 (11.5%)	156 (15.6%)	19 (1.9%)	33 (3.3%)	0 (0.0%)	92 (9.2%)	189 (18.9%)	42 (4.2%)	226 (22.6%)	128 (12.8%)
	bird	135 (13.5%)	43 (4.3%)	56 (5.6%)	50 (5.0%)	1 (0.1%)	106 (10.6%)	367 (36.7%)	28 (2.8%)	178 (17.8%)	36 (3.6%)
	cat	106 (10.6%)	54 (5.4%)	33 (3.3%)	87 (8.7%)	0 (0.0%)	195 (19.5%)	300 (30.0%)	33 (3.3%)	139 (13.9%)	53 (5.3%)
	deer	83 (8.3%)	33 (3.3%)	39 (3.9%)	54 (5.4%)	0 (0.0%)	140 (14.0%)	464 (46.4%)	26 (2.6%)	129 (12.9%)	32 (3.2%)
	dog	90 (9.0%)	26 (2.6%)	40 (4.0%)	62 (6.2%)	0 (0.0%)	283 (28.3%)	319 (31.9%)	39 (3.9%)	111 (11.1%)	30 (3.0%)
	frog	63 (6.3%)	53 (5.3%)	26 (2.6%)	40 (4.0%)	0 (0.0%)	114 (11.4%)	532 (53.2%)	27 (2.7%)	107 (10.7%)	38 (3.8%)
	horse	117 (11.7%)	56 (5.6%)	26 (2.6%)	55 (5.5%)	1 (0.1%)	148 (14.8%)	247 (24.7%)	91 (9.1%)	159 (15.9%)	100 (10.0%)
	ship	140 (14.0%)	59 (5.9%)	14 (1.4%)	26 (2.6%)	0 (0.0%)	59 (5.9%)	79 (7.9%)	14 (1.4%)	529 (52.9%)	80 (8.0%)
	truck	103 (10.3%)	101 (10.1%)	13 (1.3%)	31 (3.1%)	0 (0.0%)	40 (4.0%)	179 (17.9%)	42 (4.2%)	267 (26.7%)	224 (22.4%)

شکل ۱۴ - ماتریس آشتفتگی SGD روی داده تست

برای SGD با مقدار batch برابر  $32$  در سوال ۲.۱ تحلیل کردیم و به دقت  $22.410\%$  رسیدیم که SGD بدتر است. اما سرعت آموزش بسیار بهتر از SGD می‌باشد و نوسانات تابع هزینه نیز کمتر است. در ادامه برای بررسی بهتر اندازه بچ سه مقدار  $64$  و  $500$  و  $16$  را بررسی می‌کنیم.

شکل زیر نمودار برای اندازه بج ۵۰۰ را نشان می‌دهد. در این حالت نوسان تابع کمتر بوده و سرعت آموزش نیز بالا است.



شکل ۱۵- نمودار دقت و هزینه با اندازه بج ۵۰۰

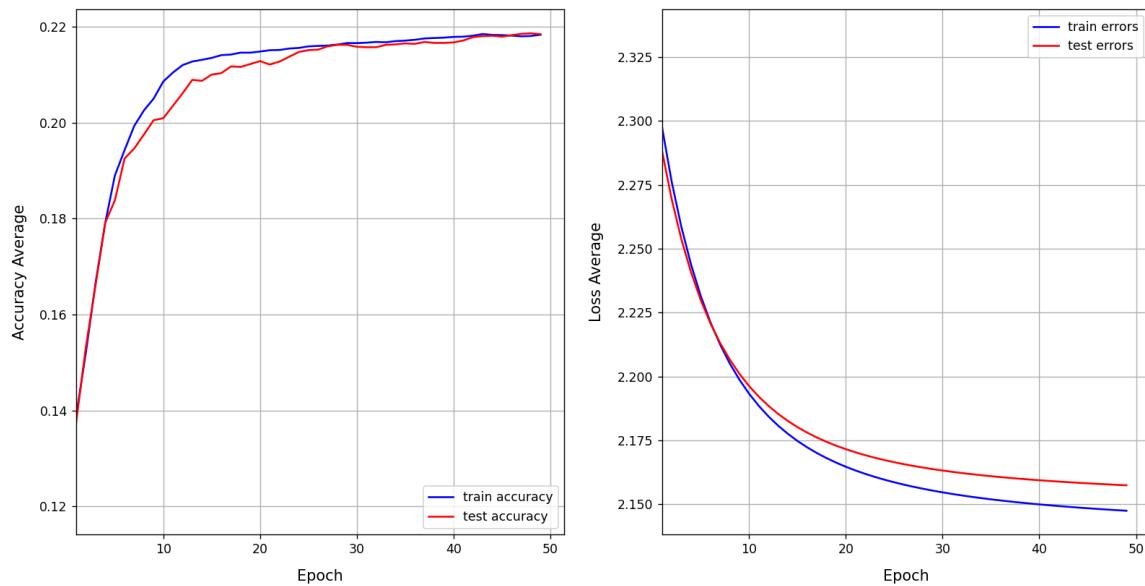
در این حالت به دقت ۱۹.۰۷ رسیدیم. ماتریس آشفتگی را در شکل زیر مشاهده می‌کنید.

Confusion Matrix											
True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
	airplane	286 (28.6%)	36 (3.6%)	38 (3.8%)	27 (2.7%)	8 (0.8%)	62 (6.2%)	46 (4.6%)	31 (3.1%)	298 (29.8%)	168 (16.8%)
	automobile	174 (17.4%)	109 (10.9%)	61 (6.1%)	15 (1.5%)	24 (2.4%)	104 (10.4%)	76 (7.6%)	48 (4.8%)	149 (14.9%)	240 (24.0%)
	bird	205 (20.5%)	29 (2.9%)	147 (14.7%)	37 (3.7%)	16 (1.6%)	112 (11.2%)	161 (16.1%)	66 (6.6%)	124 (12.4%)	103 (10.3%)
	cat	143 (14.3%)	60 (6.0%)	124 (12.4%)	65 (6.5%)	38 (3.8%)	177 (17.7%)	125 (12.5%)	54 (5.4%)	66 (6.6%)	148 (14.8%)
	deer	146 (14.6%)	42 (4.2%)	150 (15.0%)	43 (4.3%)	38 (3.8%)	132 (13.2%)	225 (22.5%)	72 (7.2%)	58 (5.8%)	94 (9.4%)
	dog	144 (14.4%)	49 (4.9%)	113 (11.3%)	51 (5.1%)	41 (4.1%)	247 (24.7%)	157 (15.7%)	62 (6.2%)	53 (5.3%)	83 (8.3%)
	frog	98 (9.8%)	49 (4.9%)	126 (12.6%)	59 (5.9%)	41 (4.1%)	89 (8.9%)	266 (26.6%)	77 (7.7%)	55 (5.5%)	140 (14.0%)
	horse	146 (14.6%)	54 (5.4%)	105 (10.5%)	41 (4.1%)	35 (3.5%)	163 (16.3%)	70 (7.0%)	66 (6.6%)	94 (9.4%)	226 (22.6%)
	ship	241 (24.1%)	63 (6.3%)	30 (3.0%)	17 (1.7%)	3 (0.3%)	61 (6.1%)	23 (2.3%)	16 (1.6%)	294 (29.4%)	252 (25.2%)
	truck	139 (13.9%)	95 (9.5%)	50 (5.0%)	26 (2.6%)	9 (0.9%)	58 (5.8%)	43 (4.3%)	42 (4.2%)	149 (14.9%)	389 (38.9%)

شکل ۱۶- ماتریس آشفتگی با اندازه بج ۵۰۰

شکل زیر نمودار در حالت اندازه بج ۶۴ را نشان می‌دهد. در این حالت سرعت آموزش از ۵۰۰ کمتر و نوسانات بیشتر است.

Accuracy and Loss plot for train and validation data



شکل ۱۷- نمودار با اندازه بج

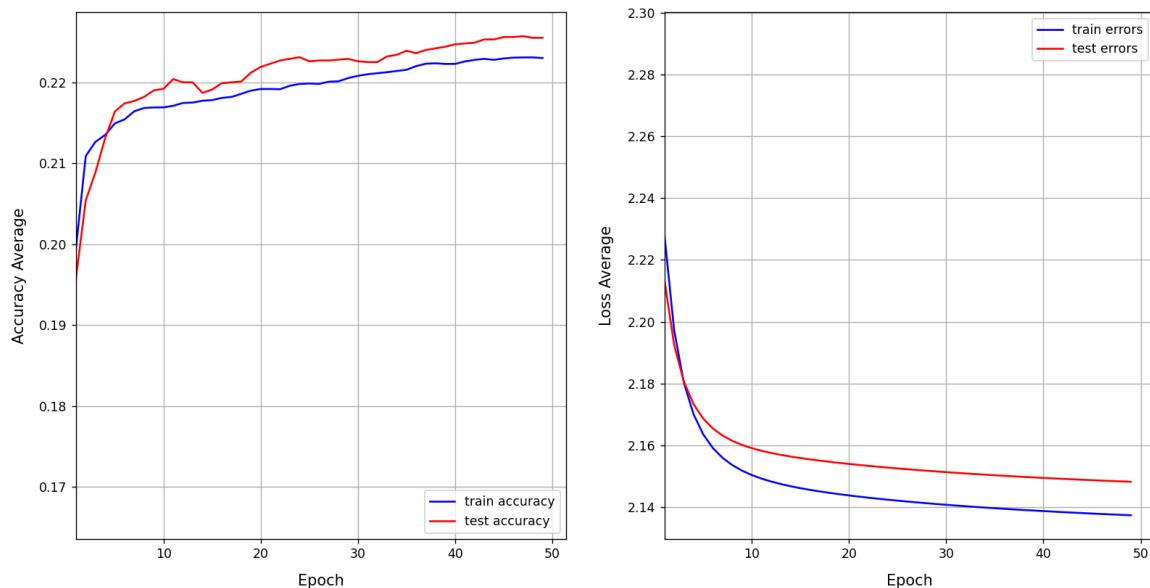
در این حالت به دقت ۲۲.۱۵۰ رسیدیم. شکل زیر ماتریس آشتفتگی در این حالت را نشان می‌دهد.

Confusion Matrix											
True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
	airplane	350 (35.0%)	48 (4.8%)	28 (2.8%)	21 (2.1%)	17 (1.7%)	59 (5.9%)	61 (6.1%)	86 (8.6%)	212 (21.2%)	118 (11.8%)
	automobile	144 (14.4%)	116 (11.6%)	34 (3.4%)	11 (1.1%)	23 (2.3%)	115 (11.5%)	103 (10.3%)	73 (7.3%)	162 (16.2%)	219 (21.9%)
	bird	158 (15.8%)	45 (4.5%)	105 (10.5%)	30 (3.0%)	48 (4.8%)	151 (15.1%)	198 (19.8%)	84 (8.4%)	106 (10.6%)	75 (7.5%)
	cat	150 (15.0%)	47 (4.7%)	49 (4.9%)	54 (5.4%)	43 (4.3%)	253 (25.3%)	157 (15.7%)	68 (6.8%)	89 (8.9%)	90 (9.0%)
	deer	98 (9.8%)	32 (3.2%)	69 (6.9%)	30 (3.0%)	83 (8.3%)	205 (20.5%)	260 (26.0%)	70 (7.0%)	85 (8.5%)	68 (6.8%)
	dog	105 (10.5%)	41 (4.1%)	57 (5.7%)	26 (2.6%)	70 (7.0%)	332 (33.2%)	147 (14.7%)	89 (8.9%)	73 (7.3%)	60 (6.0%)
	frog	81 (8.1%)	43 (4.3%)	59 (5.9%)	28 (2.8%)	60 (6.0%)	169 (16.9%)	320 (32.0%)	83 (8.3%)	63 (6.3%)	94 (9.4%)
	horse	120 (12.0%)	46 (4.6%)	37 (3.7%)	27 (2.7%)	49 (4.9%)	171 (17.1%)	138 (13.8%)	147 (14.7%)	95 (9.5%)	170 (17.0%)
	ship	194 (19.4%)	58 (5.8%)	19 (1.9%)	17 (1.7%)	9 (0.9%)	77 (7.7%)	32 (3.2%)	45 (4.5%)	349 (34.9%)	200 (20.0%)
	truck	109 (10.9%)	79 (7.9%)	16 (1.6%)	18 (1.8%)	18 (1.8%)	54 (5.4%)	91 (9.1%)	84 (8.4%)	172 (17.2%)	359 (35.9%)

شکل ۱۸- ماتریس آشتفتگی برای اندازه بج

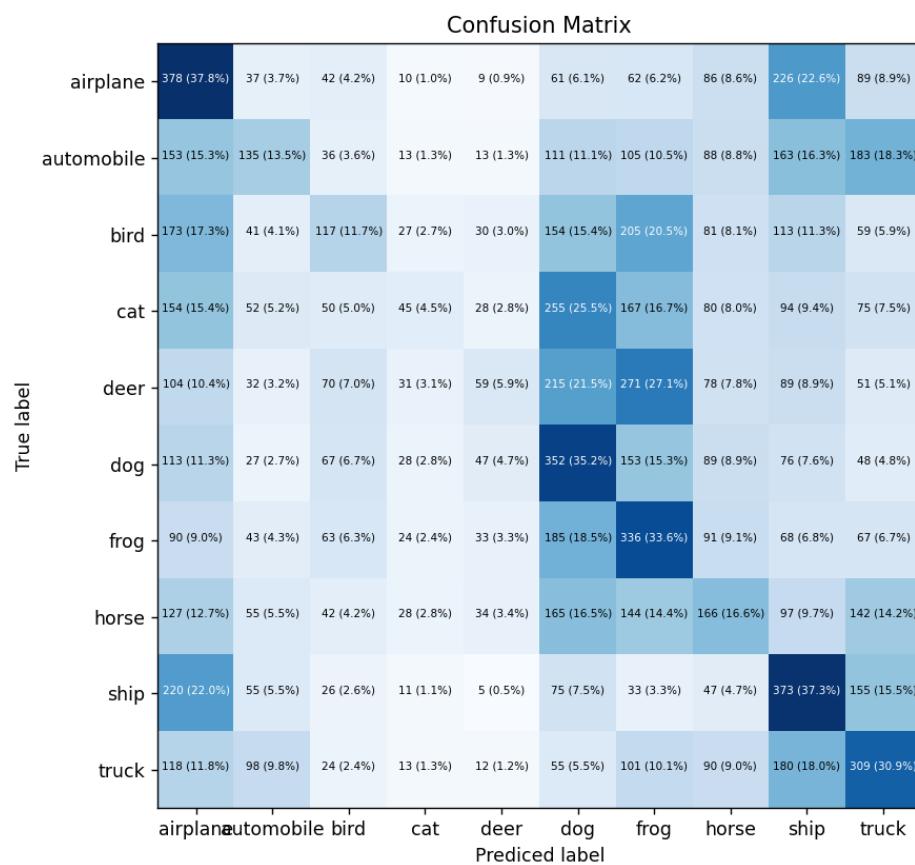
در آخر نیز اندازه بج ۱۶ را آزمایش می‌کنیم. در این حالت نمودار به شکل زیر است. سرعت آموختش در این حالت پایین بوده و نوسانات نیز بیشتر است.

Accuracy and Loss plot for train and validation data



شکل ۱۹- نمودار دقت و هزینه با اندازه بج ۱۶

دقت به دست آمده در این حالت ۲۲.۷۰۰ است. این دقت از اندازه بج ۶۴ بیشتر می‌باشد.



