گزارش پروژه شبکه‌های عصبی مصنوعی

هوش محاسباتی

دکتر حسین کارشناس

آدرینا ابراهیمی ۹۹۳۶۲۳۰۰۲

کیان مجلسی ۹۹۳۶۱۳۰۵۱

ریپازیتوری گیت‌هاب پروژه

(ریپازیتوری گیت‌هاب پس از آخرین ارائه در دسترس همگان قرار می‌گیرد.)

اردیبهشت ۱۴۰۲

فهرست

۱- مقدمه

این پروژه با هدف توسعه یک سیستم مبتنی بر شبکه‌های عصبی به منظور حل مسئله شناسایی داده‌های موجود در مجموعه تصاویر CIFAR10 انجام شده است که در سه بخش مختلف با تمرکز جداگانه هر کدام روی آموزش، معماری و کاربرد شبکه‌های عصبی توسعه داده شده است. بخش‌های مختلف پروژه عبارت‌اند از: دسته‌بندی تصاویر با شبکه عصبی، جستجوی معماری عصبی برای شناسایی الگو و خوشه‌بندی به کمک شبکه‌های عصبی.

۲- بخش اول: دسته‌بندی تصاویر با شبکه عصبی

در این بخش باید یک شبکه عصبی پیش‌خور برای دسته‌بندی تصاویر آموزش داده شود. معماری این شبکه از دو قسمت اصلی استخراج‌کننده ویژگی و دسته‌بند تشکیل می‌شود. قسمت اول، هر تصویر را به عنوان ورودی می‌گیرد و آن را به برداری در یک فضای ویژگی نگاشت می‌کند. این بردار، بازنمایی یکتایی از تصویر ورودی است که می‌توان از آن برای وظایف مختلفی از جمله دسته‌بندی استفاده کرد. به این منظور باید از شبکه کانولوشنی ResNet34 با وزن‌های از پیش آموزش دیده بر روی مجموعه داده ImageNet به عنوان استخراج‌کننده ویژگی استفاده کنیم. همچنین در این بخش نیازی به آموزش قسمت استخراج کننده ویژگی نیست و پارامترهای آن نباید در فرآیند آموزش تغییر یابند. در قسمت دوم، بردار ویژگی دریافت شده در ورودی باید به یکی از دسته‌های از پیش مشخص شده نگاشت شود. به این منظور در لایه آخر این شبکه دسته‌بند باید از تابع فعال‌سازی SoftMax استفاده کنیم تا برای هر دسته یک احتمال تعیین شود. این شبکه را با بکارگیری تابع خطا Categorical Cross-Entropy آموزش می‌دهیم.

۲-۱- قسمت الف

در قسمت الف این بخش از ما خواسته شده است از یک شبکه عصبی MLP سه لایه (یک لایه مخفی و یک لایه خروجی) به عنوان دسته‌بند استفاده کنیم. تعداد نورون‌های لایه مخفی را ۲۰ در نظر گرفته و برای آن‌ها از تابع فعالسازی ReLU استفاده می‌کنیم. این شبکه را به وسیله الگوریتم پس انتشار خطا و بهینه‌ساز SGD با نرخ یادگیری ۰/۰۰۱ برای ۲۰ دور بر روی قسمت آموزشی مجموعه داده CIFAR10 آموزش داده و وزن‌های دسته‌بند را بروزرسانی می‌کنیم. سپس معیارهای صحت، ماتریس درهم‌ریختگی و امتیاز F1 مدل آموزش دیده را هم روی مجموعه آموزشی و هم مجموعه آزمایشی محاسبه می‌کنیم.

۲-۱-۱ روش انجام آزمایش در قسمت الف

پس از خواندن داده‌ها و اعمال مبدل، داده‌های آموزشی، آزمایش و اعتبارسنجی (۱۵ درصد داده‌های آموزشی) را جدا می‌کنیم. سپس، کلاس‌های لایه‌ها، توابع فعال‌سازی، هزینه، بهینه‌ساز، استخراج‌کننده‌های ویژگی و مدل شبکه عصبی را تعریف می‌کنیم (هر یک جداگانه شرح داده خواهند شد.). در مرحله بعدی، یک شی از کلاس مدل شبکه عصبی با تنظیم پارامترهای batch_size و feature_extractor ساخته و با صدا کردن متد train از این مدل به تعداد دور مورد نظر عملیات آموزش شبکه عصبی را انجام می‌دهیم. در نهایت با استفاده از متد evaluate دقت مدل را روی داده‌های آزمایش و آموزش می‌سنجیم.

۲-۱-۲ توضیح اجمالی کد در قسمت الف

۲-۱-۲-۱ مبدل

در این قسمت از مبدل Resize برای افزایش سایز تصاویر به 256x256 پیکسل به کار رفته است. همچنین از مبدل CenterCrop نیز برای برش در قسمت میانی تصویر استفاده شده است.

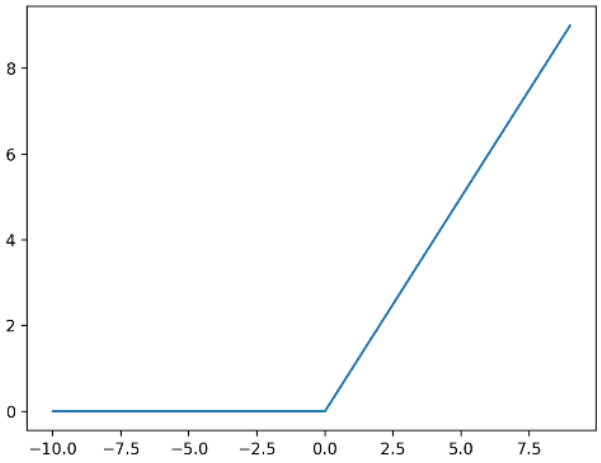
۲-۱-۲-۲ کلاس Dense

این کلاس یک لایه (شامل نورون‌هایی که تمامی ورودی‌های قبل به آن متصل‌اند) از شبکه را مشخص کرده. آرگومان‌های این کلاس عبارت است از: تعداد ورودی‌ها و تعداد نورون‌های این لایه. وزن‌ها را با توجه به فرمول زیر (هیوریستیک He) و بایاس را با برداری تمام صفر مقداردهی اولیه کرده و با استفاده از دو متد forward و backward می‌توان عملیات پیش فرستادن و پس‌انتشار را با فرمول زیر انجام داد.

$$weight\_initialize \rightarrow randn(n_{inputs}, n_{neurons}) * \sqrt{2/n_{inputs}}$$

۲-۱-۲-۳ توابع فعال‌ساز

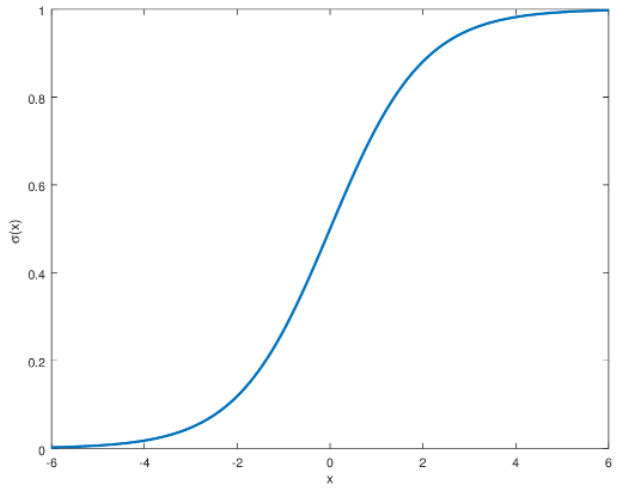
- **ReLU:** در این قسمت تابع ReLU برای عملیات پیش فرستادن و پس‌انتشار در دو متد forward و backward فرمول زیر پیاده‌سازی شده است.



شکل ۲-۱ نمودار تابع فعالیت ReLU

$$f(x) = \max(0, x) = \begin{cases} x \; if \; x > 0 \\ 0 \; otherwise \end{cases}$$

- **Sigmoid:** در این قسمت تابع Sigmoid برای عملیات پیش فرستادن و پس‌انتشار در دو متد forward و backward با فرمول زیر پیاده‌سازی شده است.



شکل ۲-۲ نمودار تابع فعالیت Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

۲-۱-۴ کلاس Categorical_Cross_Entropy_loss_Softmax

در این قسمت تابع خطا Categorical Cross Entropy به همراه تابع فعال‌ساز SoftMax برای دو عملیات پیش فرستان و پس‌انتشار با فرمول زیر پیاده‌سازی شده است.

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{s}) = -\sum_{i=1}^{batch\_size} \boldsymbol{y}^{(i)} \ln(\boldsymbol{s}^{(i)})$$

$$\boldsymbol{s}^{(i)} = \left(\frac{e^{z_1}}{\sum_{i=1}^{n} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{n} e^{z_i}}, \cdots, \frac{e^{z_n}}{\sum_{i=1}^{n} e^{z_i}}\right)$$

۲-۱-۲-۵ کلاس SGD (Stochastic Gradient Descent)

این کلاس با دریافت نرخ یادگیری عملیات به‌روزرسانی وزن‌ها و بایاس‌ها را طبق فرمول زیر انجام می‌دهد.

$$w \coloneqq w - \eta \frac{\partial J}{\partial w}$$

$$b \coloneqq b - \eta \frac{\partial J}{\partial b}$$

۲-۱-۲-۵ کلاس ResNet34

در این کلاس استخراج‌کننده ویژگی ResNet34 به کار گرفته شده است که در ابتدا یک شیء از ResNet34 ساخته شده و پس از آن لایه آخر (fc) آن جدا شده و یک شبکه عصبی از روی آن ساخته می‌شود. در متد get_features با دادن یک تصویر ویژگی‌های آن استخراج شده و آن‌ها را برمی‌گرداند. در متد get_size ابعاد خروجی استخراج‌کننده ویژگی برگردانده می‌شود.

۲-۱-۲-۶ کلاس AKModel

مدل شبکه عصبی مصنوعی در این کلاس پیاده‌سازی شده است و دارای بخش‌های زیر می‌باشد.

- **متد __init__:** در این قسمت مقادیر استخراج‌کننده ویژگی، تعداد کلاس‌های دسته‌بند، اندازه دسته‌ها و پارامتر tune که هنگام ساخت یک شی از این کلاس دریافت می‌شوند، در متغیرهای کلاس ذخیره شده و بهینه‌ساز هم تعیین می‌شوند. پس از آن اندازه ویژگی‌های ورودی که از استخراج‌کننده ویژگی گرفته شده نیز ذخیره می‌شود. در صورتی که پارامتر tune (این پارامتر در قسمت دوم پروژه استفاده شده است در صورتی که این مقدار برابر true باشد، مقداردهی اولیه به لایه‌ها و انتساب توابع فعال‌سازی انجام نمی‌شود.) مقداردهی نشده باشد، دو لایه ذکر شده در صورت پروژه به همراه توابع فعال‌ساز تعریف می‌شوند. پس از آن داده‌های آموزش، اعتبارسنجی و آزمایش روی اندازه دسته‌ها جدا شده و در متغیرهای تعریف شده ذخیره می‌شوند.

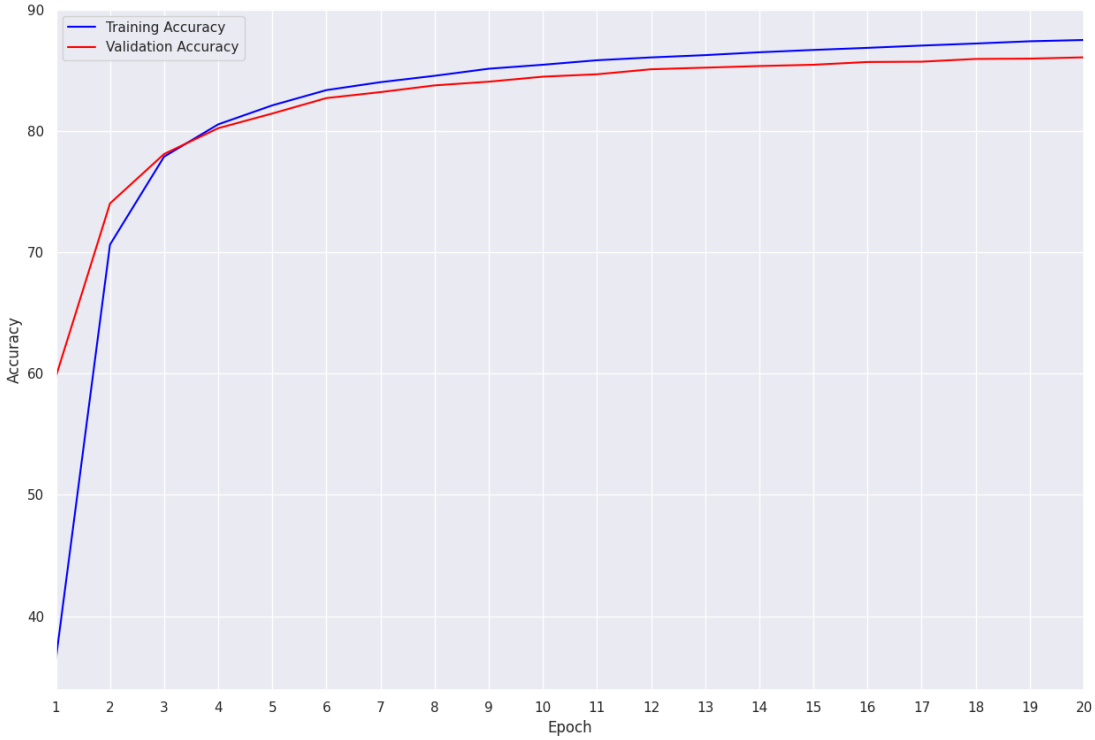
- **متد forward_propagation_:** در این متد در ابتدا در صورتی که تعداد لایه‌ها با تعداد توابع فعال‌ساز برابر نباشد یک اکسپشن برگردانده می‌شود. در غیر این صورت، ویژگی‌های داده از استخراج‌کننده ویژگی دریافت شده و به ترتیب روی هر لایه و تابع فعال‌ساز مربوط به آن عملیات forward انجام می‌شود. در صورتی که در لایه آخر باشیم، باید از متد forward کلاس Categorical Cross Entropy Loss Softmax استفاده کنیم به همین علت آرگومان‌های batch_size و class_label را نیز به عنوان پارامتر می‌دهیم.
- **متد test_val_:** در این متد دقت و هزینه برای هر دسته از داده‌های اعتبارسنجی حساب شده و در لیست‌های مربوط به آن ذخیره می‌شوند در نهایت میانگین دقت و هزینه برای کل داده‌های اعتبارسنجی برگردانده می‌شوند.
- **evaluate:** در این متد داخل یک حلقه روی داده‌های آموزش یا آزمایش یک‌بار عملیات پیش‌فرستادن را انجام داده و برچسب‌های پیش‌بینی شده را دریافت و ذخیره می‌کنیم. پس از آن، معیارهای صحت و F1 را برای کل داده‌های آموزش یا آزمایش محاسبه می‌کنیم.
- **predict:** در این متد یک بار عملیات پیش فرستادن را برای داده‌هایی که به عنوان پارامتر داده می‌شوند محاسبه کرده و برچسب‌های پیش‌بینی شده را برمی‌گردانیم.
- **train:** در این متد ابتدا لیست‌های تاریخچه را تعریف کرده و پس از آن به تعداد دورهایی که در آرگومان متد داده می‌شوند یک حلقه تعریف کرده و پس از آن روی دسته‌های مختلف عملیات پیش فرستادن را انجام داده، برچسب‌های پیش‌بینی شده را ذخیره کرده و دقت و هزینه را محاسبه می‌کنیم. حال، عملیات پس انتظار را انجام داده و پس از آن وزن‌ها و بایاس هر لایه را اپدیت می‌کنیم. در نهایت، دقت و هزینه را برای داده‌های اعتبارسنجی محاسبه کرده و مقادیر دقت و هزینه را برای داده‌های آموزش و اعتبارسنجی در لیست‌های تاریخچه ذخیره می‌کنیم.
- **plot_losses_and_accuracy:** در این متد نمودار دقت و هزینه برای داده‌های آموزش و اعتبارسنجی رسم می‌شود.
- **plot_confusion_matix:** در این متد داخل یک حلقه روی داده‌های آموزش یا آزمایش یک‌بار عملیات پیش‌فرستادن را انجام داده و برچسب‌های پیش‌بینی شده را دریافت و ذخیره می‌کنیم.

