



دانشگاه تهران
دانشکده‌گان فارابی
دانشکده‌ی مهندسی
گروه مهندسی کامپیوتر

طراحی و پیاده سازی موتور پردازش سامانه‌ی بازشناسی محتوای کارت ملی با استفاده از بینایی ماشین

نگارش:

محمد عرفان نایب آقایی

استاد راهنما:

دکتر کاظم فولادی

گزارش پروژه برای دریافت درجه‌ی کارشناسی
در رشته‌ی مهندسی کامپیوتر

شهریور ۱۴۰۲

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

فرم صورت جلسه داورى

تقدیم

به نام خداوند بخشنده و مهربان

با افتخار و افتخار بی‌نهایت، این پروژه را به مادر و پدرم اختصاص می‌دهم. از شما، که همواره در تمامی لحظات زندگی، من را با عشق و مهر خود در کنار خود داشته‌اید و تشویق‌ها و پشتیبانی‌هایتان همچون نوری در راهم بوده است، سپاسگزارم. این پروژه، نمایانگر دستاوردها و انگیزه‌هایی است که از شما به ارث برده‌ام.

محمد عرفان نایب آقایی

به نام پروردگار آسمان‌ها

تقدیم به مادر و پدر عزیزم که با عشق و پشتکار بی‌پایانشان همواره پشتیبان و منشی من در تمامی مراحل زندگی بوده‌اند. این پروژه، نتیجه زحمات و هدایت‌های شماست که همواره به من انگیزه و انرژی داده‌اید تا به بهترین خود برسم.

امیرحسین علی‌پور

تابستان ۱۴۰۲

تشکر و قدردانی

به نام عظیم و خداوندی که بخشنده و رحیم است

از اینکه به عنوان استاد راهنما در این سفر دانشی مرا همراهی کردید، بسیار قدردانی می‌کنم. دوران تحصیلی من، تحت هدایت‌ها و توجه‌های شما، به یک دورهٔ پربار و ممتع تبدیل شد. از شما یاد گرفتم که در مسیر یادگیری، انگیزه و پشتکار، دو راهنمای اصلی برای کسب دانش و موفقیت هستند.

تشکر از شما بخاطر توجه‌های پژوهشی و پیشنهادات سازنده‌تان که به من کمک کرد تا دستاوردهای نوآورانه‌ای در این پروژه داشته باشم. ترکیب تخصص فنی‌تان با طرح‌های کاربردی و مسئله‌مداری، به من امکان مهارت‌های جدید و ارزشمندی را ارائه کرد.

از شما بخاطر انگیزه‌بخشی‌های خود در مواجهه با چالش‌ها و مشکلات در این پروژه تشکر می‌کنم. شما همواره در دسترس بودید و انگیزه‌های ارزشمندتان، برای من به عنوان یک راهنما و مربی بی‌بدیل عمل کرد.

با تشکر و امتنان بی‌پایان،

محمد عرفان نایب آقایی، امیرحسین علی‌پور

تابستان ۱۴۰۲

چکیده

در این پروژه، ما به بررسی و تحلیل یک رویکرد جدید برای استخراج اطلاعات از کارتهای ملی با استفاده از مدل‌های بینایی کامپیوتر و ابزار Optical Character Recognition (OCR) یا به اختصار Tesseract پرداختیم. ابتدا، ما به جمع‌آوری و پیش‌پردازش داده‌های مرتبط با کارتهای ملی پرداخته و داده‌های بهبود یافته را به عنوان دیتاست آموزشی و ارزیابی معرفی کردیم. مرحله اول این پروژه شامل لیبل‌زدن تصاویر است که در آن، اطلاعات مرتبط با مناطق مختلف تصاویر با استفاده از ابزار Label-Studio تعیین و لیبل می‌شوند. این فرآیند از اهمیت بسزایی برخوردار است زیرا دقت و کیفیت لیبل‌زدن مستقیماً بر کارایی مدل‌های بینایی کامپیوتری و استخراج اطلاعات تأثیر می‌گذارد. در مرحله بعد، از مدل‌های مبتنی بر الگوریتم YOLOv5 برای تشخیص و دسته‌بندی اشیاء در تصاویر کارت ملی استفاده کردیم. این مدل‌ها با توانایی شناسایی اجسام به صورت موثر، قادر به تشخیص عناصر مختلف کارت ملی از جمله شماره ملی، نام، نام خانوادگی، تاریخ تولد و سایر اطلاعات مرتبط بودند. فرآیند پیش‌پردازش نیز نقش مهمی در بهبود دقت استخراج اطلاعات از تصاویر اسکن شده داشت. ما از تکنیک‌های مختلفی از جمله تغییر اندازه‌گیری، حذف نویز، اصلاح چرخش و حذف خطوط استفاده کردیم تا تصاویر بهبود یافته به مرحله تشخیص اطلاعات پردازش شوند. از آنجا که متن استخراج شده از تصاویر ممکن است حاوی اشکالات باشد، ما از ابزار Tesseract برای تشخیص و تبدیل متن تصاویر به متن قابل خواندن استفاده نمودیم. همچنین، ما تکنیک‌های پاک‌سازی متن را برای حذف کاراکترهای اضافی و بهبود خوانایی متن استخراج شده به کار بردیم. نتایج به دست آمده از این پروژه نشان داد که رویکردهای ارائه شده می‌توانند با دقت قابل قبولی اطلاعات موجود در کارتهای ملی را استخراج کرده و دسته‌بندی کنند. علاوه بر این، مقایسه میان مدل‌های YOLOv5 و YOLOv7 به ما نشان داد که YOLOv5 با دقت بالاتر و سرعت بهتری در تشخیص اشیاء عمل می‌کند. در اتمام، این پروژه به عنوان یک نمونه موفقیت‌آمیز از ترکیب تکنیک‌های تشخیص اشیاء و استخراج متن با استفاده از مدل‌های بینایی کامپیوتر و ابزار Tesseract می‌باشد و دقت و سرعت استخراج اطلاعات از تصویر را افزایش می‌دهد.

کلیدواژه‌ها: استخراج اطلاعات، لیبل، YOLOv5، کارت ملی، بینایی ماشین، استخراج متن، استخراج متن،
Training, OCR

فهرست مطالب

فصل اول: مقدمه	1
۱-۱ معرفی پروژه	۱
۱-۲ اهداف پروژه	۱
۱-۳ خروجی مورد انتظار پروژه	۲
فصل دوم: جمع آوری مجموعه داده و لیبل زدن	۵
۲-۱ جمع آوری مجموعه داده	۵
۲-۲ لیبل زدن	۵
فصل سوم: TRAINING	۹
۳-۱ بخش YOLO	۹
۳-۲ بخش آموزش	۱۰
۱-۲-۳ Import	۱۰
۲-۲-۳ Train	۱۱
۳-۲-۳ Detect	۲۱
فصل چهارم: OCR	۳۵
۴-۱ IMPORT	۳۵
۴-۲ بارگذاری مدل	۳۷
۴-۳ بریدن بخش های شناسایی شده توسط مدل	۳۸
۴-۴ شناسایی مسیر بخش های بریده شده	۳۹
۴-۵ PREPROCESS	۴۲
۴-۶ آموزش فونت کارت ملی به TESSERACT	۵۲
۴-۷ تابع OCR	۶۵
فصل پنجم: خلاصه و نتیجه گیری	۶۷
۵-۱ خلاصه	۶۷
۵-۲ نتیجه گیری	۶۹
فصل ششم: کارهای آینده	۷۳

۲۵.....منابع و مراجع

۷۷.....پیوست‌ها

فصل اول : مقدمه

۱-۱ معرفی پروژه

در این پروژه، قصد داریم از طریق استفاده از تکنیک‌های بینایی کامپیوتر و یادگیری عمیق، اطلاعات موجود در کارت ملی را به‌طور اتوماتیک و دقیق استخراج کنیم. با استفاده از شبکه‌های عصبی کانوایولوشنی (CNN) و تکنیک‌های پردازش تصویر، امکان تشخیص و استخراج اطلاعات از تصاویر کارت ملی به‌طور خودکار و بدون نیاز به دخالت دستی فراهم می‌شود.

مراحل اصلی پروژه:

۱. جمع‌آوری داده‌ها: برای آموزش مدل، نیاز به دسترسی به مجموعه‌ای از تصاویر کارتهای ملی با اطلاعات متفاوت داریم. این تصاویر باید شامل تصاویر با کیفیت متفاوت و متغیره باشند.
۲. پیش‌پردازش تصاویر: تصاویر جمع‌آوری شده باید قبل از ورود به مدل پیش‌پردازش شوند. این مرحله شامل تغییر اندازه، تصحیح روشنایی و کاهش نویز است.
۳. طراحی و آموزش مدل: این بخش شامل انتخاب یک معماری مناسب از شبکه‌های عصبی کانوایولوشنی برای تشخیص و استخراج اطلاعات از تصاویر است. مثلاً می‌توانید یک شبکه CNN را برای تشخیص شماره ملی، نام و نام خانوادگی، تاریخ تولد و... آموزش دهید.
۴. تست و ارزیابی: پس از آموزش مدل، باید آن را بر روی داده‌های تست ارزیابی کرده و عملکرد آن را از نظر دقت، صحت و سرعت بررسی کنید.
۵. استخراج اطلاعات در واقعیت: پس از آموزش و ارزیابی مدل، می‌توانید آن را برای استخراج اطلاعات از تصاویر کارتهای ملی واقعی استفاده کنید. مدل باید بتواند اطلاعات مورد نیاز را از تصاویر استخراج کرده و به‌طور خودکار نمایش دهد.
۶. رابط کاربری: در نهایت، می‌توانید یک رابط کاربری ساده طراحی کنید که به کاربران اجازه دهد تصاویر کارت ملی خود را بارگذاری کنند و اطلاعات مورد نیاز را از مدل استخراج کنند.

۲-۱ اهداف پروژه

پروژه استخراج اطلاعات کارت ملی با استفاده از مدل‌های بینایی کامپیوتر اهداف متعددی را دنبال می‌کند که به طور جامع در زیر توضیح داده شده‌اند:

۱. تشخیص اجزای کارت ملی: یکی از اهداف اصلی این پروژه تشخیص و تمییزبخشی اجزای مختلف کارت ملی است. این شامل تشخیص محل شماره ملی، نام و نام خانوادگی، تصویر شناسنامه و سایر اطلاعات مهم موجود در کارت ملی می‌شود.

۲. استخراج اطلاعات دقیق: با استفاده از مدل‌های بینایی کامپیوتر، هدف دقیق‌ترین استخراج اطلاعات از تصویر کارت ملی است. این اطلاعات می‌توانند شماره ملی، نام، نام خانوادگی، تاریخ تولد و سایر مشخصات فردی باشند.

۳. دستیابی به دقت بالا: یکی از چالش‌های اصلی در این پروژه دستیابی به دقت بالا در تشخیص و استخراج اطلاعات از تصاویر است. هدف این است که مدل با دقت بسیار بالا و بدون اشتباه به تشخیص و استخراج اطلاعات بپردازد.

۴. مقاومت در برابر تغییرات: پروژه باید به‌طور موثر با تغییرات مختلف در تصاویر کارت ملی مانند تغییر زاویه دید، نورپردازی متفاوت و کیفیت متغیر مقابله کند.

۵. حفظ حریم خصوصی: یکی از اهمیت‌های اصلی این پروژه، حفظ حریم خصوصی افراد است. تضمین محرمانگی و امنیت اطلاعات شخصی که از تصاویر استخراج می‌شوند، بسیار حیاتی است.

۶. کاهش زمان و تلاش انسانی: این پروژه به انسان‌ها کمک می‌کند تا از تلاش زیادی برای وارد کردن دستی اطلاعات کارت ملی جلوگیری کنند. این باعث صرفه‌جویی در زمان و انرژی انسانی می‌شود.

۷. کاربردهای وسیع: نتایج این پروژه می‌توانند در انواع فرآیندهای احراز هویت، ثبت‌نام، تایید هویت در دنیای دیجیتال و سایر فعالیت‌های مشابه مورد استفاده قرار گیرند.

۸. توسعه فناوری: این پروژه به عنوان یک زمینه جذاب برای توسعه فناوری‌های پردازش تصویر و یادگیری عمیق در حوزه هوش مصنوعی می‌تواند عمل کند.

۹. تحقیقات پژوهشی: پروژه می‌تواند به عنوان موضوعی برای تحقیقات و پژوهش‌های آینده در زمینه پردازش تصویر، تشخیص الگو و هوش مصنوعی عمل کند.

با ترکیب این اهداف، پروژه استخراج اطلاعات کارت ملی با استفاده از مدل‌های بینایی کامپیوتر به عنوان یک پروژه کاربردی و پیشرفته می‌تواند به حل چالش‌های مختلف در حوزه احراز هویت و مدیریت اطلاعات شخصی کمک کند.

۳-۱ خروجی مورد انتظار پروژه

خروجی مورد انتظار از این پروژه، مجموعه‌ای از اطلاعات مهم و حیاتی از کارت ملی فرد مورد نظر است که از تصویر کارت ملی استخراج می‌شوند. این اطلاعات شامل جزئیات مختلفی از شخصیت فرد هستند که به طور معمول در هر کارت ملی وجود دارند. در زیر توضیحات دقیق‌تری از هر یک از این اطلاعات آمده است:

1. شماره ملی: شماره ملی یک عدد دسته‌ای است که به هر فرد در کشور اختصاص دارد. این شماره به عنوان یک شناسه منحصر به فرد استفاده می‌شود و در فرآیندهای مختلف از احراز هویت تا تعیین حقوق و وظایف اجتماعی استفاده می‌شود.
 2. تصویر کارت ملی: این تصویر نمایان‌گر نمونه‌ای واقعی از کارت ملی فرد است. این تصویر معمولاً به شکل دیجیتالی نمایش داده می‌شود و می‌تواند تشخیص چهره فرد را نیز شامل شود.
 3. نام و نام خانوادگی: نام کامل و دقیق فرد صاحب کارت ملی. این اطلاعات به عنوان هویت فرد در تمامی موارد احراز هویت و تایید مشخصات استفاده می‌شود.
 4. تاریخ تولد: تاریخ تولد فرد به صورت کامل با سال، ماه و روز ارائه می‌شود. این تاریخ به عنوان عاملی اساسی در تعیین سن و هویت فرد استفاده می‌شود.
 5. نام پدر: این اطلاعات به عنوان جزء دیگری از شناسایی فردی مورد استفاده قرار می‌گیرند.
 6. پایان اعتبار کارت ملی: تاریخی که نشان‌دهنده پایان اعتبار کارت ملی است. این تاریخ نشان‌دهنده زمانی است که کارت ملی فرد منقضی می‌شود یا نیاز به تجدید شده است.
- با جمع‌آوری و استخراج این اطلاعات از تصویر کارت ملی، خروجی نهایی پروژه به صورت یک مجموعه دقیق و کامل از اطلاعات شخصی فرد خواهد بود. این اطلاعات می‌توانند در فرآیندهای مختلف احراز هویت، ثبت‌نام، تجزیه و تحلیل اطلاعات شخصی و سایر فعالیت‌ها مورد استفاده قرار گیرند.

