

بسم الله الرحمن الرحيم

گزارش کار پروژه - فاز دوم

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

حسین انیسی - 40215203

محمد یوسف شریف - 40119693

امیررضا پاک روح - 40215763

دیتا پایپلاین

در فاز دوم ساختار دیتا پایپلاین باز طراحی شد . برای جلوگیری از تصادفی بودن نتایج مقدار seed ثابت تنظیم شد تا تقسیم داده ها در هر اجرا یکسان باشد و رفتار مدل در فاز train پایدار تر باشد و مقایسه پذیری نتایج بیشتر باشد

تقسیم داده ها به صورت Stratified بر اساس merged category id انجام شد تا نسبت منصفانه ای از هر کلاس برداشته شود . 80 درصد داده ها برای Train 15 درصد برای Validation و 5 درصد برای Test در نظر گرفته شد

برای مقاومت مدل در برابر شرایط مختلف Augmentation به صورت زیر اعمال شد

RandomResizedCrop (224)

RandomHorizontalFlip (p=0.5)

RandomRotation (درجه ± 15)

ColorJitter (روشنایی، کنتراست و اشباع رنگ)

نرمال سازی با مقادیر استاندارد ImageNet

دیتا لودر(واسط بین GPU و دیتاست) برای آموزش رو یک RTX3060 به صورت زیر بهینه سازی شد

Batch Size = 256

Num Workers = 4

فعال سازی pin_memory

فعال سازی persistent_workers

تنظیم prefetch_factor = 2

تلاش کردیم از Sharding و webdataset استفاده کنیم تا سرعت بهبود پیدا کند اما به دلیل باگ های ایجاد شده صرف نظر شد

دیتا لودر از ProductImageDataset استفاده میکند که تصاویر را با استفاده از read imge میخواند و خروجی به صورت imge , lable است

معماری مدل

در این بخش معماری مدل نسبت به بیسلاین بهبود داده ایم به این صورت که از Transfer Learning استفاده میشود تا مدل با تکیه بر ویژگی‌های از قبل یادگرفته‌شده روی ImageNet، با دقت بالاتر و همگرایی سریع‌تر دسته‌بندی تصاویر محصولات را انجام دهد.

آموزش در دو مرحله انجام شده است

1-Warm-up

در ابتدای آموزش کل Backbone فریز می‌شود و فقط لایه FC تا Head سریع با کلاس‌های جدید سازگار شود و وزن‌های pretrained از دست نروند

2- Fine-tuning

در ایپوک دوم لایه 4 از فریز خارج میشود و در ایپوک چهارم لایه سوم نیز فعال می‌شود تا ابتدا بخش سطح بالا بهینه شود و پس از آن فیچرهای عمیق‌تر بهینه شود این روش از اورفیت ناگهانی نیز جلوگیری می‌کند

از CrossEntropyLoss با Label Smoothing 0.1 استفاده شده

برای بخش بهینه‌سازی از AdamW با $\text{weight_decay}=1e-4$ استفاده شده است.

دو گروه نرخ یادگیری (Param Groups) تعریف شده:

نرخ یادگیری بزرگ‌تر: Head (fc)

نرخ یادگیری کوچک‌تر: Backbone (layer4)

این کار باعث می‌شود head سریع‌تر یاد بگیرد و backbone با تغییرات کنترل‌شده fine-tune شود.

نرخ یادگیری بر اساس زمان و به طور تدریجی و نرم تغییر می کند و از CosineAnnealingLR استفاده شده است تا از نوسانات بیش از حد در انتهای آموزش جلوگیری شود و البته این روش در Transfer Learning عملکرد بهتری دارد

در هر اپوک یک چک پوینت ذخیره میشود و اگر Validation Accuracy افزایش یافته باشد به عنوان بهترین مدل ذخیره می شود. نگاشت دسته ها نیز ذخیره میشود.

در نهایت یک ارزیابی انجام میشود و loss , Accuracy, Best Validation Accuracy گزارش می شود و نمودار Train Loss , Validation Loss, Accuracy ذخیره می شود

ارزیابی نهایی کد (Evaluation)

ابتدا معماری مدل بازسازی میشود و وزن های ذخیره شده در Best-Model لود میشود و مدل در حالت eval قرار میگیرد تا Dropout غیر فعال شود و BatchNorm در حالت inference عمل کند

در تابع clean_df اقدامات زیر انجام شده است:

- حذف دسته هایی با کمتر از 25000 نمونه
- نمونه گیری دقیق 25000 تصویر از هر کلاس (Balanced Dataset)
- بازنگاشت شناسه کلاس ها به بازه 0 تا N-1
- حذف تصاویر ناموفق دانلود شده
- حذف مسیرهای نامعتبر

این مورد تعادل داده ها و حفظ لیبل درست داده ها را مدیریت می کند

تقسیم داده ها با همان Random Seed قبل انجام میشود

از هیچ آگمنتیشنی استفاده نمیشود تا مدل صرفا بر اساس تصویر ورودی دسته بندی را حدس بزند

در `torch.no_grad` تصویر را به مدل می دهیم و خروجی `logits` دریافت می شود و با استفاده از `argmax` کلاس حدس زده شده مشخص میشود (بیشترین احتمال) و این اطلاعات ذخیره میشود

با `classification_report` معیار های `Precision, Recall, F1-score, Support` را در یک فایل `txt` ذخیره می شود

تصویر `classification_report` نیز ذخیره می شود

رابط کاربری

در انتهای کار با استفاده از `Gradio` یک دمو برای پروژه طراحی شده تا قابلیت تست عکس های خارج از دیتاست اولیه نیز وجود داشته باشد

مانند بخش قبل مدل نهایی بازسازی میشود و فایل های مورد نیاز لود میشود

`get_class_names` نیز دسته بندی ها را همانطور که در کد های قبلی اسفنده شده بارگزاری میکند

دقیقا مثل مرحله ارزیابی موارد زیر انجام میشود

به `Resize 256`

به `CenterCrop 224`

به `Tensor` تبدیل

با `ImageNet` میانگین و انحراف معیار `Normalize`

در `classify_product` تصویر از `Gradio` خوانده میشود و به فرمت `PIL` تبدیل میکنیم سپس ترنسفورم را اعمال و به مدل تحویل می دهیم

با استفاده از `Softmax` احتمال هر کلاس محاسبه می شود و خروجی به صورت یک دیکشنری که شامل نام کلاس و احتمال آن است برگردانده میشود

سه کلاسی که بیشترین احتمال را دارند نمایش می دهیم