پس از آن، با برچسب‌های پیش‌بینی شده و برچسب‌های واقعی ماتریس درهم‌ریختگی را ایجاد می‌کنیم.

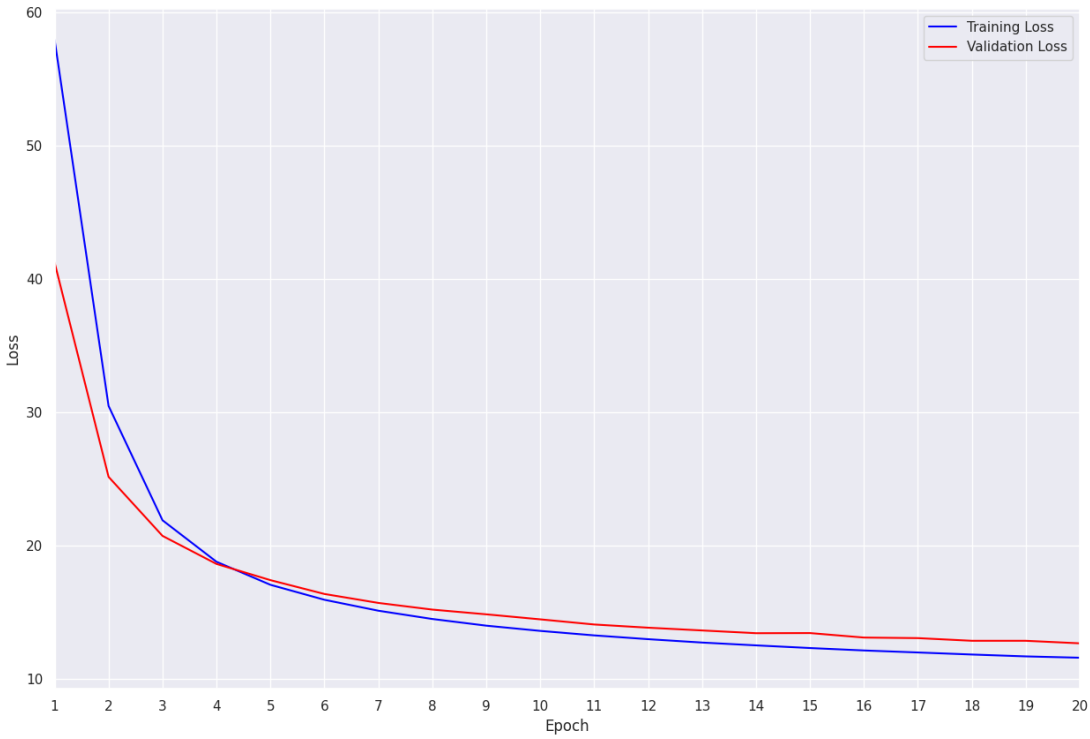
۳-۱-۲ تحلیل نتایج قسمت الف

۱-۳-۱-۲ اندازه دسته ۳۲، تعداد دور ۲۰ و نرخ یادگیری ۰/۰۰۱

- برای پارامترهایی که در عنوان ذکر شده‌اند، به ترتیب نمودار صحت و هزینه برای داده‌های آموزش و اعتبارسنجی هنگام آموزش مدل به شکل زیر می‌باشد. مشاهده می‌شود صحت برای داده‌های اعتبارسنجی همراه صحت برای داده‌های آموزش افزایش یافته و به همگرایی نسبی رسیده‌اند؛ برای همین اطمینان حاصل می‌شود که بیش‌برازش رخ نداده است. همچنین هزینه برای داده‌های اعتبارسنجی نیز همراه هزینه برای داده‌های آموزش کاهش یافته و به همگرایی نسبی رسیده‌اند؛ بنابراین مجددا می‌توان نتیجه گرفت بیش‌برازشی رخ نداده‌است.



شکل ۲-۳ نمودار صحت برای داده‌های آموزش و اعتبارسنجی



شکل ۲-۴ نمودار هزینه برای داده‌های آموزش و اعتبارسنجی

- معیار صحت و امتیاز F1 برای داده‌های آموزش و آزمایش به شرح زیر می‌باشد. (دقیقا خروجی‌های حاصل از اجرای کد قرار داده شده‌اند.)

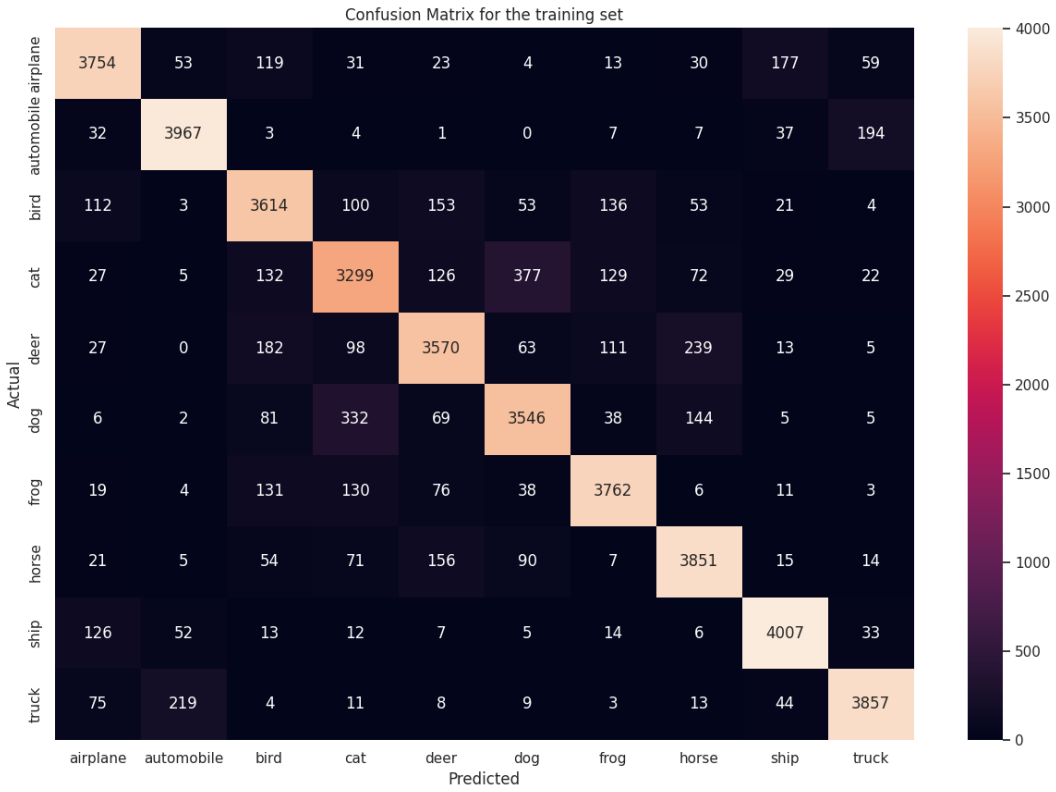
Accuracy score on training set: 0.8759294117647058 → %87

F1 score on training set: 0.875691218506928 → %87

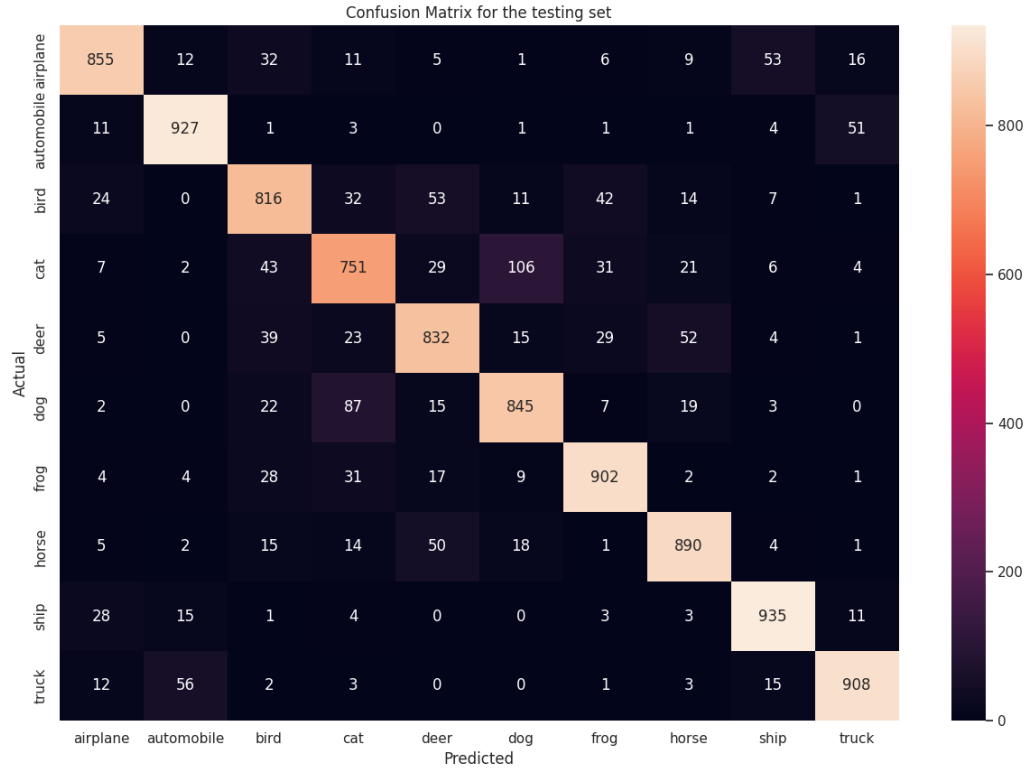
Accuracy score on testing set: 0.8661 → %86

F1 score on testing set: 0.8657603519981125 → %86

- ماتریس درهم‌ریختگی برای داده‌های آموزش و آزمایش به شکل زیر می‌باشد. مشاهده می‌شود در هر دو داده‌های آموزش و آزمایش مدل برای تشخیص سگ و گربه نسبت به سایر برچسب‌ها عملکرد ضعیف‌تری دارد و حدود ۷۰۰ داده را به اشتباه پیش‌بینی کرده است. علاوه بر این مورد، مدل در تشخیص اسب و آهو همچنین اتومبیل و تراکتور اشتباهات بیشتری داشته است.

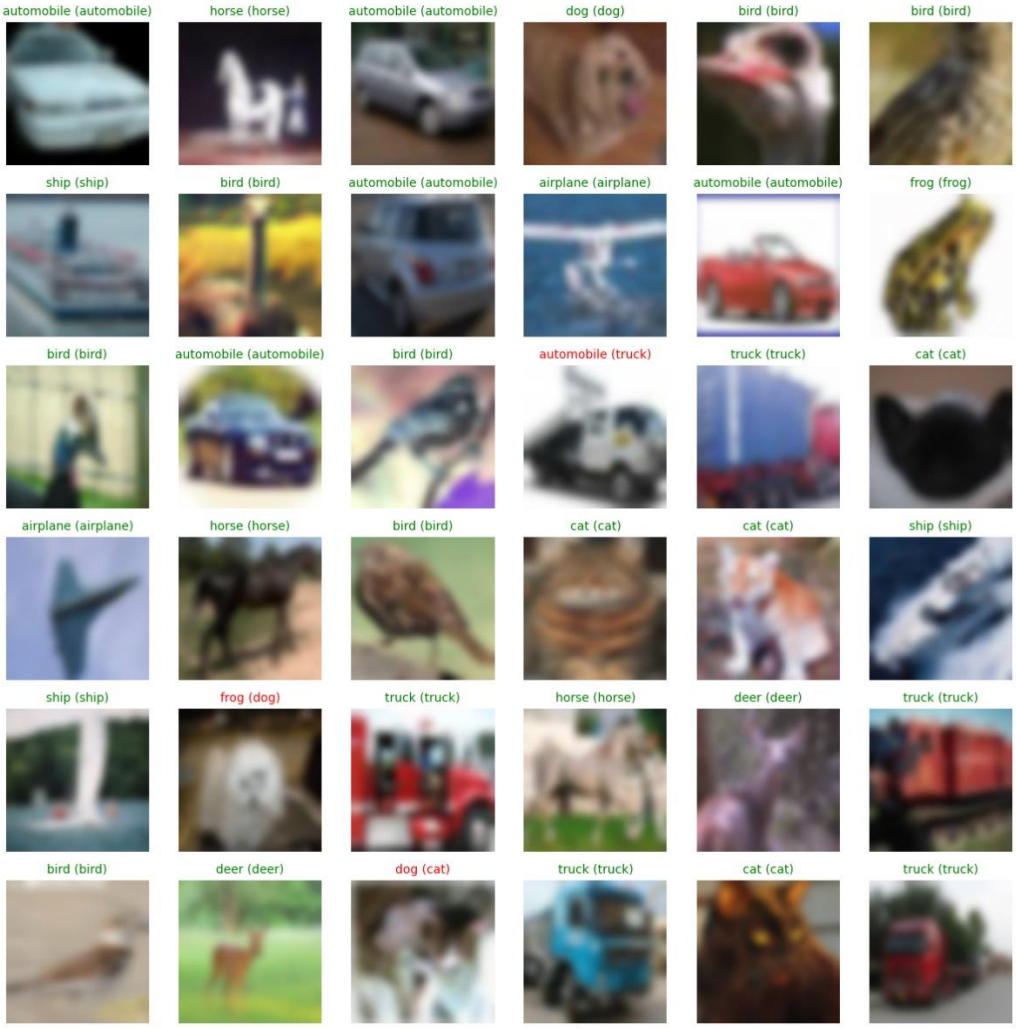


شکل ۲-۵ ماتریس درهم ریختگی برای داده‌های آموزش



شکل ۲-۶ ماتریس درهم ریختگی برای داده‌های آزمایش

- ۳۶ نمونه تصادفی از تصاویر برچسب‌گذاری شده به شکل زیر می‌باشند. برچسب واقعی عبارت داخل پرانتز و برچسب پیش‌بینی شده عبارت خارج پرانتز است. مشاهده می‌شود تعداد کمی از داده‌ها به اشتباه دسته‌بندی شده‌اند.

