پرسش ۲ -پیاده سازی یک سیستم طبقه بندی خودرو با استفاده از VGG16 و SVM

۲-۱ مقدمه

در سال‌های اخیر با رشد سریع حوزه‌ی یادگیری عمیق، مدل‌های زیادی برای تحلیل و طبقه‌بندی تصاویر معرفی شده‌اند. یکی از کاربردهای جالب این مدل‌ها، شناسایی و دسته‌بندی خودروها از روی تصاویر آن‌هاست. این کار می‌تونه در زمینه‌هایی مثل سیستم‌های کنترل ترافیک، پارکینگ‌های هوشمند، تحلیل رفتار راننده‌ها و حتی بازارهای فروش خودرو نقش مهمی داشته باشه.

هدف این پروژه طراحی یک سیستم طبقه‌بندی خودرو بوده که بتونه بین مدل‌های مختلف خودروهای تویوتا تفاوت قائل بشه. برای این کار از مدل VGG16استفاده شده تا ویژگی‌های تصویری (features) از عکس‌ها استخراج بشن، و بعد این ویژگی‌ها به یک مدل SVMداده شدن تا کار طبقه‌بندی (classification)رو انجام بده.

در کنار این مدل ترکیبی(VGG + SVM) ، عملکرد چند مدل دیگه هم بررسی شده؛ از جمله AlexNet و یک مدل ساده‌ی CNN که طراحی شده تا با بقیه مقایسه بشه. همچنین، نسخه‌های مختلفی از مدل SVMبا کرنل‌های Linearو RBFآزمایش شدن تا ببینیم کدوم ترکیب عملکرد بهتری داره.

در ادامه‌ی این گزارش مراحل آماده‌سازی داده‌ها، استخراج ویژگی، آموزش مدل‌ها و تحلیل نتایج آورده شده. هدف اینه که ببینیم کدوم روش برای طبقه‌بندی خودروهای تصویری بهتر جواب می‌ده و چرا.

استفاده از نسخه GPU-SVM با کتابخانهRAPIDS

در ابتدای کار تصمیم بر این بود که برای پیاده‌سازی مدل Support Vector Machine (SVM) از کتابخانه‌ی معروف scikit-learnاستفاده شود. اما به‌دلیل اینکه استخراج ویژگی‌ها با استفاده از مدل VGG16منجر به تولید داده‌هایی با ابعاد نسبتاً بالا شده بود، زمان آموزش مدل SVM روی CPU به‌طور محسوسی زیاد بود. علاوه بر این، در برخی مراحل از اجرای مدل، به دلیل حجم بالای ویژگی‌ها، با مشکلات حافظه نیز مواجه شدیم.

در تلاش برای تسریع روند آموزش و بهینه‌سازی مصرف منابع، تصمیم گرفتیم از نسخه‌ی GPU-based الگوریتم SVM استفاده کنیم. در این مرحله، کتابخانه‌ی cuMLکه بخشی از پلتفرم **RAPIDS**است، مورد استفاده قرار گرفت. این کتابخانه از GPU برای شتاب‌دهی به محاسبات استفاده می‌کند و نسخه‌ای از SVMرا ارائه می‌دهد که بسیار سریع‌تر از نسخه‌ی مرسوم در scikit-learnاجرا می‌شود.

فرآیند نصب RAPIDS و راه‌اندازی آن در محیط Google Colab به راحتی با استفاده از اسکریپت آماده‌ی این پروژه در GitHub انجام شد. بعد از نصب و بارگذاری داده‌ها، مدل SVM با کرنل RBF آموزش داده شد و پیش‌بینی روی داده‌های تست انجام گرفت. خوشبختانه نتایج نهایی از نظر دقت، precision، recallو F1 Score با نتایج مدل CPU کاملاً قابل مقایسه بود، با این تفاوت که زمان آموزش به شکل چشم‌گیری کاهش پیدا کرد.

این تجربه نشان داد که در پروژه‌هایی که با حجم داده‌ی بالا سروکار داریم، استفاده از راهکارهای GPU مانند cuMLمی‌تواند گزینه‌ی بسیار خوبی برای افزایش کارایی باشد.

مدیریت حافظه در مرحله استخراج ویژگی

در مراحل ابتدایی پروژه، زمانی که تصمیم گرفته شد از مدل‌های VGG16 و AlexNet برای استخراج ویژگی‌های تصویری استفاده شود، یکی از چالش‌های اصلی، محدودیت منابع سخت‌افزاری در محیط Google Colabبود. با توجه به این‌که هر تصویر باید از طریق مدل عبور می‌کرد تا ویژگی‌های آن استخراج شود، و از طرفی تعداد تصاویر در هر کلاس نیز قابل توجه بود، اجرای همزمان تمام داده‌ها باعث کرش کردن runtime در Colab به دلیل کمبود RAM می‌شد.

در ابتدا برای حل این مشکل، داده‌ها به صورت دستی به چند batch کوچک‌تر تقسیم شدند و هر بار فقط یک بخش از آن‌ها از مدل عبور داده می‌شد. این روش اگرچه موقتی بود و امکان ادامه کار را فراهم می‌کرد، اما هم از نظر زمانی پرهزینه بود و هم احتمال بروز خطاهای انسانی در مدیریت داده‌ها را افزایش می‌داد.

در ادامه‌ی پروژه، با آشنایی بیشتر با کتابخانه‌ی PyTorch، از ساختار استاندارد آن یعنی Datasetو DataLoaderاستفاده شد. با تعریف یک کلاس Dataset سفارشی برای مجموعه تصاویر پروژه، و سپس استفاده از DataLoader برای ارسال داده‌ها به مدل به صورتbatch ، هم مشکل کمبود حافظه برطرف شد و هم فرآیند استخراج ویژگی ساختارمند و قابل مدیریت‌تر شد.