فصل دوم: جمع آوری مجموعه داده و لیبل زدن

۱-۲ جمع آوری مجموعه داده

با استفاده از دریافت تصاویر کارت ملی (۵۰ عدد) از یک شرکت و جمع آوری تصاویر کارت ملی دوستان و اطرافیان (۱۵۰ عدد) اطلاعات مورد نیاز برای ادامه کار و لیبل زدن را جمع آوری کردیم و پس از آن به لیبل زدن، پرداختیم.

۲-۲ لیبل زدن

با استفاده از لیبل استودیو شروع به لیبل زدن برای تصاویر میکنیم.

لیبل استودیو (Label Studio) به عنوان یک چارچوب منبع باز برچسب گذاری و تشخیص اطلاعات در داده های تصویری و متنی، ابزاری قدرتمند است که برای توسعه دهندگان و تیم های متخصص در زمینه هایی از جمله پردازش تصویر، پردازش متن، و یادگیری عمیق ارزشمند است. این ابزار به شما امکان می دهد داده های مختلف را با هدف برچسب گذاری و تشخیص اطلاعات، به صورت تعاملی مدیریت کنید.

لیبل استودیو به ویژه در پروژه هایی با محتوای چندرسانه ای یا متنی که نیاز به تشخیص و برچسب گذاری الگوها، اجسام یا اطلاعات مشخص دارند، بسیار موثر و کاربردی است. با استفاده از یک رابط کاربری ساده و متشکل از ابزارهای تعاملی، تیم ها و توسعه دهندگان می توانند به راحتی داده ها را برچسب گذاری کرده و اطلاعات مرتبط با آنها را تشخیص دهند.

ویژگی ها و امکانات برجسته ای که لیبل استودیو ارائه می کند عبارتند از:

۱. پشتیبانی از چند منبع داده: این ابزار قابلیت استفاده از چندین منبع داده متنوع مثل تصاویر، متن ها و داده های مختلف دیگر را فراهم می کند. این امر به شما اجازه می دهد تا برای پروژه های مختلف از انواع مختلف داده استفاده کنید.

۲. سفارشی سازی و تنظیمات چندرسانه ای: واسط کاربری لیبل استودیو قابلیت سفارشی سازی را ارائه می دهد، به طوری که می توانید آن را به نیازهای خاص پروژه تنظیم کنید. متون توضیحی، نمونه های ارجاع و دستورات مربوط به برچسب گذاری و تشخیص را اضافه کرده و تنظیمات را بهینه کنید.

۳. ارتباط با مدل‌های یادگیری عمیق: لیبل استودیو امکان اتصال به مدل‌های یادگیری عمیق را دارد. این به این معناست که می‌توانید خروجی‌های برچسب‌گذاری شده را به مدل‌ها وارد کرده و نتایج پس از تحلیل به مدل‌ها بازخورد دهید.

۴. پشتیبانی از چندین کاربر: این ابزار به چندین کاربر اجازه می‌دهد که به صورت همزمان در پروژه‌های برچسب‌گذاری شرکت کنند. این امر می‌تواند همکاری و هماهنگی بین اعضای تیم را تسهیل کند.

۵. پشتیبانی از انواع تسک‌ها و مهارت‌ها: لیبل استودیو انواع مختلف تسک‌های برچسب‌گذاری و تشخیص را پشتیبانی می‌کند. این تسک‌ها شامل برچسب‌گذاری تصاویر، تشخیص الگو در تصاویر، تحلیل متن و تصویر، وظایف مرتبط با پردازش داده و سایر موارد می‌شود.

۶. مدیریت دقیق داده‌ها و تگ‌ها: این ابزار به شما امکان می‌دهد داده‌ها را با استفاده از تگ‌ها و برچسب‌ها دسته‌بندی و مدیریت کنید. این امر می‌تواند به مرتب‌سازی و تجزیه‌تحلیل بهتر داده‌ها کمک کند.

۷. امکان ذخیره داده‌های برچسب‌گذاری شده: داده‌های برچسب‌گذاری شده به صورت یک پرونده مجزا ذخیره می‌شوند. این امر به شما این امکان را می‌دهد تا داده‌های برچسب‌گذاری شده را در مراحل بعدی پروژه مورد استفاده قرار دهید.

۸. پشتیبانی از انواع تصاویر و فرمت‌ها: لیبل استودیو از انواع مختلف تصاویر از جمله تصاویر دو بعدی، تصاویر پانوراما و تصاویر ماهواره‌ای پشتیبانی می‌کند.

با این توضیحات، می‌توان نتیجه گرفت که لیبل استودیو به عنوان یک ابزار کارآمد، انعطاف‌پذیر و کامل برای برچسب‌گذاری و تشخیص اطلاعات در داده‌های تصویری و متنی بسیار مناسب است. با توجه به امکانات متنوع آن، این ابزار می‌تواند به توسعه‌دهندگان و تیم‌های متخصص در زمینه‌های مختلف بهبود و کمک کند.

سپس با انجام مراحل زیر روند کامل و دقیق برچسب‌زدن ۷ لیبل (تصویر کارت ملی، شماره ملی، نام، تاریخ تولد، نام خانوادگی، نام پدر و پایان اعتبار) در لیبل استودیو را انجام می‌دهیم:

اولین مرحله بعد از نصب لیبل استودیو، شما باید یک پروژه جدید ایجاد میکنیم. این پروژه به عنوان یک محیط مجزا برای برچسب‌گذاری و مدیریت داده‌های خود عمل می‌کند. نوع داده‌هایی که می‌خواهیم برچسب بزنیم را تعیین می‌کنیم، در اینجا تصاویر کارت‌های ملی. سپس در مرحله بعدی، داده‌های تصاویر کارت ملی مختلف را به پروژه اضافه می‌کنیم. می‌توانیم از راه‌های مختلفی مانند بارگذاری دستی یا اتصال به منابع داده خارجی از جمله پایگاه‌های داده استفاده کنیم. در مرحله بعدی تعریف لیبل‌ها را انجام می‌دهیم، برای هر یک از ۷ لیبل مذکور (تصویر کارت ملی، شماره ملی، نام، تاریخ تولد، نام خانوادگی،

نام پدر، پایان اعتبار) یک برچسب تعریف می‌کنیم. این برچسب‌ها به عنوان شناسه‌ها یا نماینده‌های مختلف اطلاعات که قصد داریم در تصاویر برچسب بزنیم، عمل می‌کنند. با انجام مراحل قبلی به مرحله شروع برچسب‌گذاری می‌رسیم. حالا می‌توانیم به واسطه تصاویر کارت ملی برخوردهای برچسب‌گذاری کنیم. برای هر تصویر، اطلاعات مذکور مانند شماره ملی، نام، تاریخ تولد و سایر اطلاعات را با استفاده از

برچسب‌های تعریف شده مشخص می‌کنیم. حالا باید نتایج را ذخیره کنیم. با هر برچسب‌گذاری انجام شده، نتایج به طور خودکار ذخیره می‌شوند. این اطلاعات می‌توانند به عنوان فایل‌های گزارش یا پایگاه‌های داده‌ای برای مراحل بعدی مورد استفاده قرار بگیرند. در گام بعدی تحلیل و استفاده از نتایج را داریم که با داده‌های برچسب‌گذاری شده، می‌توانیم تجزیه‌تحلیل‌های مختلفی انجام دهیم. این نتایج ممکن است بهبود مدل‌های یادگیری عمیق خود، تجزیه‌تحلیل مشخصات جمعیتی یا مهارت‌ها کمک کنند.

این فرآیند به ما امکان می‌دهد اطلاعات مورد نیاز خود را از داده‌ها استخراج کرده و مدل‌های یادگیری ماشینی را بهبود ببخشیم.

بعد از انجام مراحل بالا یک خروجی Yolo می‌گیریم که در بخش Train آن را توضیح می‌دهیم و ویژگی‌ها و مشخصات آن را ذکر می‌کنیم.

بعد از آنکه خروجی yolo را دریافت کردیم این خروجی به ما ۲ فولدر می‌دهد. یکی از فولدرها images و دیگری labels است برای هر کدام از این فولدرها دوباره ۲ عدد فولدر (Train, val) درست می‌کنیم. labels فایل text هستند و اسم آن‌ها دقیقا با اسم عکس‌ها یکسان می‌باشد. در فایل‌های label مختصات لیبل‌ها به صورت یک آرایه مشخص شده است. که ۷ تا ردیف برای هر عکس وجود دارد که داخل هر کدام ۴ تا مختصات دارد که هر کدام از ردیف‌ها برای یکی از ۷ لیبل ما می‌باشد.

فصل سوم: Training

بعد از طی مراحل گفته شده به مرحله training رسیدیم که در این مرحله ما با استفاده از YOLO، train کردیم.

۱-۳ بخش YOLO

YOLO (You Only Look Once) ورژن ۵: تشخیص اشیاء با دقت و سرعت بالا

YOLO یک مدل شناخته شده در زمینه تشخیص اشیاء در تصاویر و ویدئوهاست. ورژن ۵ یک ارتقاء از مدل YOLO است که به دقت بیشتر و سرعت بهتر در تشخیص اشیاء می پردازد. این مدل توسط تیم تحقیقاتی در حوزه یادگیری عمیق توسعه داده شده است.

ویژگی های کلیدی YOLO v5:

۱. سرعت بالا و تشخیص به صورت بهینه: یکی از ویژگی های مهم YOLO v5، سرعت بسیار بالا در تشخیص اشیاء است. به دلیل معماری بهینه سازی شده و بهبودهای سیستمی، مدل توانسته است بهبود قابل توجهی در سرعت پردازش داشته باشد.

۲. دقت بهبود یافته: علاوه بر سرعت، YOLO v5 نیز به بهبود دقت در تشخیص اشیاء توجه داشته است. این بهبود دقت می تواند از طریق تغییرات در معماری شبکه، تکنیک های آموزش مدل و بهره برداری از ترفندهای جدید حاصل شده باشد.

۳. معماری بهینه شده: YOLO v5 از معماری جدید و بهینه شده ای برخوردار است که ترکیبی از مدل های EfficientDet و YOLO است. این معماری باعث افزایش کارایی و کاربردی بودن مدل شده است.

۴. تناسب با دستگاه های مختلف: YOLO v5 از معماری انعطاف پذیری برخوردار است که به طور پیش فرض بر روی دستگاه های مختلف اجرا می شود. این ویژگی به توسعه دهندگان اجازه می دهد که از مدل برای کاربردهای مختلف مانند امنیت، رباتیک، خودروهای خودران و غیره استفاده کنند.

۵. مشخصه های اختصاصی: YOLO v5 از مشخصه های اختصاصی مانند PANet (Path Aggregation Network) استفاده می کند. این مشخصه ها باعث بهبود دقت تشخیص اشیاء و مشکل اشتباهات احتمالی در تصاویر می شوند.

تحولات و پیشرفت ها:

مدل YOLO v5 یک مثال از تحقیق و توسعه پیشرفته در حوزه یادگیری عمیق است که به دقت بیشتر و سرعت بهتر در تشخیص اشیاء توجه دارد. توسعه‌دهندگان در تلاشند که با بهبود مدل‌ها و اعمال تغییرات جدید، تجربه بهتری را برای کاربران فراهم کنند.

۳-۲ بخش آموزش

Import ۱-۲-۳

```
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
!pip install -qr requirements.txt comet_ml # install

import torch
import utils
display = utils.notebook_init() # checks
```

تصویر ۳-۱ نصب پیکربندی‌ها و وارد کردن ماژول‌ها

- کلون مخزن: این دستور با استفاده از ابزار کامند لاین در محیطی که اجرا می‌کنید، مخزن مدل YOLO v5 را از آدرس گیت‌هاب "<https://github.com/ultralytics/yolov5>" کلون (کپی) می‌کند. این کار به شما امکان می‌دهد که به تمام کد و فایل‌های مرتبط با این مخزن دسترسی پیدا کنید.
 - تغییر مسیر: این دستور داخل مخزن کلون شده "yolov5" وارد می‌شود. به این ترتیب، تمام دستورات بعدی در مسیر این مخزن اجرا خواهند شد.
 - نصب پیکربندی‌ها: در این قسمت، پیکربندی‌ها و ماژول‌های مورد نیاز برای اجرای مدل YOLO v5 نصب می‌شوند. فایل "requirements.txt" حاوی لیستی از ماژول‌ها و ویژگی‌های مورد نیاز برای اجرای مدل است. در اینجا، پس از نصب پیکربندی‌های مورد نیاز، ماژول "comet_ml" نیز نصب می‌شود که اینجا به عنوان محیطی برای رصد و مدیریت آزمایش‌های یادگیری ماشین استفاده می‌شود.
 - وارد کردن ماژول‌ها: در این مرحله، ماژول‌های مورد نیاز برای اجرای مدل YOLO v5 وارد می‌شوند. همچنین ماژول "utils" که مربوط به کاربردهای مختلف و کمکی در مدل YOLO است، وارد می‌شود. سپس با استفاده از تابع "notebook_init" مشخص می‌شود که آیا محیط اجرایی از نوع نوت‌بوک است یا نه، و اگر اینطور باشد، محیط چشم‌اندازی ایجاد می‌شود.
- با اجرای این دستورات، شما محیطی آماده دارید که می‌توانید از آن برای اجرای و آزمایش مدل YOLO v5 استفاده کنید. این کارها به شما امکان می‌دهد تا به راحتی مدل را آموزش دهید، داده‌های تست را تشخیص دهید و نتایج را مشاهده کنید.