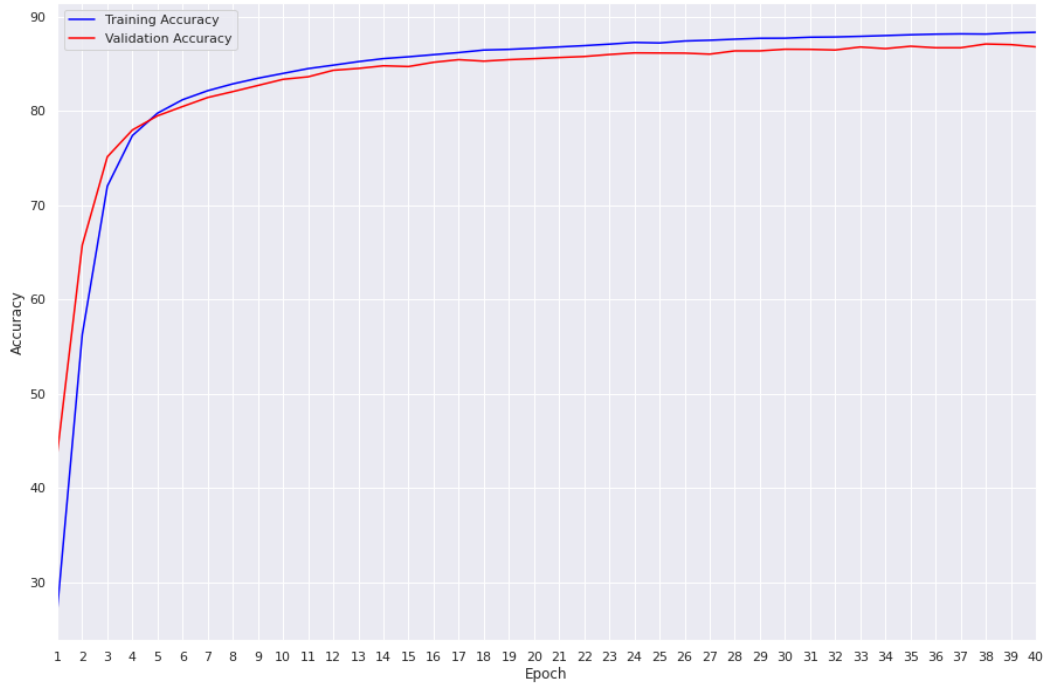


شکل ۲-۷ نمونه تصاویر برچسب زده شده

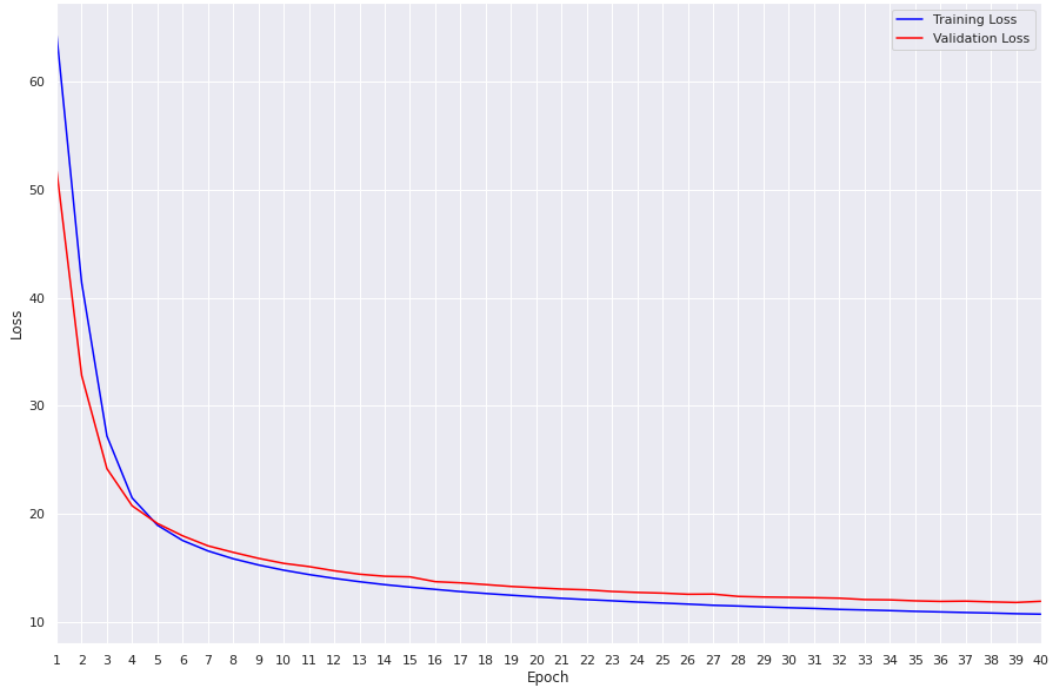
۲-۱-۳-۲ اندازه دسته ۳۲، تعداد دور ۴۰ و نرخ یادگیری ۰/۰۰۱

- برای پارامترهایی که در عنوان ذکر شده‌اند، به ترتیب نمودار صحت و هزینه برای داده‌های آموزش و اعتبارسنجی هنگام آموزش مدل به شکل زیر می‌باشد. مشاهده می‌شود صحت برای داده‌های اعتبارسنجی همراه صحت برای داده‌های آموزش افزایش یافته و به همگرایی رسیده‌اند؛ برای همین اطمینان حاصل می‌شود که بیش‌برازش رخ نداده است. همچنین هزینه

برای داده‌های اعتبارسنجی نیز همراه هزینه برای داده‌های آموزش کاهش یافته و به همگرایی رسیده‌اند؛ بنابراین مجددا می‌توان نتیجه گرفت بیش‌برازشی رخ نداده‌است.



شکل ۲-۸ نمودار صحت برای داده‌های آموزش و اعتبارسنجی



شکل ۲-۹ نمودار هزینه برای داده‌های آموزش و اعتبارسنجی

- معیار صحت و امتیاز F1 برای داده‌های آموزش و آزمایش به شرح زیر می‌باشد. (دقیقا خروجی‌های حاصل از اجرای کد قرار داده شده‌اند.)

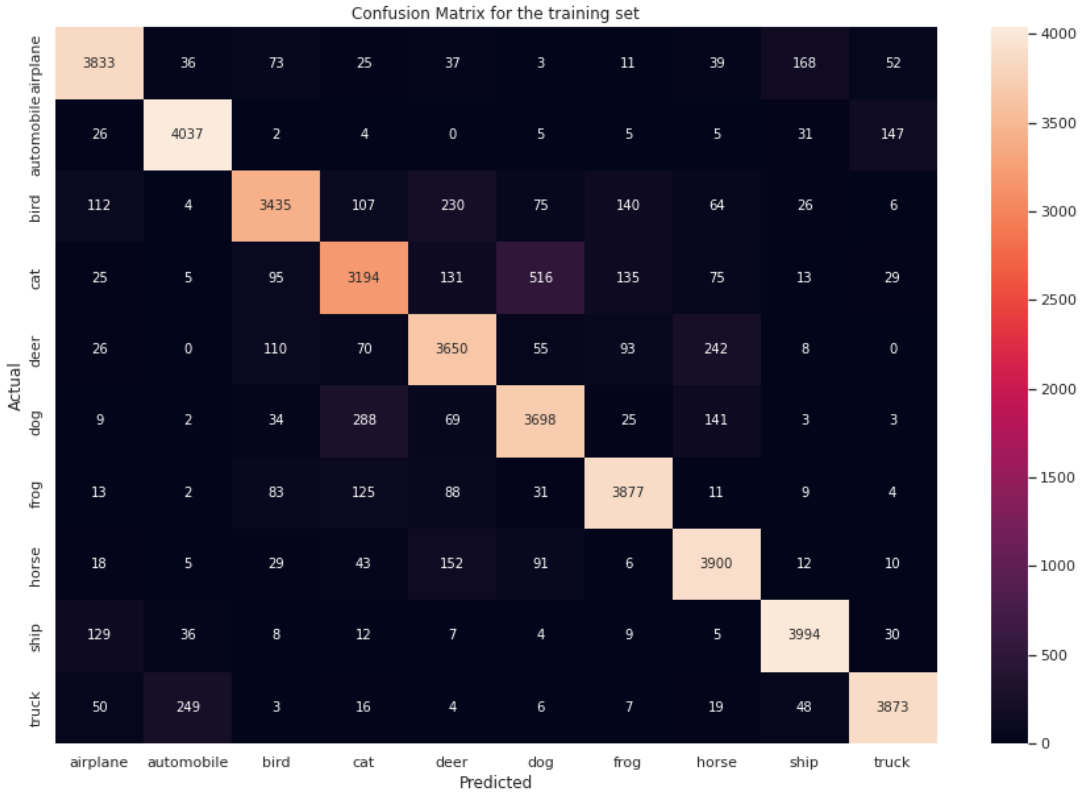
Accuracy score on training set: 0.8821411764705882 → %88

F1 score on training set: 0.8816964685081791 → %88

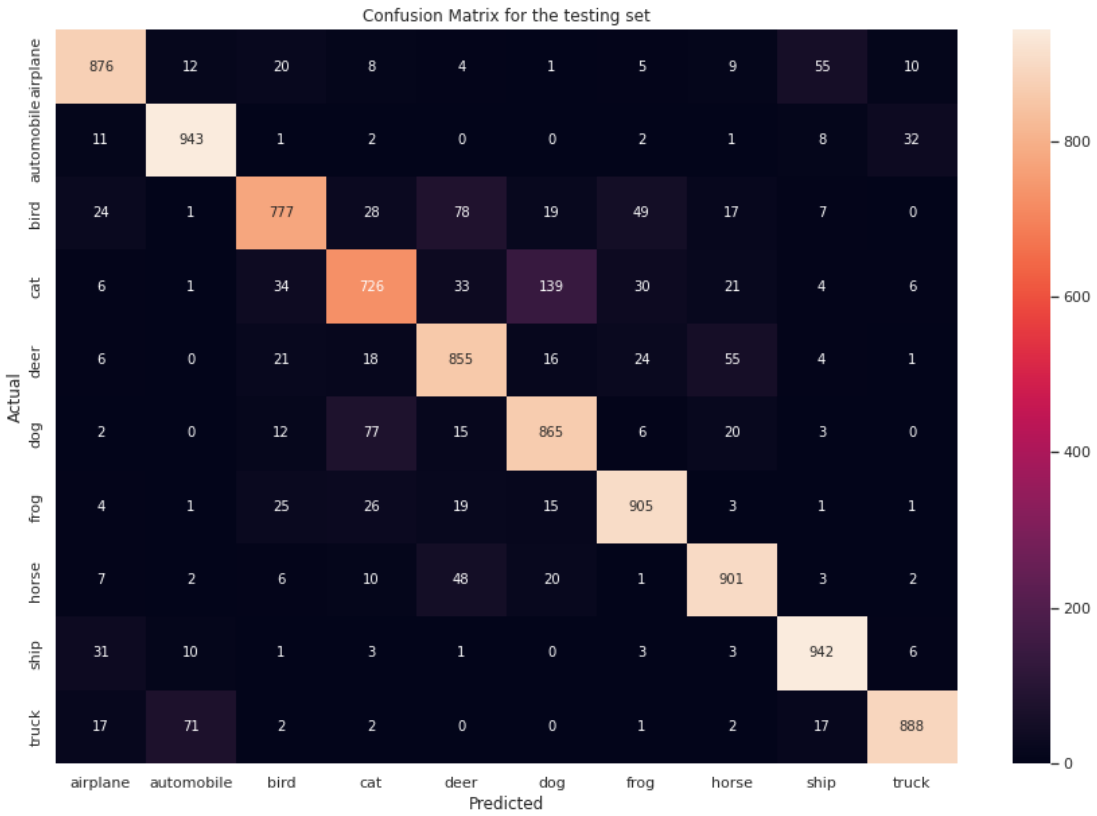
Accuracy score on testing set: 0.8678 → %86

F1 score on testing set: 0.867088010968589 → %86

- ماتریس درهم‌ریختگی برای داده‌های آموزش و آزمایش به شکل زیر می‌باشد. مشاهده می‌شود در هر دو داده‌های آموزش و آزمایش همچنان مدل برای تشخیص سگ و گربه نسبت به سایر برچسب‌ها عملکرد ضعیف‌تری دارد و حدود ۸۰۰ داده را به اشتباه پیش‌بینی کرده است. علاوه بر این مورد، مدل در تشخیص اسب و آهو به همراه آهو و پرنده، همچنین اتومبیل و تراکتور اشتباهات بیشتری داشته است. با توجه به اینکه وزن‌دهی داده‌ها تصادفی است این امکان وجود دارد که حتی در این تعداد دور نیز مدل عملکرد بدتری روی این کلاس‌ها داشته باشد، اما عملکرد مدل در مجموع نسبت به حالت قبل بهتر شده است.

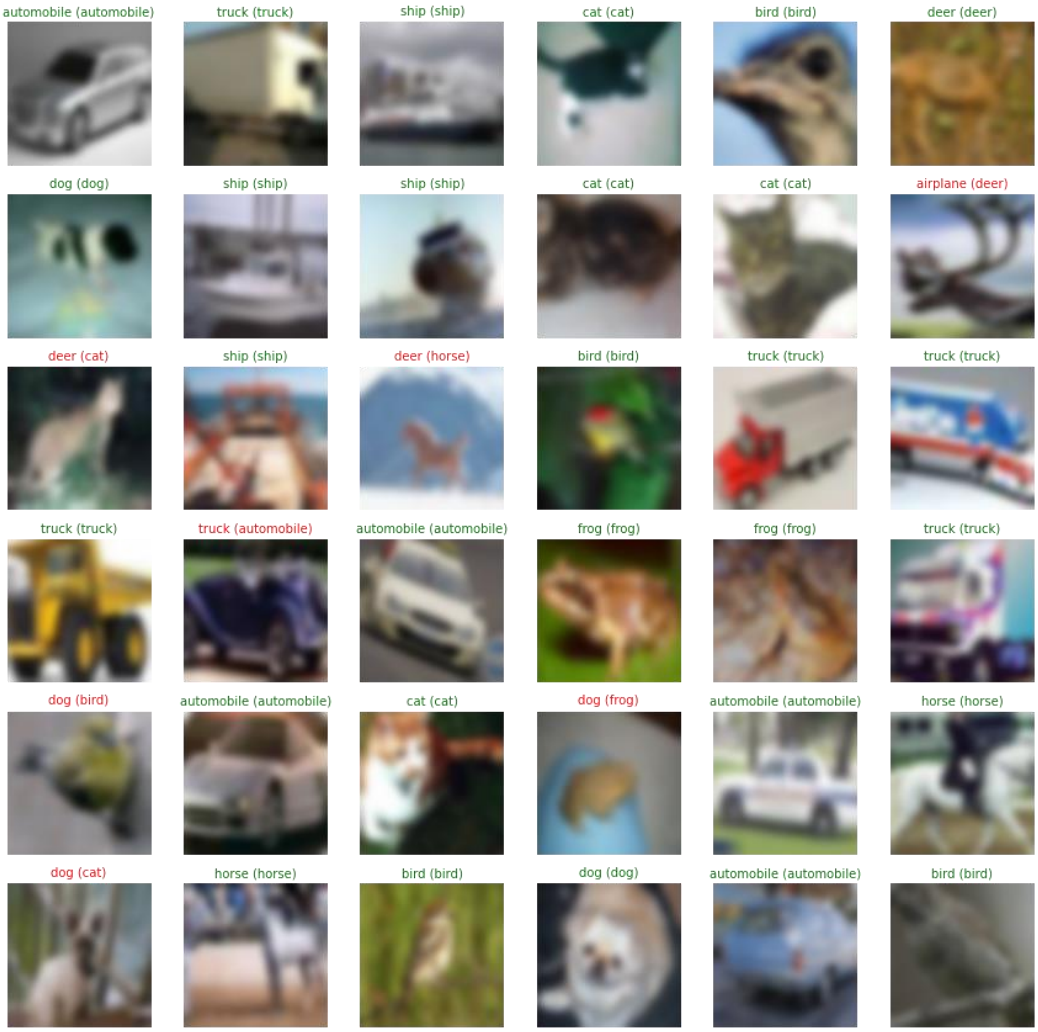


شکل ۲-۱۰ ماتریس درهم ریختگی برای داده‌های آموزش



شکل ۲-۱۱ ماتریس درهم ریختگی برای داده‌های آزمایش

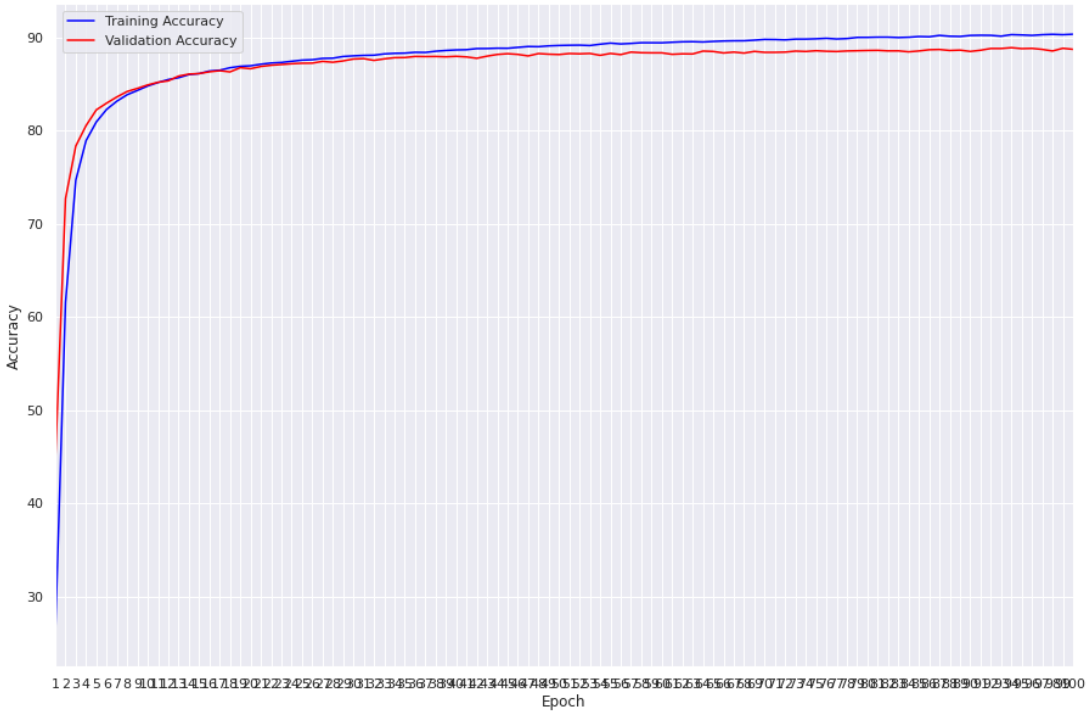
- ۳۶ نمونه تصادفی از تصاویر برچسب‌گذاری شده به شکل زیر می‌باشند. برچسب واقعی عبارت داخل پرانتز و برچسب پیش‌بینی شده عبارت خارج پرانتز است. مشاهده می‌شود مدل روی دسته‌های ذکر شده در قسمت قبل عملکرد ضعیف‌تری دارد و نتوانسته دسته‌ها را به درستی پیش‌بینی کند.