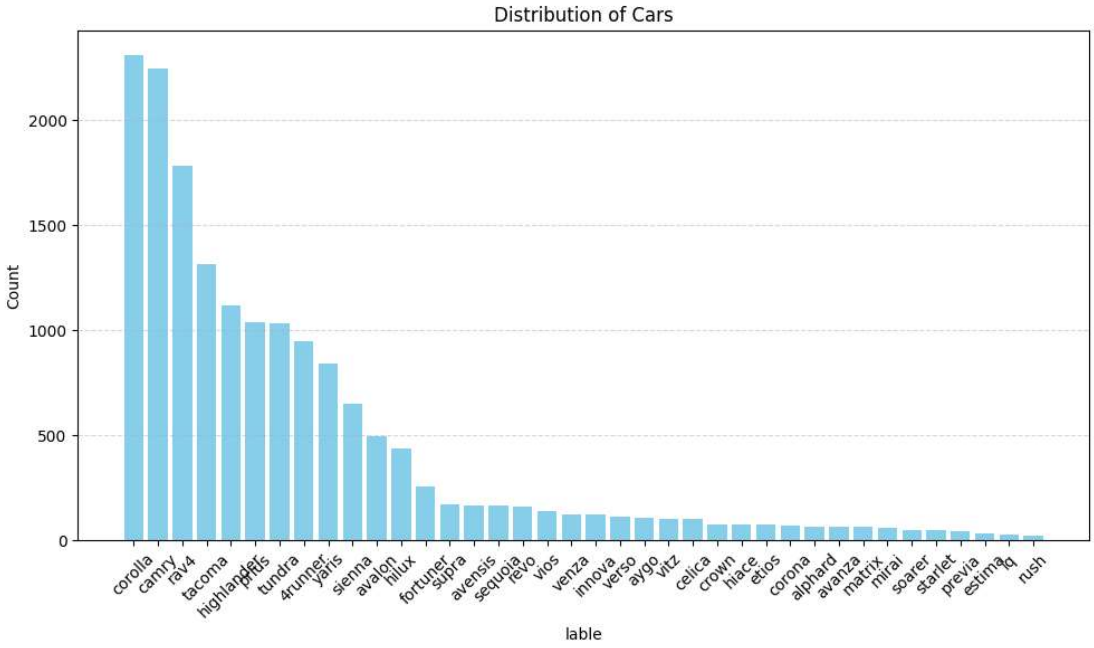
استفاده از DataLoaderاین امکان را داد تا بدون نگرانی از حجم زیاد داده‌ها، ویژگی‌های تمامی تصاویر به صورت batch و بهینه از مدل CNN استخراج شود. این تجربه‌ی فنی یکی از نقاط قوت اصلی پروژه بود که باعث شد مراحل پردازش و آموزش با اطمینان و پایداری بیشتری پیش بروند.

۲-۲ پیش پردازش داده ها

دیتا های مربوط به خودروهای تویوتا به طور کامل دانلود شدند و چیزی که حائز اهمیت است این است که در سایت ذکر شده است که دیتا ها با دقت ۹۵ درصد لیبل گذاری شده است.

حجم دانلود شده حدود ۲ گیگابایت است که از طریق محیط Google Colab دانلود و در دایرکتوری مربوط به Google Drive ذخیره شدند تا در زمان هایی که ریسورس های colab بدلیل محدودیت گرفته می شود همچنان در دسترس و قابل استفاده باشند.

تعداد عکس های مربوط به هر کلاس خودرو یا توزیع دیتاست از طریق نمودار زیر قابل مشاهده است.



شکل ۲-نمودار توزیع تصاویر خودرو های دیتاست

همانطور که صورت سوال خواسته بود ۱۰ تا از مدل خودرو به صورت دلخواه ، که در اینجا از ۱۰ خودرو با بیشترین توزیع را انتخاب کردیم و لیبل های آن ها را به مقادیر عددی ۰ تا ۹ تغییر داده شد.

برای شروع، مجموعه‌ای از تصاویر مربوط به ۱۰ مدل مختلف خودروهای تویوتا انتخاب شد. این مدل‌ها شامل corolla، camry، rav4، tacoma، highlander، prius، tundra، runner۴، yaris و sienna هستند که به ترتیب با لیبل‌های عددی ۰ تا ۹ مشخص شدند.

اولین مرحله در پردازش داده‌ها، resize کردن تمام تصاویر به اندازه‌ی ۲۲۴ در ۲۲۴ پیکسل بود. این اندازه همون سایزیه که مدل‌هایی مثل VGG16 انتظار دارن. بعد از تغییر اندازه، مقدار پیکسلی تصاویر به صورت

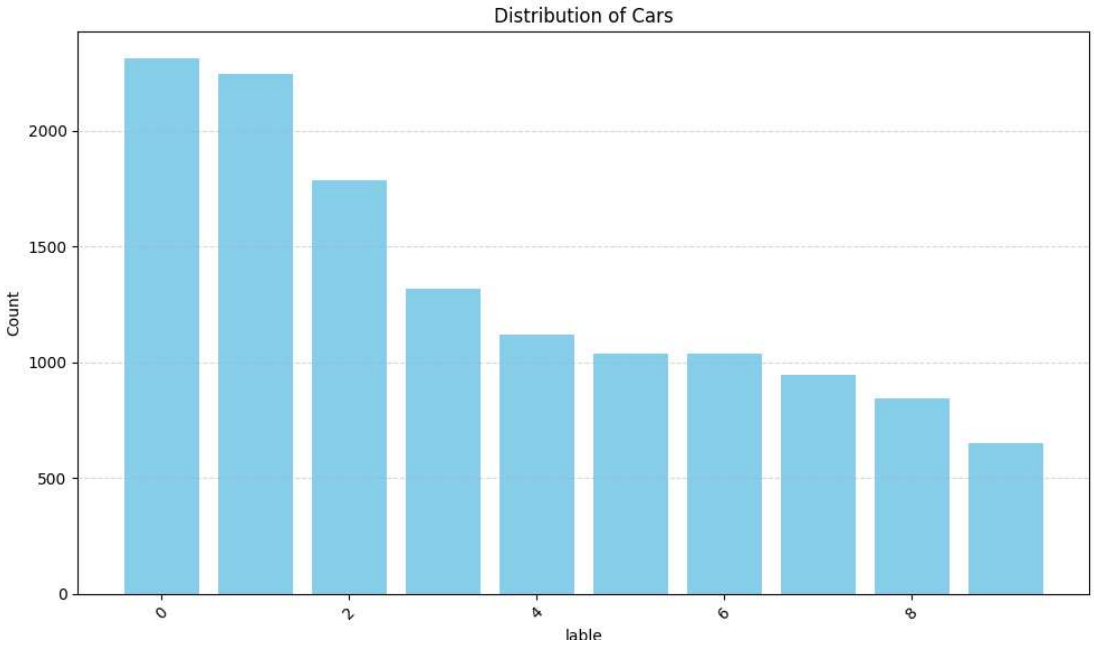
عددی بین صفر و یک نرمال‌سازی شد. این کار کمک می‌کنه تا مدل بتونه بهتر یاد بگیره و از نوسانات بزرگ عددی در ورودی جلوگیری بشه.

