

## تشخیص دسته بندی کالا از طریق عکس با استفاده از دیپ لرنینگ

حسین انیسی - 40215203

محمد یوسف شریف - 40119693

امیررضا پاک روح - 40215763

### پیش پردازش داده ها و تغییر در ساختار دسته بندی ها

دیتاست ما در ابتدا دارای 248 دسته بندی مختلف بود که برخی از آن ها تفاوت های جزئی داشتند و همچنین بعضی از دسته بندی ها دارای تعداد بسیار کمی از محصولات بودند چون این موارد میتواند دقت مدل را کاهش دهد یا مدل نسبت به برخی از دسته بندی ها سوگیری کند در ابتدای کار یک پایپلاین سه مرحله ای برای اصلاح ساختار دیتاست اجرا شد

مرحله اول: ادغام معنایی دسته ها با NLP

create\_category\_mappings.py یک فایل csv کا شامل شناسه و و نام دسته بندی هاست را دریافت میکند (categories.csv) و برای هر نامه دسته با استفاده از مدل all-MiniLM-L6-v2 در کتابخانه sentence-transformers یک embedding عددی استخراج می شود که نمایش برداری نام دسته بندی ها هستند

سپس با استفاده از الگوریتم k-mins بردار های تولید شده را خوشه بندی میکنیم

در نهایت به 50 خوشه میرسیم که هر کدام نماینده یک دسته بندی جدید است category\_mapping.csv خروجی ان کد است که شامل شناسه اولیه و جدید و نام دسته بندی است

مرحله دوم اعمال نگاشت:

در مرحله بعد یعنی در کد `apply_category_mapping.py` نگاشت رو محصولات اعمال می شود `products.csv` و `category_mapping.csv` وردی کد است که با استفاده از یک دیکشنری از شناسه اولیه به شناسه جدید محصولات به دسته بندی جدید نگاشت میکند و در فایل `products_cleaned.csv` خروجی می دهد که شامل `asin/title/imgURL/merged category id` است

مرحله سوم: فیلتر کردن دسته بندی های کم نمونه

در این بخش تعداد محصولات هر دسته شمارش می شود دسته بندی هایی که کم تر از 5000 محصول دارند حذف می شوند در اخر برای اینکه شماره دسته بندی ها پیوسته باشد برچسب گذاری از ابتدا انجام میشود خروجی کد باز نویسی `products_cleaned.csv` و `category_mapping.csv` است.

### **تحلیل دیتاست و اعتبارسنجی پایپ لاین آماده سازی داده ها**

در این مرحله ابتدا ساختار و توزیع دسته بندی خام بررسی شد (که مناسب نبود)

سپس دسته بندی های جدید اعتبارسنجی شد و نمودار توزیع نمونه ها رسم شد بررسی اعتبار لینک تصاویر: 200 نمونه تصادفی انتخاب میشود سپس درخواست `http` با یک تای اوت ارسال میشود و دنیهای وضعیت بررسی شد و نتیجه این شد که اکثر لینک ها معتبر هستند

در بخش دیگر به طور تصادفی از هر دسته دو تصویر نهایش داده می شود تا کیفیت و تمایز تصویر به صورت بصری بررسی شود

1000 تصویر به طور تصادفی انتخاب شد و عرض ارتفاع هرود بررسی قرار گرفت نتیجه این است که پراکندگی ابعاد تصویر زیاد است که در مراحل بعد این مشکل را حل میکنیم

بیست تصویر تصادفی انتخاب میکنیم و رنگ غالب هر دسته را استخراج میکنیم نتیجه این است چون پس زمینه اکثر محصولات سفید است ممکن است مدل به

این مورد وابسته باشد که در مرحله آگمنتیشن برای این مورد راه حل هایی در نظر میگیریم

تصمیم عدم استفاده از سگمنتیشن: را اعتبار سنجی میکنیم چون کنتراست بین محصول و پس زمینه بالاست و هزینه این بخش نیز زیاد است پس زمینه را حذف نمیکنیم و به جای آن در مرحله آگمنتیشن راه حل هایی را در نظر میگیریم

### **طراحی استراتژی Data Augmentation و مدیریت عدم تعادل کلاس ها**

در این مرحله ابعاد همه تصاویر را به  $224 \times 224$  استرچ میکنیم که با معماری هایی مثل ResNet, EfficientNet سازگار تر است  
هر تصویر را حداکثر 20 درجه چرخش می دهیم و فضا های خالی ایجاد شده را با رنگ سفید پر میکنیم

تصویر حداکثر 10 درصد در راستای عمودی و افقی جابجا میکنیم  
کمی تغییرات نوری در تصویر اعمال میکنیم  
با احتمال 50 دسر تصویر را آینه میکنیم  
از مقادیر استاندارد ImageNet استفاده میکنیم

همه این اعمال برای افزایش مقاومت مدل نسبت به شرایط خاص استدیویی است در نهایت به صورت تصادفی یک تصویر را با تغییرات اعمال شده نمایش میدهیم تا کیفیت خروجی از لحاظ بصری بررسی شود  
مرحله دوم:

مقابله با عدم تعادل کلاس ها تعداد نمونه های هر کلاس ها را محاسبه میکنیم و آن ها را بر اساس فراوانی مرتب میکنم و سپس کلاس ها را دهک ها و چارک ها تقسیم میکنیم و وزن آگمنتیشن را بر اساس این چارک ها تعیین میکنیم تا تعادل مدل حفظ شود

## طراحی و پیاده سازی مدل Baseline

در این مرحله یک مدل پایه را ترین میکنیم تا صحت عملکرد مدل بررسی شود  
برای این مرحله برای هر کلاس از 25000 نمونه استفاده کردیم و کلاس هایی که  
کمتر از این تعداد نمونه داشتند کنار گذاشته شد

نمونه از طریق URL دانلود میشوند و با استفاده از MD5 به آن ها نام یکتا  
اختصاص می دهیم . همچنین داده ها کش میشوند تا از دانلود مجدد  
جلوگیری شود .