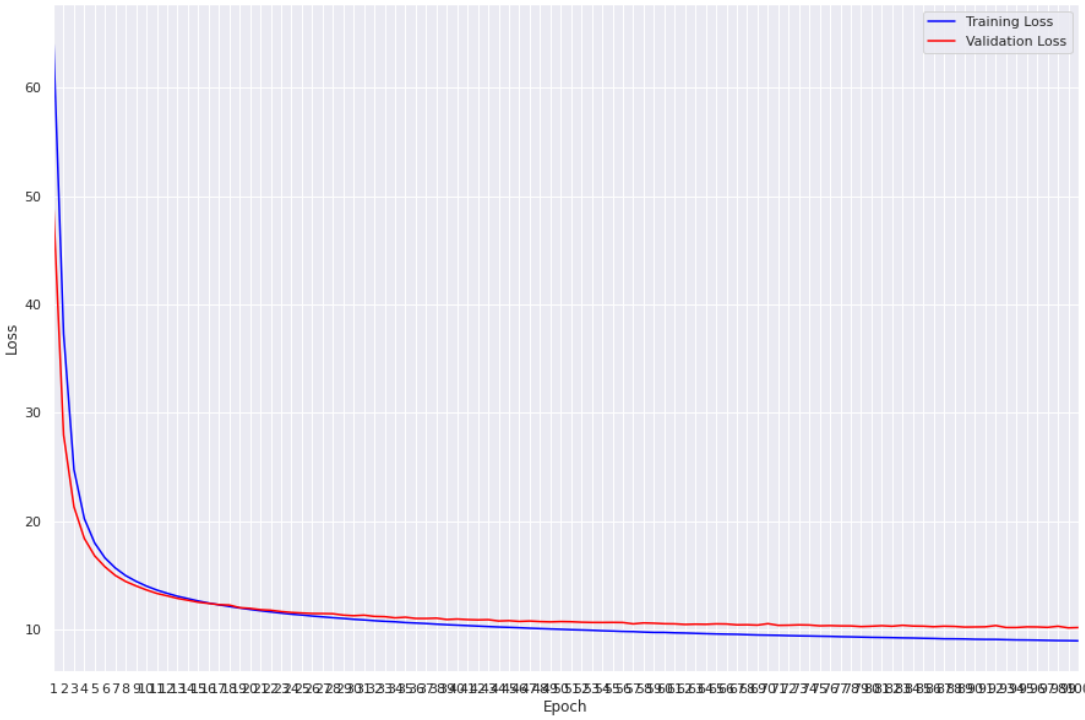
شکل ۲-۱۲ نمونه تصاویر برچسب زده شده

۲-۱-۳-۳ اندازه دسته ۳۲، تعداد دور ۱۰۰ و نرخ یادگیری ۰/۰۰۱

- برای پارامترهایی که در عنوان ذکر شده‌اند، به ترتیب نمودار صحت و هزینه برای داده‌های آموزش و اعتبارسنجی هنگام آموزش مدل به شکل زیر می‌باشد. مشاهده می‌شود صحت برای داده‌های اعتبارسنجی همراه صحت برای داده‌های آموزش افزایش یافته و به همگرایی رسیده‌اند؛ برای همین اطمینان حاصل می‌شود که بیش‌برازش رخ نداده است. همچنین هزینه برای داده‌های اعتبارسنجی نیز همراه هزینه برای داده‌های آموزش کاهش یافته و به همگرایی رسیده‌اند؛ بنابراین مجددا می‌توان نتیجه گرفت بیش‌برازشی رخ نداده‌است.



شکل ۲-۱۳ نمودار صحت برای داده‌های آموزش و اعتبارسنجی



شکل ۲-۱۴ نمودار هزینه برای داده‌های آموزش و اعتبارسنجی

- معیار صحت و امتیاز F1 برای داده‌های آموزش و آزمایش به شرح زیر می‌باشد. (دقیقا خروجی‌های حاصل از اجرای کد قرار داده شده‌اند.)

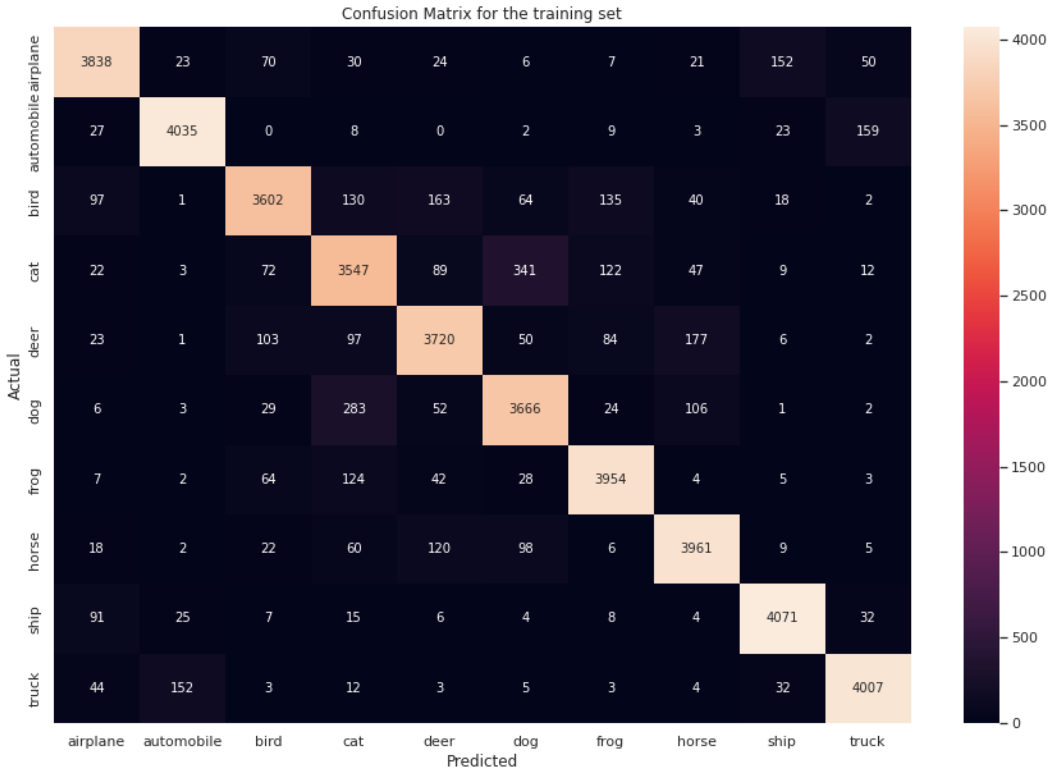
Accuracy score on training set: 0.9035529411764706 → %90

F1 score on training set: 0.9034699105892284 → %90

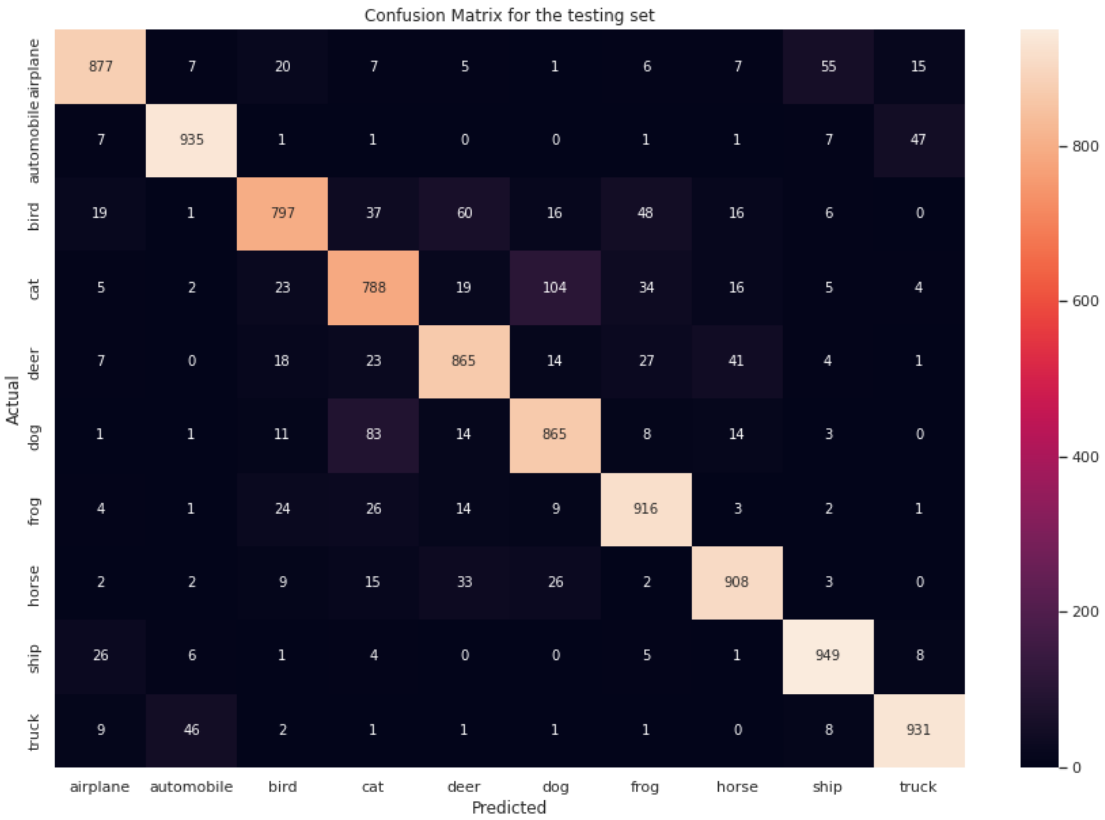
Accuracy score on testing set: 0.8831 → %88

F1 score on testing set: 0.8827635523034382 → %88

- ماتریس درهم‌ریختگی برای داده‌های آموزش و آزمایش به شکل زیر می‌باشد. مشاهده می‌شود در هر دو داده‌های آموزش و آزمایش همچنان مدل برای تشخیص سگ و گربه نسبت به سایر برچسب‌ها عملکرد ضعیف‌تری دارد و حدود ۶۰۰ داده را به اشتباه پیش‌بینی کرده است. علاوه بر این مورد، مدل در تشخیص اسب و آهو به همراه آهو و پرنده، همچنین اتومبیل و تراکتور اشتباهات بیشتری داشته است. با توجه به اینکه وزن‌دهی داده‌ها تصادفی است این امکان وجود دارد که حتی در این تعداد دور نیز مدل عملکرد بدتری روی این کلاس‌ها داشته باشد، اما عملکرد مدل در مجموع نسبت به حالت‌های قبل بهتر شده است.

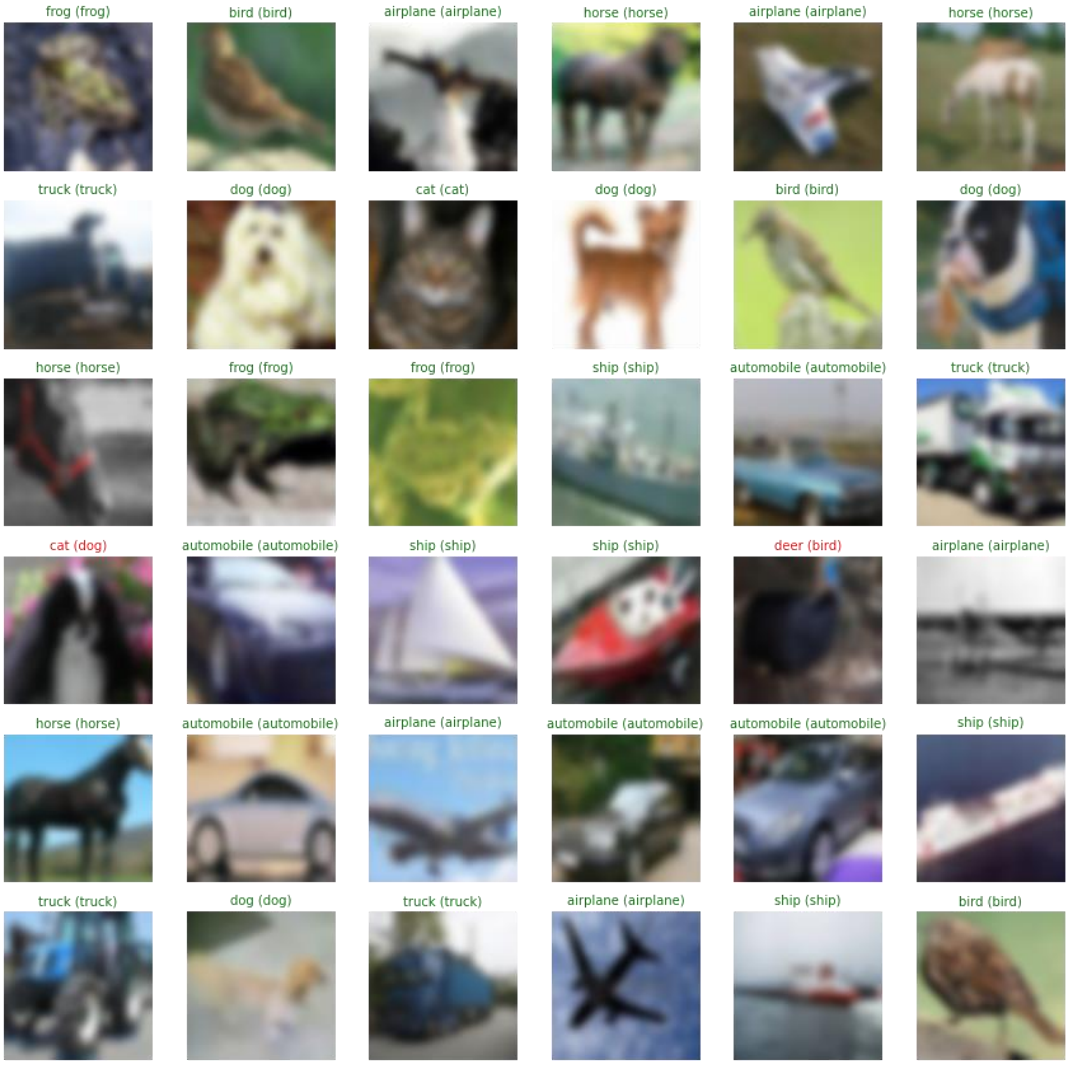


شکل ۲-۱۵ ماتریس درهم ریختگی برای داده‌های آموزش



شکل ۲-۱۶ ماتریس درهم ریختگی برای داده‌های آزمایش

- ۳۶ نمونه تصادفی از تصاویر برچسب‌گذاری شده به شکل زیر می‌باشند. برچسب واقعی عبارت داخل پرانتز و برچسب پیش‌بینی شده عبارت خارج پرانتز است. مشاهده می‌شود عملکرد مدل بسیار بهتر از دو حالت قبل است، هرچند ممکن است در نمونه‌های دسته‌بندی شده دیگر تعدادی دسته‌بندی اشتباه موجود باشد.



شکل ۲-۱۷ نمونه تصاویر برچسب زده شده

۲-۲- قسمت ب (اختیاری)

در قسمت ب این بخش از ما خواسته شده است از یک شبکه عصبی MLP سه لایه (یک لایه مخفی و یک لایه خروجی) به عنوان دسته‌بند استفاده کنیم. تعداد نورون‌های لایه مخفی را ۲۰ در نظر گرفته و برای آن‌ها از تابع فعالسازی ReLU استفاده می‌کنیم. این شبکه را به وسیله الگوریتم تکامل عصبی برای ۲۰ دور بر روی قسمت آموزشی مجموعه داده CIFAR10 آموزش داده و وزن‌های دسته‌بند را بروزرسانی می‌کنیم. سپس معیارهای صحت، ماتریس درهم‌ریختگی و امتیاز F1 مدل آموزش دیده را هم روی مجموعه آموزشی و هم مجموعه آزمایشی محاسبه می‌کنیم.