نکته‌ی مهمی که بعد از بررسی دیتاست مشخص شد، این بود که تعداد تصاویر در کلاس‌های مختلف برابر نیست. مثلاً بعضی مدل‌ها چند صد تصویر داشتن ولی بعضی دیگه فقط چند ده تصویر. این عدم تعادل می‌تونه باعث بشه مدل فقط روی کلاس‌های پرتکرار تمرکز کنه و بقیه رو نادیده بگیره.

برای حل این مشکل از روش Oversampling همراه با Data Augmentation استفاده شد. یعنی برای کلاس‌هایی که تصویر کم‌تری داشتن، با ایجاد تغییرات جزئی روی تصاویر موجود، تصاویر جدید تولید شد. این تغییرات شامل چرخش تصویر، فلیپ کردن، تغییرات رنگ و نور و... بودن. این کار هم باعث افزایش تعداد داده‌ها شد و هم باعث شد مدل با تنوع بیشتری از تصاویر روبه‌رو بشه که به یادگیری بهتر کمک می‌کنه.

در نهایت، با اضافه شدن تصاویر جدید، دیتاست به‌صورت نسبتاً متوازن بین کلاس‌ها تقسیم شد و مدل می‌تونه آموزش عادلانه‌تری ببینه.

طبق مقاله اندازه ی تصاویر را به ۲۲۴ در ۲۲۴ تغییر دادیم و پس از تغییر سایز به وسیله نرمال سازی مقادیر پیکسلی را به بین ۰ و ۱ تبدیل کردیم.

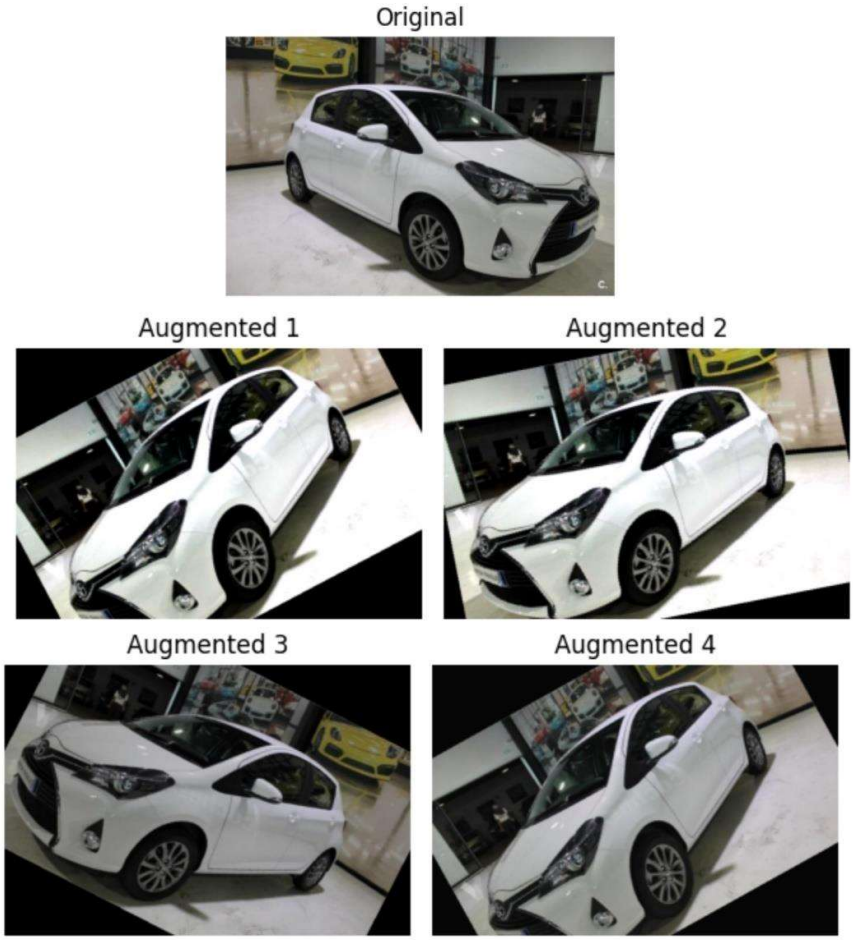


شکل ۳ – توزیع کلاس ها پس از ریسایز و تغییر لیبل ها

همانطور که در شکل بالا مشاهده میشود توزیع تعداد تصاویر در کلاس ها ، داری توزیع بالانس نیست و به اصطلاح داده ها نامتوازن توزیع شده اند.

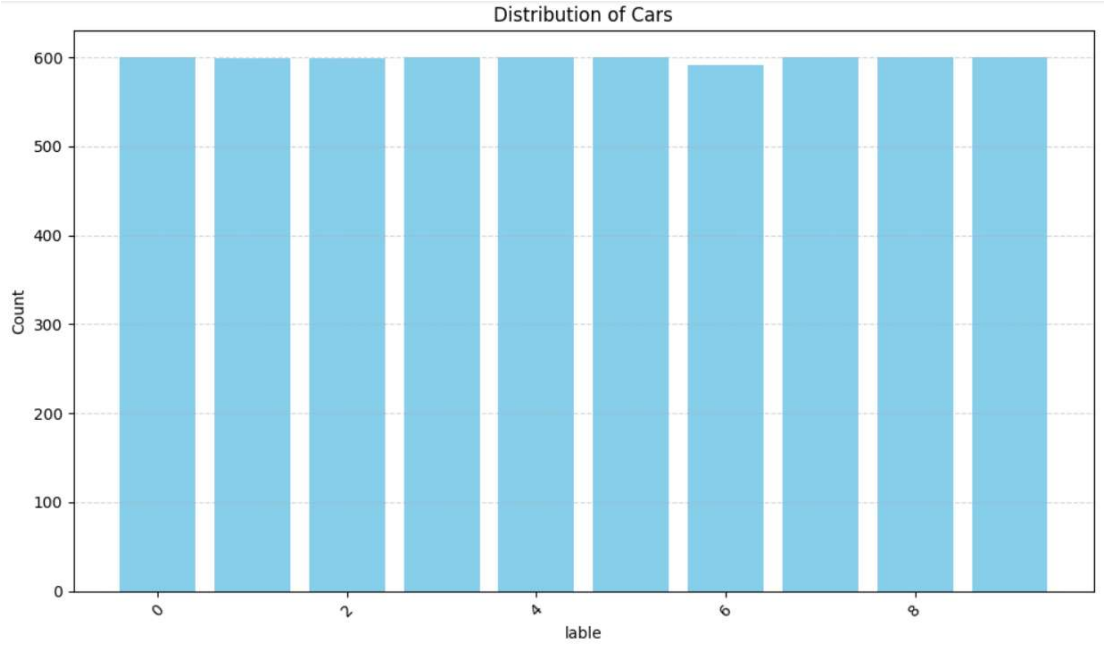
برای بالانس کردن دیتاست میتوان از ۳ تکنیک استفاده کرد:

- UnderSampling : این روش سریع و ساده است ولی استفاده از این روش میتواند باعث از بین رفتن اطلاعات مفید کلاس های پرتکرار شود.
- Oversampling : در این روش برخلاف روش قبل ، اطلاعات یا داده ها حفظ میشوند. ولی احتمال overfiting را افزایش میدهد.
- Oversampling با Data Augmentation : این روش تنوع بالا ، حفظ تعادل را در داده ها به ارمغان می آورد و همچنین به تعمیم مدل کمک میکند. عیب این روش نیازمندی آن به زمان محاسباتی بیشتر نسبت به دو روش دیگر است.



شکل ۴ – نمایش تصویری نحوه Agumentaion

ما برای برقراری توازن در داده ها از روش DataAugmentation با Oversampling استفاده کردیم. و دلیل این انتخاب ، وجود زمان کافی برای محاسبات بود .



شکل ۵ – توزیع کلاس ها بعد از نرمالسازی

3-2 استخراج ویژگی ها

در این مرحله مدل های VGG16 و AlexNet بارگزاری شدند و با استفاده از لایه های کانولوشنی آن ها ویژگی مربوط به هر تصویر در هر مدل را به صورت جداگانه بدست آوردیم و بعد از یک بعدی سازی (flatten) آن ها را در درایو ذخیره کردیم.

بعد از آماده‌سازی و نرمال‌سازی تصاویر، مرحله‌ی بعدی مربوط به استخراج ویژگی‌هاست. هدف از این مرحله اینه که قبل از آموزش مدل طبقه‌بندی‌کننده) مثل(SVM ، ویژگی‌های معنادار و فشرده‌ای از تصاویر به دست بیاریم که نماینده‌ی خوبی از محتویات اون تصویر باشن.

برای این کار از دو مدل معروف در حوزه‌ی بینایی ماشین استفاده شد VGG16 :و .AlexNetهر دو این مدل‌ها از نوع convolutional neural network (CNN) هستن و روی دیتاست ImageNet آموزش دیدن. به همین دلیل، بدون نیاز به آموزش مجدد، می‌تونن ویژگی‌های خیلی خوبی از تصاویر استخراج کنن. استفاده از این مدل‌ها به صورت از پیش آموزش‌دیده (pretrained) باعث می‌شه هم در زمان صرفه‌جویی بشه، هم نتایج بهتری بگیریم.

در این پروژه، فقط از بخش کانولوشنی (convolutional layers) این مدل‌ها استفاده شد. یعنی بخش fully connectedکه در انتهای مدل وجود داره و برای classification اصلی مدل طراحی شده، کنار گذاشته شد. دلیل این کار اینه که ما خودمون قصد داشتیم ویژگی‌های خروجی رو جداگانه به SVM بدیم و classification رو با اون انجام بدیم.

ویژگی‌های خروجی از این لایه‌ها معمولاً به شکل آرایه‌هایی چندبعدی هستن که نماینده‌ی الگوهای مختلف تصویرن (مثل لبه‌ها، بافت‌ها، شکل‌ها و...). این آرایه‌ها بعد از استخراج، به یک بردار یک‌بعدی تبدیل شدن تا بشه اونا رو به مدل SVM داد. این کار رو اصطلاحاً flattenکردن می‌گن.

در نهایت، بردارهای ویژگی استخراج‌شده برای همه‌ی تصاویر ذخیره شدن تا در مراحل بعدی برای آموزش و ارزیابی مدل‌ها استفاده بشن. چون استخراج ویژگی نسبتاً زمان‌بره، این کار فقط یک‌بار انجام شد و نتایج در حافظه‌ی دائمی) مثل (Google Drive ذخیره شدن.

۴-۲ آموزش و ارزیابی مدل

۱. در این مرحله به وسیله ویژگی های استخراج شده در مرحله قبل لایه های fully connected مربوط به مدل های VGG16 و AlexNet آموزش داده شدند. و نتایج به صورت زیر قابل مشاهده اند.

برای ارزیابی عملکرد مدل‌های طبقه‌بندی، معمولاً از چند معیار رایج استفاده می‌شود که مهم‌ترین آن‌ها عبارت‌اند از Precision، Recall و F1 Score. این معیارها به ما کمک می‌کنند تا علاوه بر بررسی دقت کلی مدل (Accuracy)، بتوانیم عملکرد آن را در شناسایی صحیح نمونه‌های هر کلاس به‌صورت جزئی‌تر بررسی کنیم.

Precision (دقت)

Precision نشان می‌دهد از بین تمام تصاویری که مدل به عنوان «یک کلاس خاص» (مثلاً مدل Corolla) پیش‌بینی کرده، چه درصدی از آن‌ها واقعاً متعلق به همان کلاس بوده‌اند.

$$Precision = \frac{TP}{\mathrm{TP} + \mathrm{FP}}$$

فرمول ۱-۰

به زبان ساده، اگر مدل در پیش‌بینی کلاس‌ها خیلی سخت‌گیر باشد و فقط در مواقعی که مطمئن است پیش‌بینی کند، معمولاً precision بالاتری خواهد داشت. اما ممکن است برخی نمونه‌ها را اصلاً شناسایی نکند. در مسئله‌ی ما، اگر مدل فقط خودروهایی را به عنوان «RAV4» تشخیص دهد که کاملاً شبیه به نمونه‌های آموزشی RAV4 هستند، precisionبالا می‌رود ولی ممکن است بعضی RAV4های واقعی را نادیده بگیرد.