شکل ۲۰- ماتریس آشتفتگی با اندازه بج ۱۶

در مجموع با کاهش اندازه بج، سرعت یادگیری کاهش یافته و نوسانات در تابع هزینه و دقت نیز افزایش می‌باید. اما هر چه اندازه بج کمتر باشد، generalization بیشتر می‌شود. در جدول زیر نتایج ارزیابی هر مدل را مشاهده می‌کنید.

جدول ۶- مقایسه نتایج اندازه بج روی داده تست

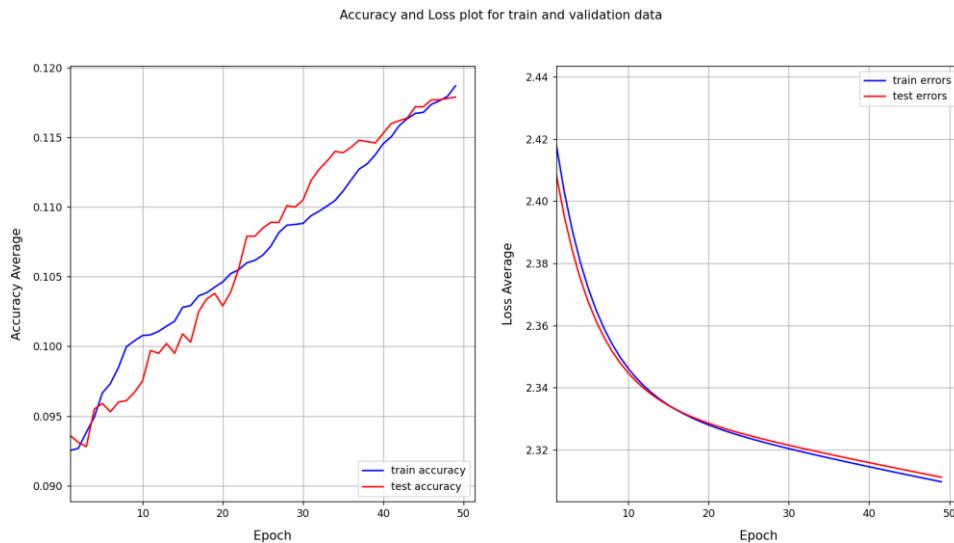
Batch size	۱۶	۶۴	۵۰۰
Accuracy	۲۲.۷	۲۲.۱۵	۱۹.۰۷
precision	۲۲.۷	۲۲.۱۵	۱۹.۰۷
recall	۲۲.۷	۲۲.۱۵	۱۹.۰۷
F1 score	۲۲.۷	۲۲.۱۵	۱۹.۰۷
Loss	۲.۱۳۶	۲.۱۴۸	۲.۲۲۳

با توجه به نتایج جدول بالا می‌توان دریافت که با کاهش اندازه بج، به دقت بهتری می‌رسیم. البته این مسئله عمومیت ندارد و نرخ یادگیری نیز تاثیرگذار است. اما با نرخ یادگیری ۱،۰۰۰،۰۰۰.۱، اندازه بج کمتر به نتیجه بهتری منتهی می‌شود. اما با توجه به اینکه با کاهش اندازه بج سرعت یادگیری کاهش می‌یابد، یک trade off بین این مقادیر وجود دارد.

## سوال (۱.۷)

یکی از پارامترهای مهم در آموزش مدل نرخ یادگیری است. این پارامتر را برابر ۱،۰۰.۱، ۰.۱ و ۱۰۰ مقدار اولیه قرار دهید و خطأ و دقت را بررسی کنید.

مقدار اولیه نرخ یادگیری ۰.۰۰۱ بود که با آن به دقت ۰.۰۱ رسیدیم. حال نرخ یادگیری را ۰.۰۱ مقدار اولیه یعنی  $10^{-5}$  قرار می‌دهیم. همانطور که در نمودار زیر مشاهده می‌کنید، با این نرخ یادگیری به دقت بالایی نمی‌رسیم. زیرا وزن‌ها با مقادیر بسیار کوچک به روزرسانی شده و به نقطه optimal نزدیک می‌شوند. بنابراین اگر بخواهیم از این نرخ یادگیری استفاده کنیم و به دقت بالایی بررسیم، باید تعداد ایپاک‌ها را نیز خیلی بیشتر کنیم.



شکل ۲۱- نمودار دقت و زیان با نرخ یادگیری ۰.۰۰۰۰۱

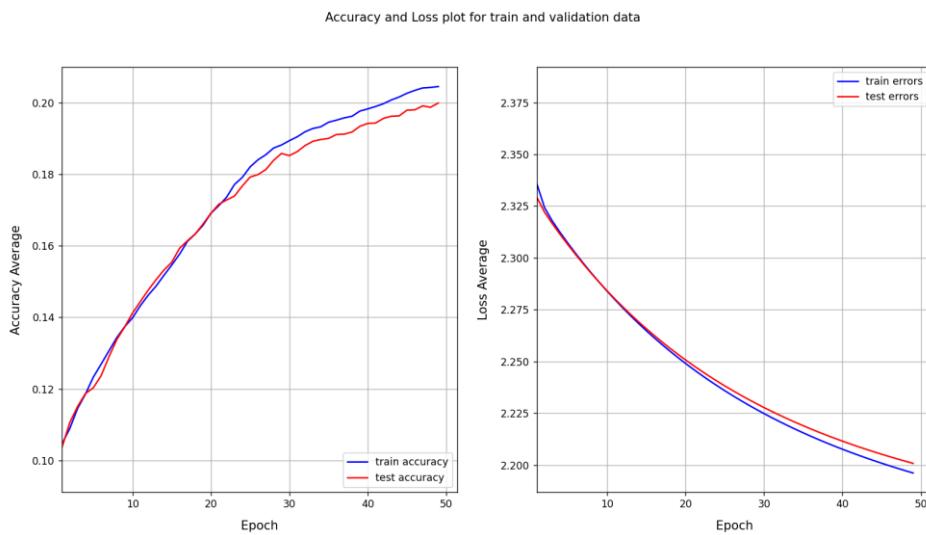
روی داده‌های تست به دقت ۱۱.۳۱۰ رسیدیم که نسبت به نتایجی که تاکنون گرفتیم، کمتر است. شکل زیر ماتریس آشتفتگی را نشان می‌دهد. همانطور که می‌بینید، مدل هنوز به خوبی آموزش ندیده است و عمدۀ تصاویر را در کلاس هوایپیما برچسب گذاری کرده است.

Confusion Matrix											
True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
	airplane	192 (19.2%)	68 (6.8%)	46 (4.6%)	44 (4.4%)	13 (1.3%)	77 (7.7%)	21 (2.1%)	54 (5.4%)	256 (25.6%)	229 (22.9%)
	automobile	229 (22.9%)	81 (8.1%)	75 (7.5%)	71 (7.1%)	18 (1.8%)	58 (5.8%)	33 (3.3%)	107 (10.7%)	197 (19.7%)	131 (13.1%)
	bird	278 (27.8%)	39 (3.9%)	122 (12.2%)	41 (4.1%)	20 (2.0%)	61 (6.1%)	52 (5.2%)	160 (16.0%)	113 (11.3%)	114 (11.4%)
	cat	269 (26.9%)	62 (6.2%)	125 (12.5%)	52 (5.2%)	25 (2.5%)	71 (7.1%)	37 (3.7%)	139 (13.9%)	84 (8.4%)	136 (13.6%)
	deer	266 (26.6%)	43 (4.3%)	134 (13.4%)	35 (3.5%)	15 (1.5%)	44 (4.4%)	72 (7.2%)	211 (21.1%)	69 (6.9%)	111 (11.1%)
	dog	298 (29.8%)	52 (5.2%)	99 (9.9%)	31 (3.1%)	41 (4.1%)	46 (4.6%)	47 (4.7%)	214 (21.4%)	66 (6.6%)	106 (10.6%)
	frog	294 (29.4%)	32 (3.2%)	147 (14.7%)	62 (6.2%)	10 (1.0%)	25 (2.5%)	69 (6.9%)	190 (19.0%)	70 (7.0%)	101 (10.1%)
	horse	227 (22.7%)	56 (5.6%)	102 (10.2%)	71 (7.1%)	34 (3.4%)	88 (8.8%)	24 (2.4%)	130 (13.0%)	105 (10.5%)	163 (16.3%)
	ship	183 (18.3%)	113 (11.3%)	28 (2.8%)	55 (5.5%)	15 (1.5%)	50 (5.0%)	21 (2.1%)	43 (4.3%)	258 (25.8%)	234 (23.4%)
	truck	182 (18.2%)	72 (7.2%)	85 (8.5%)	142 (14.2%)	13 (1.3%)	50 (5.0%)	21 (2.1%)	59 (5.9%)	210 (21.0%)	166 (16.6%)

شکل ۲۲- ماتریس آشتفتگی برای نرخ یادگیری ۰.۰۰۰۰۱

در گام بعدی، نرخ یادگیری ۰.۰۰۰۰۱ را آزمایش می‌کنیم. این نرخ یادگیری از نرخ یادگیری مرحله قبل بیشتر است، بنابراین توقع داریم که فرآیند آموزش با سرعت بیشتری انجام شود و در ۵۰ ایپاک به دقت

بهتری نسبت به حالت قبل برسیم. شکل زیر نمودار دقت و هزینه را در این حالت نشان می‌دهد.



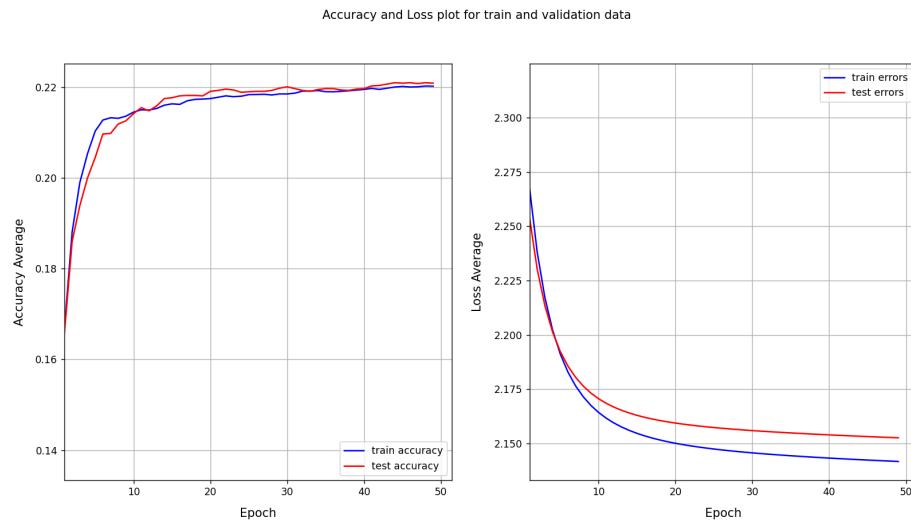
شکل ۲۳- نمودار دقت و هزینه نرخ یادگیری ۰۰۰۱

همانطور که انتظار داشتیم، به دقت بالاتری در داده‌های اعتبارسنجی نسبت به حالت قبل رسیدیم. زیرا در هر بار به روزرسانی وزن‌ها، مقدار جهش بیشتر بوده و حرکت به سمت نقطه optimal بیشتر صورت گرفته است. ماتریس آشفتگی را در شکل زیر می‌بینید. نسبت به حالت قبلی، پیشرفت مشهودی مشاهده می‌کنیم. بهترین کلاس برای کامیون است اما از کلاس‌های دیگر نیز تا حد قابل قبولی پیش‌بینی درست صورت گرفته است.

Confusion Matrix											
True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
	airplane	315 (31.5%)	45 (4.5%)	31 (3.1%)	26 (2.6%)	10 (1.0%)	55 (5.5%)	59 (5.9%)	33 (3.3%)	265 (26.5%)	161 (16.1%)
	automobile	161 (16.1%)	116 (11.6%)	53 (5.3%)	17 (1.7%)	33 (3.3%)	117 (11.7%)	86 (8.6%)	27 (2.7%)	142 (14.2%)	248 (24.8%)
	bird	173 (17.3%)	42 (4.2%)	132 (13.2%)	38 (3.8%)	27 (2.7%)	142 (14.2%)	187 (18.7%)	35 (3.5%)	108 (10.8%)	116 (11.6%)
	cat	141 (14.1%)	59 (5.9%)	89 (8.9%)	67 (6.7%)	44 (4.4%)	226 (22.6%)	144 (14.4%)	21 (2.1%)	72 (7.2%)	137 (13.7%)
	deer	113 (11.3%)	38 (3.8%)	117 (11.7%)	41 (4.1%)	56 (5.6%)	179 (17.9%)	256 (25.6%)	35 (3.5%)	63 (6.3%)	102 (10.2%)
	dog	117 (11.7%)	48 (4.8%)	86 (8.6%)	50 (5.0%)	55 (5.5%)	295 (29.5%)	178 (17.8%)	37 (3.7%)	56 (5.6%)	78 (7.8%)
	frog	86 (8.6%)	48 (4.8%)	92 (9.2%)	54 (5.4%)	56 (5.6%)	125 (12.5%)	308 (30.8%)	37 (3.7%)	55 (5.5%)	139 (13.9%)
	horse	130 (13.0%)	55 (5.5%)	74 (7.4%)	43 (4.3%)	54 (5.4%)	192 (19.2%)	97 (9.7%)	41 (4.1%)	90 (9.0%)	224 (22.4%)
	ship	212 (21.2%)	57 (5.7%)	20 (2.0%)	28 (2.8%)	7 (0.7%)	62 (6.2%)	27 (2.7%)	19 (1.9%)	311 (31.1%)	257 (25.7%)
	truck	117 (11.7%)	88 (8.8%)	38 (3.8%)	27 (2.7%)	16 (1.6%)	60 (6.0%)	63 (6.3%)	30 (3.0%)	143 (14.3%)	418 (41.8%)

شکل ۲۴- ماتریس آشفتگی با نرخ یادگیری ۰۰۰۱

سپس از نرخ یادگیری ۰.۰۰۱ استفاده می‌کنیم که همان نتایج قسمت ۱.۲ به دست می‌آید. شکل زیر نمودار دقت و هزینه را نشان می‌دهد. دقت به دست آمده در داده اعتبار سنجی از دو حالت قبلی بهتر است.



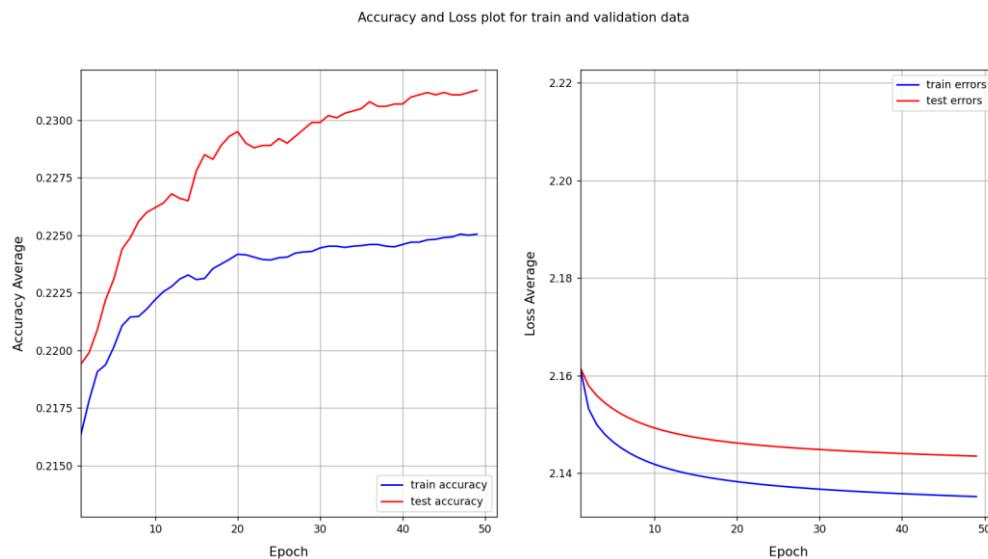
شکل ۲۵- نمودار دقت و هزینه با نرخ یادگیری ۰.۰۰۱

دقت به دست آمده در این قسمت بر روی داده‌های تست، ۲۲.۴۱۰ است. این دقت از دقت به دست آمده در دو قسمت قبل، بهتر است که نشان می‌دهد افزایش نرخ یادگیری در ۵۰ ایپاک اول به بهبود مدل کمک می‌کند. ماتریس آشفتگی نیز نشان می‌دهد که تعداد پیش‌بینی‌های درست برای هر کلاس افزایش داشته است.