Train ۲-۲-۳

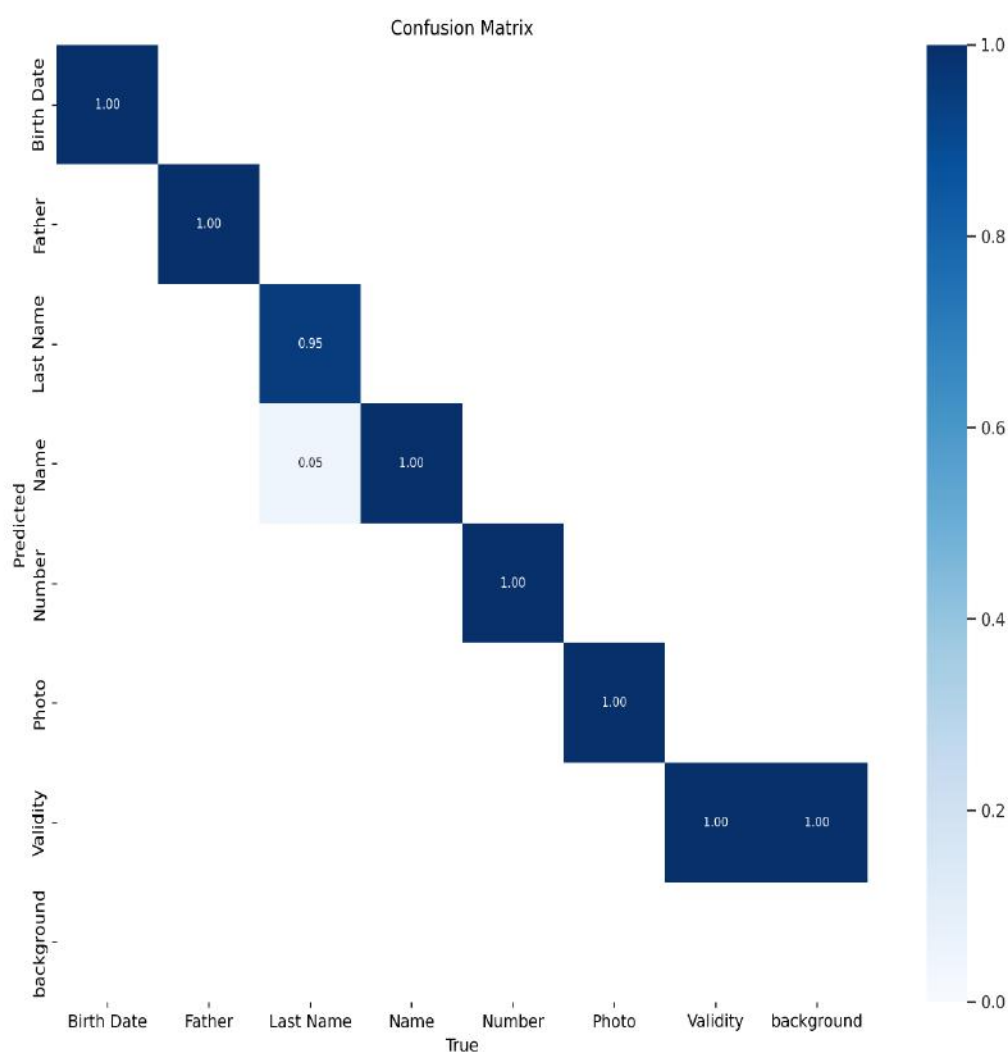
```
python train.py --img 1024 --batch 8 --epochs 100 --data custom_data.yaml --weights yolov5s.pt --cache
```

تصویر ۲-۳ دستور اجرای فایل Train

کد بالا با استفاده از دستورات زبان برنامه‌نویسی پایتون یک مدل YOLO v5 را برای آموزش با داده‌های سفارشی شما تنظیم می‌کند. در اینجا هر پارامتر و دستوری را به طور جداگانه توضیح می‌دهم:

- **python train.py!**: این دستور به معنی اجرای فایل "train.py" با استفاده از مفسر پایتون است. با اجرای این دستور، مراحل آموزش مدل YOLO v5 آغاز می‌شود.
- **--img 1024**: این پارامتر نشان می‌دهد که تصاویر ورودی به مدل چه ابعادی داشته باشند. در اینجا تصاویر به ابعاد ۱۰۲۴x۱۰۲۴ پیکسل تنظیم می‌شوند.
- **--batch 8**: در مدل‌های یادگیری عمیق، آموزش معمولاً به صورت دسته به دسته (batch-wise) انجام می‌شود. این به این معنی است که به جای آموزش شبکه روی تصاویر یکی یکی، داده‌ها به گروه‌هایی با تعداد تعیین شده تصویر (دسته) تقسیم می‌شوند و شبکه روی هر دسته‌ای از داده‌ها به صورت همزمان آموزش می‌بیند. این پارامتر تعیین می‌کند که هر دسته آموزش (بچ) چند تصویر در خود داشته باشد. در اینجا هر بچ شامل ۸ تصویر است.
- **--epochs 100**: این پارامتر تعیین می‌کند که مراحل آموزش به مدت چند دوره اجرا شود. ابتدا با ۱۰ دوره آموزش را انجام دادیم ولی نتیجه دلخواه را نگرفتیم به همین منظور دوره آموزش را ۱۰ تا ۱۰ افزایش دادیم تا به دوره ۱۰۰ رسیدیم که نتیجه دلخواه را گرفتیم.
- **--data custom_data.yaml**: این پارامتر مشخص می‌کند که فایل پیکربندی مربوط به داده‌های سفارشی شما چیست. در اینجا فایل "custom_data.yaml" که اطلاعات مربوط به مسیرها و تنظیمات داده‌ها را دارد، استفاده می‌شود. این فایل شامل مسیر مورد نظر برای Train و همچنین برای تست (Val) می‌باشد. علاوه بر آن یک آرایه به اسم name شامل اسامی لیبل‌ها نیز دارد.
- **--weights yolov5s.pt**: در یادگیری عمیق، معمولاً از مدل‌های پیش‌آموزش‌شده به عنوان ابتدایی برای آموزش مدل خود استفاده می‌شود. این مدل‌های پیش‌آموزش‌شده را می‌توان از طریق پارامتر **weights** به مدل داد. این دستور به مدل نشان می‌دهد که از وزن‌های اولیه مدل "yolov5s.pt" به عنوان نقطه شروع برای آموزش استفاده کند. وزن‌های اولیه مدل‌ها معمولاً از روی دیتاست‌های بزرگ و معتبری یادگیری شده‌اند و در طی فرآیند پیش‌آموزش، قابلیت‌های مهمی در تشخیص الگوها را یاد گرفته‌اند. با استفاده از این وزن‌ها، مدل از ابتدای آموزش به نقطه بهتری از فضای جستجوی مدل‌ها شروع می‌کند و ممکن است به سرعت به جواب‌های بهتری برسد.
- **--cache**: در مواردی که مدل‌های یادگیری عمیق بر روی داده‌های حجیم آموزش داده می‌شوند، استفاده از حافظه کش (Cache) می‌تواند کمک کننده باشد. Cache به طور خلاصه یک منطقه از حافظه

کامپیوتر است که داده‌ها به صورت موقتی در آن ذخیره می‌شوند تا دسترسی سریع‌تر و بهبود سرعت پردازش داده‌ها را فراهم کند. پارامتر `--cache` به مدل اجازه می‌دهد که از حافظه کش جهت ذخیره و مدیریت داده‌های میانی در طول فرآیند آموزش استفاده کند. این کار باعث می‌شود که داده‌های محاسباتی در حافظه موقتی ذخیره شده و از دیسک به صورت مکرر خوانده نشوند که سرعت پردازش را افزایش می‌دهد. استفاده از `Cache` به ویژه زمانی مفید است که دسترسی به داده‌های اصلی (مثلاً تصاویر آموزشی) از دیسک طولانی و زمان‌بر باشد. با داشتن داده‌ها در حافظه کش، این دسترسی‌های زمان‌بر کاهش می‌یابد و مدل با سرعت بیشتری می‌تواند به آموزش بپردازد.



تصویر ۳-۳ Confusion Matrix

ماتریس اشتباهات (Confusion Matrix) یک ابزار ارزیابی در زمینه مدل‌های دسته‌بندی است که به تحلیل دقیق‌تر عملکرد مدل بر روی داده‌های آزمایشی کمک می‌کند. این ماتریس به صورت جدولی

نشان می‌دهد که مدل چه تعداد داده‌های مثبت و منفی را به درستی تشخیص داده و چه تعداد را به اشتباه تشخیص داده است.

از این ماتریس می‌توان معیارهای مختلفی را محاسبه کرد که به تحلیل عملکرد مدل کمک می‌کنند، از جمله:

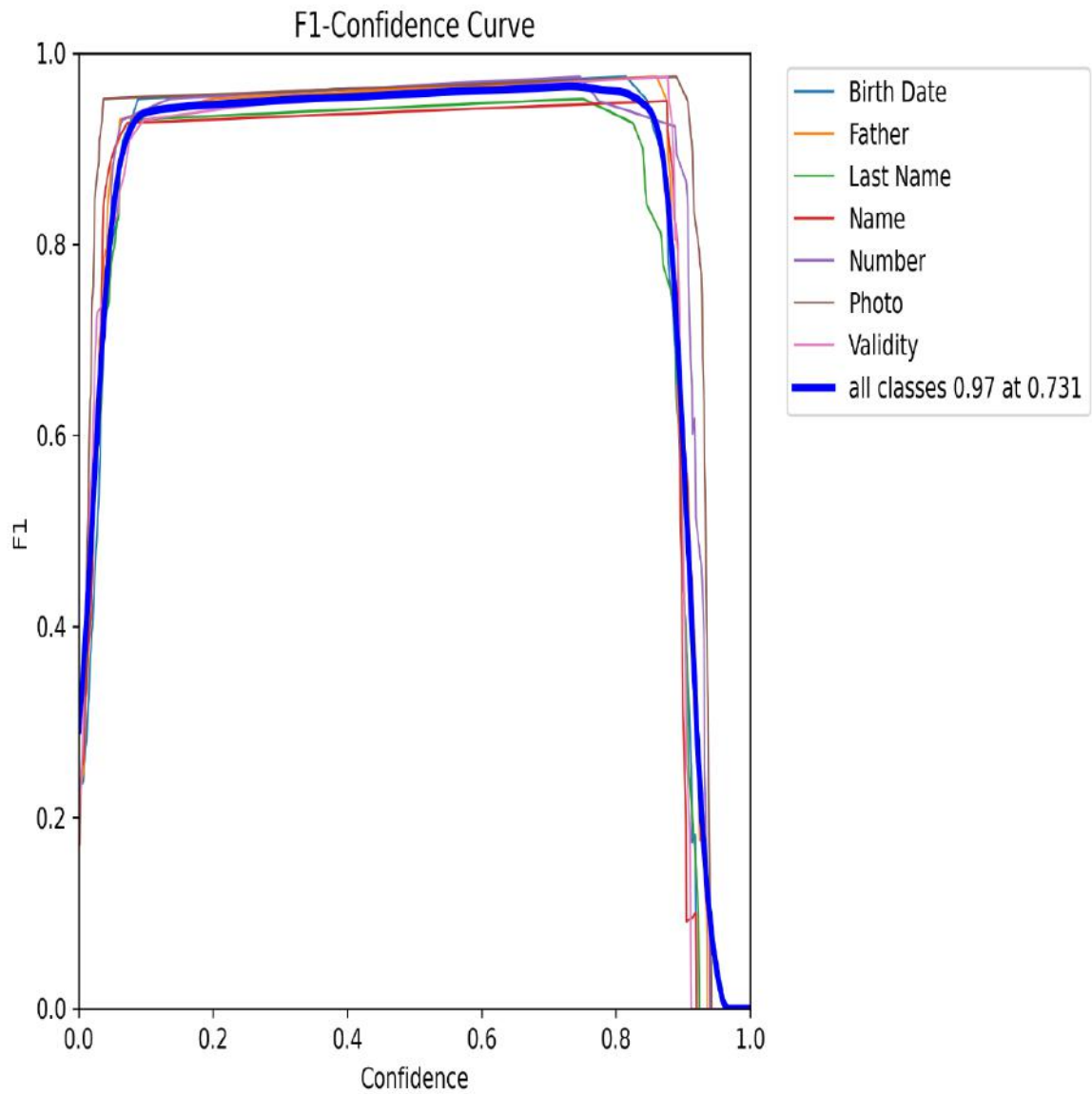
- دقت (Precision): نسبت تعداد مثبت‌های درست به تعداد کل مثبت‌های تشخیص داده شده توسط مدل. دقت نشان می‌دهد که از تمام مواردی که مدل مثبت تشخیص داده است، چه تعدادی واقعاً مثبت بوده‌اند.

- بازیابی (Recall): نسبت تعداد مثبت‌های درست به تعداد کل مثبت‌های واقعی. بازیابی نشان می‌دهد که مدل چه تعداد از مثبت‌های واقعی را تشخیص داده و به مثبت‌هایی که در داده‌های واقعی وجود دارند، پاسخ داده است.

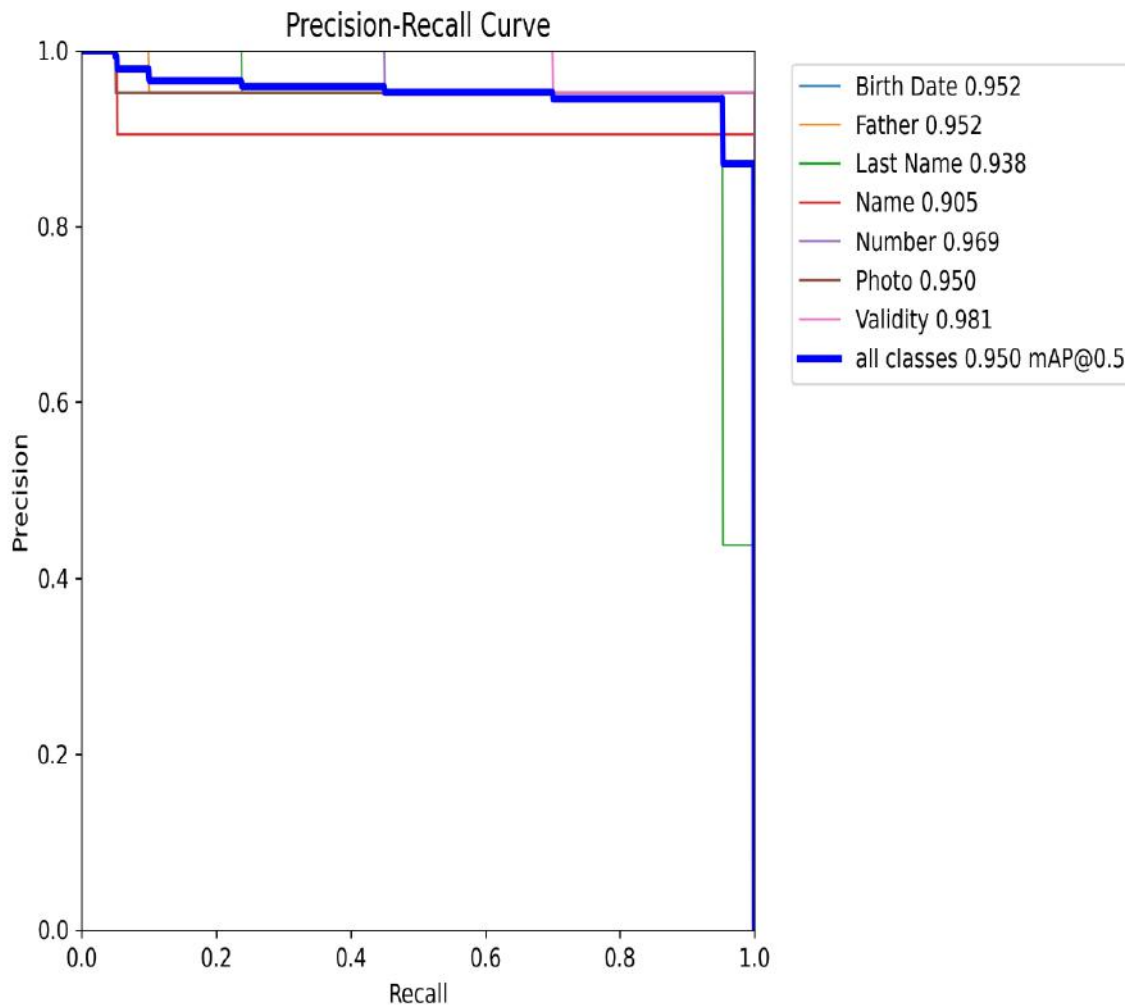
- F1-Score: میانگین هارمونیک دقت و بازیابی. این معیار مفید است زمانی که تعادل بین دقت و بازیابی اهمیت دارد.