بازخوانی (Recall)

Recallنشان می‌دهد مدل از بین تمام تصاویری که واقعاً به یک کلاس خاص تعلق داشته‌اند، چه تعداد را به درستی شناسایی کرده است.

$$Recall = \frac{TP}{\mathrm{TP} + \mathrm{FN}}$$

فرمول ۲-۰

معمولا با افزایش میزان precision باعث کاهش Recall می شود و برعکس.

اگر مدل تمایل داشته باشد تا بیشتر پیش‌بینی کند (حتی با اطمینان کمتر)، معمولاً recall بالا می‌رود. این یعنی احتمال اینکه مدل، تصویر یک خودروی RAV4 را اشتباهی مثلاً «Highlander» تشخیص دهد(False Negative) ، کمتر خواهد شد.

رابطه‌ی بین Precision وRecall

بین این دو معیار یک trade-offطبیعی وجود دارد؛ افزایش یکی معمولاً باعث کاهش دیگری می‌شود. به همین دلیل، برای رسیدن به یک ارزیابی متعادل، از معیار ترکیبی F1 Scoreاستفاده می‌شود که میانگین‌هارمونیک این دو مقدار است:

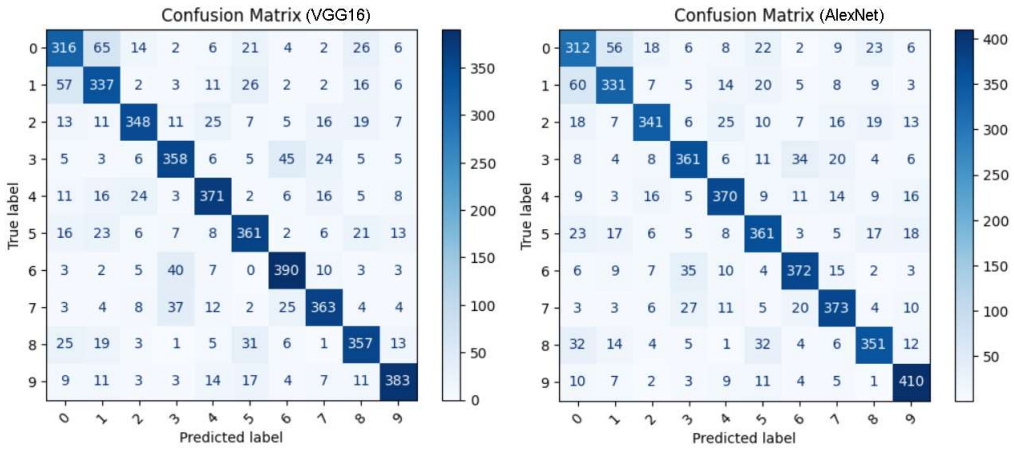
$$f1 - score = \frac{Precision * Recall}{Precision + Recall} * 2$$

فرمول ۳-۰

F1 Score زمانی بیشترین مقدار خود را دارد که precision و recall به یک اندازه بالا باشند. در پروژه‌ی ما، این معیار کمک می‌کند تا عملکرد مدل‌ها را به‌صورت کلی‌تری ارزیابی کنیم و مدل‌هایی که هم کم‌ترین اشتباه را دارند و هم بیشترین شناسایی صحیح را انجام می‌دهند، بهتر تشخیص داده شوند.

مدل	Accuracy	Precision	Recall	F1 Score
AlexNet	0.7752	0.7759	0.7751	0.7751
VGG16	0.7756	0.7768	0.7756	0.7758

جدول ۴-ارزیابی مدل های AlexNet و VGG16



شکل - 6 ماتریس در هم ریختگی مربوط به دو مدل پس آموزش

هر دو مدل در تشخیص برخی کلاس‌ها عملکرد قابل توجهی داشتند. به‌خصوص کلاس‌های ۳، ۴، ۶، ۷ و ۹ به خوبی از سوی هر دو مدل شناسایی شدند:

- کلاس ۶ در VGG16 با ۳۹۰ پیش‌بینی صحیح و در AlexNet با ۳۷۲ مورد، از بالاترین دقت‌ها برخوردار بود.
- کلاس ۴ در هر دو مدل عملکرد بالایی داشت 371) در VGG و ۳۷۰ در.(AlexNet
- کلاس ۳ و ۷ نیز با مقادیر پیش‌بینی صحیح بالا، نشان‌دهنده‌ی توانایی مدل‌ها در شناسایی این خودروها هستند.
- کلاس ۹ بیشترین دقت را در هر دو مدل داشت، به‌ویژه در AlexNet که با ۴۱۰ پیش‌بینی صحیح، بیشترین مقدار در کل ماتریس‌ها بود

در مقابل، هر دو مدل در تشخیص کلاس‌های ۰ و ۱ دچار چالش بودند:

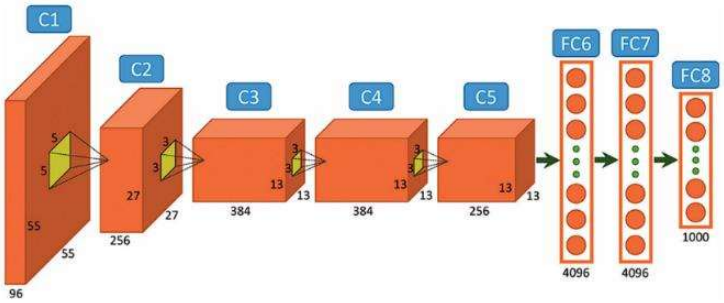
- در:VGG16
  - کلاس ۰ به اشتباه ۶۵ بار به عنوان کلاس ۱ پیش‌بینی شده است.
  - کلاس ۱ نیز ۵۷ بار به اشتباه به عنوان کلاس ۰ طبقه‌بندی شده است.
- در:AlexNet
  - کلاس ۰ با ۵۶ مورد اشتباه در کلاس ۱
  - کلاس ۱ با ۶۰ مورد اشتباه در کلاس ۰

این تقارن در اشتباهات نشان می‌دهد که تمایز بین این دو کلاس برای هر دو مدل دشوار بوده و به احتمال زیاد شباهت‌های ظاهری بین خودروهای این دو کلاس (مثلاً فرم کلی، رنگ یا زاویه عکس) باعث بروز این خطا شده است.