۲-۲-۱ روش انجام آزمایش در قسمت ب

پس از خواندن داده‌ها و اعمال مبدل، داده‌های آموزشی، آزمایش و اعتبارسنجی (۱۵ درصد داده‌های آموزشی) را جدا می‌کنیم. سپس، کلاس‌های لایه‌ها، توابع فعال‌سازی، هزینه، بهینه‌ساز، استخراج‌کننده‌های ویژگی، مدل شبکه عصبی، کروموزوم پارامترهای شبکه عصبی و الگوریتم تکاملی را تعریف می‌کنیم (هر یک جداگانه شرح داده شده‌اند و قسمت‌های جدید شرح داده خواهند شد.). در مرحله بعدی، یک شی از کلاس الگوریتم‌های با تنظیم پارامترهای population_size، max_generation، p_mutation و p_crossover ساخته و با صدا کردن متد run از این مدل به تعداد بیشینه نسل‌های مورد نظر عملیات ساخت جمیت نسل بعد، پیدا کردن بهترین کروموزم هر نسل و بهترین کروموزوم کل نسل‌ها را انجام می‌دهیم.

۲-۲-۲ توضیح اجمالی کد در قسمت ب

۲-۲-۲-۱ کلاس Chromosome_AK

در این کلاس کروموزوم‌ها برای الگوریتم تکاملی تعریف شده‌اند که ژن‌ها در واقع لیستی از لایه‌های شبکه عصبی‌اند که باید وزن‌ها و بایاس آن‌ها را به گونه‌ای تغییر داد که مقدار تابع برازندگی را بیشینه کنند. متدهای این کلاس به شرح زیر است.

- **متد __init__:** در این متد برازندگی در ابتدا برابر None قرار داده شده و ژن‌ها به صورت لیستی از دولایه (پنهان و خروجی) با تعداد ورودی و نورون‌های مشخص شده تعریف می‌شوند.
- **متد __repr__:** از این متد برای نمایش برازندگی هر کروموزوم استفاده می‌شود.
- **متد mutate:** در این متد عملیات جهش گاوسی به این صورت انجام می‌شود که بر اساس توضیح نرمال با میانگین صفر و انحراف معیار وزن‌ها و بایاس‌ها به صورت جداگانه مقداری را به آن‌ها اضافه می‌کند.
- **متد recombinate:** این متد با دریافت دو والد، دو فرزند حاصل از تقطیع چند نقطه‌ای تولید می‌کند. به این صورت که سطرهای ماتریس وزن‌های هر لایه فرزندان با احتمالی سطر ماتریس وزن‌های هر لایه والد اول و با احتمالی و همین مقدار را از والد دوم به ارث می‌برند.
- **متد calculate_fitness:** در این متد یک شی از مدل شبکه عصبی با استفاده از لایه‌های کروموزوم ساخته شده و معیار صحت و امتیاز F1 این شبکه با استفاده از متد evaluate مدل به دست می‌آید. در نهایت برازندگی کروموزوم را برابر با معیار صحت شبکه حاصل قرار می‌دهیم.

۲-۲-۲-۲ کلاس EvolutionaryAlgorithm_AK

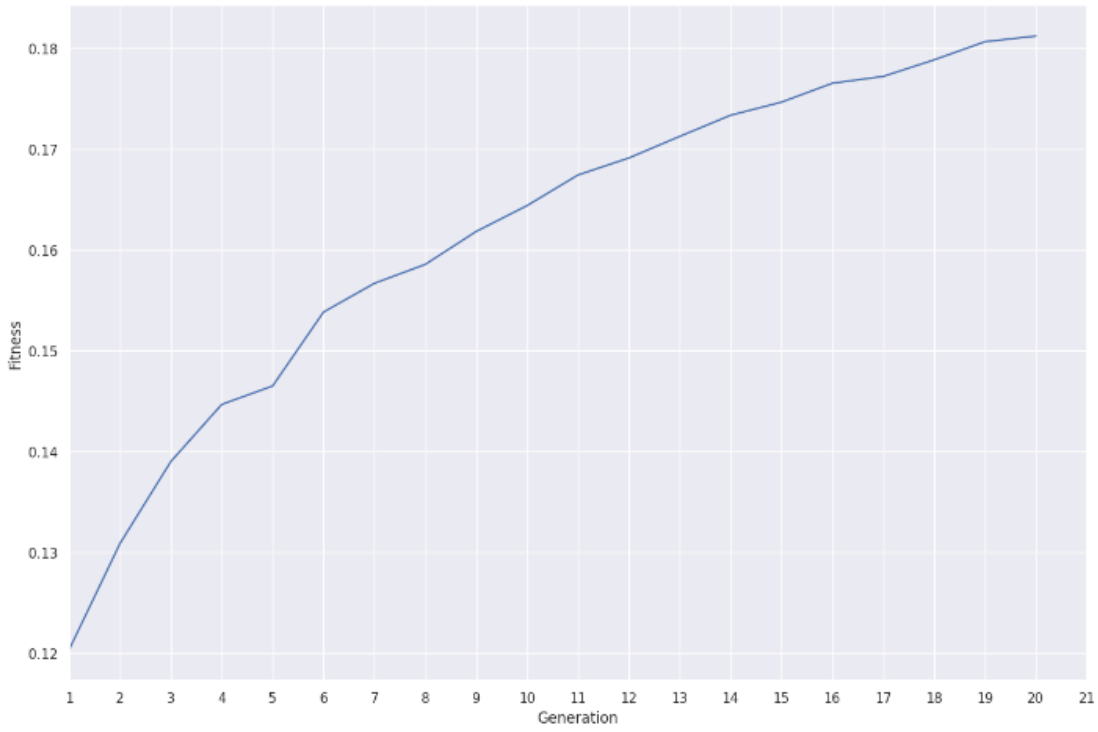
در این کلاس عملیات‌های لازم برای الگوریتم تکاملی مانند تولید جمعیت اولیه، جهش، بازترکیب، انتخاب، تولید نسل بعدی و انتخاب بهترین کروموزوم انجام می‌شود. متدهای این کلاس به شرح زیر است.

- **متد __init__:** این متد مقداردهی‌های اولیه که در پارامتر ساخت شئ داده شده‌اند را انجام داده و پس از آن لیست‌های جمعیت، فرزندان، تاریخچه برازندگی و بهترین را تعریف می‌کنیم.
- **متد initialize_population:** در این متد ابتدا لیست جمعیت را خالی کرده و پس از آن با استفاده از یک حلقه به طول اندازه جمعیت، کروموزم ساخته و آن‌ها را به لیست جمعیت اضافه می‌کنیم. و بهترین نسل اول را ذخیره می‌کنیم.
- **متد mutation:** در این متد با احتمال جهشی که داریم، متد جهش از شئ کروموزوم صدا زده شده و عملیات جهش انجام می‌شود.
- **متد recombination:** این متد با دریافت دو والد با احتمال بازترکیبی که داریم، متد بازترکیب از شئ کروموزوم صدا زده شده و عملیات بازترکیب انجام می‌شود. در نهایت فرزندان تولید شده به لیست فرزندان اضافه می‌شوند.
- **متد tournament_selection:** در این متد انتخاب با استفاده از روش tournament انجام شده و از سه کروموزومی که با جایگذاری به صورت رندوم انتخاب می‌شوند، کروموزومی که بیشترین برازندگی را دارد به عنوان والد انتخاب می‌شود.
- **متد generate_next_generation:** در این متد عملیات ساخت یک نسل انجام می‌شود. ابتدا لیست فرزندان خالی شده، سپس با استفاده از روش انتخاب دو والد انتخاب کرده و روی آن‌ها عملیات بازترکیب را انجام می‌دهیم. پس از آن، عملیات جهش را انجام داده و برازندگی را برای تمامی فرزندان محاسبه می‌کنیم. در نهایت جمعیت فرزندان را به جمعیتی که داشتیم اضافه کرده و به تعداد population_size بهترین آن‌ها را برای جمعیت نسل بعد ذخیره می‌کنیم.
- **متد run:** در این متد جمعیت اولیه تشکیل شده و به تعداد حداکثر نسل‌ها عملیات تولید جمعیت نسل بعد، اضافه کردن میانگین برازندگی هر نسل به لیست تاریچه برازندگی‌ها، پیدا کردن بهترین کروموزوم هر نسل و پیدا کردن بهترین کروموزوم تمامی نسل‌ها انجام می‌شود.

۲-۲-۳ تحلیل نتایج قسمت ب

۲-۲-۳-۱ اندازه جمعیت ۵۰، حداکثر نسل‌ها ۲۰، احتمال جهش ۰/۹، احتمال بازترکیب ۰/۱

- نمودار میانگین معیار صحت در مجموعه آزمایش در هر نسل به شکل زیر می‌باشد. مشاهده می‌شود در ۲۰ نسل الگوریتم به همگرایی نرسیده و در تعداد نسل بیشتر می‌تواند بهبود یابد.



شکل ۲-۱۸ نمودار میانگین صحت (برازندگی) در هر نسل

- معیار صحت و امتیاز F1 برای داده‌های آموزش و آزمایش به شرح زیر می‌باشد. (دقیقا خروجی‌های حاصل از اجرای کد قرار داده شده‌اند.)

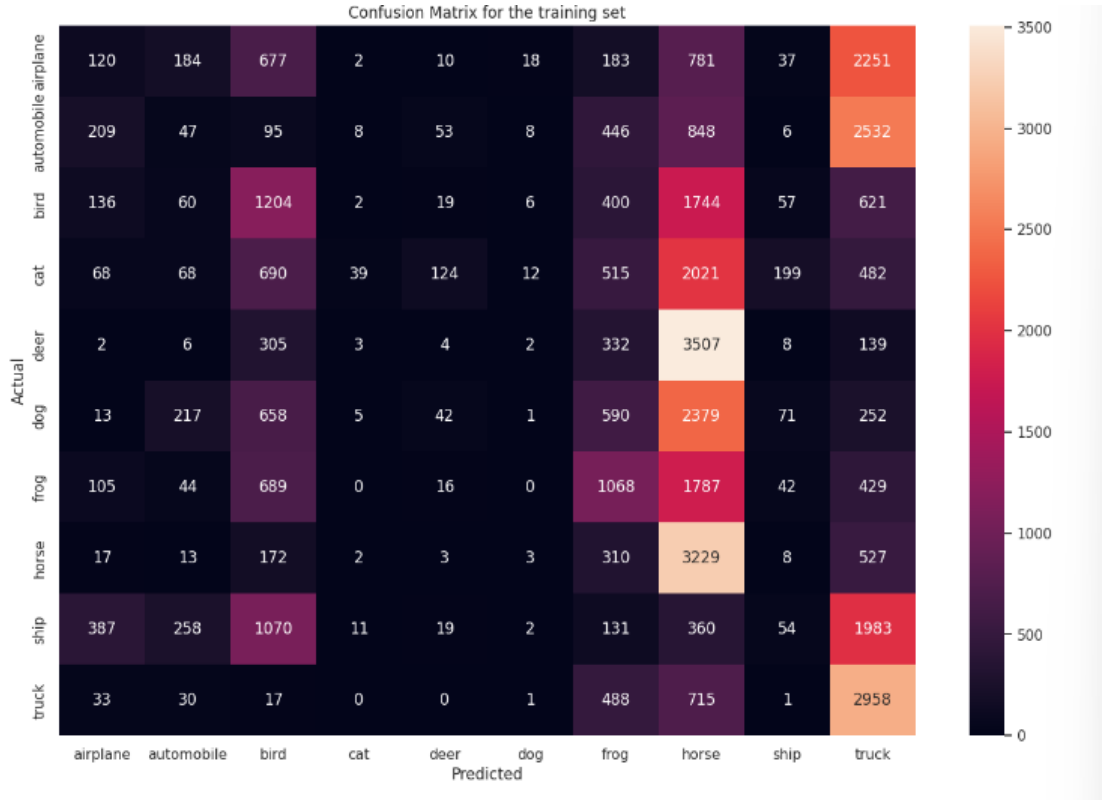
  Accuracy score on training set: 0.20527058823529412 → %20

  F1 score on training set: 0.1254680937346548 → %12

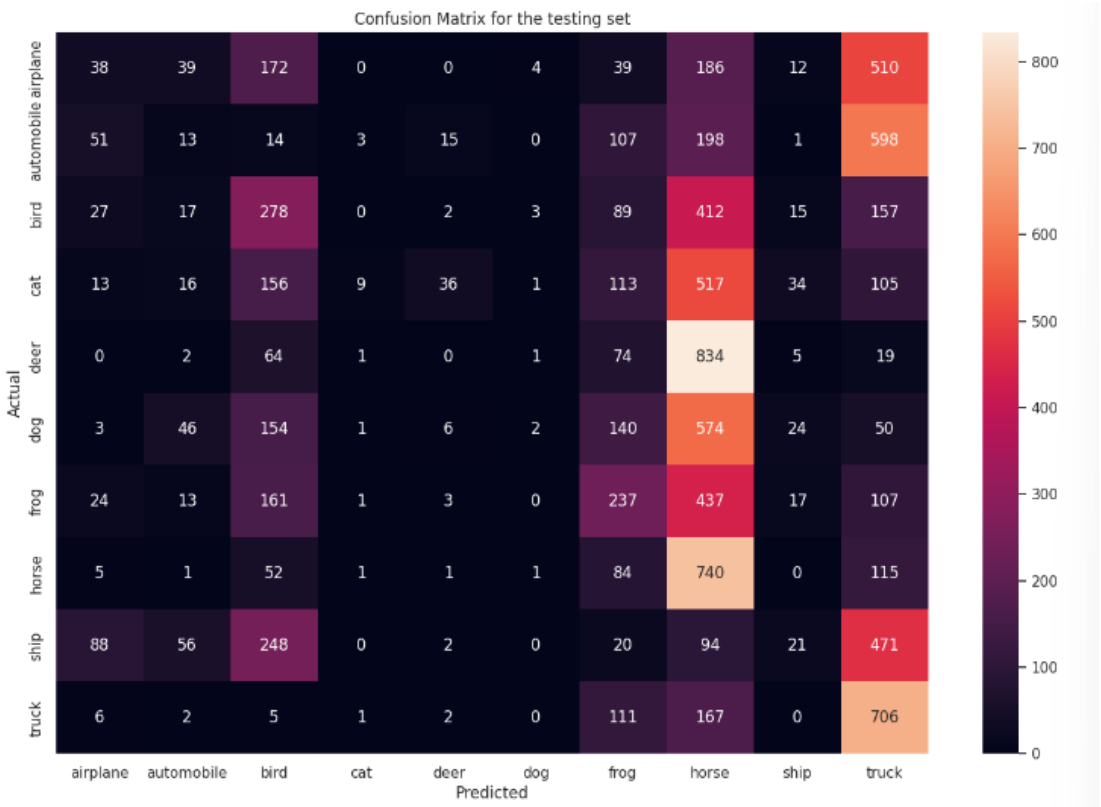
  Accuracy score on testing set: 0.2044 → %20

  F1 score on testing set: 0.1272436505922226 → %12

- ماتریس درهم‌ریختگی برای داده‌های آموزش و آزمایش به شکل زیر می‌باشد. مشاهده می‌شود مدل به درستی عمل نکرده و تنها برای تشخیص پرنده، غورباقه، اسب و تراکتور عملکرد بهتری دارد.