- ماتریس درهم‌ریختگی (Confusion Matrix): یک جدول که نمایش‌دهنده تعداد دقیق اشتباهات نوع ۱ و نوع ۲ بر اساس واقعیت و پیش‌بینی مدل است.

ماتریس اشتباهات به عنوان یک ابزار قدرتمند در ارزیابی کارایی مدل‌های دسته‌بندی، ارتقاء مدل و تنظیم پارامترها مورد استفاده قرار می‌گیرد.



تصویر ۳-۴ F1-Confidence Curve



تصویر ۳-۵ Precision-Recall curve

منحنی دقت-بازیابی (Precision-Recall Curve) یک ابزار ارزیابی برای مدل‌های دسته‌بندی با توجه به احتمال‌های پیش‌بینی شده توسط مدل است. این منحنی به بررسی تعادل میان دقت (Precision) و بازیابی (Recall) مدل برای مقادیر مختلف آستانه‌های تصمیم‌گیری می‌پردازد.

برای درک بهتر، به توضیح مختصری از دقت و بازیابی نیاز داریم:

- دقت (Precision): نسبت تعداد نمونه‌های واقعی مثبتی که به درستی به عنوان مثبت تشخیص داده شده‌اند، به تعداد کل نمونه‌هایی که به عنوان مثبت تشخیص داده شده‌اند. به عبارت دیگر:

$$\text{دقت} = \frac{\text{تعداد مثبت‌های واقعی که به درستی تشخیص داده شده‌اند}}{\text{تعداد کل مثبت‌های تشخیص داده شده‌اند}}$$

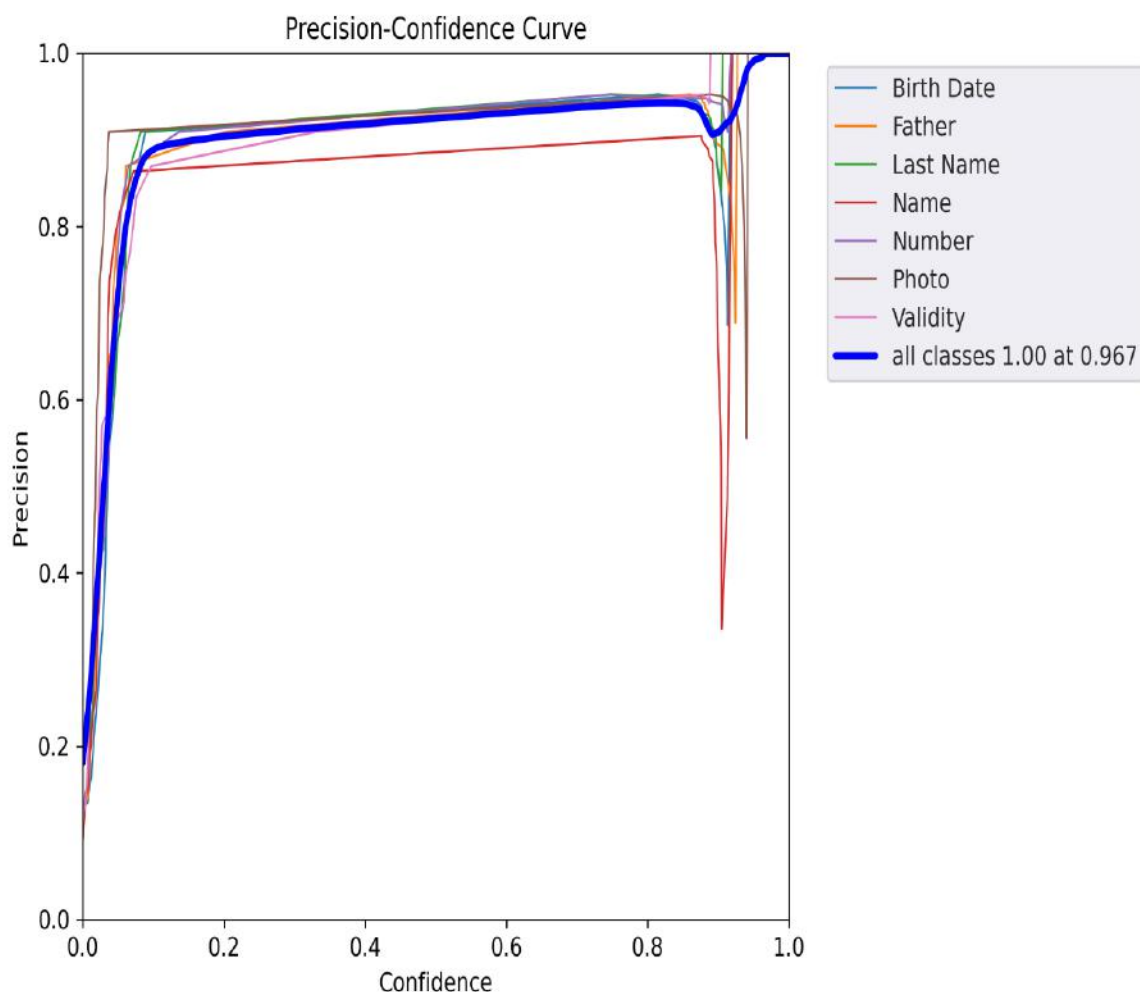
- بازیابی (Recall): نسبت تعداد نمونه‌های واقعی مثبتی که به درستی به عنوان مثبت تشخیص داده شده‌اند، به تعداد کل نمونه‌های واقعی مثبت. به عبارت دیگر:

بازیابی = $\frac{\text{تعداد مثبت‌های واقعی که به درستی تشخیص داده شده‌اند}}{\text{تعداد کل مثبت‌های واقعی}}$

منحنی دقت-بازیابی با بررسی تعادل بین دقت و بازیابی مدل در طول تمام آستانه‌های تصمیم‌گیری مختلف ساخته می‌شود. این منحنی به شکل نموداری ارائه می‌شود که محور افقی آن بازیابی و محور عمودی آن دقت است.

معمولاً در این نمودار، هر نقطه نمایانگر تعادلی بین دقت و بازیابی برای یک آستانه تصمیم‌گیری مشخص است. برای مدل‌های با عملکرد عالی، منحنی دقت-بازیابی به سمت نقاط بالا و راست متمایل خواهد شد. اگر نقاط این منحنی به نقطه (۱،۱) نزدیک شود، به این معناست که مدل برای تمام آستانه‌ها دقت و بازیابی بالا دارد.

منحنی دقت-بازیابی برای انتخاب بهترین آستانه تصمیم‌گیری و ارزیابی دقیق‌تر عملکرد مدل بسیار مفید است، به خصوص زمانی که تعادل میان دقت و بازیابی برای مسئله‌ی خاصی اهمیت دارد.



تصویر ۳-۶ Precision-Confidence Curve

منحنی دقت-مطمئن (Precision-Confidence Curve) یک ابزار ارزیابی است که در مدل‌های شبکه‌های عصبی شناسایی شیء مانند YOLOv5 به کار می‌رود. این منحنی به بررسی تعادل میان دقت دسته‌بندی (Precision) و مطمئن یا اعتماد (Confidence) پیش‌بینی‌های مدل برای مقادیر مختلف آستانه‌های اعتماد می‌پردازد.

برای درک بهتر، به توضیح مختصری از دقت دسته‌بندی و مطمئنی یا اعتماد پیش‌بینی نیاز داریم:

- دقت دسته‌بندی (Precision): نسبت تعداد نمونه‌های واقعی مثبتی که به درستی به عنوان مثبت تشخیص داده شده‌اند، به تعداد کل نمونه‌هایی که به عنوان مثبت تشخیص داده شده‌اند. به عبارت دیگر:

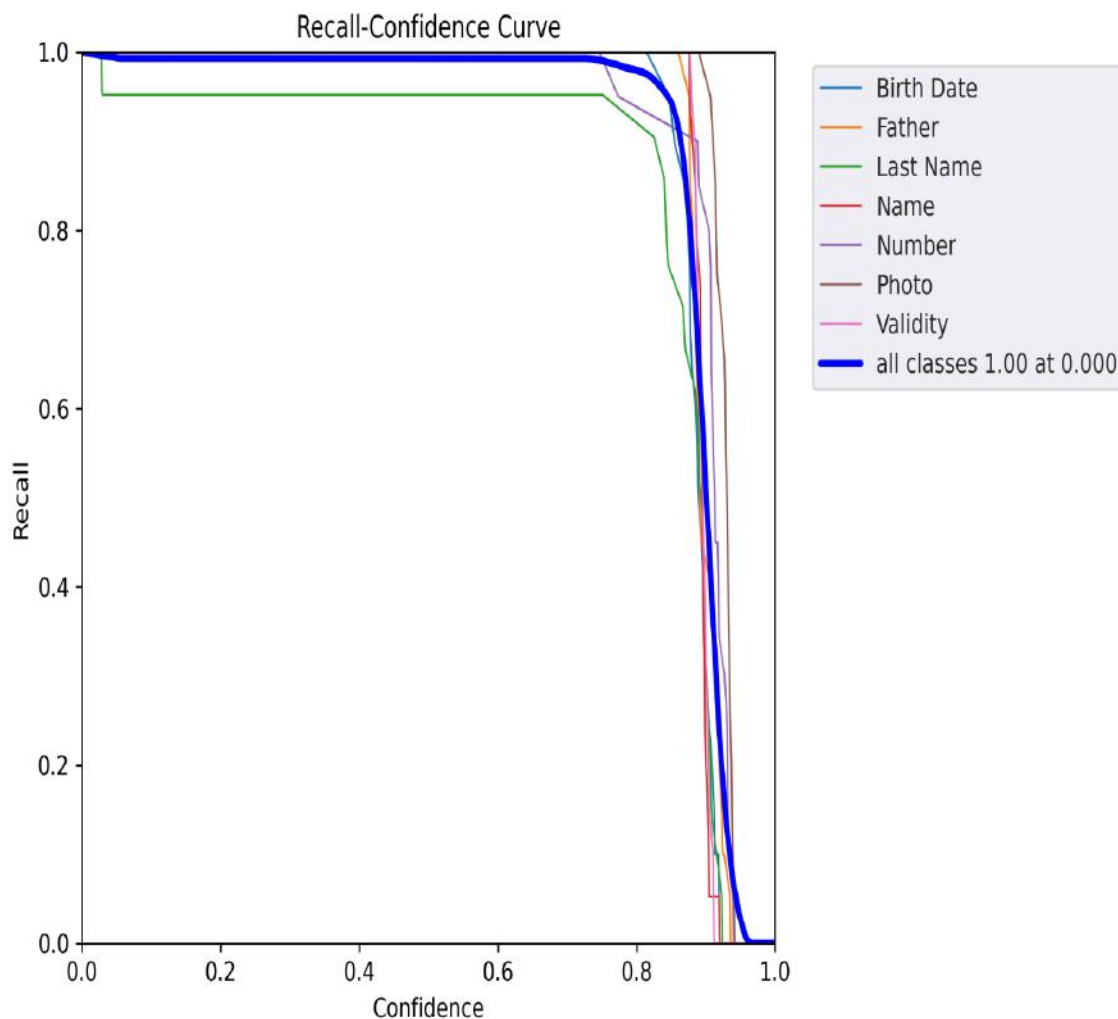
$$\text{دقت} = (\text{تعداد مثبت‌های واقعی که به درستی تشخیص داده شده‌اند}) / (\text{تعداد کل مثبت‌های تشخیص داده شده‌اند})$$

- مطمئنی یا اعتماد (Confidence): میزان اعتمادی که مدل به پیش‌بینی‌های خود دارد. در مدل‌های شبکه‌های عصبی شناسایی شیء، هر پیش‌بینی با یک احتمال اعتماد مرتبط است که نشان‌دهنده مطمئنی مدل در صحت پیش‌بینی است. مقدار این اعتماد در دسته‌های مختلف نیز متفاوت است.

منحنی دقت-مطمئنی تحلیلی از تأثیر تغییرات آستانه اعتماد مدل روی دقت دسته‌بندی مدل در انواع دسته‌ها ارائه می‌دهد. این منحنی نموداری است که محور افقی آن میزان اعتماد مدل به پیش‌بینی‌های خود (آستانه اعتماد) و محور عمودی آن دقت دسته‌بندی مدل را نمایش می‌دهد.

هر نقطه در این منحنی نمایانگر تعادلی بین دقت دسته‌بندی مدل و مطمئنی اعتماد آستانه‌ای خاص است. این منحنی برای مدل‌های با دقت و مطمئنی خوب، به سمت نقاط بالا و راست متمایل خواهد شد. به عبارت دیگر، مدل‌هایی که در تشخیص شیء خود دقت و اعتماد بالا دارند، در نقاط بالا و راست این منحنی قرار می‌گیرند.

منحنی دقت-مطمئنی می‌تواند به ما کمک کند تا آستانه‌های اعتماد مناسب برای مدل را انتخاب کنیم تا دقت دسته‌بندی مدل در حداکثر بیشینه باشد. همچنین، این منحنی می‌تواند نشان‌دهنده این باشد که آیا مدل به خوبی تعادل میان دقت و مطمئنی را حفظ می‌کند یا خیر.