۲. اصلی ترین تفاوت بین AlexNet و VGG Net فارغ از نتایج:

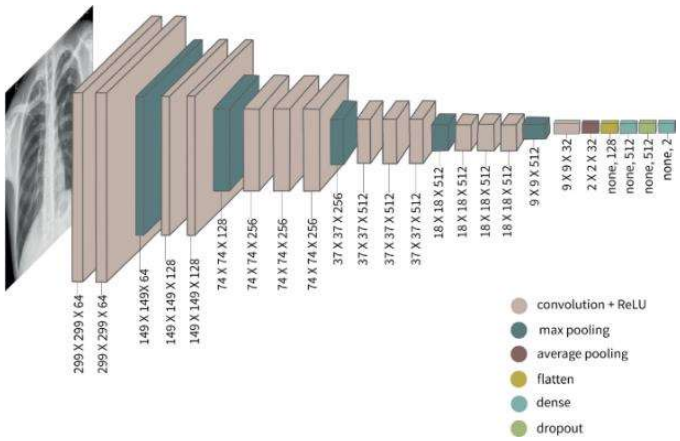
میتوان گفت که اصلی ترین تفاوت بین طراحی و عمق لایه های کانولوشن است. شبکه AlexNet با ۵ لایه کانولوشن کم عمق تر نسبت به شبکه VGG16 با ۱۳ لایه کانولوشنی است .

در مورد تفاوت این دو مدل میتوان به این نکته توجه کرد که مدل AlexNet از فیلتر های متنوع و بزرگ استفاده میکند ولی VGG16 از یک نوع فیلتر ۳ در ۳ فقط استفاده میکند.



شکل ۷ – تصویر از مدل AlexNet

مدل VGG دارای حجم بیشتر و سرعت پایین اما AlexNet دارای حجم کمتر و سرعت بالا تر ولی از لحاظ دقت مدل VGG عملکرد بهتری دارد.



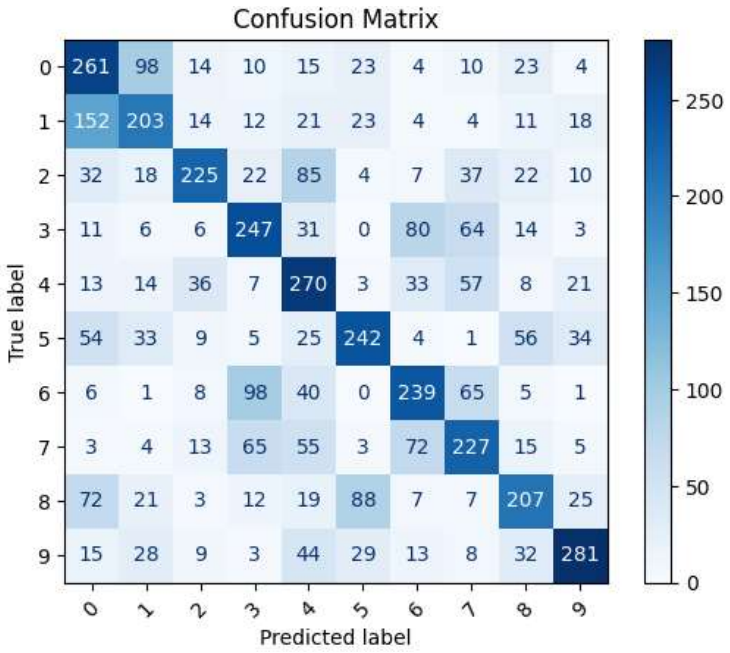
شکل ۸ – تصویر شماتیک مدل VGG16

۳. یک مدل CNN دلخواه در این بخش طراحی شد که دارای ۴ بلوک کانولوشن ، و ۳ لایه Fully Connected است ، لایه های کانولوشنی دارای تابع فعال ساز Relu و تکنیک BatchNorm برای تسریع و پایداری آموزش ، و MaxPooling برای کاهش ابعاد است. و به منظور کاهش Overfiting از Dropout استفاده شده است.

پس از آموزش نتایج به صورت زیر قابل مشاهده است:

معیار	مقدار
Accuracy	0.5198
Precision	0.5324
Recall	0.5198
F1 Score	0.5218

جدول ۵ - ارزیابی مدلCNN



شکل-9 ماتریس در هم ریختگی مربوط به مدل CNN

ماتریس درهم‌ریختگی مربوط به مدل ساده‌ی CNN طراحی‌شده در پروژه، ضعف‌هایی در دقت پیش‌بینی را نشان می‌دهد، که عمدتاً به دلیل ساده بودن ساختار آن است.

کلاس‌های ۰، ۱ و ۹ با دقت بهتری نسبت به سایر کلاس‌ها طبقه‌بندی شده‌اند.

- کلاس ۹: ۲۸۱ پیش‌بینی صحیح
- کلاس ۰: ۲۶۱ پیش‌بینی صحیح
- کلاس ۱: ۲۰۳ پیش‌بینی درست (هرچند با اشتباه زیاد)
- کلاس‌های ۲، ۳، ۶، ۷ و ۸ پیش‌بینی‌های بسیار پراکنده‌ای دارند.
- کلاس ۶، برای مثال فقط ۲۳۹ مورد به درستی تشخیص داده شده و باقی در کلاس‌های دیگر پراکنده شده‌اند.
- کلاس ۲ و ۳ نیز به‌طور مکرر با سایر کلاس‌ها اشتباه گرفته شده‌اند.

مدل ساده‌ی CNN در تفکیک مرزهای بین کلاس‌ها ضعیف‌تر عمل کرده است. این موضوع نشان می‌دهد که برای دیتاست‌های واقعی و پیچیده، استفاده از مدل‌های از پیش آموزش‌دیده یا عمیق‌تر، تأثیر چشم‌گیری در بهبود دقت دارند.

۴. مدل پیشنهادی مقاله به این صورت تعریف شده است که با ترکیب VGG و SVM به اینصورت که لایه های کانولوشنی مدل VGG وظیفه خروج feature هارا به عهده داشته باشد و SVM وظیفه Classification این feature هارا به عهده بگیرد.

برای این منظور یک مدل VGG دانلود و با استفاده از بخش کانولوشنی آن ویژگی هر تصویر را استخراج کردیم و با استفاده از آن مدل SVM را آموزش دادیم.