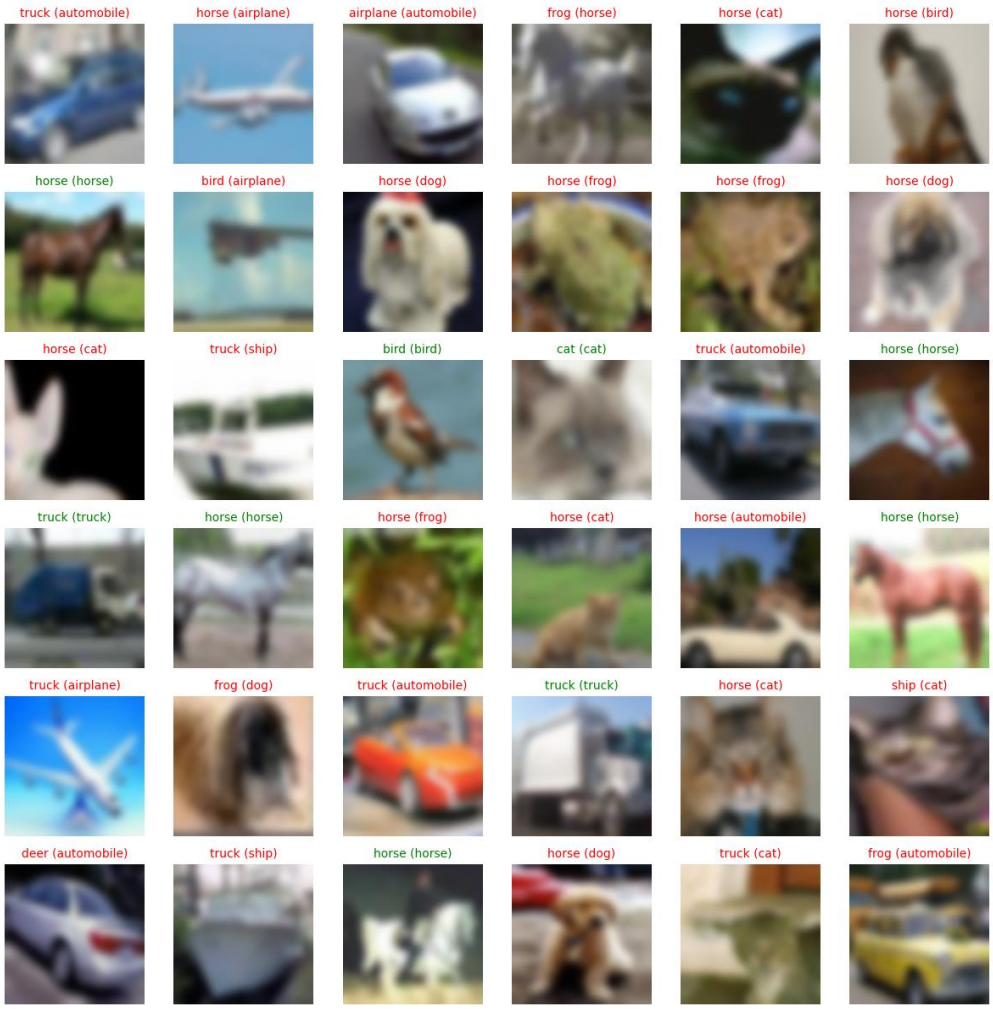


شکل ۲-۱۹ ماتریس درهم ریختگی برای داده‌های آموزش



شکل ۲-۲۰ ماتریس درهم ریختگی برای داده‌های آزمایش

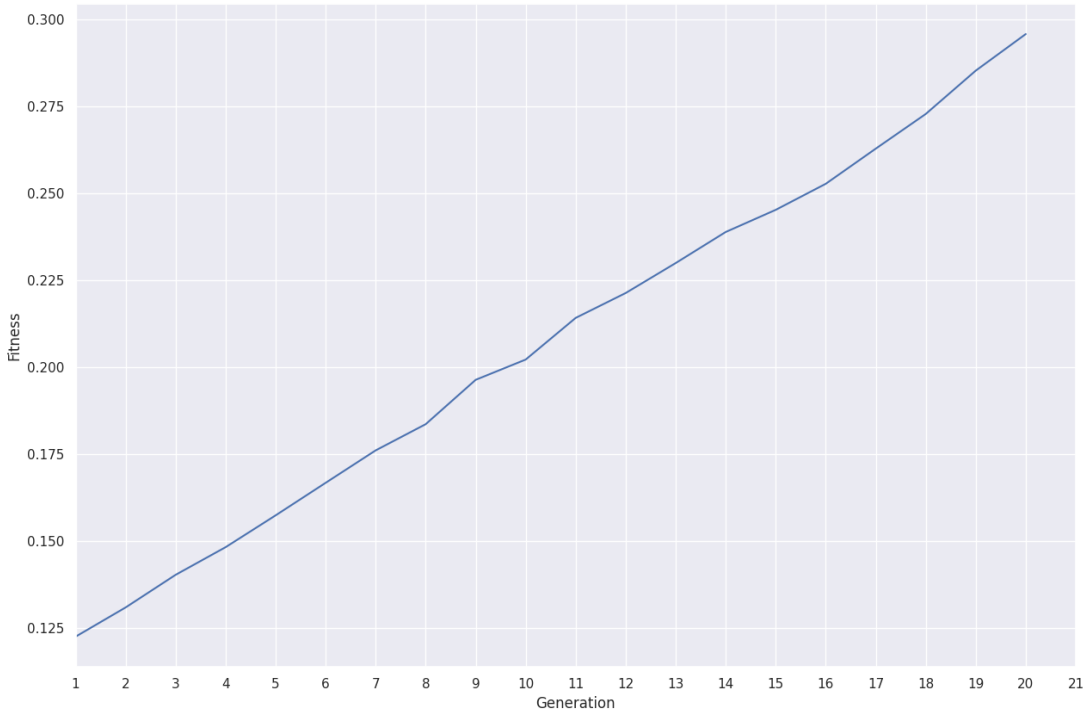
- ۳۶ نمونه تصادفی از تصاویر برچسب‌گذاری شده به شکل زیر می‌باشند. برچسب واقعی عبارت داخل پرانتز و برچسب پیش‌بینی شده عبارت خارج پرانتز است. مشاهده می‌شود دقت مدل بسیار پایین و تعداد برچسب‌هایی که به اشتباه پیش‌بینی شده‌اند، بیشتر است.



شکل ۲-۲۱ نمونه تصاویر برچسب زده شده

۲-۲-۳-۲ اندازه جمعیت ۵۰، حداکثر نسل‌ها ۲۰، احتمال جهش ۰/۱، احتمال بازترکیب ۰/۹

- نمودار میانگین معیار صحت در مجموعه آزمایش در هر نسل به شکل زیر می‌باشد. مشاهده می‌شود در ۲۰ نسل الگوریتم به همگرایی نرسیده و در تعداد نسل بیشتر می‌تواند بهبود یابد. از آن‌جایی که دقت مدل با استفاده از روش پس انتشار خطا نزدیک به ۹۰ درصد می‌رسید، برای رسیدن به چنین دقتی با استفاده از الگوریتم تکامل عصبی به تعداد نسل‌های بسیار زیاد و طبیعتا زمان بسیار بیشتری نیاز است.



شکل ۲-۲۲ نمودار میانگین صحت (برازندگی) در هر نسل

- معیار صحت و امتیاز F1 برای داده‌های آموزش و آزمایش به شرح زیر می‌باشد. (دقیقا خروجی‌های حاصل از اجرای کد قرار داده شده‌اند.)

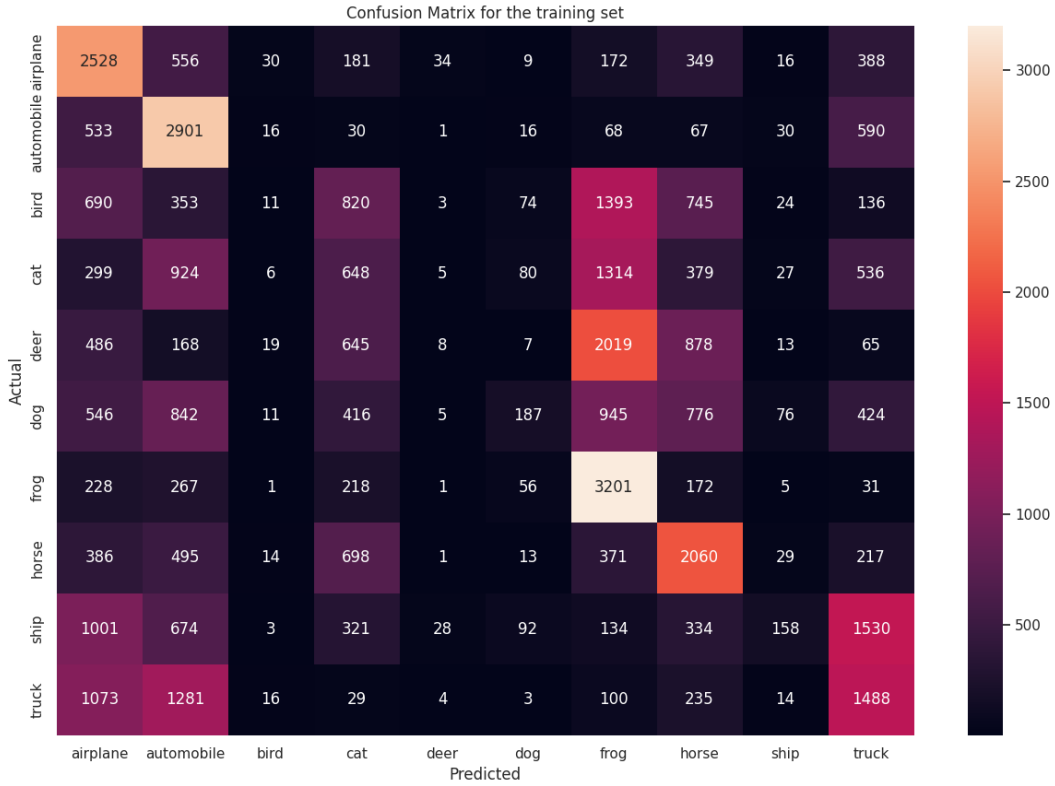
Accuracy score on training set: 0.3103529411764706 → %31

F1 score on training set: 0.2354427210244278 → %23

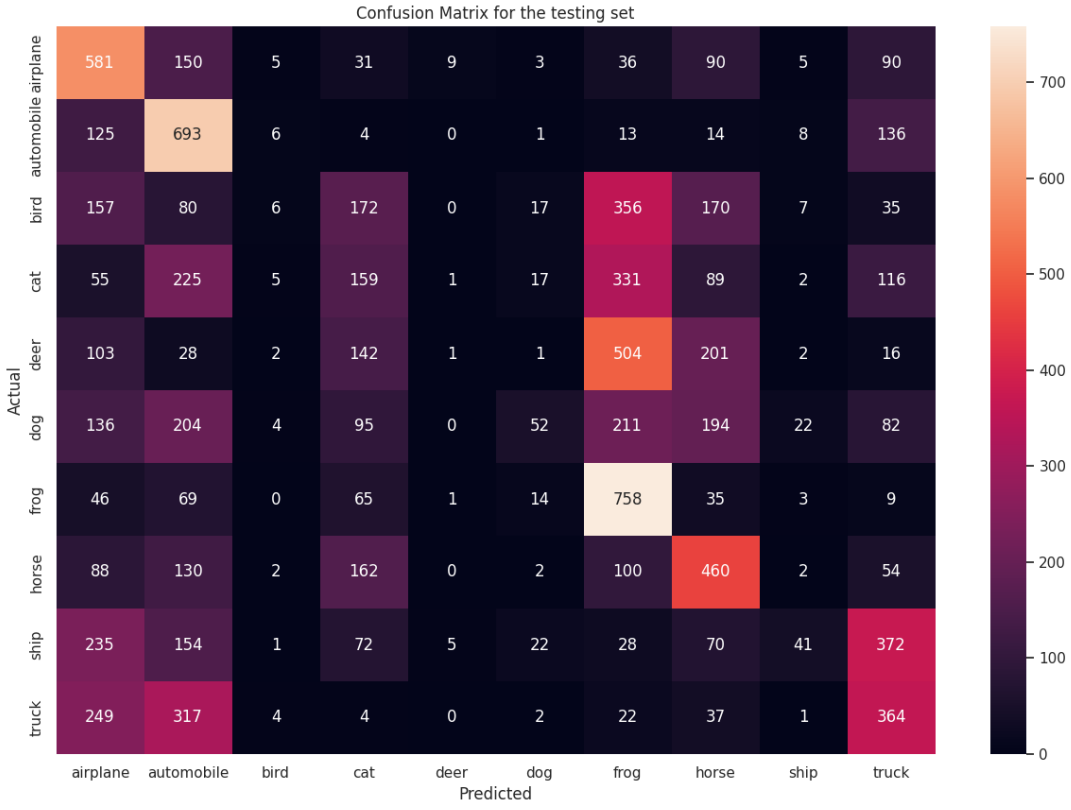
Accuracy score on testing set: 0.3115 → %31

F1 score on testing set: 0.23818402170079211 → %23

- ماتریس درهم‌ریختگی برای داده‌های آموزش و آزمایش به شکل زیر می‌باشد. مشاهده می‌شود مدل به طور کلی به درستی عمل نکرده ولی برای تشخیص هواپیما، اتومبیل، غورباقه، اسب و تراکتور عملکرد بهتری داشته است.

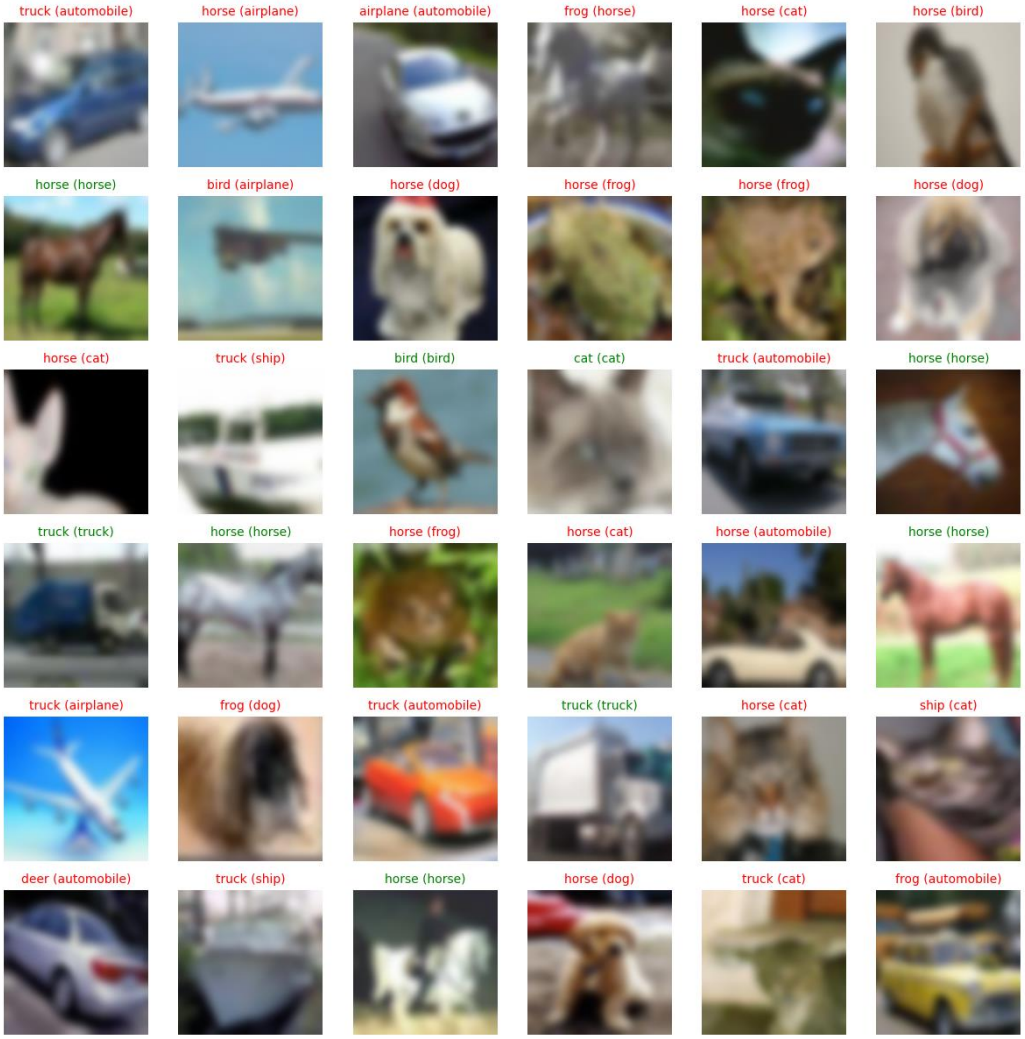


شکل ۲-۲۳ ماتریس درهم ریختگی برای داده‌های آموزش



شکل ۲-۲۴ ماتریس درهم ریختگی برای داده‌های آزمایش

- ۳۶ نمونه تصادفی از تصاویر برچسب‌گذاری شده به شکل زیر می‌باشند. برچسب واقعی عبارت داخل پرانتز و برچسب پیش‌بینی شده عبارت خارج پرانتز است. مشاهده می‌شود دقت مدل بسیار پایین اما بهتر از حالت قبل بوده و تعداد برچسب‌هایی که به اشتباه پیش‌بینی شده‌اند، نسبت به حالت قبل کمتراند.