کلاس ProductImageDataset پیاده سازی شد که داده ها را از URL یا کش  
میخواند و (image\_tensor, label) را برمیگرداند .

مراحل اگنتیشن در بیس لاین نیز پیاده سازی شد

معماری بیس لاین SimpleCnn است که بلاک کانگلوشن شامل موارد زیر است

Conv2D

BatchNorm

ReLU

MaxPooling

AdaptiveAvgPool

Linear Layer

تنظیمات Trin به این صورت است :

Optimizer: Adam

Learning Rate:  $1e-3$

Loss Function: CrossEntropyLoss

Epochs: 10

Mixed Precision Training (AMP)

GradScaler برای جلوگیری از underflow

استفاده از CUDA در صورت وجود

حلقه Trin

اجرای forward pass

محاسبه loss

backward با GradScaler

به روز رسانی وزن ها

ارزیابی روی validation set

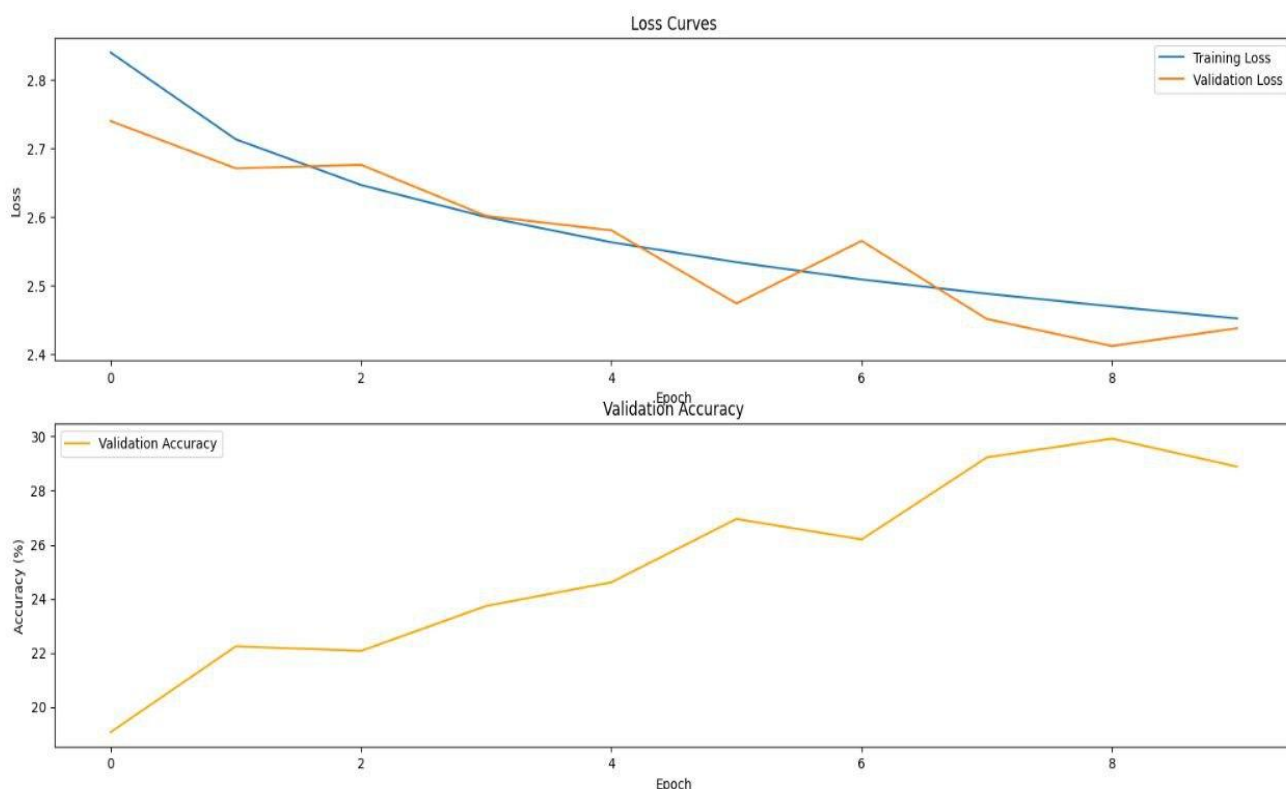
ذخیره یک فایل pth. در انتهای هر اپوک

در هر ایپوک Training Loss / Validation Loss / Validation Accuracy ثبت می شوند

## تحلیل عملکرد

اگر Train loss , Val loss کاهش و Accuracy افزایشی باشد مدل در حال یادگیری صحیح است

اگر Train loss کاهش و Val loss افزایشی باشد اور فیت رخ داده است و اگر هر دو بالا بمانند اندفیت رخ داده



کاهش یکنواخت Training loss در 10 ایپوک مشهاده می شود و نوسان غیر عادی نداریم پس مدل به طور عادی در حال یادگیری است شیب کاهش loss در ایپوک های آخر کم شده است

Val loss در اکثر ایپوک ها کاهش بوده و کمترین مقدار در ایپوک 9 است Accuracy نیز در ابتدا رشد سریع و شدت رشد کم کم کاهش یافته و بهترین مقدار ایپوک 9 به مقدار 30 درصد است

چون اختلاف Train loss , Vall loss اور فیت رخ نداده اما ممکن است اندفیت  
ملایمی رخ داده باشد