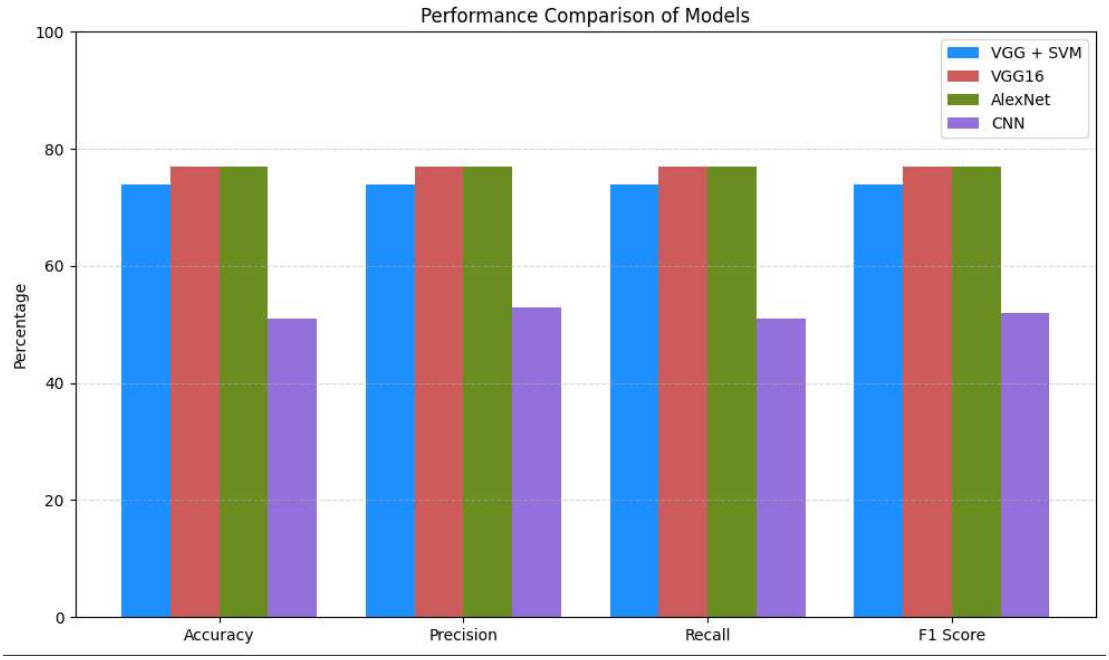
نتایج رو داده های تست به صورت زیر قابل مشاهده هستند:

معیار	مقدار
Accuracy	0.7453
Precision	0.7464
Recall	0.7453
F1 Score	0.7457

جدول ۶ - ارزیابی مدل VGG+SVM

۵-۲ تحلیل نتایج

معیار های عملکردی محاسبه شد هاز مرحله های قبل به صورت جدول زیر قابل مشاهده است.



شکل ۱۰- نمایش گرافیکی عملکرد به دست آمده توسط مدل های مختلف برای مجموعه داده تویوتا

بعد از آموزش مدل‌ها و اجرای مرحله‌ی ارزیابی روی داده‌های تست، مشخص شد که مدل‌های CNN کامل، یعنی AlexNet وVGG16 ، بهترین عملکرد رو داشتن .دقت (accuracy) این دو مدل حدود ۷۷ درصد بود که از سایر روش‌های آزمایش‌شده بهتر بود.

مدل VGG16 با دقت ۰.۷۷۵۶ و AlexNet با دقت ۰.۷۷۵۲ تونستن تصاویر خودرو رو با کیفیت مناسبی طبقه‌بندی کنن. این دو مدل از ابتدا تا انتها شامل لایه‌های کانولوشنی و fully connected بودن و روی ویژگی‌هایی که خودشون استخراج کرده بودن آموزش دیدن.

در کنار این دو مدل، ترکیب VGG16 با SVM هم آزمایش شد؛ به این صورت که ویژگی‌ها از لایه‌های کانولوشنی VGG استخراج شدن و سپس به یک مدل SVM داده شدن. در این حالت، با استفاده از کرنل Linear، دقت به حدود ۰.۷۴ رسید و با کرنل RBF کمی بهتر شد (حدود ۰.۷۵). هرچند این مقادیر نزدیک به مدل‌های CNN کامل بودن، ولی در نهایت ترکیب VGG + SVM نتونست عملکردی بهتر از خود مدل VGG16یا AlexNet ارائه بده.

این موضوع نشون می‌ده که گاهی اوقات وقتی مدل CNN به‌صورت end-to-end آموزش ببینه، بهتر می‌تونه ارتباط بین ویژگی‌ها و کلاس‌ها رو یاد بگیره. در مقابل، وقتی feature extraction و classification از هم جدا بشن، یه مقدار از هماهنگی بین این دو بخش کم می‌شه و ممکنه تأثیر منفی بزاره، به‌خصوص وقتیfeature ها خیلی پیچیده‌ان.

مدل CNN ساده‌ای هم که در این پروژه طراحی شد) با ۴ بلاک کانولوشن و چند لایه‌ی fully connected) عملکرد ضعیف‌تری داشت و به دقتی حدود ۵۲ درصد رسید. این مدل بدون استفاده از pretraining بود، و با توجه به تعداد نسبتاً محدود تصاویر، نتونست به اندازه‌ی مدل‌های بزرگ‌تر یاد بگیره.

در مجموع:

- VGG16 و AlexNet بهترین عملکرد رو داشتن.
- VGG + SVM نتایج نسبتاً خوبی داشت ولی نتونست از مدل‌های اصلی CNN جلو بزنه.
- مدل ساده‌ی طراحی‌شده برای تمرین و تست خوب بود ولی به بهینه‌سازی بیشتری نیاز داره.

این نتایج نشون می‌دن که انتخاب بین مدل‌های end-to-end و مدل‌های ترکیبی feature extractor + classifier باید با توجه به نوع داده‌ها، حجم آن‌ها و هدف نهایی پروژه انجام بشه.

با مقایسه‌ی چهار مدل، مشاهده می‌شود که **VGG16 و AlexNet** در تشخیص کلاس‌ها عملکرد بهتری دارند، به‌ویژه در کلاس‌هایی که تفاوت‌های بصری مشخص‌تری دارند. اما حتی این مدل‌ها نیز در تمایز بین برخی کلاس‌ها (مثلاً کلاس ۰ و ۱) که ظاهری مشابه دارند، دچار چالش هستند.