شکل ۲-۲۵ نمونه تصاویر برچسب زده شده

۳- بخش دوم: جستجوی معماری عصبی برای شناسایی الگو

در این بخش باید با استفاده از الگوریتم‌های تکامل مقادیر مناسب پارامترهای که ذکر می‌شوند، را برای مسئله دسته‌بندی تصاویر CIFAR10 پیدا کرده و شبکه طراحی شده را باالگوریتم پس‌انتشار خطا آموزش دهیم.

در واقع الگوریتم پس‌انتظار خطا با جستجوی پارامترهای شبکه (وزن‌ها و بایاس‌ها) و الگوریتم تکامل عصبی با جستجوی ابرپارامترهای شبکه به صورت ترکیبی با یکدیگر عمل می‌کنند. در جدول زیر مقادیر ممکن برای ابرپارامترهای شبکه نشان داده شده‌است که باید در تکامل عصبی مدنظر قرار گیرند تا بهترین ترکیب ممکن این مقادیر با استفاده از الگوریتم‌های تکاملی به دست آید. معیار برازندگی در این مسئله صحت رو مجموعه آزمایش است و طراحی بهینه، معماری‌ای است که بیشترین صحت روی مجموعه آموزشی را داشته باشد.

ابرپارامتر	مقادیر ممکن
شبکه استخراج‌کننده ویژگی	ResNet18 – ResNet34 – Vgg11
تعداد لایه‌های مخفی MLP	۰ – ۱ – ۲
تعداد نورون‌ها در هر لایه مخفی	۱۰ – ۲۰ – ۳۰
تابع فعال‌سازی در هر لایه مخفی	ReLU - Sigmoid
اندازه دسته	عدد طبیعی توانی از ۲
نرخ یادگیری	عدد حقیقی

۳-۱- روش انجام آزمایش

پس از خواندن داده‌ها و اعمال مبدل، داده‌های آموزشی، آزمایش و اعتبارسنجی (۱۵ درصد داده‌های آموزشی) را جدا می‌کنیم. سپس، کلاس‌های لایه‌ها، توابع فعال‌سازی، هزینه، بهینه‌ساز، استخراج‌کننده‌های ویژگی، مدل شبکه عصبی، کروموزوم ابرپارامترهای شبکه عصبی و الگوریتم تکاملی را تعریف می‌کنیم (هر یک جداگانه شرح داده شده‌اند و قسمت‌های جدید شرح داده خواهند شد.). در مرحله بعدی، یک شی از کلاس الگوریتم‌های با تنظیم پارامترهای population_size، max_generation، p_mutation و p_crossover ساخته و با صدا کردن متد run از این مدل به تعداد بیشینه نسل‌های مورد نظر عملیات ساخت جمیت نسل بعد، پیدا کردن بهترین کروموزم هر نسل و بهترین کروموزوم کل نسل‌ها را انجام می‌دهیم.

۳-۲- توضیح اجمالی کد

۳-۲-۱ کلاس Chromosome

در این کلاس کروموزوم‌ها برای الگوریتم تکاملی تعریف شده‌اند که ژن‌ها در واقع ابرپارامترهای شبکه عصبی‌اند که باید آن‌ها را به گونه‌ای تغییر داد که مقدار تابع برازندگی (معیار صحت روی مجموعه آموزش) را بیشینه کنند. متدهای این کلاس به شرح زیر است.

- **متد __init__:** در این متد مقادیر استخراج‌کننده ویژگی، تعداد لایه‌های پنهان، نرخ یادگیری، اندازه دسته و برازندگی در ابتدا برابر None قرار داده شده و ژن‌ها به صورت لیستی از ویژگی‌های ذکر شده تعریف می‌شوند. همچنین لیست تعداد نورون‌های هر لایه پنهان و توابع فعال‌سازی تعریف می‌شوند.
- **متد __repr__:** از این متد برای نمایش ژن‌ها و برازندگی هر کروموزوم استفاده می‌شود.
- **متد mutate:** در این متد عملیات جهش برای تک‌تک ژن‌ها به صورت جداگانه انجام می‌شود. برای استخراج‌کننده ویژگی از بین سه استخراج‌کننده ویژگی مشخص شده یکی را تصادفی انتخاب می‌کنیم. برای تعداد لایه‌های پنهان نیز از بین سه مقدار ذکر شده یکی را تصادفی انتخاب می‌کنیم. برای جهش نرخ یادگیری با احتمالی آن را ضربدر سه یا تقسیم بر سه می‌کنیم. در مورد جهش اندازه دسته نیز با احتمال آن را ضربدر دو یا تقسیم بر دو می‌کنیم و در صورتی که این مقدار کمتر از یک شد، آن را به یک تغییر می‌دهیم. برای تعداد نورون‌های هر لایه، به صورت تصادفی از بین مقادیر داده شده یکی را انتخاب می‌کنیم و در نهایت، برای انتخاب تابع فعال‌ساز برای هر لایه با احتمالی تابع فعال‌ساز ReLU یا Sigmoid را انتخاب می‌کنیم.
- **متد recombinate:** این متد با دریافت دو والد، عملیات بازترکیب را برای ژن‌ها انجام می‌دهد. به این صورت که هر کدام استخراج‌کننده ویژگی، تعداد لایه‌های پنهان و اندازه دسته یک والد را به ارث می‌برند. در بازترکیب نرخ یادگیری، به دلیل اینکه نرخ یادگیری به سمت والد خاصی میل نکند، ابتدا یک ضریب آلفا انتخاب کرده و یک ترکیب خطی از دو نرخ یادگیری والدین را برای فرزندان قرار می‌دهیم. برای بازترکیب تعداد لایه‌های پنهان و تابع فعال‌ساز، در هر کروموزوم ابتدا لیست تعداد نورون‌های پنهان هر لایه و توابع فعال‌ساز از والدین را به صورت جداگانه با هم ترکیب کرده و پس از آن به صورت تصادفی و به تعداد لایه‌های پنهان خود کروموزوم، عددی را برای تعداد نورون‌های لایه پنهان و توابع فعال‌ساز برای لایه را انتخاب می‌کنیم.
- **متد build_model:** در این متد یک شئ از مدل شبکه عصبی مصنوعی ساخته شده و استخراج‌کننده ویژگی به همراه اندازه دسته به عنوان آرگومان‌های ابتدایی به آن داده می‌شوند. سپس داده‌های آموزش، اعتبارسنجی و آزمایش (که با استفاده از مبدل داده شده در قالب دستیاران آموزشی استخراج شده‌اند،) را جدا می‌کنیم. پس از آن، نرخ یادگیری بهینه‌ساز، تعداد

ویژگی‌های ورودی، توابع فعال‌ساز و لایه‌ها را تعریف و مقداردهی می‌کنیم. در نهایت لایه خروجی را به لایه‌های مدل شبکه عصبی اضافه می‌کنیم.

- **متد calculate_fitness:** در این متد به تعداد بیشینه دفعات اجرا یک شی از مدل شبکه عصبی ساخته شده و به تعداد بیشینه دورهای مد نظر آموزش داده شده و معیار صحت این شبکه با استفاده از متد evaluate مدل به دست می‌آیند. در نهایت برازندگی کروموزوم را برابر با میانگین معیارهای صحت شبکه حاصل از تعداد بیشینه دفعات اجرا قرار می‌دهیم و مدل ساخته شده را حذف می‌کنیم.

۳-۲-۲ کلاس EvolutionaryAlgorithm

در این کلاس عملیات‌های لازم برای الگوریتم تکاملی مانند تولید جمعیت اولیه، جهش، بازترکیب، انتخاب، تولید نسل بعدی و انتخاب بهترین کروموزوم انجام می‌شود. متدهای این کلاس به شرح زیر است.

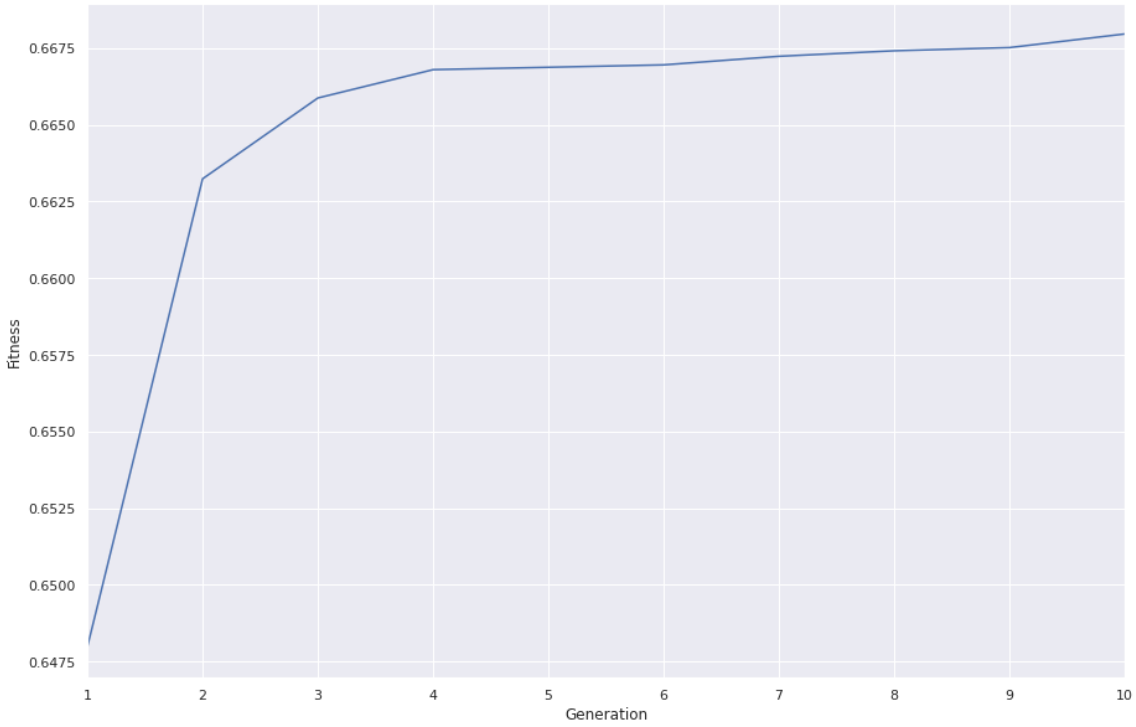
- **متد __init_:** این متد مقداردهی‌های اولیه که در پارامتر ساخت شئ داده شده‌اند را انجام داده و پس از آن لیست‌های جمعیت، فرزندان، تاریخچه برازندگی و بهترین را تعریف می‌کنیم.
- **متد initialize_population:** در این متد ابتدا لیست جمعیت را خالی کرده و پس از آن با استفاده از یک حلقه به طول اندازه جمعیت، کروموزم با استخراج‌کننده ویژگی، تعداد لایه‌های پنهان، نرخ یادگیری، اندازه دسته، تعداد نورون‌های لایه مخفی و توابع فعال‌سازی تصادفی با مقادیر مجاز ساخته و آن‌ها را به لیست جمعیت اضافه می‌کنیم. پس از محاسبه برازندگی تمامی کروموزوم‌ها، بهترین نسل اول را ذخیره می‌کنیم.
- **متد mutation:** در این متد با احتمال جهشی که داریم، متد جهش از شئ کروموزوم صدا زده شده و عملیات جهش انجام می‌شود.
- **متد recombination:** این متد با دریافت دو والد با احتمال بازترکیبی که داریم، متد بازترکیب از شئ کروموزوم صدا زده شده و عملیات بازترکیب انجام می‌شود. در نهایت فرزندان تولید شده به لیست فرزندان اضافه می‌شوند.
- **متد tournament_selection:** در این متد انتخاب با استفاده از روش tournament انجام شده و از پنج کروموزومی که با جایگذاری به صورت رندوم انتخاب می‌شوند، کروموزومی که بیشترین برازندگی را دارد به عنوان والد انتخاب می‌شود.

- **متد generate_next_generation:** در این متد عملیات ساخت یک نسل انجام می‌شود. ابتدا لیست فرزندان خالی شده، سپس با استفاده از روش انتخاب دو والد انتخاب کرده و روی آن‌ها عملیات بازترکیب را انجام می‌دهیم. پس از آن، عملیات جهش را انجام داده و برازندگی را برای تمامی فرزندان محاسبه می‌کنیم. در نهایت جمعیت فرزندان را به جمعیتی که داشتیم اضافه کرده و به تعداد population_size بهترین آن‌ها را برای جمعیت نسل بعد ذخیره می‌کنیم.
- **متد run:** در این متد جمعیت اولیه تشکیل شده و به تعداد حداکثر نسل‌ها عملیات تولید جمعیت نسل بعد، اضافه کردن میانگین برازندگی هر نسل به لیست تاریچه برازندگی‌ها، پیدا کردن بهترین کروموزوم هر نسل و پیدا کردن بهترین کروموزوم تمامی نسل‌ها انجام می‌شود.

۳-۳ تحلیل نتایج بخش دوم

۳-۳-۱ اندازه جمعیت ۱۰، حداکثر نسل‌ها ۱۰، احتمال جهش ۰/۳، احتمال بازترکیب ۰/۷

- نمودار برازندگی (میانگین معیار صحت در مجموعه آزمایش) در هر نسل به شکل زیر می‌باشد. مشاهده می‌شود الگوریتم تا حدودی به همگرایی رسیده اما همچنان در تعداد نسل‌های بیشتر می‌تواند امکان بهبود داشته باشد.



شکل ۳-۱ نمودار میانگین صحت (برازندگی) در هر نسل

- بهترین معماری‌ای که به دست آمده است ویژگی‌های زیر را دارد. (دقیقا خروجی‌های حاصل از اجرای کد قرار داده شده‌اند.)

features_extractor_network: VGG11

number_of_hidden_layers: 1

learning_rate: 0.04168599959979155

batch_size: 64

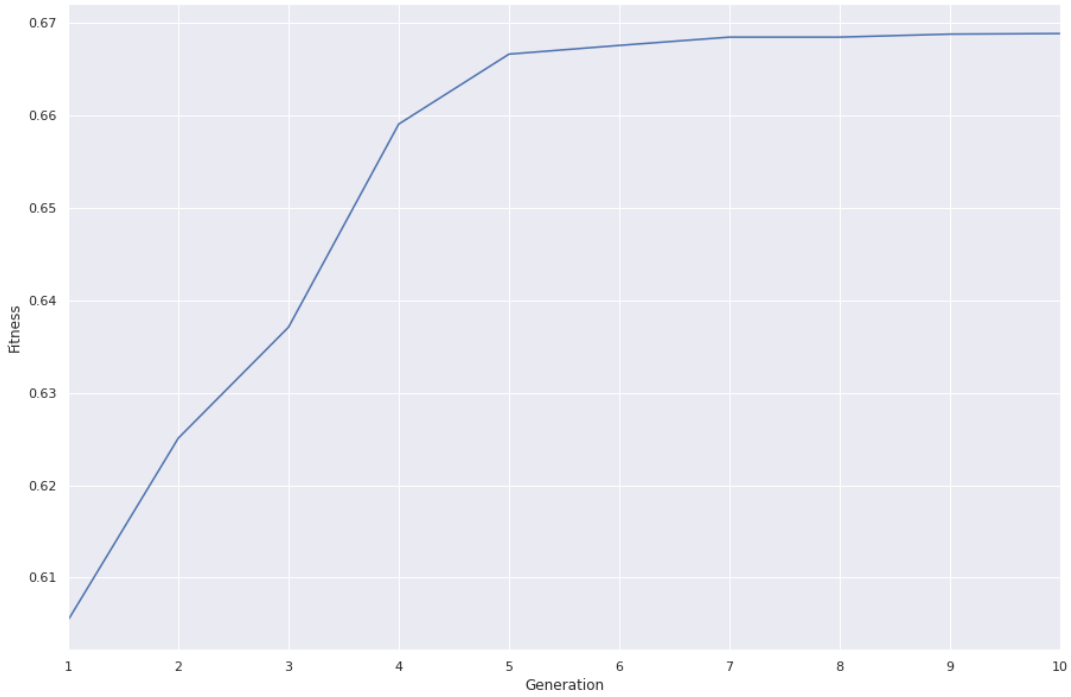
fitness: 0.6697000000000001

number_of_neurons_per_layer: [30]

activation_functions_per_layer: [ReLU]

۳-۳-۲ اندازه جمعیت ۱۰، حداکثر نسل‌ها ۱۰، احتمال جهش ۰/۷، احتمال بازترکیب ۰/۳

- نمودار برازندگی (میانگین معیار صحت در مجموعه آزمایش) در هر نسل به شکل زیر می‌باشد. مشاهده می‌شود الگوریتم به همگرایی رسیده است.