تصویر ۳-۷ Recall-Confidence Curve

منحنی بازخوانی-مطمئنی (Recall-Confidence Curve) یک ابزار مهم در ارزیابی عملکرد مدل‌های تشخیص شیء مانند YOLOv5 است. این منحنی به ما کمک می‌کند تا درک بهتری از تعادل بین بازخوانی (Recall) و مطمئنی یا اعتماد (Confidence) پیش‌بینی‌های مدل برای مقادیر مختلف آستانه‌های اعتماد پیدا کنیم.

برای توضیح بهتر، نیاز به درک مفاهیم بازخوانی و مطمئنی داریم:

- بازخوانی (Recall): نسبت تعداد نمونه‌های واقعی مثبتی که به درستی به عنوان مثبت تشخیص داده شده‌اند، به تعداد کل نمونه‌های واقعی مثبت در دیتاست. به عبارت دیگر:

بازخوانی = $\frac{\text{تعداد مثبت‌های واقعی که به درستی تشخیص داده شده‌اند}}{\text{تعداد کل مثبت‌های واقعی}}$

- مطمئنی یا اعتماد (Confidence): این مفهوم در مدل‌های تشخیص شیء به اهمیت ویژه‌ای دسترسی دارد. مطمئنی نشان‌دهنده اعتماد مدل به پیش‌بینی‌های خود در مورد هر شیء است. این مقدار معمولاً به صورت احتمال ارائه می‌شود که مدل به شیء مربوطه وجود دارد.

منحنی بازخوانی-مطمئنی نمایشگری از تعادل بین بازخوانی مدل و مطمئنی اعتماد در آستانه‌های مختلف است. در این منحنی، محور افقی مطمئنی یا اعتماد مدل را نشان می‌دهد و محور عمودی بازخوانی مدل را نمایش می‌دهد.

هر نقطه در این منحنی نمایانگر تعادلی میان بازخوانی و مطمئنی برای یک آستانه خاص است. منحنی بازخوانی-مطمئنی نشان می‌دهد که با افزایش آستانه‌های اعتماد، چگونه بازخوانی مدل تغییر می‌کند. به عبارت دیگر، چگونه مدل در شناسایی اشیاء وجود دارند و چگونه تمایل دارد اشیاء را از دست ندهد. منحنی بازخوانی-مطمئنی نشان‌دهنده عملکرد مدل در مواجهه با چالش‌ها و تصمیمات برنامه‌ریزی در حوزه تشخیص شیء است. این منحنی به ما کمک می‌کند تا توازن مناسبی بین بازخوانی و مطمئنی را برای کاربردها و مسائل خاص ایجاد کنیم.



تصویر ۳-۸ نمونه لیبیل زده شده



تصویر ۳-۹ نمونه لیبل زده شده



تصویر ۳-۱۰ نمونه لیبل زده شده

Detect ۳-۲-۳

```
python detect.py --weights runs/train/exp/weights/best.pt --img 1024 --conf 0.30 --source ../train_data/images/val
```

تصویر ۳-۱۱ دستور اجرای فایل Detect

این کد با ورودی‌های مدل آموزش دیده شده، تصاویری که در بخش val قرار دارد را پردازش می‌کند و تلاش می‌کند اشیاء مختلف موجود در تصاویر را با استفاده از مرزهای مستطیلی (Bounding Box) شناسایی کند.

- `python detect.py`: این دستور باعث اجرای فایل `detect.py` می‌شود که توابع مورد نیاز برای تشخیص اشیاء در تصاویر با استفاده از مدل آموزش دیده شده YOLO v5 را اجرا می‌کند.
- `--weights runs/train/exp/weights/best.pt`: این پارامتر وزن‌های آموزش دیده شده در مدل را مشخص می‌کند. مسیر `runs/train/exp/weights/best.pt` به بهترین وزن‌های آموزش دیده شده در طول فرآیند آموزش اشاره دارد.
- `--img 1024`: با این پارامتر، اندازه تصاویر ورودی به مدل مشخص می‌شود. تصاویر به ابعاد 1024×1024 پیکسل تغییر اندازه داده می‌شوند تا به مدل ارائه شوند.
- `--conf 0.30`: بطور دقیق‌تر، پارامتر `--conf` یک پارامتر مهم در مدل‌های تشخیص شیء مبتنی بر YOLO می‌باشد. این پارامتر به عنوان آستانه‌ای برای اعتماد به نفس (confidence threshold) عمل می‌کند که تصمیم می‌دهد کدام دسته‌های اشیاء توسط مدل تشخیص داده شوند و کدام دسته‌ها صرفاً نادیده گرفته شوند. در مدل‌های تشخیص شیء مانند YOLO، هر شیء‌ای که توسط مدل تشخیص داده می‌شود، با یک مربع مستطیلی (Bounding Box) نمایش داده می‌شود. همچنین، هر Bounding Box با یک امتیاز اعتماد همراه است که نشان‌دهنده اطمینان مدل از تشخیص صحیح شیء مربوطه است. این امتیاز اعتماد به نفس بر اساس تطابق میان اطلاعات موجود در تصویر و اطلاعات یادگیری شده توسط مدل تعیین می‌شود. با پارامتر `--conf` می‌توانید تعیین کنید که مدل تنها اشیاء که امتیاز اعتماد بیشتری از مقدار مشخص شده دارند، را نمایش دهد. به عبارت دیگر، اشیاء که مدل با اطمینان بیشتری تشخیص داده است. این کار می‌تواند کمک کند تا نتایج تشخیص اشیاء دقیق‌تر و قابل اطمینان‌تر باشند. در اینجا، مقدار ۰٫۳۰ به مدل می‌گوید که اشیاء که امتیاز اعتماد بیشتری از ۰٫۳۰ دارند، نمایش داده شوند. به عبارت دیگر، اشیاء که مدل با اطمینان حداقل ۳۰ درصدی تشخیص داده‌اند، در نتیجه نمایش داده می‌شوند. توجه داشته باشید که تنظیم مقدار این پارامتر می‌تواند تأثیر قابل توجهی بر نتایج تشخیص داشته باشد، بنابراین باید با دقت تنظیم شود تا تعادل میان دقت و میزان اشتباه در تشخیص اشیاء حفظ شود.

- `source ../train_data/images/val--`: این پارامتر مشخص می‌کند که تصاویری که می‌خواهید مدل بر روی آن‌ها اجرا شود، در کجا قرار دارند. مسیر `train_data/images/val/..` به تصاویر اعتبارسنجی (Validation) اشاره دارد. با اجرای این دستور، مدل با بهترین وزن‌های آموزش دیده شده خود روی تصاویر اعتبارسنجی اجرا می‌شود. مدل سعی می‌کند اشیاء موجود در تصاویر را با استفاده از مرزهای مستطیلی شناسایی کند. نتایج به شما نمایش داده می‌شوند تا بتوانید ارزیابی کنید که مدل چگونه عمل کرده است. این کار به منظور ارزیابی کیفیت تشخیص اشیاء با مدل آموزش دیده شده انجام می‌شود.

```
# Start training
t0 = time.time()
nb = len(train_loader) # number of batches
nw = max(round(hyp['warmup_epochs'] * nb), 100) # number of warmup iterations, max(3 epochs, 100 iterations)
# nw = min(nw, (epochs - start_epoch) / 2 * nb) # limit warmup to < 1/2 of training
last_opt_step = -1
maps = np.zeros(nc) # mAP per class
results = (0, 0, 0, 0, 0, 0, 0, 0) # P, R, mAP@.5, mAP@.5-.95, val_loss(box, obj, cls)
scheduler.last_epoch = start_epoch - 1 # do not move
scaler = torch.cuda.amp.GradScaler(enabled=amp)
stopper, stop = EarlyStopping(patience=opt.patience), False
compute_loss = ComputeLoss(model) # init loss class
callbacks.run('on_train_start')
LOGGER.info(f'Image sizes {imgsz} train, {imgsz} val\n'
            f'Using {train_loader.num_workers * WORLD_SIZE} data loader workers\n'
            f'Logging results to {colorstr('bold', save_dir)}\n'
            f'Starting training for {epochs} epochs...')
for epoch in range(start_epoch, epochs): # epoch -----
    callbacks.run('on_train_epoch_start')
    model.train()
```

تصویر ۳-۱۲ تنظیمات اولیه و شروع مراحل Train

این قسمت از کد با آغاز دوره‌ی آموزش شروع می‌شود و مراحل مربوط به تنظیمات اولیه و شروع دوره‌ی آموزش را انجام می‌دهد. بیایید به صورت تفصیل به هر قسمت بپردازیم:

۱. `t0 = time.time()`: در این خط، زمان شروع آموزش ذخیره می‌شود تا بتوانیم مدت زمان آموزش را محاسبه کنیم.

۲. `nb = len(train_loader)`: تعداد دسته‌ها (بچه‌ها) در لودر آموزش (`train_loader`) به دست می‌آید. این تعداد مشخص می‌کند که چند دسته از داده‌ها برای هر دوره‌ی آموزش وجود دارد.

۳. `nw = max(round(hyp['warmup_epochs'] * nb), 100)`: تعداد تکرارهای گرم‌شدن (`warm-up`) محاسبه می‌شود. این تعداد تعیین‌کننده‌ی تعداد گام‌های آموزشی در ابتدای فرآیند آموزش است که نرخ یادگیری را از صفر به مقدار اصلی تنظیم می‌کند. در این جا، تعداد دوره‌های گرم‌شدن

(warm-up epochs) ضربدر تعداد دسته‌ها محاسبه می‌شود. اگر این تعداد کمتر از ۱۰۰ باشد، مقدار ۱۰۰ به جای آن انتخاب می‌شود.

۴. `last_opt_step = -1`: این متغیر نشان‌دهنده‌ی آخرین گام به‌روزرسانی بهینه‌سازی است و در ابتدا به ۱- تنظیم می‌شود.

۵. `maps = np.zeros(nc)`: یک آرایه‌ی با اندازه تعداد دسته‌ها (nc) با مقادیر صفر ایجاد می‌شود. این آرایه برای ذخیره‌سازی معیارهای ارزیابی (مانند mAP) برای هر دسته به کار می‌رود.

۶. `results = (0, 0, 0, 0, 0, 0, 0)`: یک مجموعه از اعداد صفر برای نتایج ارزیابی مختلف (دقت، بازخوانی، mAP با معیارهای مختلف و تلفات ارزیابی) تعریف می‌شود.

۷. `scheduler.last_epoch = start_epoch - 1`: شمارنده‌ی دوره‌های انجام‌شده برای برنامه‌ریزی‌کننده‌ی نرخ یادگیری (scheduler) تنظیم می‌شود. مقدار اولیه این شمارنده برابر با شروع دوره‌ها (start_epoch) منهای یک می‌شود تا اولین دوره به درستی در نظر گرفته شود.

۸. `scaler = torch.cuda.amp.GradScaler(enabled=amp)`: یک مقیاس‌بند گرادیان (GradScaler) برای استفاده از تکنیک Mixed Precision تعریف می‌شود. این تکنیک بهینه‌سازی استفاده از حافظه بهتر و افزایش سرعت آموزش را هدف دارد.

۹. `stopper, stop = EarlyStopping(patience=opt.patience), False`: یک شیء از نوع توقف زودهنگام (EarlyStopping) برای کنترل وقفه زودهنگام در آموزش تعریف می‌شود. تعداد صبر (patience) به عنوان یک تنظیم قابل تنظیم به آن داده می‌شود. همچنین، متغیری به نام stop با مقدار اولیه False تعریف می‌شود که به کار می‌رود تا مشخص کند آیا آموزش باید متوقف شود یا خیر.

۱۰. `compute_loss = ComputeLoss(model)`: یک نمونه از کلاس محاسبه تلفات (ComputeLoss) برای مدل تعریف می‌شود. این کلاس برای محاسبه تلفات مختلف (تلفات جعبه، تلفات شیء و تلفات دسته) استفاده می‌شود.

۱۱. `callbacks.run('on_train_start')`: اجرای متد 'on_train_start' از ماژول بازخورد (callbacks)، که این متد به‌طور اختیاری می‌تواند در آموزش اجرا شود.

۱۲. `LOGGER.info(...)`: یک پیام اطلاعاتی در ورودی به صورت چاپی (log) نمایش داده می‌شود. این پیام اطلاعات مرتبط با اندازه تصاویر آموزشی و اعتبارسنجی، تعداد کارگرهای لودر داده آموزش، مسیر ذخیره نتایج و مدت زمان آموزش می‌باشد.

۱۳. `for epoch in range(start_epoch, epochs)`: یک حلقه تکراری برای پیمایش از تمام دوره‌های آموزش از شروع تا انتها تعریف می‌شود.

۱۴. `callbacks.run('on_train_epoch_start')`: اجرای متد 'on_train_epoch_start' از ماژول بازخورد (callbacks) برای هر دوره. این متد نیز به‌صورت اختیاری می‌تواند در آموزش اجرا شود.

۱۵. `model.train()`: مدل شبکه عصبی به حالت آموزش درآمده و آماده برای محاسبه‌ی گرادیان‌ها و بهینه‌سازی می‌شود.

```
# Update image weights (optional, single-GPU only)
if opt.image_weights:
    cw = model.class_weights.cpu().numpy() * (1 - maps) ** 2 / nc # class weights
    iw = labels_to_image_weights(dataset.labels, nc=nc, class_weights=cw) # image weights
    dataset.indices = random.choices(range(dataset.n), weights=iw, k=dataset.n) # rand weighted idx

# Update mosaic border (optional)
# b = int(random.uniform(0.25 * imgsz, 0.75 * imgsz + gs) // gs * gs)
# dataset.mosaic_border = [b - imgsz, -b] # height, width borders

mloss = torch.zeros(3, device=device) # mean losses
if RANK != -1:
    train_loader.sampler.set_epoch(epoch)
    pbar = enumerate(train_loader)
    LOGGER.info(('\n' + '%11s' * 7) % ('Epoch', 'GPU_mem', 'box_loss', 'obj_loss', 'cls_loss', 'Instances', 'Size'))
    if RANK in {-1, 0}:
        pbar = tqdm(pbar, total=nb, bar_format=TQDM_BAR_FORMAT) # progress bar
    optimizer.zero_grad()
    for i, (imgs, targets, paths, _) in pbar: # batch -----
        callbacks.run('on_train_batch_start')
        ni = i + nb * epoch # number integrated batches (since train start)
        imgs = imgs.to(device, non_blocking=True).float() / 255 # uint8 to float32, 0-255 to 0.0-1.0
```

تصویر ۳-۱۳ تغییرات و به‌روزرسانی‌ها

در این بخش از کد، تغییرات و به‌روزرسانی‌های مختلفی در مرحله‌های مختلف آموزش اعمال می‌شوند. بیاید هر بخش را به تفصیل توضیح دهیم:

۱. به‌روزرسانی وزن‌های تصاویر (تنها در یک GPU):

اگر تنظیم `opt.image_weights` فعال باشد، این بخش اجرا می‌شود. در اینجا، وزن‌های تصاویر بر اساس وزن‌های دسته‌ها تغییر می‌کند. مقادیر وزن تصاویر بر اساس وزن‌های دسته‌ها محاسبه می‌شوند. در واقع، وزن تصاویر برای کلاس‌هایی با معیارهای کمتر به میزان بیشتری تغییر می‌کند. این عمل باعث می‌شود که مدل بیشتر به کلاس‌های کمتر توجه کند.