برای بهبود عملکرد مدل‌ها پیشنهاد می‌شود:

۱. **Data Augmentation هدفمندتر**
   - استفاده از augmentation خاص برای کلاس‌هایی که بیشتر اشتباه گرفته می‌شوند. مثلاً برای کلاس‌هایی که از زوایای مختلف سخت تشخیص داده می‌شن، چرخش و تغییر زاویه می‌تونه کمک‌کننده باشه.

۲. **Fine-tuning دقیق‌تر روی VGG و AlexNet**
   - به‌جای استفاده از فقط feature extractor ، می‌توان چند لایه‌ی آخر مدل را مجدداً روی دیتاست خود آموزش داد تا دقت مدل بیشتر شخصی‌سازی شود.

۳. **افزایش داده برای کلاس‌های ضعیف‌تر**
   - کلاس‌هایی مثل ۲، ۶ یا ۸ که پراکندگی بیشتری دارند، ممکن است به دلیل کم بودن داده یا تنوع پایین در آموزش باشند.

۴. **استفاده از معماری‌های پیشرفته‌تر مثل ResNet یا EfficientNet**

- این مدل‌ها به‌طور خاص طراحی شدن تا دقت بیشتری با پارامتر کمتر ارائه بدن، و در پروژه‌های شبیه به این نتایج بهتری داشته‌اند.

۵. **تحلیل ویژگی‌های اشتباه‌شده(Misclassification Analysis)**

- بررسی بصری نمونه‌هایی که مدل اشتباه تشخیص داده می‌تونه کمک کنه بفهمیم آیا اشکال از شباهت ظاهریه یا کیفیت تصویر یا حتی لیبل‌گذاری.

۲-۶ امتیازی

۱. مدل پیشنهادی مقاله یعنی VGG + SVM را که در مرحله قبل با کرنل Linear پیاده سازی شده بود در این مرحله با استفاده کرنل های RBF , Sigmoid, Polynomial آموزش داده شد و نتایج به صورت زیر قابل مشاهده است.

کرنل SVM	Accuracy	Precision	Recall	F1 Score
Linear	0.7453	0.7464	0.7453	0.7457
RBF	0.7563	0.7598	0.7563	0.7573
Polynomial	0.7475	0.7553	0.7474	0.7480
Sigmoid	0.6624	0.6661	0.6624	0.6635

جدول ۷ - ارزیابی با کرنل های متفاوت

همان‌طور که مشاهده می‌شود، کرنل RBF بهترین عملکرد را در بین تمام گزینه‌ها داشته است، که با توجه به خاصیت انعطاف‌پذیری بالای این کرنل در مدل‌سازی مرزهای غیرخطی، نتیجه‌ی قابل انتظاری است. کرنل Polynomial نیز عملکردی نسبتاً نزدیک به RBF داشته و از کرنل Linear کمی بهتر عمل کرده است.

در مقابل، کرنل Sigmoid عملکرد نسبتاً ضعیف‌تری نشان داده است .این کرنل در بسیاری از کاربردهای عملی نسبت به سایر گزینه‌ها دقت پایین‌تری دارد و معمولاً کمتر توصیه می‌شود، مگر در شرایط خاص.

این نتایج نشان می‌دهند که برای داده‌هایی که از طریق مدل‌های عمیق مانند VGG16 استخراج می‌شوند و ممکن است دارای مرزهای غیرخطی پیچیده باشند، استفاده از کرنل‌های غیرخطی مانند RBFیا Polynomial می‌تواند منجر به بهبود دقت مدل شود.

تفاوتی که قابل مشاهده است این است که دقت کرنل linear معمولا کمتر از RBF است دلیل این موضوع هم وجود مرز های غیرخطی در RBF است .

سرعت آموزش در Linear به طور قابل توجه ای بیشتر از RBF بود.

به طور کلی کرنل Linear برای داده هایی با ویژگی های زیاد یا تفکیک پذیری خطی مناسب است و کرنل RBF مناسب برای داده های پیچیده یا با مرز های کلاس نامنظم مناسب است.

اگر داده ها ساده و به خوبی قابل تفکیک باشه ، linear kernel کافیست ولی اگر داده ها پیچیده یا مرز های کلاس ها غیر خطی باشند ، RBF Kernel عملکرد بهتری دارد.

استخراج ویژگی های از طریق مدل هایی مثل VGG و AlexNet میتواند مدل SVM را درگیر پیچیدگی ها تصاویر نشود و داده ها به طور خلاصه و مفید تر به مدل SVM داده میشود. این گونه مدل عملکرد دقیق تری خواهد داشت.این مورد زمانی خود را نشان میدهد که مرز بین کلاس ها پیچیده باشد.

در این پروژه، هدف اصلی طراحی و پیاده‌سازی یک سیستم طبقه‌بندی خودرو بر اساس تصویر بود که با استفاده از مدل‌های از پیش آموزش‌دیده مانند VGG16و AlexNetو همچنین ترکیب این مدل‌ها با الگوریتم SVMانجام شد. برای این کار مراحل مختلفی مثل پیش‌پردازش داده‌ها، متعادل‌سازی کلاس‌ها باdata augmentation ، استخراج ویژگی‌ها و در نهایت آموزش و ارزیابی مدل‌ها انجام گرفت.

نتایج نهایی نشان داد که مدل‌های کاملCNN یعنی VGG16 و AlexNet در این مسئله عملکرد بهتری نسبت به مدل‌های ترکیبی مثل VGG + SVM داشتند. این تفاوت می‌تواند به دلیل هماهنگی بهتر بین بخش استخراج ویژگی و طبقه‌بندی در معماری‌های end-to-end باشد. با این حال، استفاده از SVM به عنوان طبقه‌بند نیز عملکرد قابل قبولی ارائه داد، به‌خصوص در شرایطی که هدف کاهش پیچیدگی مدل یا تفکیک دقیق‌تر با حجم داده کمتر باشد.

همچنین مشخص شد که استفاده از data augmentation نقش مهمی در بهبود عملکرد مدل‌ها دارد، مخصوصاً زمانی که داده‌ها نامتوازن باشند. بالانس کردن کلاس‌ها باعث شد مدل‌ها آموزش بهتری ببینند و عملکرد پایدارتری داشته باشند.