Confusion Matrix											
True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
	airplane	359 (35.9%)	44 (4.4%)	38 (3.8%)	17 (1.7%)	14 (1.4%)	60 (6.0%)	59 (5.9%)	89 (8.9%)	217 (21.7%)	103 (10.3%)
	automobile	146 (14.6%)	130 (13.0%)	37 (3.7%)	13 (1.3%)	20 (2.0%)	109 (10.9%)	101 (10.1%)	86 (8.6%)	166 (16.6%)	192 (19.2%)
	bird	166 (16.6%)	40 (4.0%)	102 (10.2%)	28 (2.8%)	45 (4.5%)	149 (14.9%)	200 (20.0%)	89 (8.9%)	112 (11.2%)	69 (6.9%)
	cat	150 (15.0%)	51 (5.1%)	51 (5.1%)	49 (4.9%)	44 (4.4%)	250 (25.0%)	156 (15.6%)	78 (7.8%)	88 (8.8%)	83 (8.3%)
	deer	97 (9.7%)	32 (3.2%)	69 (6.9%)	30 (3.0%)	84 (8.4%)	205 (20.5%)	261 (26.1%)	77 (7.7%)	87 (8.7%)	58 (5.8%)
	dog	99 (9.9%)	36 (3.6%)	56 (5.6%)	29 (2.9%)	72 (7.2%)	341 (34.1%)	144 (14.4%)	97 (9.7%)	74 (7.4%)	52 (5.2%)
	frog	82 (8.2%)	43 (4.3%)	59 (5.9%)	27 (2.7%)	63 (6.3%)	172 (17.2%)	320 (32.0%)	94 (9.4%)	67 (6.7%)	73 (7.3%)
	horse	121 (12.1%)	48 (4.8%)	40 (4.0%)	27 (2.7%)	48 (4.8%)	163 (16.3%)	140 (14.0%)	163 (16.3%)	93 (9.3%)	157 (15.7%)
	ship	203 (20.3%)	54 (5.4%)	23 (2.3%)	11 (1.1%)	8 (0.8%)	76 (7.6%)	33 (3.3%)	48 (4.8%)	368 (36.8%)	176 (17.6%)
	truck	106 (10.6%)	96 (9.6%)	22 (2.2%)	16 (1.6%)	14 (1.4%)	54 (5.4%)	96 (9.6%)	93 (9.3%)	178 (17.8%)	325 (32.5%)

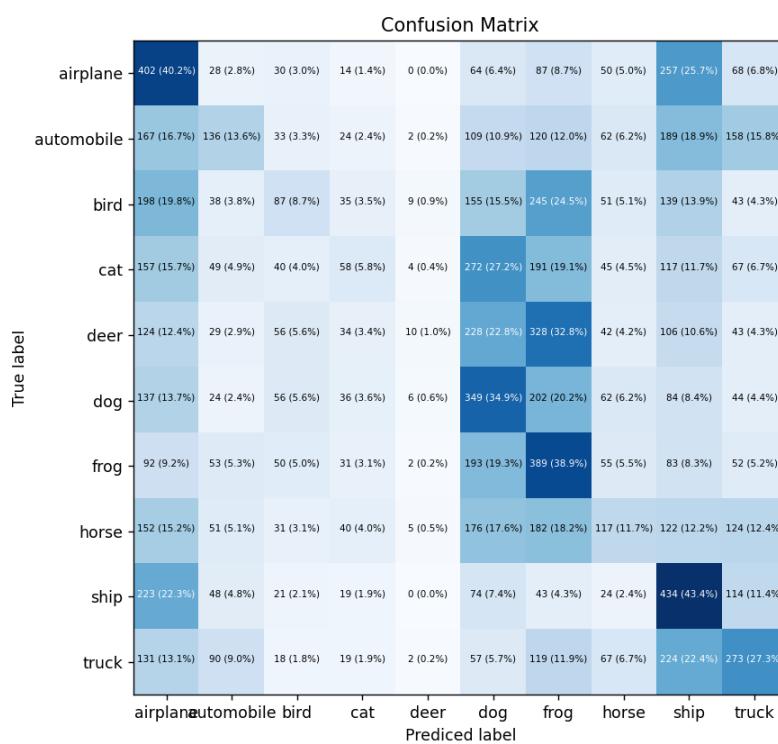
شکل ۲۶- ماتریس آشفتگی برای نرخ یادگیری ۰.۰۰۱

حال نرخ یادگیری را مجدد کاهش می‌دهیم و آن را به ۰.۰۱ می‌رسانیم. نمودار دقت و زیان در شکل زیر آمده است. تابع هزینه رو به کاهش و دقت رو به افزایش است. اما فاصله مقادیر در داده آموزش و اعتبارسنجی نسبت به حالت‌های قبل بیشتر شده است.



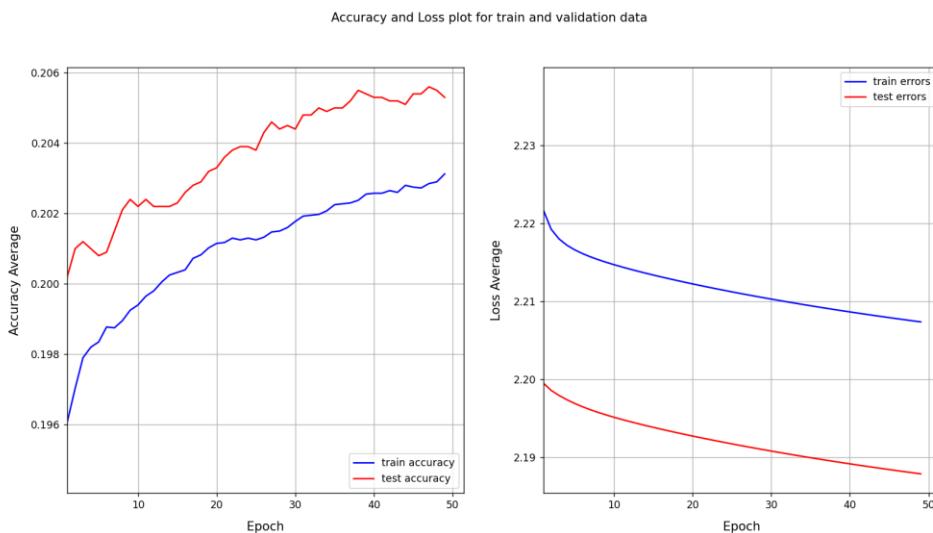
شکل ۲۷- نمودار دقت و هزینه برای نرخ یادگیری ۰.۰۱

در این حالت به دقت ۲۲.۵۵٪ بر روی مجموعه داده تست رسیدیم که کمی از حالت‌های قبل بیشتر است.



شکل ۲۸- ماتریس آشفتگی برای نرخ یادگیری ۰.۰۱

در نهایت نیز نرخ یادگیری را تا مقدار ۰.۱ کاهش می‌دهیم و مسئله را بررسی می‌کنیم. با نگاه کردن به نتیجه فرآیند آموزش، می‌بینیم که در ابتدا با آموزش به خوبی انجام شده و نمودار هزینه کاهش یافته است. اما به دلیل بالا بودن نرخ یادگیری، اگر تعداد ایپاک‌ها را افزایش دهیم، فرآیند آموزش دیگر خوب نخواهد بود. زیرا با نزدیک شدن به نقطه optimal نیاز داریم که جهش‌های کوچک داشته باشیم در صورتی که این نرخ یادگیری این فرآیند را برای ما انجام نمی‌دهد.



شکل ۲۹- نمودار دقت و هزینه برای نرخ یادگیری ۰.۱

دقت به دست آمده بر روی مجموعه دادگان تست ۰.۲۹۰ است که از حالت قبل کمتر است. در این حالت نیز آموزش کامل نشده است. با بررسی ماتریس آشفتگی نیز می‌توان این مسئله را دریافت. زیرا، عمدۀ تصاویر در کلاس قورباغه طبقه‌بندی شده‌اند.

Confusion Matrix											
True label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	
	airplane	84 (8.4%)	136 (13.6%)	12 (1.2%)	24 (2.4%)	0 (0.0%)	29 (2.9%)	330 (33.0%)	6 (0.6%)	359 (35.9%)	20 (2.0%)
	automobile	25 (2.5%)	285 (28.5%)	12 (1.2%)	33 (3.3%)	0 (0.0%)	66 (6.6%)	311 (31.1%)	16 (1.6%)	189 (18.9%)	63 (6.3%)
	bird	23 (2.3%)	87 (8.7%)	32 (3.2%)	58 (5.8%)	0 (0.0%)	50 (5.0%)	549 (54.9%)	16 (1.6%)	164 (16.4%)	21 (2.1%)
	cat	22 (2.2%)	102 (10.2%)	21 (2.1%)	75 (7.5%)	0 (0.0%)	105 (10.5%)	511 (51.1%)	20 (2.0%)	115 (11.5%)	29 (2.9%)
	deer	15 (1.5%)	64 (6.4%)	18 (1.8%)	37 (3.7%)	0 (0.0%)	56 (5.6%)	692 (69.2%)	10 (1.0%)	99 (9.9%)	9 (0.9%)
	dog	22 (2.2%)	77 (7.7%)	25 (2.5%)	52 (5.2%)	0 (0.0%)	179 (17.9%)	524 (52.4%)	20 (2.0%)	91 (9.1%)	10 (1.0%)
	frog	9 (0.9%)	93 (9.3%)	8 (0.8%)	31 (3.1%)	0 (0.0%)	42 (4.2%)	719 (71.9%)	11 (1.1%)	70 (7.0%)	17 (1.7%)
	horse	38 (3.8%)	114 (11.4%)	23 (2.3%)	43 (4.3%)	0 (0.0%)	94 (9.4%)	424 (42.4%)	56 (5.6%)	152 (15.2%)	56 (5.6%)
	ship	24 (2.4%)	202 (20.2%)	8 (0.8%)	35 (3.5%)	0 (0.0%)	46 (4.6%)	187 (18.7%)	7 (0.7%)	454 (45.4%)	27 (2.7%)
	truck	22 (2.2%)	223 (22.3%)	11 (1.1%)	29 (2.9%)	0 (0.0%)	22 (2.2%)	288 (28.8%)	22 (2.2%)	248 (24.8%)	135 (13.5%)

شکل ۳۰- ماتریس آشفتگی برای نرخ یادگیری ۰.۱

به طور کلی با مشاهداتی که تاکنون داشتیم و به کمک جدول زیر، می‌توان گفت نرخ یادگیری سرعت انطباق مدل با مسئله را کنترل می‌کند. نرخ یادگیری کوچک‌تر به تعداد ایپاک بیشتری نیاز دارد زیرا، در هر گام تغییرات کوچک‌تری در به روزرسانی وزن‌ها ایجاد می‌شود. در حالی که نرخ‌های یادگیری بزرگ‌تر منجر به تغییرات سریع می‌شوند و به تعداد ایپاک‌های کمتری نیاز دارند. علاوه بر این، نرخ یادگیری خیلی زیاد می‌تواند باعث شود که مدل خیلی سریع به یک راه حل غیربهینه همگرا شود، در حالی که نرخ یادگیری بسیار کوچک می‌تواند باعث گیر کردن فرآیند شود. با این تعاریف، به نظر می‌رسد بهترین نرخ یادگیری با ثابت بودن سایر پارامترها ۰.۰۱ است.

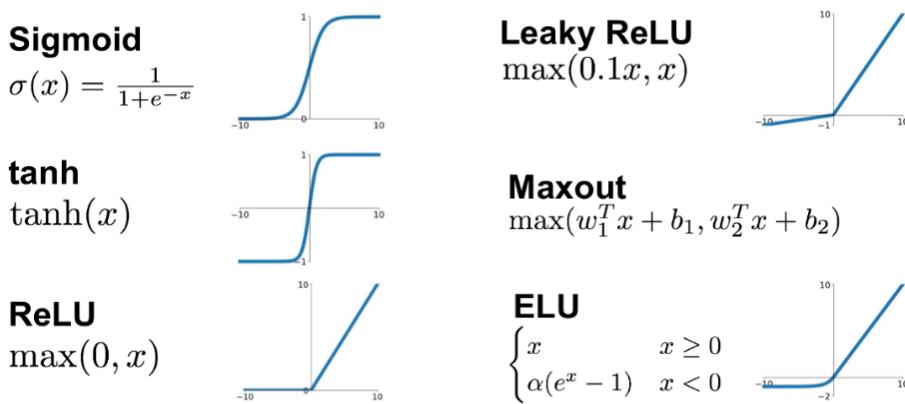
جدول ۷- مقایسه نتایج ارزیابی روی داده تست با نرخ یادگیری متفاوت

Learning rate	1e-5	1e-4	1e-3	0.01	0.1
Accuracy	۱۱.۳۱۰	۲۰.۵۹۰	۲۲.۴۱۰	۲۲.۵۵۰	۲۰.۲۹۰
precision	۱۱.۳۱۰	۲۰.۵۹۰	۲۲.۴۱۰	۲۲.۵۵۰	۲۰.۲۹۰
recall	۱۱.۳۱۰	۲۰.۵۹۰	۲۲.۴۱۰	۲۲.۵۵۰	۲۰.۲۹۰
F1 score	۱۱.۳۱۰	۲۰.۵۹۰	۲۲.۴۱۰	۲۲.۵۵۰	۲۰.۲۹۰
Loss	۲.۲۳۱	۲.۲۱۷	۲.۱۴۲	۲.۱۳۴	۲.۲۱۰

## سوال (۱.۸)

تابع فعال ساز ذکر شده در جدول ۱ را با یکدیگر از نظر تئوری مقایسه کنید. همچنین پس از پیاده سازی، نتایج آن را مقایسه کنید.

در شکل زیر انواع تابع فعال ساز به همراه نمودار آنها را مشاهده می کنید.