۲. به‌روزرسانی مرز:

در این بخش، مرز به‌روزرسانی می‌شود. مرز به ترکیب تصاویر مختلف برای تولید دسته‌های بزرگتر و متنوع‌تر استفاده می‌شود.

۳. محاسبه تلفات میانگین و تنظیم‌های مربوطه:

- `mloss = torch.zeros(3, device=device)`: یک تانسور صفر با ابعاد (۳) و بر روی دستگاه (device) فعلی ایجاد می‌شود. این تانسور برای ذخیره تلفات میانگین (به‌روزرسانی در هر دسته) مورد استفاده قرار می‌گیرد.

- `if RANK != -1`: اگر شماره‌ی رتبه‌بندی (RANK) مختلف از -۱ باشد (که در مدل DDP برای موازی‌سازی استفاده می‌شود)، نمونه‌های لودر داده‌آموزش با تنظیم شماره دوره به‌روزرسانی می‌شوند.
- `pbar = enumerate(train_loader)`: یک شیء تقدیم شده‌ی لودر داده‌های آموزش به‌عنوان یک شمارنده پیمایشی ایجاد می‌شود.

۴. نمایش وضعیت به‌روزرسانی در میان دسته‌ها:

- `LOGGER.info(...)`: یک پیام اطلاعاتی در ورودی به صورت ایجاد اپی (log) نمایش داده می‌شود. این پیام شامل اطلاعاتی مانند شماره دوره، حافظه‌ی GPU، تلفات باکس، تلفات شیء، تلفات دسته، تعداد نمونه‌ها و اندازه تصاویر مورد استفاده در هر دسته است.

- اگر RANK برابر با -۱ یا ۰ باشد (در مدل DDP به عنوان رتبه‌بندی اجرای موازی)، نوار پیشرفت (progress bar) با استفاده از تابع `tqdm` ایجاد می‌شود.

۵. مقداردهی اولیه بهینه‌سازی و شروع دسته‌ها:

- `optimizer.zero_grad()`: تمام گرادیان‌های بهینه‌سازی را صفر می‌کند تا برای محاسبه‌ی گرادیان‌های جدید در هر دسته آماده باشد.

- `for i, (imgs, targets, paths, _) in pbar`: ...: یک حلقه تکراری برای پیمایش از دسته‌های داده‌ها ایجاد می‌شود. برای هر دسته:

- `callbacks.run('on_train_batch_start')`: اجرای متد 'on_train_batch_start' از `callbacks`. این متد نیز اختیاری است و برای هر دسته اجرا می‌شود.

- `ni = i + nb * epoch`: تعداد دسته‌های یکپارچه‌شده تا آن دسته را از زمان شروع آموزش تا کنون محاسبه می‌شود.

```

# Warmup
if ni <= nw:
    xi = [0, nw] # x interp
    # compute_loss.gr = np.interp(ni, xi, [0.0, 1.0]) # iou loss ratio (obj_loss = 1.0 or iou)
    accumulate = max(1, np.interp(ni, xi, [1, nbs / batch_size])).round()
    for j, x in enumerate(optimizer.param_groups):
        # bias lr falls from 0.1 to lr0, all other lrs rise from 0.0 to lr0
        x['lr'] = np.interp(ni, xi, [hyp['warmup_bias_lr'] if j == 0 else 0.0, x['initial_lr'] * lf(epoch)])
        if 'momentum' in x:
            x['momentum'] = np.interp(ni, xi, [hyp['warmup_momentum'], hyp['momentum']])

# Multi-scale
if opt.multi_scale:
    sz = random.randrange(int(imgsz * 0.5), int(imgsz * 1.5) + gs) // gs * gs # size
    sf = sz / max(imgs.shape[2:]) # scale factor
    if sf != 1:
        ns = [math.ceil(x * sf / gs) * gs for x in imgs.shape[2:]] # new shape (stretched to gs-multiple)
        imgs = nn.functional.interpolate(imgs, size=ns, mode='bilinear', align_corners=False)

```

تصویر ۳-۱۴ آماده سازی داده‌ها

– `imgs = imgs.to(device, non_blocking=True).float() / 255`: تصاویر ورودی به دستگاه مورد نظر منتقل شده و از `uint8` به `float32` تبدیل می‌شوند (در بازه‌ی ۰ تا ۱).

در این بخش از کد، مراحل برای تنظیمات اختصاصی و آماده‌سازی داده‌ها قبل از آموزش شبکه انجام می‌شود. بیاید هر قسمت را به تفصیل بررسی کنیم:

۱. (Warmup):

در این بخش، تنظیمات مربوط به گرم‌شدن (یادگیری ابتدایی با نرخ کم) برای شبکه تعیین می‌شود. این کار به شبکه کمک می‌کند تا از ابتدای آموزش به یادگیری مناسب بپردازد.

– `if ni <= nw`: اگر تعداد دسته‌های یادگیری تا کنون (`ni`) کمتر یا مساوی تعداد دسته‌های گرم‌شدن (`nw`) باشد، گام‌های مربوط به گرم‌شدن انجام می‌شود.

– `xi = [0, nw]`: یک محدوده برای تعداد دسته‌ها برای تابع ترازوئیدی تعیین می‌شود.

– `accumulate = max(1, np.interp(ni, xi, [1, nbs / batch_size])).round()`: تعداد گام‌های تکرار دسته‌ها (`accumulate`) بر اساس تابع ترازوئیدی با تعداد دسته‌های گذشته‌شده (`ni`) و تعداد دسته‌های کل (`nbs`) تعیین می‌شود.

- `for j, x in enumerate(optimizer.param_groups):` ... برای هر گروه از پارامترهای بهینه‌ساز، نرخ یادگیری (`lr`) و مومنتوم مرتبط با گرم‌شدن به تدریج تغییر می‌کند.

- برای پارامتر اول (`bias`) نرخ یادگیری از `warmup_bias_lr` (نرخ اندازه‌گیری بایاس) تا مقدار اولیه تغییر می‌کند.

- برای سایر پارامترها نرخ یادگیری از `*` تا مقدار اولیه خود تغییر می‌یابد.

- اگر مومنتوم در گروه موجود باشد، مقدار مومنتوم نیز از مقدار گرم‌شدن (`warmup_momentum`) به مقدار مومنتوم اصلی (`momentum`) تغییر می‌کند.

۲. مقیاس چندگانه (`Multi-scale`):

اگر تنظیم `opt.multi_scale` فعال باشد، تصاویر ورودی در اندازه‌های مختلف تغییر اندازه می‌شوند. این کار به شبکه کمک می‌کند تا در اندازه‌های مختلف اشیا را تشخیص دهد.

- `sz = random.randrange(int(imgsz * 0.5), int(imgsz * 1.5) + gs) // gs * gs` یک اندازه تصادفی بین نصف اندازه تصاویر ورودی (`imgsz`) تا ۱,۵ برابر آن انتخاب می‌شود. سپس به مقداری بزرگ‌تر از اندازه گام‌های شبکه (`gs`) تقسیم شده و به بالا گرد می‌شود.

- `sf = sz / max(imgs.shape[2:])` فاکتور مقیاس (`scale factor`) بر اساس اندازه‌ی جدید (`sz`) و بزرگ‌ترین اندازه از ابعاد تصاویر ورودی محاسبه می‌شود.

- `if sf != 1`: اگر فاکتور مقیاس متفاوت از ۱ باشد (یعنی اندازه تصاویر تغییر کرده است):

- `ns = [math.ceil(x * sf / gs) * gs for x in imgs.shape[2:]]` اندازه‌های جدید (`ns`) برای

تصاویر ورودی محاسبه می‌شوند. این اندازه‌ها به گونه‌ای تغییر می‌کنند که به مضرب گام‌های شبکه (`gs`) نزدیک‌ترین عدد صحیح به بالا باشند.

`imgs = nn.functional.interpolate(imgs, size=ns, mode='bilinear', align_corners=False)`: تصاویر ورودی با استفاده از تابع `interpolate` اندازه‌بندی مجدد می‌شوند به طوری که به‌بالا گردند و کیفیت حفظ شود.

```
# Forward
with torch.cuda.amp.autocast(amp):
    pred = model(imgs) # forward
    loss, loss_items = compute_loss(pred, targets.to(device)) # loss scaled by batch_size
    if RANK != -1:
        loss *= WORLD_SIZE # gradient averaged between devices in DDP mode
    if opt.quad:
        loss *= 4.

# Backward
scaler.scale(loss).backward()

# Optimize - https://pytorch.org/docs/master/notes/amp_examples.html
if ni - last_opt_step >= accumulate:
    scaler.unscale_(optimizer) # unscale gradients
    torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=10.0) # clip gradients
    scaler.step(optimizer) # optimizer.step
    scaler.update()
    optimizer.zero_grad()
    if ema:
        ema.update(model)
    last_opt_step = ni
```

تصویر ۳-۱۵ محاسبه پیش‌بینی‌ها و تلفات و بهینه‌سازی

در این قسمت از کد، مراحل محاسبه‌ی پیش‌بینی‌ها، محاسبه‌ی تلفات و بهینه‌سازی شبکه انجام می‌شوند. بیا ببینیم هر بخش را تفصیلی بررسی کنیم:

۱. پیش‌بینی (Forward):

– `with torch.cuda.amp.autocast(amp):` با استفاده از مدل `"torch.cuda.amp.autocast"` به وضوح پیش‌بینی‌ها (Forward) انجام می‌شود. این کار بهینه‌سازی‌هایی برای استفاده از میزان دقیق انجام پیش‌بینی‌ها در GPU را انجام می‌دهد.

– `pred = model(imgs)` با اعمال تصاویر ورودی به مدل، پیش‌بینی‌های شبکه برای اشیا مختلف انجام می‌شود.

- `loss, loss_items = compute_loss(pred, targets.to(device))` با استفاده از مدل پیش‌بینی‌شده و برچسب‌های واقعی، تلفات (`loss`) محاسبه می‌شود. همچنین، تلفات‌های جزئی نیز (`loss_items`) بازگردانده می‌شود.

- `if RANK != -1`: اگر شماره‌ی رتبه‌بندی (`RANK`) مختلف از -۱ باشد (در مدل `DDP` برای موازی‌سازی)، تلفات بر اساس تعداد دستگاه‌های موازی‌سازی (`WORLD_SIZE`) می‌شود تا مقدار میانگین گرادیان‌ها بین دستگاه‌ها تساوی شود.

- `if opt.quad: loss *= 4`: اگر تنظیم `opt.quad` فعال باشد، تلفات با یک ضریب ۴ ضرب می‌شود. این اختیاری است و تغییرات اضافه در تلفات اعمال می‌کند.

۲. بازگشت (`Backward`):

- `scaler.scale(loss).backward()`: گرادیان‌ها با استفاده از تابع `scale` و به اشکال مقیاس‌دار به عقب منتقل می‌شوند. این عمل از امکانات معماری `Mixed Precision` یا `AMP` استفاده می‌کند تا گرادیان‌ها به شکل مناسب مقیاس‌دار شوند.

۳. بهینه‌سازی (`Optimize`):

- `if ni - last_opt_step >= accumulate`: ... اگر تعداد گام‌های تکرار دسته‌ها (`accumulate`) از زمان آخرین بهینه‌سازی تا کنون (`last_opt_step`) بیشتر یا مساوی شده باشد، مراحل بهینه‌سازی انجام می‌شود.

- `scaler.unscale_(optimizer)`: گرادیان‌های مقیاس‌دار را به حالت اصلی باز می‌گرداند.

- `torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=10.0)`: گرادیان‌ها محدود به حداکثر مقدار نرم (`norm`) مشخص (۱۰٫۰) می‌شوند تا از افزایش گرادیان‌های خیلی بزرگ جلوگیری شود.

- `scaler.step(optimizer)`: با استفاده از گرادیان‌ها، بهینه‌سازی پارامترهای شبکه انجام می‌شود.

- `scaler.update()`: اطلاعات مقیاس‌سازی به‌روزرسانی می‌شود.

- `optimizer.zero_grad()`: گرادیان‌های بهینه‌ساز را صفر می‌کند تا برای محاسبه‌ی گرادیان‌های جدید در دسته‌های بعدی آماده باشد.

- `if ema: ema.update(model)`: اگر تنظیم `ema` فعال باشد (استفاده از میانگین متحرک توزیع معکوس برای بهبود نتایج آموزش)، میانگین متحرک به‌روزرسانی می‌شود.

– `last_opt_step = ni`: آخرین مرحله بهینه‌سازی به مقدار کنونی تعداد دسته‌های یکپارچه‌شده (`ni`) تغییر می‌کند تا در مراحل آینده استفاده شود.

```
# Mutate
mp, s = 0.8, 0.2 # mutation probability, sigma
npr = np.random
npr.seed(int(time.time()))
g = np.array([meta[k][0] for k in hyp.keys()]) # gains 0-1
ng = len(meta)
v = np.ones(ng)
while all(v == 1): # mutate until a change occurs (prevent duplicates)
    v = (g * (npr.random(ng) < mp) * npr.randn(ng) * npr.random() * s + 1).clip(0.3, 3.0)
    for i, k in enumerate(hyp.keys()): # plt.hist(v.ravel(), 300)
        hyp[k] = float(x[i + 7] * v[i]) # mutate

# Constrain to limits
for k, v in meta.items():
    hyp[k] = max(hyp[k], v[1]) # lower limit
    hyp[k] = min(hyp[k], v[2]) # upper limit
    hyp[k] = round(hyp[k], 5) # significant digits

# Train mutation
results = train(hyp.copy(), opt, device, callbacks)
callbacks = Callbacks()
# Write mutation results
keys = ('metrics/precision', 'metrics/recall', 'metrics/mAP_0.5', 'metrics/mAP_0.5:0.95', 'val/box_loss',
        'val/obj_loss', 'val/cls_loss')
print_mutation(keys, results, hyp.copy(), save_dir, opt.bucket)
```

تصویر ۳-۱۶ ایجاد تغییرات تصادفی در تنظیمات آموزش

در این بخش از کد، مرحله‌ای برای ایجاد تغییرات تصادفی در تنظیمات هایپرپارامترها (تنظیمات جزئیات آموزش)، آموزش با تنظیمات تغییر یافته، و ثبت نتایج بهبودها انجام می‌شود. بیاید به تفصیل هر بخش را تشریح کنیم:

۱. تغییرات تصادفی در هایپرپارامترها (`Mutate`):

– `mp, s = 0.8, 0.2`: مقدارهای ثابت برای احتمال تغییر (`mutation probability`) و واریانس (`sigma`) تنظیم می‌شوند.