شکل ۳-۲ نمودار میانگین صحت (برازندگی) در هر نسل

- بهترین معماری‌ای که به دست آمده است ویژگی‌های زیر را دارد. (دقیقا خروجی‌های حاصل از اجرای کد قرار داده شده‌اند.)

features_extractor_network: VGG11

number_of_hidden_layers: 1

learning_rate: 0.08999999999999998

batch_size: 128

fitness: 0.6696799999999999

number_of_neurons_per_layer: [30]

activation_functions_per_layer: [ReLU]

۴- بخش سوم: خوشه‌بندی به کمک شبکه‌های عصبی

در این بخش باید به خوشه‌بندی بردارهای استخراج شده از شبکه ResNet34 بپردازیم. مجموعه آموزشی CIFAR10 را بدون برچسب در نظر گرفته و با استفاده از شبکه از پیش آموزش دیده ResNet34، بردار ویژگی را برای هر یک از تصاویر این مجموعه استخراج می‌کنیم. سپس آن‌ها به به کمک شبکه SOM که در لایه خروجی ۱۰ نورون دارد، با آموزشی به تعداد ۲۰ دور و در هر سه ذکر شده خوشه‌بندی می‌کنیم.

٤-۱- روش انجام آزمایش

پس از خواندن داده‌ها و اعمال مبدل، داده‌های آموزشی، آزمایش و اعتبارسنجی (۱۵ درصد داده‌های آموزشی) را جدا می‌کنیم. سپس، کلاس‌های لایه‌ها، توابع فعال‌سازی، هزینه، بهینه‌ساز، استخراج‌کننده‌های ویژگی، مدل شبکه عصبی، خوشه‌بند SOM را تعریف می‌کنیم (هر یک جداگانه شرح داده شده‌اند و قسمت‌های جدید شرح داده خواهند شد.). در مرحله بعدی، یک شی از کلاس SOM با تنظیم پارامترهای epochs، learning_rate، map_mask و neighborhood_radius ساخته و با صدا کردن متد train از این مدل به تعداد دورهای مورد نظر وزن‌های هر نورون را با دیدن نمونه‌های آموزشی به‌روز می‌کنیم.

٤-۲- توضیح اجمالی کد

۴-۲-۱ کلاس SOM

در این کلاس یک شبکه نگاشت خودسازمانده که یک شبکه دو لایه با ساختار همسایگی برای نورون‌های خروجی است، تعریف شده و از یادگیری رقابتی برای آموزش استفاده شده است. در این شیوه از آموزش، وزن‌های نورون خروجی برنده و همسایه‌های آن برای نزدیک شدن به داده ورودی تطبیق پیدا می‌کنند.

- **متد __init__:** در این متد مقادیر تعداد دورها، نرخ یادگیری، نقشه نورون‌های خروجی و شعاع همسایگی مقداردهی اولیه شده‌اند.
- **متد initialize_weights:** در این متد وزن‌ها با توجه به نقشه نورون‌های خروجی به صورت تصادفی مقداردهی می‌شوند.
- **متد find_bmu:** در این متد با توجه به نزدیکی هر نورون به ورودی نورون برنده مشخص شده و فاصله به متخصات نورون برنده تبدیل می‌شود.
- **متد train:** در هر دور، یک تصویر به شبکه ResNet34 داده شده و بردار ویژگی آن تصویر دریافت و به یک شئ Numpy تبدیل می‌شود. سپس با کمک تابع find_bmu، نورونی که به این داده نزدیک‌تر است مختصاتش دریافت می‌شود. در نهایت تمامی نورون‌هایی که در همسایه این نورون (نورون برنده) هستند، وزن‌هایشان به داده نزدیک می‌شود. مقدار نزدیک شدن وزن‌ها به داده با پارامتر learning_rate کنترل شده و برای شناسایی همسایه‌های نورون برنده، از فاصله منهتن استفاده می‌کنیم.
- **متد create_map:** در این متد دو ماتریس به ابعاد صفحه نورون‌ها ایجاد می شود. در self.map به هر نورون یک لیست خالی در ابتدا اختصاص داده می‌شود. سپس در حلقه‌ای، برای هر داده آموزشی نورون برنده آن پیدا شده و آن داده آموزشی به لیست آن نورون برنده اضافه می‌شود. در نهایت برای ایجاد self.label_map از تابع اکثریت استفاده می‌کنیم و دسته‌ای که بیشترین تکرار را در لیست هر نورون دارد به عنوان برچسب آن نورون قرار می‌دهیم.
- **متد print_map_report:** گزارشی از مختصات هر نورون، بیشترین کلاس تکرار شده در آن، تعداد نمونه‌ها و فرکانس هر کلاس به صورت تفکیک شده در آن نورون ارائه می‌دهد.
- **متد evaluate:** معیار صحت و امتیاز F1 را بر روی داده‌های آزمایش مشخص می‌کند.
- **متد plot_label_map :** در این متد خوشه هر نورون رو شکل نشان داده می‌شود.

- **متد plot_som:** در این متد u_matrix برای نورون‌ها تشکیل و رسم می‌شود. هر چه رنگ تیره‌تر باشد یعنی نورون‌ها به یک‌دیگر نزدیک‌تراند.
- **متد euclidean_distance:** این متد فاصله اقلیدسی دو بردار را با توجه به فرمول زیر برمی‌گرداند.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

- **متد manhattan_distance:** این متد فاصله منهتن دو بردار را با توجه به فرمول زیر برمی‌گرداند.

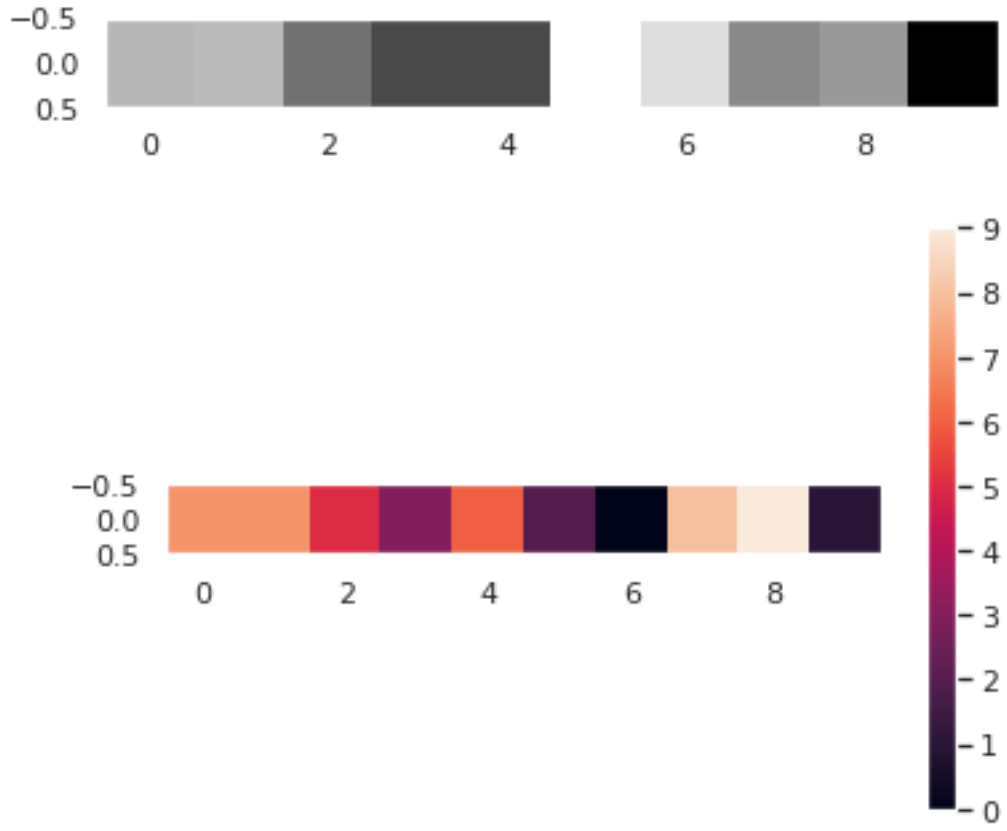
$$d(p,q) = \sum_{i=1}^{n}|p_i - q_i|$$

۴-۳ تحلیل نتایج بخش سوم

۴-۳-۱ الف) نورون‌های خروجی در یک الگوی یک بعدی، قطر همسایگی ۱

معیار صحت برابر ۵۰ درصد و امتیاز F1 برابر ۴۵ درصد.

- **تصویر بردارهای وزن برای نگاشت ویژگی** (هر نورون چه کلاسی را نشان می‌دهد.)



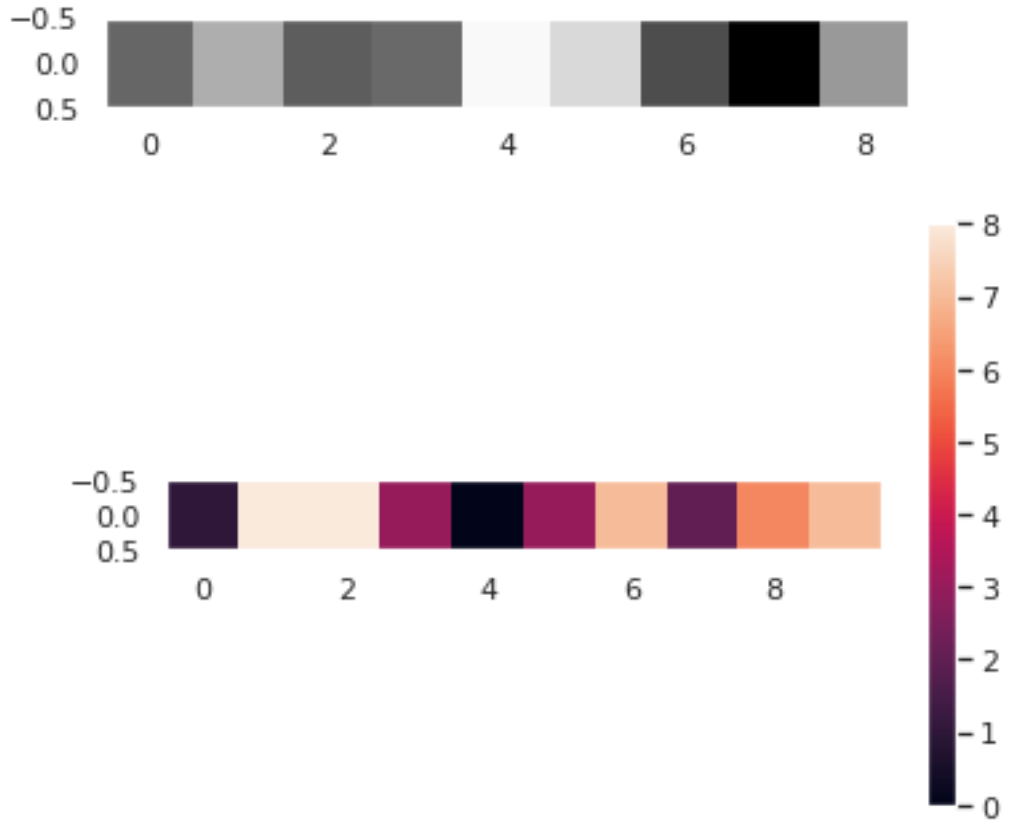
شکل ۴-۱ تصویر بردارهای وزن نگاشت ویژگی

- **پراکندگی برچسب‌های مختلف هر خوشه:** شماره نورون (خوشه)، تعداد نمونه‌ها در هر خوشه، تعداد نمونه‌های دسته‌های مختلف در خوشه به تفکیک، شماره دسته‌ای که بیشترین تعداد نمونه را دارد و فرکانس هر دسته در فایلی به نام som_p1.txt به پیوست ارسال می‌گردد.

٤-٣-٢ ب) نورون‌های خروجی در یک الگوی یک بعدی، قطر همسایگی ۳

معیار صحت برابر ۳۷ درصد و امتیاز F1 برابر ۲۹ درصد.

- **تصویر بردارهای وزن برای نگاشت ویژگی (هر نورون چه کلاسی را نشان می‌دهد.)**



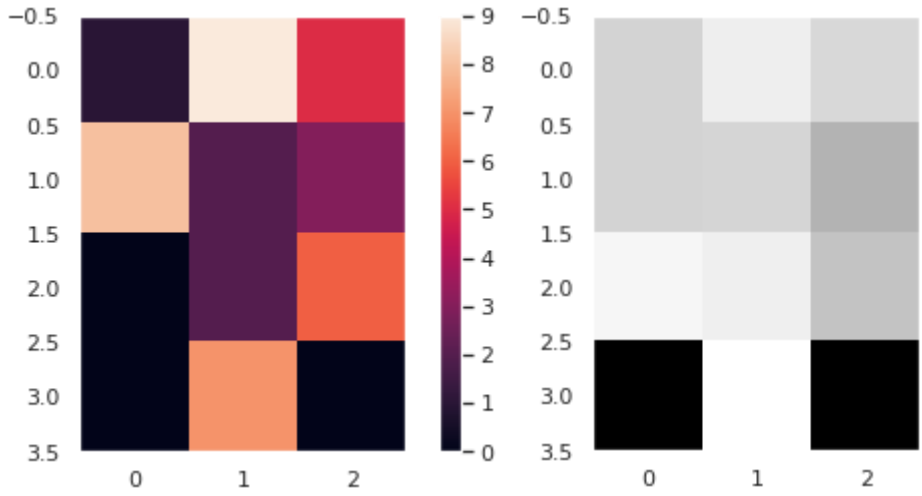
شکل ۴-۲ تصویر بردارهای وزن نگاشت ویژگی

- **پراکندگی برچسب‌های مختلف هر خوشه:** شماره نورون (خوشه)، تعداد نمونه‌ها در هر خوشه، تعداد نمونه‌های دسته‌های مختلف در خوشه به تفکیک، شماره دسته‌ای که بیشترین تعداد نمونه را دارد و فرکانس هر دسته در فایلی به نام som_p2.txt به پیوست ارسال می‌گردد.

٤-٣-٣ پ) نورون‌های خروجی در یک الگوی دو بعدی به شکل ذکر شده، قطر همسایگی ۱

معیار صحت برابر ٤٦ درصد و امتیاز F1 برابر ۳۹ درصد.

- **تصویر بردارهای وزن برای نگاشت ویژگی (هر نورون چه کلاسی را نشان می‌دهد.)**



شکل ۴-۳ تصویر بردارهای وزن نگاشت ویژگی

- **پراکندگی برچسب‌های مختلف هر خوشه:** شماره نورون (خوشه)، تعداد نمونه‌ها در هر خوشه، تعداد نمونه‌های دسته‌های مختلف در خوشه به تفکیک، شماره دسته‌ای که بیشترین تعداد نمونه را دارد و فرکانس هر دسته در فایلی به نام som_p3.txt به پیوست ارسال می‌گردد.

۵- منابع

Computational Intelligence, A Methodological Introduction 3rd edition: Rudolf Kruse, Sanaz Mostaghim, Christian Borgelt, Christian Braune, Matthias Steinbrecher

Understanding Self-Organising Map Neural Network with Python Code | by Ken Moriwaki | Towards Data Science

Derivative of the Softmax Function and the Categorical Cross-Entropy Loss | by Thomas Kurbiel | Towards Data Science