شکل ۳۱- تابع فعال ساز

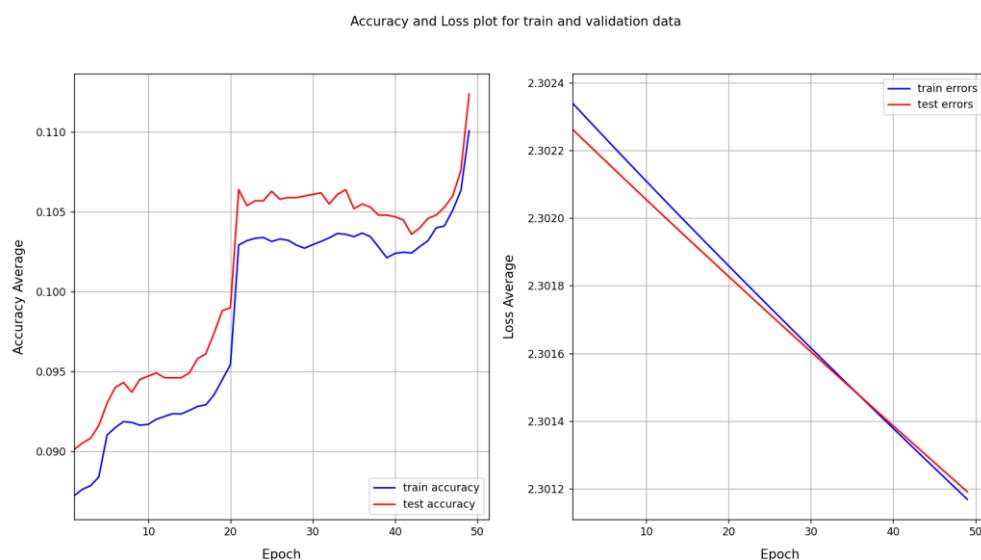
دلیل اصلی استفاده از تابع سیگموئید این است که بین صفر و یک است. بنابراین، برای مدل هایی که باید احتمال را به عنوان خروجی پیش بینی کنیم، استفاده می شود. یکی از ایرادات این تابع این است که مشتق تابع نیست اما یکنواخت است. همچنین، تابع سیگموئید منجر به اشباع شدن می شود و گرادیان را از بین می برد. یکی دیگر از مشکلات سیگموئید این است که zero-centered است. به همین خاطر تمام خروجی ها مثبت می شود و گرادیان ها یا همیشه مثبت هستند یا منفی. این مسئله رسیدن به نقطه بهینه را سخت می کند. در آخر نیز به دلیل محاسبه  $\exp$  حجم محاسباتی سنگینی داشته و کند است.

تابع  $\tanh$  نیز مانند سیگموئید است اما شرایط بهتری دارد. محدوده تابع  $\tanh$  از  $-1$  تا  $1$  است. بنابراین مشکل zero-centered را ندارد. اما دو مشکل دیگر همچنان وجود دارد. این تابع یکنواخت است اما مشتق آن یکنواخت نیست. تابع  $\tanh$  عمدها برای طبقه بندی بین دو کلاس استفاده می شود.

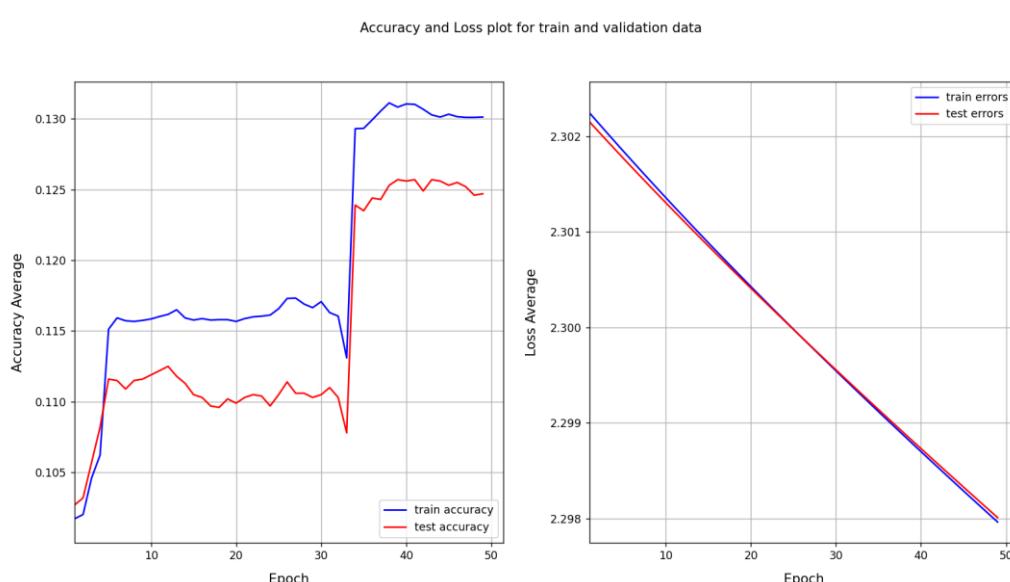
ReLU در حال حاضر پر استفاده ترین تابع فعال ساز است. همانطور که در شکل بالا می بینید، وقتی  $Z$  کمتر از صفر باشد  $f(z) = 0$  است و زمانی که  $Z$  بیشتر یا برابر با صفر باشد  $f(z) = z$  است. این تابع و مشتق آن هر دو یکنواخت هستند. اما مسئله این است که تمام مقادیر منفی بلا فاصله صفر می شوند که توانایی مدل را برای برازش یا آموزش صحیح از داده ها کاهش می دهد. این بدان معناست که هر ورودی منفی که به تابع فعال سازی  $ReLU$  داده می شود، مقدار را بلا فاصله در نمودار به صفر تبدیل می کند، که به نوبه خود با نگاشت مناسب مقادیر منفی، بر نمودار حاصل تأثیر می گذارد.  $ReLU$  هیچ کدام از مشکلات سیگموئید را ندارد.

محاسبات موجود در تابع ReLU، ساده و صرفاً یک مقایسه است. به همین خاطر محاسبات در بخش تابع غیرخطی با استفاده از ReLU نسبت به سایر توابع غیرخطی مانند tanh و سیگموئید با سرعت بیشتری انجام می‌شود. از طرفی فرآیند آموزش با این تابع نسبت به سایر توابع غیرخطی سریع‌تر است. توابع غیرخطی tanh و سیگموئید در مقادیر خیلی بزرگ یا کوچک به اشباع می‌رسند و این باعث می‌شود که گرادیان این توابع به سمت صفر میل کند. درنتیجه کل فرآیند آموزش با سرعت پایین‌تری نسبت به ReLU انجام می‌شود.

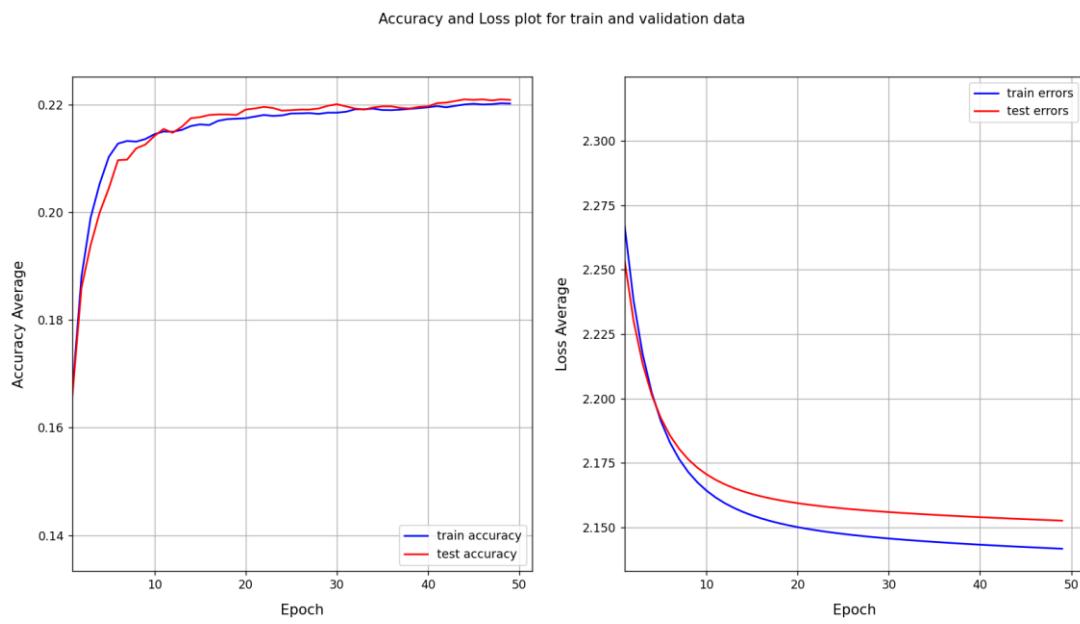
تابع Leaky ReLU به افزایش دامنه عملکرد ReLU کمک می‌کند. ضریب این تابع عموماً ۰.۰۱ است. بنابراین محدوده Leaky ReLU پی‌نهایت تا بی‌نهایت است. این تابع ماهیت یکنواخت دارد. همچنین، مشتق آن نیز ماهیت یکنواخت دارد. در ادامه این  $\text{ReLU}^+$  تابع را به صورت عملی آزمایش کرده و نتایج را مقایسه می‌کنیم.



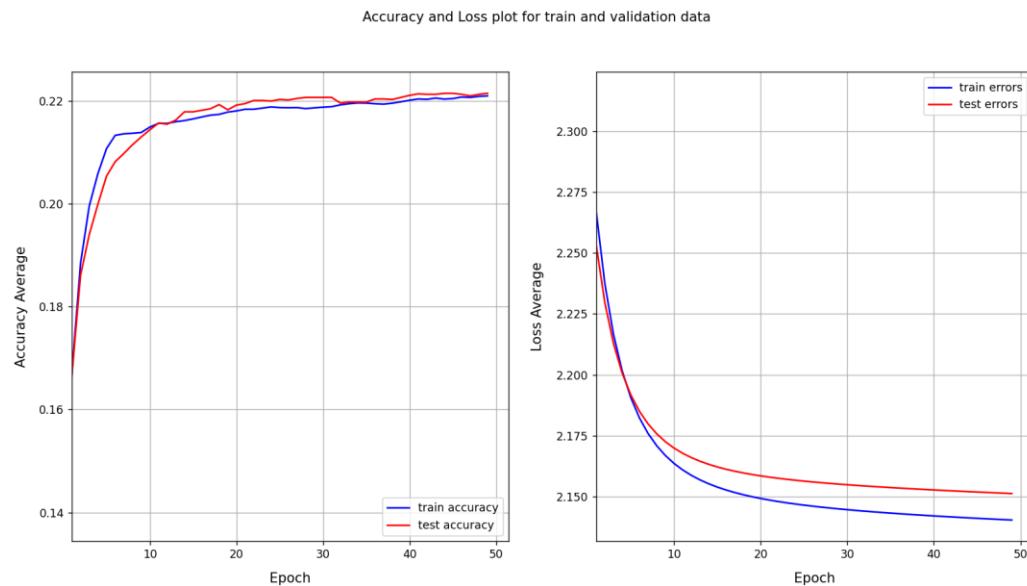
شکل ۳۲- نمودار دقت و هزینه با تابع سیگموئید



شکل ۳۳- نمودار دقت و هزینه با تابع Tanh



شکل ۳۴- نمودار دقت و هزینه با تابع ReLu



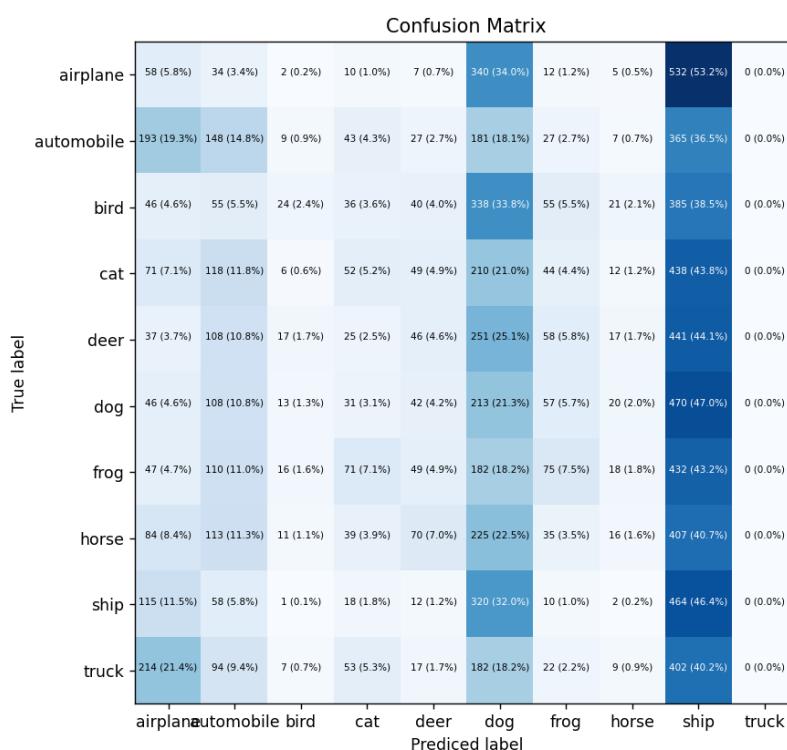
شکل ۳۵- نمودار دقت و هزینه با تابع LeakyReLU

همانطور که در نتایج بالا نیز مشخص است، توابع سیگوئید و Tanh نتایج ضعیفی داشته‌اند. اما ReLu و LeakyReLU عملکرد بهتری داشتند. علاوه بر نتایج به دست آمده، سرعت پردازش در ReLu و LeakyReLU بسیار بیشتر از دو تابع دیگر است. عملکرد LeakyReLU و ReLu تقریباً مشابه بوده و به دقت یکسانی نیز رسیدند. جدول زیر نتایج ارزیابی روی داده تست را نشان می‌دهد.

جدول ۸- مقایسه ارزیابی دادگان تست با توابع فعال ساز متفاوت

Activation Function	Sigmoid	Tanh	ReLU	LeakyRelu
Accuracy	۱۰.۹۶۰	۱۲.۷۹۰	۲۲.۴۱۰	۲۲.۴
precision	۱۰.۹۶۰	۱۲.۷۹۰	۲۲.۴۱۰	۲۲.۴
recall	۱۰.۹۶۰	۱۲.۷۹۰	۲۲.۴۱۰	۲۲.۴
F1 score	۱۰.۹۶۰	۱۲.۷۹۰	۲۲.۴۱۰	۲۲.۴
Loss	۲.۳۱۲	۲.۲۹۹	۲.۱۴۲	۲.۱۵۸

همانطور که در جدول بالا مشخص است، عملکرد با توابع ReLu و LeakyReLu بهتر است و به دقت بالاتری رسیدیم. اما در این مسئله و با این تعداد ایپاک تفاوت آنچنانی میان ReLu و LeakyReLu وجود ندارد. ماتریس‌های آشتفتگی نیز برای هر ۴ تابع فعال ساز رسم شده‌است.