– `npr = np.random`: یک نمونه‌برداری تصادفی از کتابخانه NumPy ایجاد می‌شود.

– `npr.seed(int(time.time()))`: بر اساس زمان فعلی، نمونه‌برداری تصادفی مقداردهی اولیه می‌شود.

– `g = np.array([meta[k][0] for k in hyp.keys()])` – آرایه‌ای حاوی مقادیر اولیه هایپرپارامترها ایجاد می‌شود.

– `ng = len(meta)`: تعداد هایپرپارامترها به عنوان ابعاد بردارها محاسبه می‌شود.

– `v = np.ones(ng)`: یک بردار با ابعاد هایپرپارامترها با مقدار یک ایجاد می‌شود.

– `while all(v == 1)`: تا زمانی که تمامی اعضای بردار `v` برابر یک باشند، عملیات زیر انجام می‌شود (این کار تکرار تغییرات تصادفی تا زمانی است که تغییر واقعی رخ دهد را انجام می‌دهد):

– `v = (g * (npr.random(ng) < mp) * npr.randn(ng) * npr.random() * s + 1).clip` –

(۳,۰): تغییرات تصادفی را در هایپرپارامترها ایجاد می‌کند. مقادیر مربوط به هر هایپرپارامتر با توجه به احتمال تغییر (`mp`)، مقادیر اولیه (`g`)، مقادیر تصادفی ایجاد شده (`npr.randn(ng)`)، و واریانس تنظیم شده (`s`) محاسبه و تغییر می‌کنند. سپس به مقدارهای معتبر محدود می‌شوند.

– `for i, k in enumerate(hyp.keys()): hyp[k] = float(x[i + 7] * v[i])` – مقادیر جدید محاسبه‌شده (`v`) تغییر می‌کنند.

۲. محدودسازی به محدوده‌ها (Constrain to limits):

– برای هر هایپرپارامتر، مقدار محاسبه‌شده باید در محدوده‌هایی که از قبل تعیین شده است، محدود شود.

۳. آموزش با تغییرات (Train mutation):

– از تابع `train` برای آموزش با هایپرپارامترهای تغییر یافته (`hyp`) استفاده می‌شود. نتایج آموزش به متغیر `results` اختصاص می‌یابد.

۴. بازیابی تنظیمات اصلی (Callbacks and Metrics):

– `callbacks = Callbacks()`: تنظیمات اصلی کالبک‌ها و معیارها بازیابی می‌شود.

– `keys = ('metrics/precision', 'metrics/recall', 'metrics/mAP_0.5', 'metrics/mAP_0.5:0.95', 'val/box_loss', 'val/obj_loss', 'val/cls_loss')` – کلیدهای مربوط به معیارهای مختلف آموزش تعیین می‌شود.

۵. چاپ نتایج تغییر (Print Mutation Results):

– `print_mutation(keys, results, hyp.copy(), save_dir, opt.bucket)` تابع `print_mutation` برای چاپ نتایج بهبودها با توجه به معیارها و هایپرپارامترهای تغییر یافته صدا زده می‌شود.



تصویر ۳-۱۷ نمونه لیبل زده شده



تصویر ۳-۱۸ نمونه لیبل پیش‌بینی شده توسط مدل



تصویر ۳-۱۹ نمونه لیبل زده شده



تصویر ۳-۲۰ نمونه لیبل پیش‌بینی شده توسط مدل

فصل چهارم: OCR

Import ۱-۴

```
!apt install tesseract-ocr -q
!apt install poppler-utils -q
!apt install tesseract-ocr-fas -q
!pip install -q pytesseract
!pip install -q googletrans==3.1.0a0
!pip install -q pdf2image

import torch
import os
import re
import cv2
import argparse
import imutils
import random
import pytesseract
import nltk
import pandas as pd
import io
import numpy as np
from PIL import Image
from pytesseract import Output
from google.colab.patches import cv2_imshow
```

تصویر ۴-۱ نصب و وارد کردن بسته‌ها و کتابخانه‌ها

به طور خلاصه، این بخش از کد برای نصب و وارد کردن بسته‌ها و کتابخانه‌ها در محیط اجرایی Google Colab طراحی شده است. دستورات `apt install` برای نصب نرم‌افزارهای خارجی و دستورات `pip install` برای نصب پکیج‌های پایتون استفاده می‌شوند.

۱. نصب بسته‌های مورد نیاز:

– `!apt install tesseract-ocr -q`: نصب نرم‌افزار Tesseract-OCR برای تشخیص متون در تصاویر.

– `!apt install poppler-utils -q`: نصب ابزارهای مرتبط با کتابخانه Poppler برای کار با فایل‌های PDF.

– `!apt install tesseract-ocr-fas -q`: نصب پشتیبانی از زبان فارسی برای Tesseract-OCR.

– `!pip install -q pytesseract`: نصب پکیج pytesseract برای استفاده از واسطه Tesseract در پایتون.

– `!pip install -q googletrans==3.1.0a0`: نصب نسخه‌ی مشخصی از googletrans برای ترجمه‌ی

متون با استفاده از Google Translate.

– `!pip install -q pdf2image`: نصب پکیج `pdf2image` برای تبدیل صفحات PDF به تصاویر.

۲. وارد کردن کتابخانه‌ها و پکیج‌ها:

– `import torch`: وارد کردن کتابخانه `PyTorch` برای استفاده از قابلیت‌های یادگیری ماشین.

– `import os`: وارد کردن کتابخانه `os` برای اجرای دستورات سیستمی و مدیریت فایل‌ها و دایرکتوری‌ها.

– `import re`: وارد کردن کتابخانه `re` برای استفاده از عبارات باقاعده (`Regular Expressions`).

– `import cv2`: وارد کردن کتابخانه `OpenCV` برای پردازش تصاویر.

– `import argparse`: وارد کردن کتابخانه `argparse` برای پردازش آرگومان‌های خط فرمان.

– `import imutils`: وارد کردن کتابخانه `imutils` برای استفاده از ابزارهای پردازش تصویر.

– `import random`: وارد کردن کتابخانه `random` برای ایجاد اعداد تصادفی.

– `import pytesseract`: وارد کردن کتابخانه `pytesseract` برای استفاده از `Tesseract` در پایتون.

– `import nltk`: وارد کردن کتابخانه `nltk` برای پردازش متن و متن کاوی.

– `import pandas as pd`: وارد کردن کتابخانه `pandas` برای کار با داده‌های جدولی.

– `import io`: وارد کردن کتابخانه `io` برای کار با جریان‌های ورودی/خروجی.

– `import numpy as np`: وارد کردن کتابخانه `numpy` برای محاسبات علمی با آرایه‌ها.

– `from PIL import Image`: وارد کردن کلاس `Image` از کتابخانه `PIL (Python Imaging Library)`

برای کار با تصاویر.

– `from pytesseract import Output`: وارد کردن کلاس `Output` از کتابخانه `pytesseract` برای

مدیریت خروجی `Tesseract`.

– `from google.colab.patches import cv2_imshow`: وارد کردن تابع `cv2_imshow` برای

نمایش تصاویر در محیط `Google Colab`.

به این ترتیب، کتابخانه‌ها و پکیج‌های مورد نیاز برای تجزیه و تحلیل تصاویر، پردازش متن، کار با داده‌های جدولی و ترجمه متون به صورت مفصل نصب و وارد می‌شوند. این تدابیر اولیه ضروری هستند تا کد به درستی اجرا شود.

2-4 بارگذاری مدل

```
model_name='/content/drive/MyDrive/yolov5/runs/train/exp/weights/best.pt'
model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_name, force_reload=True)
model.conf = 0.60
```

تصویر ۴-۲ بارگیری یک مدل شبکه عصبی

این بخش از کد برای بارگیری یک مدل شبکه عصبی با استفاده از فریم‌ورک PyTorch و کتابخانه YOLOv5 طراحی شده است. بیایید به تفصیل هر قسمت از کد بپردازیم:

۱. `model_name =` `"/content/drive/MyDrive/yolov5/runs/train/exp/weights/best.pt"` در این قسمت، مسیر به فایل مدل مورد نظر تعیین می‌شود. این مسیر نشان می‌دهد که مدل برای آموزش و ذخیره سازی مدل در مسیر مشخصی قرار گرفته است.

۲. `model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_name, force_reload=True)` در این قسمت، مدل YOLOv5 از مسیر مشخص شده (`model_name`) بارگیری می‌شود. از تابع `torch.hub.load` برای بارگیری مدل از هاب PyTorch (`hub`) استفاده می‌شود. مشخصات مدل به شکل زیر تعیین می‌شود:

- `'ultralytics/yolov5'`: این پارامتر نشان‌دهنده نام هاب و مخزن مرتبط با مدل YOLOv5 در شبکه‌های اجتماعی می‌باشد.

- `'custom'`: این پارامتر نشان‌دهنده نوع مدل YOLOv5 است که در اینجا از یک مدل اختصاصی استفاده می‌شود.

- `path=model_name`: مسیر فایل مدل که قبلاً تعیین شده است، به عنوان مسیر برای بارگیری مدل استفاده می‌شود.

- `force_reload=True`: با تنظیم این پارامتر به `True`، اجبار به بارگیری مدل از دیسک انجام می‌شود حتی اگر مدل در حافظه نهان موجود باشد.

۳. `model.conf = 0.60`: با این دستور، مقدار آستانه‌ی اعتماد مدل (`confidence threshold`) به `0.60` تنظیم می‌شود. این آستانه نشان‌دهنده حداقل اعتمادی است که برای تشخیص اشیاء در تصاویر استفاده می‌شود. اشیاء با اعتماد کم‌تر از این آستانه در خروجی مدل نمایش داده نخواهند شد.

در کل، این کد برای بارگیری مدل YOLOv5 از مسیر مشخصی و تنظیم آستانه اعتماد مورد استفاده قرار می‌گیرد تا امکان تشخیص اشیاء در تصاویر با دقت مشخص شده فراهم شود.

3-4 بریدن بخش‌های شناسایی شده توسط مدل

```
images_list = '/content/drive/MyDrive/train_data/images/val/2.jpg'
#for image in os.listdir(images_path):

results = model(images_list) # inference
results.print() # or .show(), .save(), .crop(), .pandas(), etc.
results.crop(save=True)
```

image 1/1: 659x1080 1 Birth Date, 1 Father, 1 Last Name, 1 Name, 1 Number, 1 Photo, 1 Validity
Speed: 1046.2ms pre-process, 402.7ms inference, 25.6ms NMS per image at shape (1, 3, 416, 640)

تصویر ۴-۳ تشخیص اشیاء

بطور کلی، این کد برای استفاده از مدل YOLOv5 به منظور تشخیص اشیاء در تصاویر طراحی شده است. حالا به تفصیل هر بخش از کد بپردازیم:

۱. `images_list = '/content/drive/MyDrive/train_data/images/val/2.jpg'` در این خط، مسیر تصویر ورودی برای تست مدل تعیین می‌شود. شما می‌توانید مسیر تصاویر خود را در محلی که تصاویر ذخیره شده‌اند، تعیین کنید.

۲. `results = model(images_list)`: با استفاده از این خط، مدل YOLOv5 بر روی تصویر ورودی اجرا می‌شود. مدل تلاش می‌کند اشیاء موجود در تصویر را تشخیص داده و مشخصات مربوط به آن‌ها را استخراج کند.

۳. `results.print()`: این خط، نتایج تشخیص اشیاء در تصویر را نمایش می‌دهد. نتایج شامل اطلاعاتی مانند موقعیت اشیاء (مثلاً مختصات گوشه‌ها)، برچسب‌های اشیاء و اعتماد مدل به تشخیص هر شیء است.

۴. `results.crop(save=True)`: با این دستور، تصاویر برش‌داده‌شده از اشیاء تشخیص‌داده‌شده در تصویر ایجاد می‌شود و در صورت تنظیم پارامتر `'save=True'`، این تصاویر برش‌داده‌شده ذخیره می‌شوند. این تصاویر برش‌داده‌شده، تصاویری هستند که فقط شامل اشیاء مشخص‌شده در تصویر اصلی هستند.

به این ترتیب، با اجرای این کد، می‌توانید مدل YOLOv5 را بر روی تصاویر مختلف اجرا کرده، اشیاء تشخیص‌داده‌شده را بازبینی کنید و در صورت نیاز تصاویر برش‌داده‌شده از اشیاء را ذخیره کنید.