شکل ۳۶- ماتریس آشتفتگی برای تابع Sigmoid

Confusion Matrix										
True label	airplane	0 (0.0%)	6 (0.6%)	9 (0.9%)	324 (32.4%)	0 (0.0%)	14 (1.4%)	2 (0.2%)	98 (9.8%)	8 (0.8%)
	automobile	452 (45.2%)	19 (1.9%)	24 (2.4%)	40 (4.0%)	249 (24.9%)	0 (0.0%)	30 (3.0%)	5 (0.5%)	157 (15.7%)
	bird	451 (45.1%)	0 (0.0%)	42 (4.2%)	28 (2.8%)	296 (29.6%)	0 (0.0%)	66 (6.6%)	8 (0.8%)	99 (9.9%)
	cat	326 (32.6%)	5 (0.5%)	22 (2.2%)	58 (5.8%)	388 (38.8%)	0 (0.0%)	52 (5.2%)	2 (0.2%)	140 (14.0%)
	deer	365 (36.5%)	7 (0.7%)	46 (4.6%)	28 (2.8%)	398 (39.8%)	0 (0.0%)	59 (5.9%)	5 (0.5%)	86 (8.6%)
	dog	300 (30.0%)	3 (0.3%)	26 (2.6%)	27 (2.7%)	445 (44.5%)	0 (0.0%)	73 (7.3%)	4 (0.4%)	114 (11.4%)
	frog	314 (31.4%)	2 (0.2%)	31 (3.1%)	55 (5.5%)	407 (40.7%)	0 (0.0%)	85 (8.5%)	5 (0.5%)	92 (9.2%)
	horse	376 (37.6%)	8 (0.8%)	31 (3.1%)	33 (3.3%)	350 (35.0%)	0 (0.0%)	36 (3.6%)	4 (0.4%)	130 (13.0%)
	ship	604 (60.4%)	6 (0.6%)	3 (0.3%)	13 (1.3%)	261 (26.1%)	0 (0.0%)	14 (1.4%)	0 (0.0%)	95 (9.5%)
	truck	509 (50.9%)	10 (1.0%)	12 (1.2%)	54 (5.4%)	210 (21.0%)	0 (0.0%)	24 (2.4%)	3 (0.3%)	139 (13.9%)
Predicted label										

شکل ۳۷- ماتریس آشفتگی برای تابع Tanh

Confusion Matrix										
True label	airplane	44 (4.4%)	38 (3.8%)	17 (1.7%)	14 (1.4%)	60 (6.0%)	59 (5.9%)	89 (8.9%)	217 (21.7%)	103 (10.3%)
	automobile	146 (14.6%)	130 (13.0%)	37 (3.7%)	13 (1.3%)	20 (2.0%)	109 (10.9%)	101 (10.1%)	86 (8.6%)	166 (16.6%)
	bird	166 (16.6%)	40 (4.0%)	102 (10.2%)	28 (2.8%)	45 (4.5%)	149 (14.9%)	200 (20.0%)	89 (8.9%)	112 (11.2%)
	cat	150 (15.0%)	51 (5.1%)	51 (5.1%)	49 (4.9%)	44 (4.4%)	250 (25.0%)	156 (15.6%)	78 (7.8%)	88 (8.8%)
	deer	97 (9.7%)	32 (3.2%)	69 (6.9%)	30 (3.0%)	84 (8.4%)	205 (20.5%)	261 (26.1%)	77 (7.7%)	87 (8.7%)
	dog	99 (9.9%)	36 (3.6%)	56 (5.6%)	29 (2.9%)	72 (7.2%)	341 (34.1%)	144 (14.4%)	97 (9.7%)	74 (7.4%)
	frog	82 (8.2%)	43 (4.3%)	59 (5.9%)	27 (2.7%)	63 (6.3%)	172 (17.2%)	320 (32.0%)	94 (9.4%)	67 (6.7%)
	horse	121 (12.1%)	48 (4.8%)	40 (4.0%)	27 (2.7%)	48 (4.8%)	163 (16.3%)	140 (14.0%)	163 (16.3%)	93 (9.3%)
	ship	203 (20.3%)	54 (5.4%)	23 (2.3%)	11 (1.1%)	8 (0.8%)	76 (7.6%)	33 (3.3%)	48 (4.8%)	368 (36.8%)
	truck	106 (10.6%)	96 (9.6%)	22 (2.2%)	16 (1.6%)	14 (1.4%)	54 (5.4%)	96 (9.6%)	93 (9.3%)	178 (17.8%)
Predicted label										

شکل ۳۸- ماتریس آشفتگی برای تابع ReLu

		Confusion Matrix									
		Predicted label									
True label	airplane	358 (35.8%)	44 (4.4%)	39 (3.9%)	16 (1.6%)	14 (1.4%)	60 (6.0%)	59 (5.9%)	89 (8.9%)	218 (21.8%)	103 (10.3%)
	automobile	145 (14.5%)	131 (13.1%)	38 (3.8%)	12 (1.2%)	22 (2.2%)	109 (10.9%)	101 (10.1%)	84 (8.4%)	166 (16.6%)	192 (19.2%)
	bird	166 (16.6%)	38 (3.8%)	103 (10.3%)	28 (2.8%)	48 (4.8%)	147 (14.7%)	199 (19.9%)	90 (9.0%)	112 (11.2%)	69 (6.9%)
	cat	149 (14.9%)	50 (5.0%)	51 (5.1%)	50 (5.0%)	43 (4.3%)	251 (25.1%)	156 (15.6%)	78 (7.8%)	89 (8.9%)	83 (8.3%)
	deer	100 (10.0%)	33 (3.3%)	72 (7.2%)	29 (2.9%)	82 (8.2%)	201 (20.1%)	261 (26.1%)	78 (7.8%)	85 (8.5%)	59 (5.9%)
	dog	100 (10.0%)	34 (3.4%)	58 (5.8%)	29 (2.9%)	72 (7.2%)	342 (34.2%)	143 (14.3%)	95 (9.5%)	74 (7.4%)	53 (5.3%)
	frog	82 (8.2%)	44 (4.4%)	59 (5.9%)	27 (2.7%)	62 (6.2%)	174 (17.4%)	319 (31.9%)	94 (9.4%)	66 (6.6%)	73 (7.3%)
	horse	120 (12.0%)	48 (4.8%)	40 (4.0%)	27 (2.7%)	50 (5.0%)	164 (16.4%)	139 (13.9%)	162 (16.2%)	92 (9.2%)	158 (15.8%)
	ship	203 (20.3%)	57 (5.7%)	23 (2.3%)	10 (1.0%)	7 (0.7%)	79 (7.9%)	33 (3.3%)	48 (4.8%)	367 (36.7%)	173 (17.3%)
	truck	106 (10.6%)	97 (9.7%)	23 (2.3%)	16 (1.6%)	14 (1.4%)	54 (5.4%)	96 (9.6%)	91 (9.1%)	177 (17.7%)	326 (32.6%)

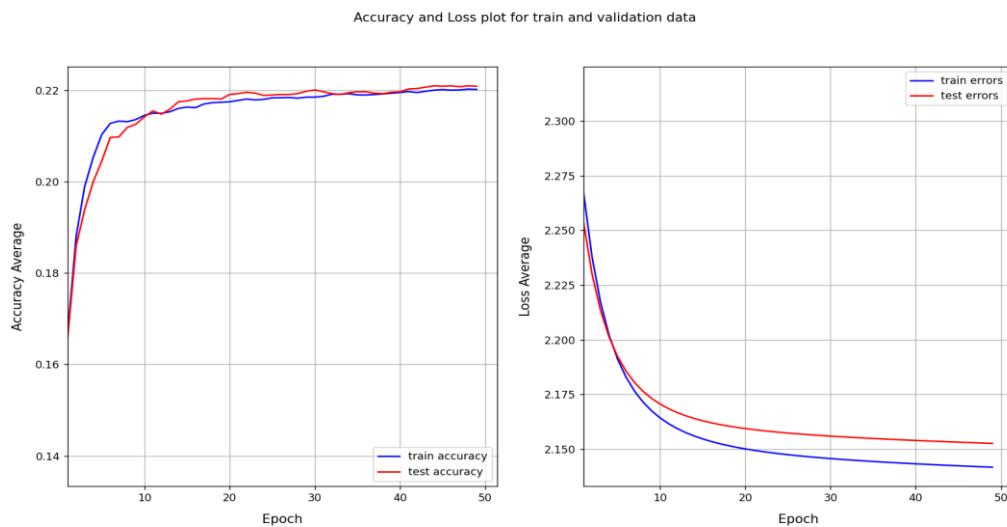
شکل -۳۹- ماتریس آشفتگی برای تابع LeakyReLU

ماتریس آشفتگی برای ReLU و LeakyReLU بسیار به هم نزدیک است.

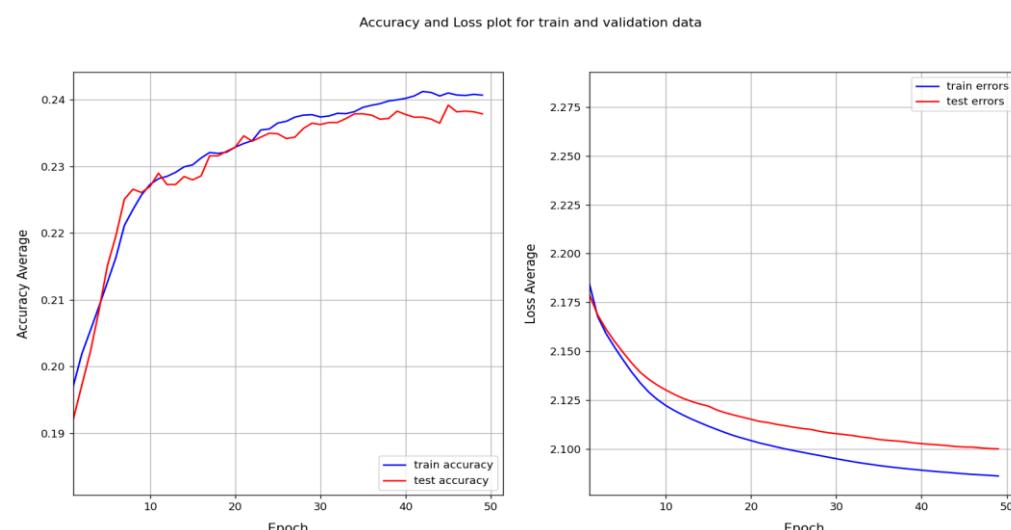
پیاده‌سازی این توابع به همراه مشتق آن‌ها در پوشه nets/ANN.py قرار گرفته است.

## سوال (۱.۹)

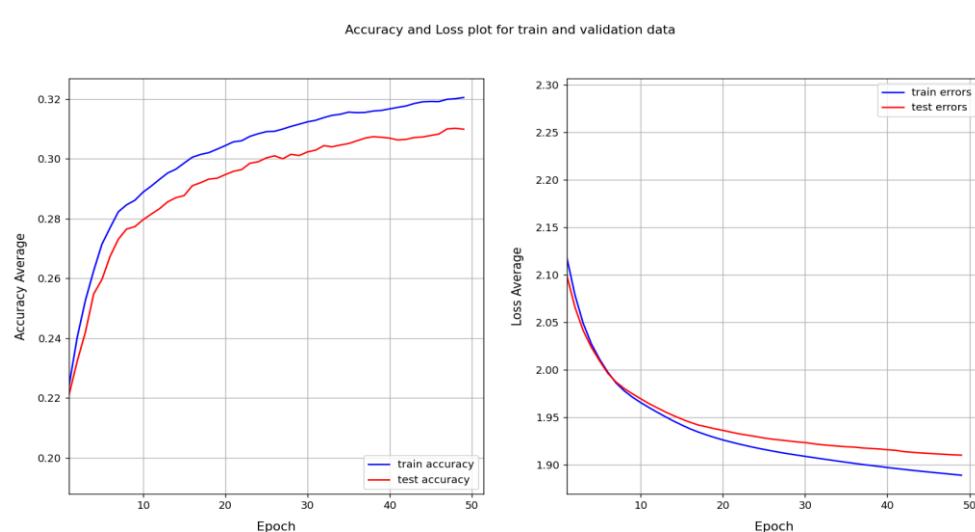
تعداد لایه‌ها مدل را از دو لایه به سه لایه افزایش دهید. خطای دقت به دست آمده بر روی هر کدام از داده‌های آموزشی، اعتبار سنجی و آزمون را نسبت به مدل دو لایه مقایسه کنید.  
برای بررسی این قسمت یکبار از دو لایه پنهان با ۱۶ نورون و یکبار از دو لایه پنهان با ۶۴ و ۱۶ نورون استفاده کرده و هر دو را با مدل یک لایه پنهان ۱۶ تایی در سوال ۱.۲ مقایسه کردیم. پس از انجام فرآیند آموزش روی هر سه مدل، نمودار دقت و هزینه برای داده‌های آموزش و اعتبار سنجی رسم شد.



شکل ۴۰- نمودار دقت و هزینه یک لایه پنهان با ۱۶ نورون



شکل ۴۱- نمودار دقت و هزینه دو لایه پنهان با ۱۶ نورون



شکل ۴۲- نمودار دقت و هزینه دو لایه پنهان با ۱۶ و ۶۴ نورون

برای مقایسه بهتر مدل‌ها دقت و زیان در جدول‌های زیر برای داده‌های آموزش، اعتبارسنجی و آزمایش آورده شده است. همانطور که مشخص است، مدل سه لایه عملکرد بهتری نسبت به مدل دولایه داشته است و به دقت بالاتر و مقدار زیان کمتری رسیدیم. این بدان معناست که مسئله طبقه‌بندی ما سخت است و مدل یک لایه قادر به حل کامل آن نمی‌باشد و برای دستیابی به دقت بهتر، باید مدل را پیچیده‌تر کنیم. علاوه بر افزایش لایه‌ها، در مدل سوم تعداد نورون‌های لایه‌ی مخفی اول را از ۱۶ به ۶۴ افزایش دادیم. در این حالت دقت بهبود بسیاری بهتری نسبت به افزایش لایه‌ها داشت و به دقت ۳۱ درصد روى داده‌های تست رسیدیم. این بهبود نیز بیانگر سخت بودن مسئله طبقه‌بندی و نیازمندی مسئله به مدل پیچیده‌تر می‌باشد. این نکته را نیز نباید فراموش کنیم که این افزایش لایه و نورون تا جایی خوب است که منجر به بیش‌برازش نشود.

جدول ۹- مقایسه دقت سه مدل

	یک لایه پنهان ۱۶ تایی	دو لایه پنهان ۱۶ تایی	دو لایه ۶۴ و ۱۶ تایی
آموزش	۲۲.۰۲	۲۴.۰۷	۳۲.۰۵
اعتبارسنجی	۲۲.۰۹	۲۳.۷۹	۳۰.۹۹
آزمون	۲۲.۴۱۰	۲۴.۵۶	۳۱.۳۲۰

جدول ۱۰- مقایسه هزینه سه مدل

	یک لایه پنهان ۱۶ تایی	دو لایه پنهان ۱۶ تایی	دو لایه ۶۴ و ۱۶ تایی
آموزش	۲.۱۴۱۷	۲.۰۸۶۱	۱.۸۸۷
اعتبارسنجی	۲.۱۵۲۶	۲.۰۹۹	۱.۹۰۹۹
آزمون	۲.۱۴۲	۲.۰۸۲	۱.۹۰۵۱

در ادامه ماتریس آشفتگی برای هر سه مدل را مشاهده می‌کنید. در مدل دولایه ۱۶ تایی و سه لایه ۱۶ تایی، بهترین طبقه‌بندی برای کلاس هوایپیما بوده است. در حالی که در مدل سه لایه ۶۴ و ۱۶ تایی بهترین طبقه‌بندی متعلق به کشتی است. نکته قابل توجه در این مقایسه این است که در دو مدل دولایه ۱۶ تایی، ۱۶۶ تا از تصاویر خودرو به عنوان کشتی طبقه‌بندی شدند. این تعداد در مدل سه لایه ۱۶ تایی

به ۱۵۸ تا کاهش یافته است. در مدل سه لایه ۶۴ و ۱۶ تایی تعداد تصاویر خودرو که به عنوان کشتی طبقه‌بندی شده‌اند، ۱۴۷ تا است. این مسئله نشان از یادگیری بهتر مدل و توانایی آن در تشخیص تصاویر تقریباً مشابه است.

		Confusion Matrix									
		Predicted label									
True label	airplane	359 (35.9%)	44 (4.4%)	38 (3.8%)	17 (1.7%)	14 (1.4%)	60 (6.0%)	59 (5.9%)	89 (8.9%)	217 (21.7%)	103 (10.3%)
	automobile	146 (14.6%)	130 (13.0%)	37 (3.7%)	13 (1.3%)	20 (2.0%)	109 (10.9%)	101 (10.1%)	86 (8.6%)	166 (16.6%)	192 (19.2%)
	bird	166 (16.6%)	40 (4.0%)	102 (10.2%)	28 (2.8%)	45 (4.5%)	149 (14.9%)	200 (20.0%)	89 (8.9%)	112 (11.2%)	69 (6.9%)
	cat	150 (15.0%)	51 (5.1%)	51 (5.1%)	49 (4.9%)	44 (4.4%)	250 (25.0%)	156 (15.6%)	78 (7.8%)	88 (8.8%)	83 (8.3%)
	deer	97 (9.7%)	32 (3.2%)	69 (6.9%)	30 (3.0%)	84 (8.4%)	205 (20.5%)	261 (26.1%)	77 (7.7%)	87 (8.7%)	58 (5.8%)
	dog	99 (9.9%)	36 (3.6%)	56 (5.6%)	29 (2.9%)	72 (7.2%)	341 (34.1%)	144 (14.4%)	97 (9.7%)	74 (7.4%)	52 (5.2%)
	frog	82 (8.2%)	43 (4.3%)	59 (5.9%)	27 (2.7%)	63 (6.3%)	172 (17.2%)	320 (32.0%)	94 (9.4%)	67 (6.7%)	73 (7.3%)
	horse	121 (12.1%)	48 (4.8%)	40 (4.0%)	27 (2.7%)	48 (4.8%)	163 (16.3%)	140 (14.0%)	163 (16.3%)	93 (9.3%)	157 (15.7%)
	ship	203 (20.3%)	54 (5.4%)	23 (2.3%)	11 (1.1%)	8 (0.8%)	76 (7.6%)	33 (3.3%)	48 (4.8%)	368 (36.8%)	176 (17.6%)
	truck	106 (10.6%)	96 (9.6%)	22 (2.2%)	16 (1.6%)	14 (1.4%)	54 (5.4%)	96 (9.6%)	93 (9.3%)	178 (17.8%)	325 (32.5%)

شکل ۴۳- ماتریس آشتفتگی مدل دو لایه ۱۶ تایی

		Confusion Matrix									
		Predicted label									
True label	airplane	456 (45.6%)	34 (3.4%)	66 (6.6%)	4 (0.4%)	30 (3.0%)	32 (3.2%)	55 (5.5%)	31 (3.1%)	212 (21.2%)	80 (8.0%)
	automobile	149 (14.9%)	155 (15.5%)	34 (3.4%)	12 (1.2%)	28 (2.8%)	95 (9.5%)	82 (8.2%)	98 (9.8%)	158 (15.8%)	189 (18.9%)
	bird	185 (18.5%)	37 (3.7%)	168 (16.8%)	15 (1.5%)	84 (8.4%)	102 (10.2%)	172 (17.2%)	92 (9.2%)	87 (8.7%)	58 (5.8%)
	cat	147 (14.7%)	53 (5.3%)	72 (7.2%)	36 (3.6%)	81 (8.1%)	202 (20.2%)	136 (13.6%)	113 (11.3%)	72 (7.2%)	88 (8.8%)
	deer	128 (12.8%)	34 (3.4%)	111 (11.1%)	14 (1.4%)	123 (12.3%)	163 (16.3%)	241 (24.1%)	60 (6.0%)	75 (7.5%)	51 (5.1%)
	dog	107 (10.7%)	33 (3.3%)	81 (8.1%)	17 (1.7%)	103 (10.3%)	280 (28.0%)	138 (13.8%)	114 (11.4%)	70 (7.0%)	57 (5.7%)
	frog	108 (10.8%)	50 (5.0%)	89 (8.9%)	11 (1.1%)	106 (10.6%)	133 (13.3%)	308 (30.8%)	70 (7.0%)	51 (5.1%)	74 (7.4%)
	horse	126 (12.6%)	47 (4.7%)	63 (6.3%)	20 (2.0%)	57 (5.7%)	133 (13.3%)	123 (12.3%)	187 (18.7%)	94 (9.4%)	150 (15.0%)
	ship	272 (27.2%)	59 (5.9%)	36 (3.6%)	5 (0.5%)	12 (1.2%)	51 (5.1%)	28 (2.8%)	24 (2.4%)	412 (41.2%)	101 (10.1%)
	truck	125 (12.5%)	96 (9.6%)	40 (4.0%)	14 (1.4%)	28 (2.8%)	42 (4.2%)	77 (7.7%)	82 (8.2%)	165 (16.5%)	331 (33.1%)

شکل ۴۴- ماتریس آشتفتگی مدل سه لایه ۱۶ تایی

		Confusion Matrix									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
True label	airplane	448 (44.8%)	35 (3.5%)	82 (8.2%)	20 (2.0%)	14 (1.4%)	21 (2.1%)	47 (4.7%)	38 (3.8%)	250 (25.0%)	45 (4.5%)
	automobile	97 (9.7%)	291 (29.1%)	24 (2.4%)	33 (3.3%)	20 (2.0%)	56 (5.6%)	126 (12.6%)	69 (6.9%)	147 (14.7%)	137 (13.7%)
	bird	197 (19.7%)	55 (5.5%)	178 (17.8%)	33 (3.3%)	89 (8.9%)	85 (8.5%)	205 (20.5%)	72 (7.2%)	63 (6.3%)	23 (2.3%)
	cat	100 (10.0%)	76 (7.6%)	80 (8.0%)	89 (8.9%)	49 (4.9%)	211 (21.1%)	217 (21.7%)	72 (7.2%)	59 (5.9%)	47 (4.7%)
	deer	98 (9.8%)	52 (5.2%)	150 (15.0%)	33 (3.3%)	164 (16.4%)	81 (8.1%)	241 (24.1%)	107 (10.7%)	50 (5.0%)	24 (2.4%)
	dog	73 (7.3%)	57 (5.7%)	112 (11.2%)	69 (6.9%)	54 (5.4%)	287 (28.7%)	170 (17.0%)	81 (8.1%)	79 (7.9%)	18 (1.8%)
	frog	49 (4.9%)	56 (5.6%)	82 (8.2%)	54 (5.4%)	54 (5.4%)	113 (11.3%)	486 (48.6%)	39 (3.9%)	47 (4.7%)	20 (2.0%)
	horse	100 (10.0%)	55 (5.5%)	81 (8.1%)	42 (4.2%)	83 (8.3%)	60 (6.0%)	123 (12.3%)	322 (32.2%)	55 (5.5%)	79 (7.9%)
	ship	154 (15.4%)	62 (6.2%)	23 (2.3%)	14 (1.4%)	13 (1.3%)	38 (3.8%)	34 (3.4%)	23 (2.3%)	573 (57.3%)	66 (6.6%)
	truck	110 (11.0%)	153 (15.3%)	20 (2.0%)	25 (2.5%)	15 (1.5%)	32 (3.2%)	104 (10.4%)	63 (6.3%)	184 (18.4%)	294 (29.4%)

شکل -۴۵ - ماتریس آشتفتگی مدل سه لایه ۶۴ و ۱۶ تابی

به طور کلی، تعداد لایه‌هایی که در یک شبکه عصبی نیاز داریم به پیچیدگی مسئله بستگی دارد. برای درک شهودی بهتر می‌توان گفت که لایه اول شبکه پیکسل‌های تصویر را به عنوان ورودی می‌گیرد و برخی از اشکال ساده مانند خطوط، منحنی‌ها و غیره را شناسایی می‌کند. لایه دوم می‌تواند این خطوط و منحنی‌ها را بگیرد و اشکالی مانند مثلث، مربع، دایره و ... را یاد بگیرد. به همین ترتیب، لایه‌های بعدی می‌توانند الگوهای بسیار پیچیده تری را شناسایی کنند. به همین دلیل است که ممکن است برای کارهای پیچیده‌تر به شبکه‌های عمیق تری نیاز داشته باشیم.

## سوال (۱.۱۰)

دلیل استفاده از تکانه در شبکه عصبی را شرح دهید. چگونه می‌توان این مفهوم را به صورت عملی پیاده کرد و پارامتر آن را تنظیم کنید؟ آیا استفاده از تکانه، باعث بهبود خطا و دقت مدل می‌شود؟

تکانه کمک می‌کند که بردارهای گرادیان در جهت صحیح حرکت کنند و در نتیجه منجر به همگرای سریعتر می‌شود. علاوه بر این، تکانه به کاهش نویز گرادیان نیز کمک می‌کند. در SGD ما مشتق دقیقتابع ضرر خود را محاسبه نمی‌کنیم و آن را در batch تخمین می‌زنیم. این بدان معناست که ما همیشه در جهت بهینه پیش نمی‌رویم، زیرا مشتقات ما نویزی هستند. بنابراین، به کمک تکانه، یعنی اضافه کردن

ضریبی از گام قبل، نویز را کاهش می‌دهیم. در مجموع، تکانه به سرعت آموزش کمک می‌کند و فرآیند آموزش را سریع‌تر می‌کند. همچنین به کمک تکانه می‌توان از نقاط مینیمم محلی فرار کرد و به نقطه بهینه رسید.

پیاده‌سازی تکانه در زمانی انجام می‌شود که می‌خواهیم وزن‌ها را به کمک گرادیان آپدیت کنیم. پیاده‌سازی به کمک فرمول زیر انجام شده است.  $\alpha$  نرخ یادگیری و  $\beta$  ضریب مومنتوم است.

$$V_t = \beta V_{t-1} + \frac{\alpha \delta l}{6W}$$

$$W = W - V_t$$

پیاده‌سازی این فرمول در فایل `ANN.py` کلاس `Layer` در پوشه `nets` انجام شده است.

در جدول زیر به ازای سه ضریب مومنتوم  $0.1$ ،  $0.5$  و  $0.9$  مدل را آموزش داده و نتایج را مقایسه کردیم. همانطور که می‌بینید، با افزایش مقدار مومنتوم، مقدار هزینه کاهش و دقت افزایش داشته است. بنابراین می‌توان گفت که اضافه کردن مومنتوم در افزایش دقت هر چند اندک، اما تاثیرگذار است.

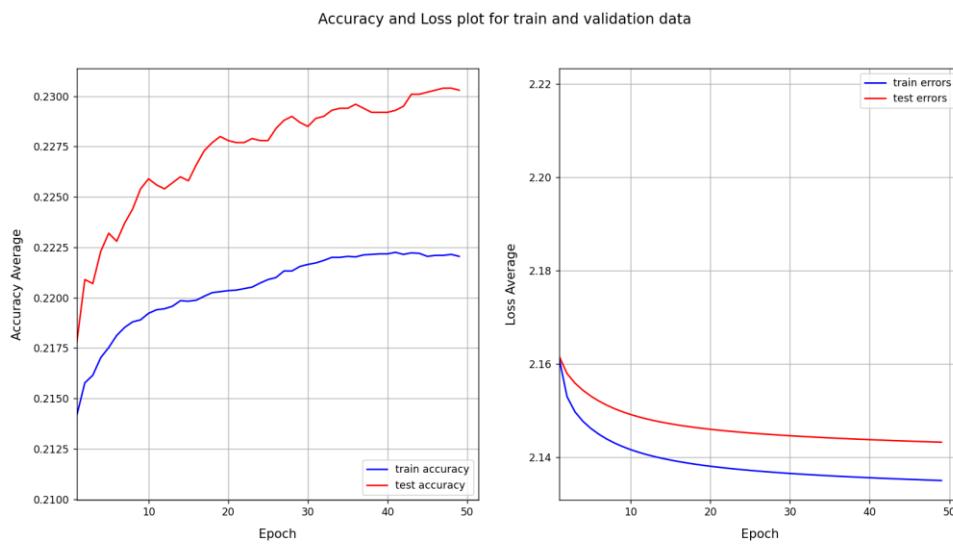
جدول ۹- مقایسه دقت به ازای مومنتوم‌های مختلف

Momentum	$0$	$0.1$	$0.5$	$0.9$
آموزش	۲۲.۰۲	۲۲.۰۶	۲۲.۳	۲۲.۲
اعتبارسنجی	۲۲.۰۹	۲۲.۰۷	۲۲.۵۲	۲۳.۰۳
آزمون	۲۲.۴۱۰	۲۲.۴۹۰	۲۲.۶۲۰	۲۲.۷۲۰

جدول ۱۰- مقایسه مقدار هزینه به ازای مومنتوم‌های مختلف

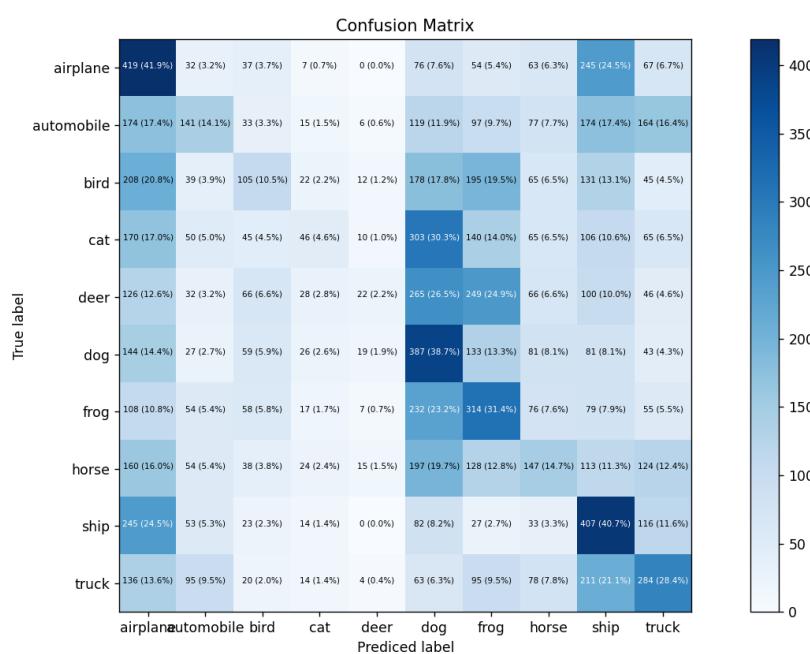
Momentum	$0$	$0.1$	$0.5$	$0.9$
آموزش	۲.۱۴۱۷	۲.۱۴۱	۲.۱۳۷۴	۲.۱۳۵
اعتبارسنجی	۲.۱۵۲۶	۲.۱۵۱۹	۲.۱۴۸۳	۲.۱۴۳۲
آزمون	۲.۱۴۲	۲.۱۴۱۳	۲.۱۳۷	۲.۱۳۰۹

همانطور که به کمک جدول بالا دریافتیم، بهترین مقدار مومنتوم ۰.۹ است. شکل زیر نمودار دقت و هزینه مربوط به آموزش با مومنتوم ۰.۹ را نشان می‌دهد.



شکل ۴۶- نمودار دقت و هزینه مدل با مومنتوم ۰.۹

با مقایسه این شکل و شکل ۲ که مربوط به آموزش مدل بدون مومنتوم است، در می‌یابیم زمانی که از مومنتوم استفاده کردیم، در ایپاک اول آپدیت وزن‌ها به گونه‌ای انجام شده است که میزان کاهش هزینه و افزایش دقت بسیار بیشتر بوده است. همچنین در بازه افزایش دقت از ۲۰ تا ۲۲ درصد، در حالتی که از مومنتوم استفاده کردیم، نمودار smooth تری نسبت به حالت بدون مومنتوم داریم که این نشان از کاهش نویز دارد. شکل زیر ماتریس آشفتگی برای آموزش با مومنتوم ۰.۹ را نشان می‌دهد.



شکل ۴۷- ماتریس آشفتگی برای مومنتوم ۰.۹

به طور کلی، مومنتوم یک ابرپارامتر است که در الگوریتم های بهینه سازی مانند نزول گرادیان تصادفی برای سرعت بخشیدن به همگرایی مدل استفاده می شود. این به بهینه ساز کمک می کند تا با افزودن کسری از بهروزرسانی قبلی به بهروزرسانی فعلی، سریع تر به سمت حداقل عملکرد ضرر حرکت کنیم. عموماً از ضریب  $0.9$  استفاده می شود. مقدار  $0.9$  به این معنی است که  $90\%$  از بهروزرسانی قبلی به بهروزرسانی فعلی اضافه می شود. مشخص شده است که این مقدار در بسیاری از برنامه های یادگیری عمیق به خوبی کار می کند، زیرا بین بیش از حد و یا کم رنگ شدن حداقل تابع ضرر تعادل برقرار می کند.

اگر مقدار تکانه خیلی کم باشد، ممکن است زمان بیشتری طول بکشد تا بهینه ساز همگرا شود، و اگر خیلی زیاد باشد، ممکن است بیش از حد شود و حول نقطه مینیمم نوسان کند. بنابراین،  $0.9$  تعادل خوبی بین این دو افراط در نظر گرفته می شود و در عمل برای بسیاری از مدل های یادگیری عمیق به خوبی کار می کند.

### سوال (۱.۱۱)

مشکل بیش برآش در شبکه های عصبی را توضیح دهید. برای مقابله با این مشکل، چه روش هایی وجود دارد؟ حداقل یکی از روش ها را پیاده سازی کنید و نتایج را بررسی کنید.

بیش برآش زمانی اتفاق می افتد که مدل دارای واریانس بالایی باشد. در این حالت، مدل در داده های آموزشی به خوبی عمل می کند اما در مجموعه ارزیابی دقت پایین و مقدار زیان بالایی دارد. به عبارتی، مدل الگوهای داده را در مجموعه داده آموزشی به خاطر می سپارد اما نمی تواند به نمونه های دیده نشده تعمیم دهد. بیش برآش در یکی از حالات زیر اتفاق می افتد:

- ❖ داده های مورد استفاده برای آموزش بسیار نویزی هستند. مدل نویز در داده های آموزشی را ضبط می کند و نمی تواند یادگیری مدل را تعمیم دهد.
- ❖ مدل دارای واریانس بالایی باشد.
- ❖ اندازه داده های آموزشی کافی نباشد و مدل بر روی داده های آموزشی محدود برای چندین دوره آموزش ببیند.

❖ معماری مدل دارای چندین لایه عصبی باشد که در کنار هم قرار گرفته اند. شبکه های عصبی عمیق پیچیده هستند و به زمان قابل توجهی برای آموزش نیاز دارند و اغلب منجر بیش برآش می شوند.

بنابراین، بیش برآش زمانی اتفاق می افتد که مدل نتواند تعمیم دهد و روی مجموعه داده آموزشی بسیار شود. روش های زیر برای جلوگیری از بیش برآش به کار گرفته می شوند. fit

• ساده سازی مدل

اولین گام در برخورد با بیش برازش، کاهش پیچیدگی مدل است. برای کاهش پیچیدگی، می‌توانیم لایه‌ها را حذف کنیم یا تعداد نورون‌ها را کاهش دهیم تا شبکه کوچک‌تر شود.

#### Early Stopping •

توقف زودهنگام نوعی منظم‌سازی در حین آموزش است. از آنجایی که همه شبکه‌های عصبی با استفاده از گرادیان نزولی یاد می‌گیرند، توقف زودهنگام تکنیکی است که برای همه مشکلات قابل استفاده است. روش گرادیان نزولی مدل را به روزرسانی می‌کند تا با هر تکرار بهتر با داده‌های آموزشی سازگار شود. پس از آن نقطه optimal، بهبود تناسب مدل با داده‌های آموزشی منجر به افزایش خطای تعمیم می‌شود. قوانین Early Stopping کمک می‌کند که فرآیند آموزش را قبل از overfitting متوقف کرد.

#### Data Augmentation •

Data Augmentation به معنای افزایش اندازه داده است که تعداد تصاویر موجود در مجموعه داده را افزایش می‌دهد. برخی از تکنیک‌های رایج آن عبارتند از: چرخش، مقیاس‌بندی، تغییر روشنایی، اضافه کردن نویز و غیره.

#### Regularization •

منظم‌سازی تکنیکی برای کاهش پیچیدگی مدل است. این کار را با اضافه کردن یک عبارت جریمه به تابع ضرر انجام می‌دهیم. رایج‌ترین تکنیک‌ها به عنوان منظم‌سازی L1 و L2 شناخته می‌شوند. هدف جریمه L1 به حداقل رساندن مقدار مطلق وزن‌ها است و هدف پنالتی L2 به حداقل رساندن مجذور بزرگی وزن‌ها است.

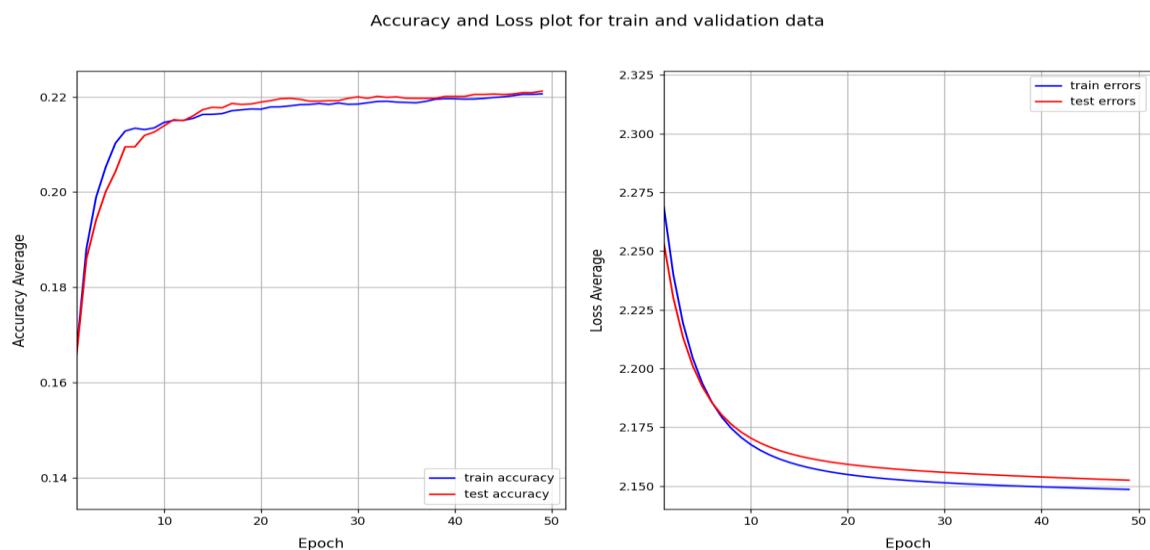
اگر داده‌ها برای مدل‌سازی دقیق پیچیده هستند، L2 انتخاب بهتری است/ زیرا می‌تواند الگوهای ذاتی موجود در داده‌ها را بیاموزد. در حالی که L1 در صورتی بهتر عمل می‌کند که داده‌ها به اندازه کافی ساده باشند تا با دقت مدل شوند.

#### Dropouts •

Dropout خود شبکه را تغییر می‌دهد. در طول آموزش در هر تکرار به طور تصادفی نورون‌ها را از شبکه عصبی جدا می‌کند. در واقع شبکه عصبی را مجبور می‌کنیم که به تمام ویژگی‌هایی که برای یک عکس استخراج می‌کند، وابسته نباشد و بتواند با مقدار کمتری از ویژگی‌ها هم یاد بگیرد.

در این سوال برای جلوگیری از بیش‌برازش از Regularization کمک گرفتیم. برای پیاده‌سازی L2 ضریبی از مجذور وزن‌های لایه آخر را به مقدار تابع زیان اضافه کردیم. هم‌چنین همان ضریب از خود وزن را نیز در هنگام به روز رسانی از وزن کم کردیم. مقدار ضریب که با نام gamma مشخص شده‌است در فایل

config.yml قابل تنظیم است. در صورتی که نیازی به Regularization نداشته باشیم، این ضریب را برابر صفر می‌گذاریم. بهترین مقداری که با توجه به مفروضات مسئله به دست آمد،  $1 \times 10^{-4}$  بود. شکل زیر نمودار دقต و هزینه را با این ضریب نشان می‌دهد.



شکل ۴۸- نمودار دقت و هزینه با regularization

بر روی داده‌های تست به دقت  $22.480$  رسیدیم و بدون L2 دقت  $22.410$  را داشتیم. اندکی بهبود در دقت حاصل شده است. اما برای بررسی بهتر نتیجه می‌توان به نمودار هزینه تابع validation و train دقت کرد. همانطور که می‌دانیم، یکی از روش‌های تشخیص بیش برآش این است که تابع هزینه validation افزایش می‌یابد درحالی که تابع هزینه آموزش کاهش می‌یابد. در این حالتی که از رگولاریزیشن استفاده کردیم، فاصله بین نمودار هزینه validation و loss کاهش یافت. شکل زیر نمودار ماتریس آشفتگی را نشان می‌دهد. چون بهبود قابل توجهی در دقت نداشتیم، تفاوت چندانی در این ماتریس ایجاد نشده است.

		Confusion Matrix									
		Predicted label									
True label	airplane	358 (35.8%)	44 (4.4%)	40 (4.0%)	17 (1.7%)	14 (1.4%)	60 (6.0%)	59 (5.9%)	88 (8.8%)	217 (21.7%)	103 (10.3%)
	automobile	146 (14.6%)	130 (13.0%)	38 (3.8%)	13 (1.3%)	21 (2.1%)	110 (11.0%)	100 (10.0%)	85 (8.5%)	165 (16.5%)	192 (19.2%)
	bird	165 (16.5%)	40 (4.0%)	103 (10.3%)	29 (2.9%)	45 (4.5%)	149 (14.9%)	200 (20.0%)	89 (8.9%)	112 (11.2%)	68 (6.8%)
	cat	149 (14.9%)	51 (5.1%)	51 (5.1%)	50 (5.0%)	43 (4.3%)	250 (25.0%)	158 (15.8%)	79 (7.9%)	87 (8.7%)	82 (8.2%)
	deer	97 (9.7%)	33 (3.3%)	69 (6.9%)	30 (3.0%)	85 (8.5%)	204 (20.4%)	261 (26.1%)	77 (7.7%)	87 (8.7%)	57 (5.7%)
	dog	99 (9.9%)	36 (3.6%)	57 (5.7%)	30 (3.0%)	73 (7.3%)	340 (34.0%)	143 (14.3%)	96 (9.6%)	74 (7.4%)	52 (5.2%)
	frog	81 (8.1%)	44 (4.4%)	61 (6.1%)	27 (2.7%)	63 (6.3%)	171 (17.1%)	322 (32.2%)	91 (9.1%)	67 (6.7%)	73 (7.3%)
	horse	120 (12.0%)	49 (4.9%)	40 (4.0%)	28 (2.8%)	47 (4.7%)	163 (16.3%)	141 (14.1%)	164 (16.4%)	93 (9.3%)	155 (15.5%)
	ship	203 (20.3%)	55 (5.5%)	23 (2.3%)	11 (1.1%)	8 (0.8%)	76 (7.6%)	33 (3.3%)	48 (4.8%)	367 (36.7%)	176 (17.6%)
	truck	106 (10.6%)	98 (9.8%)	22 (2.2%)	16 (1.6%)	14 (1.4%)	54 (5.4%)	96 (9.6%)	92 (9.2%)	178 (17.8%)	324 (32.4%)

شکل ۴۹- ماتریس آشفتگی برای regularization

برای بررسی بهتر عملکرد رگیولاریزیشن باید مدل را تا تعداد زیادی ایپاک آموزش دهیم و مسئله بیش برآذش را بررسی کنیم. متاسفانه به دلیل محدودیت سخت افزاری و زمانی امکان آموزش مدل با تعداد بالای ایپاک امکان پذیر نبود.

## سوال (۱.۱۲)

دلیل استفاده از K-fold چیست؟ این روش را به صورت عملی پیاده‌سازی کنید و نتایج را گزارش دهید.

یک روش ارزیابی است که در مقایسه با روش‌های دیگر منجر به مدلی با سوگیری کمتری می‌شود. زیرا تضمین می‌کند که هر نمونه از مجموعه داده اصلی این شانس را دارد که در فرآیند آموزش و تست ظاهر شود. اگر داده‌های ورودی محدودی داشته باشیم، این یکی از بهترین رویکردها است.

این روش شامل مراحل زیر می‌باشد:

✓ کل داده‌ها را به طور تصادفی به K فولد تقسیم می‌کنیم (مقدار K باید خیلی کوچک یا خیلی زیاد باشد، در حالت ایده‌آل ۵ تا ۱۰ را انتخاب می‌کنیم). مقدار بالاتر K منجر به مدل با سوگیری کمتری می‌شود.

✓ سپس مدل را با استفاده از K-1 تا این فولدها آموزش داده با استفاده از اخرين K فولد باقیمانده اعتبارسنجی کنید.

- ✓ این فرآیند را تا زمانی که هر K-fold به عنوان مجموعه آزمایشی استفاده شود، تکرار کنید. سپس میانگین نمرات ثبت شده خود را بگیرید. این معیار عملکرد مدل خواهد بود.

### سوال (۱.۱۳)

از تمام پارامترهای ذکر شده استفاده کرده و بهترین شبکه را آموزش و نتایج ارزیاقش دهید.

در این قسمت برای مقداردهی وزن‌های اولیه از He initialization استفاده کردیم. در این نوع مقدار دهی از فرمول زیر استفاده می‌شود. این مقدار دهی تاثیر به سزایی در افزایش دقت داشت.

$$W = randn(fan_{in}, fan_{out}) / \sqrt{\frac{fan_{in}}{2}}$$

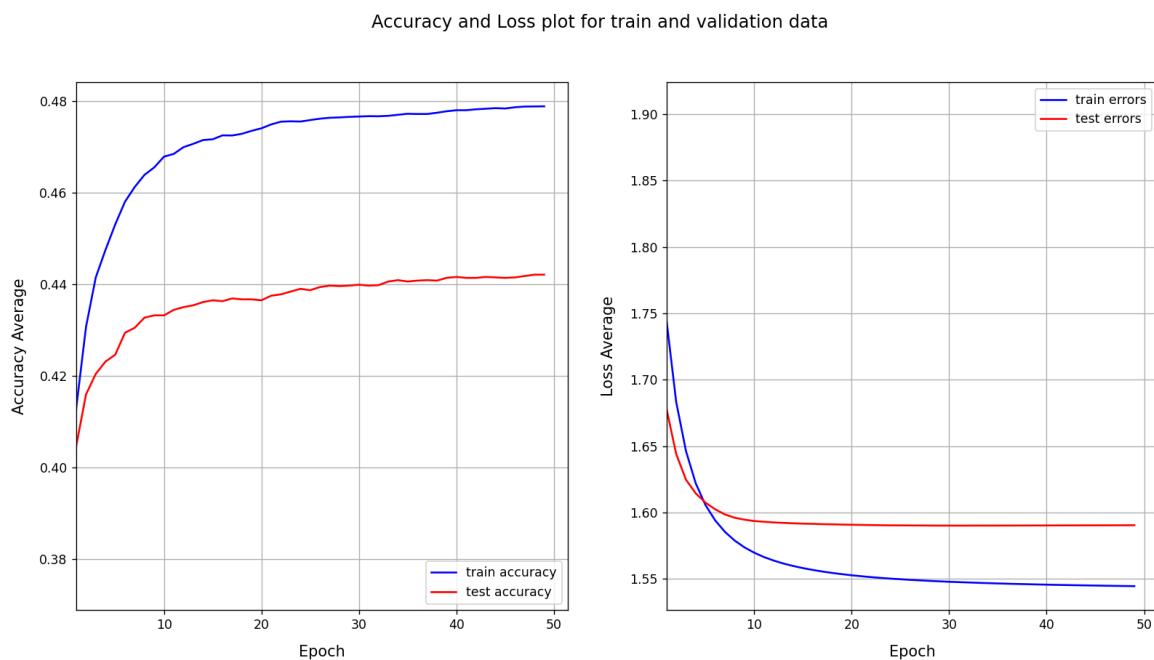
برای استفاده از این نوع مقداردهی، مقدار W\_init را برابر He قرار دهید. در غیر اینصورت مقدار آن خالی باشد.

همچنین برای بهبود مدل از learning rate scheduler کمک گرفتیم. برای این منظور config.yml پیاده‌سازی شده است. برای استفاده از learning rate scheduler Decay برابر True قرار دهید. جدول زیر سایر پارامترهای استفاده شده را نشان می‌دهد.

جدول ۱۱ - پارامترهای بهینه

پارامتر	مقدار
تعداد نورون‌ها	یک لایه ۱۵۰ نورونی و یک لایه ۶۴ نورونی
مومنتوم	۰.۹
تعداد ایپاک	۵۰
تابع هزینه	Cross-Entropy
تابع فعال‌ساز	ReLU
ضریب regularization	۰.۰۰۱
Batch size	۳۲
Learning rate اولیه	۰.۰۱

پس از آموزش مدل به کمک پارامترهای بالا، نمودار دقت و هزینه برای داده آموزش و اعتبارسنجی به صورت زیر شد.



شکل ۵۰- نمودار دقت و هزینه برای مدل بهینه

جدول زیر نتایج ارزیابی روی داده تست را نشان می‌دهد. همانطور که می‌بینیم، دقت به دست آمده بسیار بهتر از گام‌های قبل است. هرچند این دقت با دقت ایده آل فاصله دارد، اما با اضافه کردن پارامترهای دیگر و افزایش تعداد ایپاک می‌توان این دقت را افزایش داد.

جدول ۱۲- نتایج ارزیابی بر روی داده تست در مدل بهینه

Accuracy	۴۴.۱
precision	۴۴.۱
recall	۴۴.۱
F1 score	۴۴.۱
Loss	۱.۵۹۱

شکل زیر ماتریس آشفتگی داده‌های تست را نشان می‌دهد. همانطور که از ماتریس می‌توان دریافت، طبقه‌بندی چهار کلاس سگ، گربه، پرنده و گوزن از سایر کلاس‌ها سخت‌تر بوده است. تعداد زیادی از سگ‌ها به عنوان گربه و تعداد زیادی از گربه‌ها به عنوان سگ طبقه‌بندی شده‌اند. به دلیل شباهت این دو

حیوان به یکدیگر مدل در تشخیص آن‌ها ضعف داشته است. علاوه بر این، تعدادی از تصاویر کامیون نیز به عنوان ماشین طبقه‌بندی شده‌اند.

		Confusion Matrix									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
True label	airplane	511 (51.1%)	38 (3.8%)	56 (5.6%)	27 (2.7%)	38 (3.8%)	18 (1.8%)	30 (3.0%)	47 (4.7%)	171 (17.1%)	64 (6.4%)
	automobile	50 (5.0%)	523 (52.3%)	14 (1.4%)	31 (3.1%)	20 (2.0%)	23 (2.3%)	33 (3.3%)	37 (3.7%)	85 (8.5%)	184 (18.4%)
	bird	90 (9.0%)	24 (2.4%)	282 (28.2%)	82 (8.2%)	160 (16.0%)	79 (7.9%)	133 (13.3%)	85 (8.5%)	29 (2.9%)	36 (3.6%)
	cat	32 (3.2%)	38 (3.8%)	96 (9.6%)	248 (24.8%)	73 (7.3%)	178 (17.8%)	143 (14.3%)	87 (8.7%)	36 (3.6%)	69 (6.9%)
	deer	46 (4.6%)	28 (2.8%)	142 (14.2%)	56 (5.6%)	395 (39.5%)	48 (4.8%)	137 (13.7%)	80 (8.0%)	40 (4.0%)	28 (2.8%)
	dog	31 (3.1%)	20 (2.0%)	112 (11.2%)	167 (16.7%)	63 (6.3%)	344 (34.4%)	104 (10.4%)	91 (9.1%)	37 (3.7%)	31 (3.1%)
	frog	26 (2.6%)	35 (3.5%)	90 (9.0%)	72 (7.2%)	126 (12.6%)	52 (5.2%)	510 (51.0%)	38 (3.8%)	14 (1.4%)	37 (3.7%)
	horse	46 (4.6%)	40 (4.0%)	55 (5.5%)	61 (6.1%)	82 (8.2%)	81 (8.1%)	56 (5.6%)	487 (48.7%)	25 (2.5%)	67 (6.7%)
	ship	114 (11.4%)	80 (8.0%)	22 (2.2%)	27 (2.7%)	29 (2.9%)	27 (2.7%)	13 (1.3%)	24 (2.4%)	597 (59.7%)	67 (6.7%)
	truck	50 (5.0%)	192 (19.2%)	17 (1.7%)	46 (4.6%)	10 (1.0%)	22 (2.2%)	36 (3.6%)	60 (6.0%)	79 (7.9%)	488 (48.8%)

## بخش دوم - رگرسن

### سوال (۲.۱)

آیا برای این دیتافریم پیش پردازش خاصی نیاز است؟ در صورت نیاز آن را اعمال کنید.

برای بررسی داده‌ها به دستور df.info نگاهی به شرایط دیتاست می‌اندازیم. شکل زیر نتیجه این دستور را نشان می‌دهد. همانطور که مشخص است، هیچ یک از ستون‌ها مقدار nan ندارند و از این بابت دیتاست تمیز است.

RangeIndex: 10000 entries, 0 to 9999			
Data columns (total 17 columns):			
#	Column	Non-Null Count	Dtype
0	squareMeters	10000 non-null	int64
1	numberOfRooms	10000 non-null	int64
2	hasYard	10000 non-null	int64
3	hasPool	10000 non-null	int64
4	floors	10000 non-null	int64
5	cityCode	10000 non-null	int64
6	cityPartRange	10000 non-null	int64
7	numPrevOwners	10000 non-null	int64
8	made	10000 non-null	int64
9	isNewBuilt	10000 non-null	int64
10	hasStormProtector	10000 non-null	int64
11	basement	10000 non-null	int64
12	attic	10000 non-null	int64
13	garage	10000 non-null	int64
14	hasStorageRoom	10000 non-null	int64
15	hasGuestRoom	10000 non-null	int64
16	price	10000 non-null	float64

dtypes: float64(1), int64(16)  
memory usage: 1.3 MB  
None

شکل ۵۱ - ستون‌های دیتاست

اما با بررسی رنج ستون‌های مختلف، در می‌یابیم که هر ستون در رنج متفاوتی قرار دارد. به همین جهت داده‌های هر ستون باید نرمال یا استاندارد شوند. به همین جهت هر دو روش در پیاده سازی کرده ایم. اما نتیجه به دست آمده از نرمال کردن بهتر بود.

### سوال (۲.۲) پیاده‌سازی رگرسن

برای اجرای این سوال، پس از قرار دادن آدرس مکان دیتاست و قراردادن Regression به عنوان regression\_main.config.yml در train.py و cost\_function سپس test.py فراخوانی می‌شود.

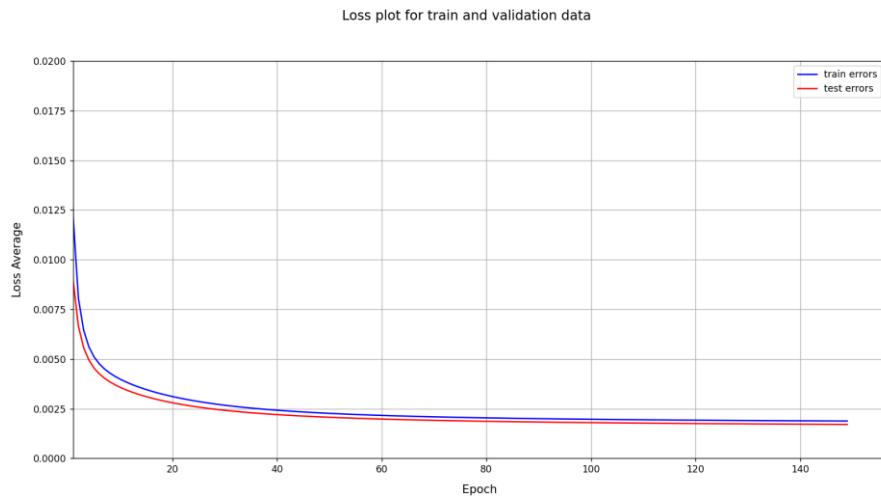
برای پیاده‌سازی رگرسن، تابع هزینه آن در losses/regression.py پیاده سازی شده است. این تابع هزینه همان MSE می‌باشد. علاوه بر این، در رگرسن در لایه آخر از تابع فعال ساز استفاده نشده است. جدول زیر پارامترهای بهینه برای آموزش مدل رگرسن را نشان می‌دهد.

جدول ۱۳ - پارامترهای بهینه برای رگرسن

پارامتر	مقدار
تعداد نورون‌ها	یک لایه ۶۴ نوروونی
مومنتوم	۰.۹
تعداد ایپاک	۱۵۰
تابع هزینه	MSE
تابع فعال‌ساز	ReLU
ضریب regularization	*
Batch size	۳۲
Learning rate اولیه	۰.۰۰۱

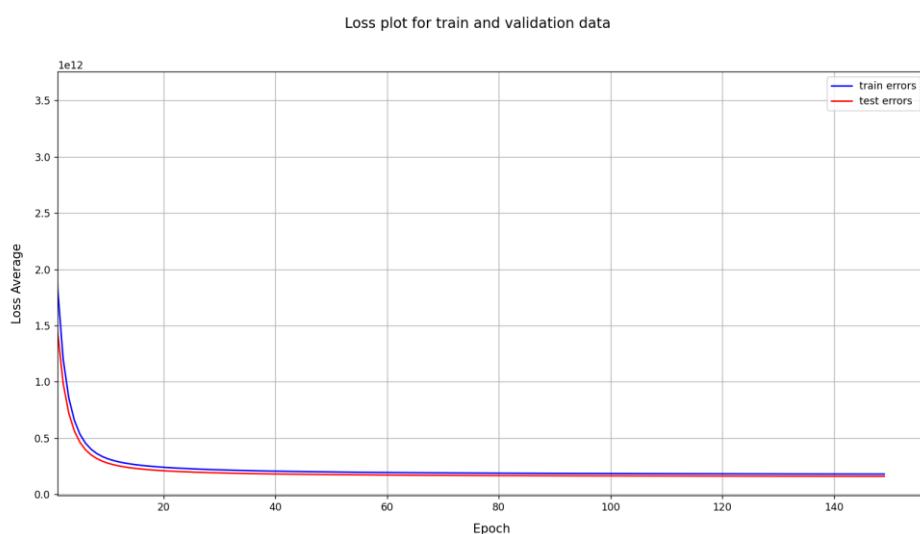
### سوال ۲.۳) نمودار هزینه

برای نمودار هزینه، یک بارستون target را نیز normal کردیم و یک بار هم بدون نرمال کردن ستون امتحان کردیم. در شکل نمودار تغییری ایجاد نمی‌شود و فقط scale متفاوتی خواهیم داشت. پس از آموزش مدل، تابع هزینه برای داده آموزش و اعتبارسنجی در حالت نرمال بودن ستون هدف، به صورت زیر شد. همانطور که می‌بینید، این نمودار نزولی است و مقدار هزینه رو به کاهش است که نشان دهنده آموزش مدل می‌باشد. بعد از ۶۰ ایپاک نمودار هزینه تقریباً ثابت شده و به کندی کاهش پیدا می‌کند. با این حال با افزایش تعداد ایپاک همچنان تابع هزینه سیر نزولی خواهد داشت.



شکل ۵۲- نمودار هزینه رگرسن با نرمال سازی ستون هدف

شکل زیر نمودار هزینه بدون نرمال سازی ستون هدف می باشد. همانطور که می بینید، شباهت زیادی با نمودار قبلی دارد و تنها scale متفاوت است.



شکل ۵۳- نمودار هزینه رگرسن بدون نرمال سازی ستون هدف

جدول زیر برای مقایسه مقدار تابع هزینه رسم شده است. همانطور که مشاهده می کنید مقدار هزینه هر سه داده بسیار کم و به هم نزدیک است.

جدول ۱۴ - مقایسه مقدار هزینه دادگان در رگرسن

۰.۰۰۱۹	داده آموزش
۰.۰۰۱۷	داده اعتبار سنجی
۰.۰۰۱۸	داده تست

جدول زیر نیز تابع هزینه در حالت نرمال نکردن ستون هدف را نشان می‌دهد. مقدار اولیه loss در ایپاک اول ۳۵۸۴۰۷۹۲۶۱۹۵۹.۳۵۲ بوده و پس از آموزش به ۱۸۲۶۲۲۷۰۷۰۵۹.۸۷۱۶ رسیده است. یعنی از سه میلیون به ۱۸۰ هزار رسیدیم و این نشان می‌دهد که مدل آموزش دیده است.

جدول ۱۵ - مقایسه مقدار هزینه دادگان در رگرسن بدون نرمال سازی

۱۸۰۱۹۶۱۳۶۵۶۵.۶۳۸۶	داده آموزش
۱۶۰۱۱۴۲۹۴۴۴۰.۳۰۹۸	داده اعتبار سنجی
۱۸۲۶۲۲۷۰۷۰۵۹.۸۷۱۶	داده تست

## مراجع

- [1] CIFAR-10 analysis with a neural network | Kaggle
- [2] Coding A Neural Network From Scratch in NumPy | by Joe Sasson | Towards Data Science
- [3] Understand Data Normalization in Machine Learning | by Zixuan Zhang | Towards Data Science
- [4] How to use Data Scaling Improve Deep Learning Model Stability and Performance - MachineLearningMastery.com
- [5] Stochastic Gradient Descent with momentum | by Vitaly Bushaev | Towards Data Science
- [6] Batch, Mini Batch & Stochastic Gradient Descent | by Sushant Patrikar | Towards Data Science
- [7] Techniques to prevent overfitting in Neural Networks | Data Science and Machine Learning | Kaggle
- [8] Deriving the Backpropagation Equations from Scratch (Part 1) | by Thomas Kurbiel | Towards Data Science
- [9] Building a Neural Network from Scratch in Python and in TensorFlow - nick becker
- [10] Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning | by Suki Lau | Towards Data Science
- [11] python - How to split/partition a dataset into training and test datasets for, e.g., cross validation? - Stack Overflow
- [12] Why and how to Cross Validate a Model? | by Sanjay.M | Towards Data Science