4-4 شناسایی مسیر بخش‌های بریده شده

```
# crop detected images
!rm -r runs/
# Crop detected objects and saved it in seprate paths

for img_path in images_list:
    image = cv2.imread(img_path, 1)
    image_name = img_path.split("/")[-1].split(".png")[0]
    H, W = image.shape[:2]

    labels_path = "/content/yolov5/runs/detect/exp/labels/"
    boxes = pd.read_csv(f"{labels_path}{image_name}.txt", delim_whitespace=True, header=None, index_col=False)

    for index, row in boxes.iterrows():
        if row[0] == 0:
            saved_path = "/content/Number/"
        elif row[0] == 1:
            saved_path = "/content/FistName/"
        elif row[0] == 2:
            saved_path = "/content/LastName/"
        elif row[0] == 3:
            saved_path = "/content/Father/"
        elif row[0] == 4:
            saved_path = "/content/Validation/"
        elif row[0] == 5:
            saved_path = "/content/BirthDate/"
        elif row[0] == 6:
            continue
```

تصویر ۴-۴ شناسایی مسیر بخش‌های بریده شده

```
(center_x, center_y, bbox_width, bbox_height) = yoloFormattocv(float(row[1]), float(row[2]),
                                                                float(row[3]), float(row[4]), H, W)

crop_img = image[center_y:bbox_height, center_x:bbox_width]
# draw rectangle
# rectangled = cv2.rectangle(image, (center_x, center_y), (bbox_width, bbox_height), color=(0,0,255), thickness=1)
# cv2.imshow(crop_img)

saved_path += (image_name + "_" + str(index) + ".png" )
cv2.imwrite(saved_path, crop_img)
detected_images_dir_BirthDate = '/content/runs/detect/exp/crops/Birth Date'
detected_images_dir_Name = '/content/runs/detect/exp/crops/Name'
detected_images_dir_LastName = '/content/runs/detect/exp/crops/Last Name'
detected_images_dir_Father = '/content/runs/detect/exp/crops/Father'
detected_images_dir_Validity = '/content/runs/detect/exp/crops/Validity'
detected_images_dir_Number = '/content/runs/detect/exp/crops/Number'
detected_images_list_BirthDate = [os.path.join(detected_images_dir_BirthDate, _file) for _file in os.listdir(detected_images_dir_BirthDate)]
detected_images_list_Name = [os.path.join(detected_images_dir_Name, _file) for _file in os.listdir(detected_images_dir_Name)]
detected_images_list_LastName = [os.path.join(detected_images_dir_LastName, _file) for _file in os.listdir(detected_images_dir_LastName)]
detected_images_list_Father = [os.path.join(detected_images_dir_Father, _file) for _file in os.listdir(detected_images_dir_Father)]
detected_images_list_Validity = [os.path.join(detected_images_dir_Validity, _file) for _file in os.listdir(detected_images_dir_Validity)]
detected_images_list_Number = [os.path.join(detected_images_dir_Number, _file) for _file in os.listdir(detected_images_dir_Number)]

rm: cannot remove 'runs/': No such file or directory
Saved 1 image to runs/detect/exp
Saved results to runs/detect/exp
```

تصویر ۴-۵ شناسایی مسیر بخش‌های بریده شده

۱. `rm -r runs/`: این دستور مختصری برای حذف پوشه `runs/` به صورت بازگشتی و همچنین فایل‌های و زیرشاخه‌های آن از سیستم فایل است.

۲. این قسمت از کد یک حلقه را آغاز می‌کند که بر روی تمام تصاویر موجود در لیست `images_list` اجرا می‌شود.

۳. `image = cv2.imread(img_path, 1)`: این دستور تصویر مورد نظر را با استفاده از کتابخانه `OpenCV` بارگیری می‌کند. پارامتر ۱ نشان‌دهنده حالت بارگیری تصویر به صورت رنگی (RGB) است.

۴. `image_name = img_path.split("/")[-1].split(".png")`: این خط برای استخراج نام تصویر از مسیر کامل آن استفاده می‌شود. با توجه به آخرین قسمت مسیر و نیز حذف پسوند `'png'`، نام تصویر استخراج می‌شود.

۵. `H, W = image.shape`: این خط ابعاد تصویر (عرض و ارتفاع) را در متغیرهای `H` و `W` ذخیره می‌کند.

۶. `labels_path = "/content/yolov5/runs/detect/exp/labels"`: در این مرحله مسیر پوشه حاوی فایل‌های مرتبط با برچسب‌ها تعیین می‌شود.

۷. `boxes = pd.read_csv(f"{labels_path}{image_name}.txt")`: از اینجا، فایل متنی مرتبط با برچسب‌های تشخیص‌داده‌شده برای هر تصویر خوانده می‌شود. این فایل شامل اطلاعات موقعیت و برچسب اشیاء تشخیص‌داده‌شده در تصویر است.

۸. قسمت حلقه `for index, row in boxes.iterrows`: مسئول اجرای دستورات برای هر ردیف در داده‌های مربوط به برچسب‌ها است.

۹. در این بخش از کد، براساس مقدار اندیس در ستون اول داده‌ها (که برچسب اشیاء را نشان می‌دهد)، مسیر ذخیره‌سازی برش‌داده‌شده اشیاء تعیین می‌شود. این مسیر در متغیر `saved_path` ذخیره می‌شود.

۱۰. متغیرهای مربوط به موقعیت مرکز و اندازه برش اشیاء از توابع `yoloFormattocv` (که به طور کامل آورده نشده‌اند) محاسبه می‌شوند.

۱۱. تصویر برش‌داده‌شده با استفاده از توابع ایندیس‌گیری نامعتبر از تصویر اصلی ایجاد می‌شود.

۱۲. تصویر برش‌داده‌شده در مسیر `saved_path` ذخیره می‌شود.

۱۳. دستورات مشابه برای اشیاء مختلف با انواع مختلف ادامه دارند.

۱۴. انتهای این بخش، لیست‌هایی از مسیرهای تصاویر برش داده‌شده برای هر نوع اشیاء (مانند تاریخ تولد، نام، نام خانوادگی و غیره) ایجاد می‌شوند.

کلیتاً، این بخش از کد برای برش داده‌شده‌های تصاویر اشیاء تشخیص داده‌شده از تصویر اصلی ایجاد می‌کند و آن‌ها را در دسته‌های مختلف ذخیره می‌کند. هر تصویر برش داده‌شده با برچسب مربوطه ذخیره می‌شود تا بتوانید تصاویر مختلف را بر اساس اشیاء تشخیص داده‌شده مشاهده کنید.

Preprocess 5-4

```
def process_image(img_path):
    temp_filename = resize_image(img_path)
    img = remove_noise_and_smooth(temp_filename)
    img = fix_rotation(img)
    img = remove_lines(img)

    return img

def resize_image(img_path):
    try:
        img = Image.open(img_path)
        length_x, width_y = img.size
        factor = max(1, int(1800 / length_x)) # 1800 for tesseract
        size = factor * length_x, factor * width_y
        im_resized = img.resize(size, Image.ANTIALIAS)

        import tempfile
        temp_file = tempfile.NamedTemporaryFile(delete=False, suffix=".TIFF")
        temp_filename = temp_file.name
        im_resized.save(temp_filename, dpi=(300, 300)) # best for OCR

        return temp_filename
    except IOError:
        print("Error while reading the file.")
```

تصویر ۴-۶ Preprocess

۱. `resize_image(img_path)`: این تابع با ورودی گرفتن مسیر تصویر، تصویر را به اندازه‌ی مناسب برای پردازش OCR (تشخیص متن) با توسعه‌ی داده‌های تصویری مانند Tesseract تغییر اندازه می‌دهد.

- ابتدا تصویر با استفاده از `Image.open(img_path)` باز می‌شود.

- سپس ابعاد اصلی تصویر با `length_x` و `width_y` ذخیره می‌شوند.

- عاملی به نام `factor` محاسبه می‌شود که به صورت حداقلی ۱ و حداکثری برابر با نسبت ۱۸۰۰ به طول تصویر است. این مقدار برای ایجاد تصویر با اندازه مناسب برای Tesseract استفاده می‌شود.

- با استفاده از `factor`، اندازه جدید تصویر به `size` محاسبه می‌شود.

- سپس تصویر اصلی به اندازه جدید تغییر اندازه داده می‌شود و در متغیر `im_resized` ذخیره می‌شود.

- یک فایل موقت با پسوند TIFF ایجاد می‌شود و تصویر تغییر اندازه داده شده با رزولوشن ۳۰۰ DPI در آن ذخیره می‌شود.

- نام فایل موقت برای استفاده‌های بعدی به عنوان خروجی تابع بازگردانده می‌شود.

۲. دستورات `remove_noise_and_smooth`، `fix_rotation` و `remove_lines` این توابع برای پردازش تصویر و بهبود وضوح تصویر و همچنین تصحیح چرخش و حذف خطوط مزاحم استفاده می‌شوند.

- `remove_noise_and_smooth`: این تابع اغلب برای حذف نویز و اصطکاک‌های کوچک در تصویر استفاده می‌شود. این ممکن است با استفاده از فیلترهای مختلفی مانند فیلتر گاوسی یا فیلتر NLMeans انجام شود.

- `fix_rotation`: اگر تصویر به صورت نامناسبی چرخش داده شده باشد، این تابع می‌تواند تصویر را به حالت صحیح بازگرداند. این امر ممکن است با تشخیص زوایای مختلف در تصویر و تنظیم مجدد اندازه‌ها انجام شود.

- `remove_lines`: این تابع برای حذف خطوط افقی یا عمودی اضافی در تصویر مورد استفاده قرار می‌گیرد. ممکن است از تبدیل هاف (Hough Transform) یا روش‌های تشخیص خطوط دیگر برای این منظور استفاده شود.

به طور کلی، این تابع‌ها به شما کمک می‌کنند تا تصویر ورودی را به وضعیت مناسبی برای پردازش تشخیص متن تبدیل کنید.

```
def remove_noise_and_smooth(img_path):
    try:
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # Applying erosion and dilation to remove the noise
        img = cv2.bitwise_not(img)
        kernel = np.ones((5, 5), np.uint8)
        img = cv2.erode(img, kernel, iterations=1)
        img = cv2.dilate(img, kernel, iterations=1)
        # show_wait_destroy('dilate', img)
        img = cv2.bitwise_not(img)

        kernel = np.ones((1, 1), np.uint8)
        opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
        closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)
        img = cv2.bitwise_or(img, closing)
        #show_wait_destroy('bitwise_or', img)

        img = apply_threshold(img, 1)
        # show_wait_destroy('threshold', img)
        img = smooth_image(img)

    return img
except IOError:
    print("Error while reading the file.")
```

تصویر ۴-۷ پیش پردازش تصویر برای بهبود تشخیص متن توسط مدل

به طور کامل، فرآیند `remove_noise_and_smooth` در این کد به منظور پیش‌پردازش تصویر برای بهبود تشخیص متن توسط مدل OCR استفاده می‌شود. دقیقتر، این فرآیند شامل چند مرحله‌ی پیش‌پردازش است که به ترتیب زیر انجام می‌شوند:

۱. خواندن تصویر و تبدیل به سیاه و سفید:

- ابتدا تصویر از مسیر داده شده با استفاده از کتابخانه `cv2` باز می‌شود.

- سپس تصویر از فضای رنگی RGB به سیاه و سفید تبدیل می‌شود (`cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`).

۲. حذف نویزها با استفاده از ارسال و پهنش:

- با استفاده از عملگرهای ارسال و پهنش، تصویر بهبود می‌یابد و نویزها حذف می‌شوند.

- ابتدا تصویر به حالت منفی تغییر می‌کند تا در ادامه بتوان از ارسال و پهنش بر روی نواحی تاریک استفاده کرد (`img = cv2.bitwise_not(img)`).

- یک ماتریس از اندازه (x5۵) با اعداد یک ایجاد می شود که برای انجام ارسال و پهنش استفاده می شود.
- ارسال و پهنش به ترتیب با استفاده از عملگرهای `erode` و `dilate` انجام می شوند تا نویزها حذف شوند. این کار به نواحی تاریک تصویر کمک می کند.
- ۳. بازگرداندن تصویر به حالت اصلی:
- تصویر به حالت انعکاسی (اصلی) برگردانده می شود تا به حالت اصلی بازگردد (`cv2.bitwise_not(img)`).
- ۴. عملیات افتتاحی و بسته شدنی:
- یک ماتریس کوچک تر (x1۱) ایجاد می شود که برای عملیات افتتاحی و بسته شدنی (`closing` و `opening`) استفاده می شود.
- این عملیات ها به ترتیب برای حذف اجزا کوچک و رفع شکستگی ها و حفره ها در تصویر استفاده می شوند.
- ۵. تبدیل تصویر به تصویر سیاه و سفید با استفاده از آستانه ای:
- با استفاده از تابع `'apply_threshold'`، تصویر به تصویر سیاه و سفید تبدیل می شود.
- در اینجا، از روش ۱ از روش های آستانه ای استفاده می شود که باعث تبدیل تصویر به سیاه و سفید با آستانه متغیر است. این کار باعث تمایز بین پس زمینه و متن می شود.
- ۶. نرم تر کردن تصویر:
- با استفاده از تابع `'smooth_image'`، تصویر نرم تر می شود.
- این کار با اعمال تکنیک فیلتر گاوسی بر روی تصویر انجام می شود که بهبود قابلیت تشخیص توسط مدل OCR را تسهیل می کند.
- اگر خطایی در خواندن تصویر یا انجام فرآیندها رخ دهد، پیام خطای مناسب نمایش داده می شود تا کاربر آگاه شود.

```
def apply_threshold(img, argument):
    switcher = {
        1: cv2.threshold(cv2.GaussianBlur(img, (9, 9), 0), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1],
        2: cv2.threshold(cv2.GaussianBlur(img, (7, 7), 0), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1],
        3: cv2.threshold(cv2.GaussianBlur(img, (5, 5), 0), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1],
        4: cv2.threshold(cv2.medianBlur(img, 5), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1],
        5: cv2.threshold(cv2.medianBlur(img, 3), 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1],
        6: cv2.adaptiveThreshold(cv2.GaussianBlur(img, (5, 5), 0), 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY, 31, 2),
        7: cv2.adaptiveThreshold(cv2.medianBlur(img, 3), 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 31,
                                2),
    }
    return switcher.get(argument, "Invalid method")
```

تصویر ۴-۸ تبدیل تصویر اصلی به تصویر سیاه و سفید

این تابع `apply_threshold` وظیفه تبدیل تصویر اصلی به تصویر سیاه و سفید با استفاده از روش‌های مختلف آستانه‌ای را داراست. این روش‌ها به منظور جدا سازی متن از پس زمینه و افزایش تمایز متن در تصویر برای تشخیص موفق‌تر توسط مدل OCR استفاده می‌شوند. در این تابع، با توجه به مقدار آرگومان `argument` که مقدار ورودی تابع است، از یکی از روش‌های تبدیل به تصویر سیاه و سفید استفاده می‌شود.

توضیحات خط به خط این تابع به این شکل است:

۱. تعریف دیکشنری `'switcher'`:

- یک دیکشنری به نام `switcher` ایجاد می‌شود که نقش معادله‌ی `switch case` در زبان‌های برنامه‌نویسی دیگر را ایفا می‌کند.

۲. استفاده از `'switcher'` برای تبدیل تصویر:

- با توجه به مقدار آرگومان `'argument'`، یکی از مقادیر ۱ تا ۷، یکی از روش‌های تبدیل تصویر انتخاب می‌شود.

- اگر مقدار آرگومان در محدوده‌ی مقادیر معتبر باشد، تابع `cv2.threshold` برای تبدیل تصویر به تصویر سیاه و سفید با استفاده از یکی از روش‌های آستانه‌ای فراخوانی می‌شود.

- برخی از روش‌های استفاده شده عبارتند از:

- تبدیل تصویر با استفاده از فیلتر گاوسی با اندازه (x9۹) به تصویر سیاه و سفید با آستانه‌ای که با استفاده از روش `Otsu` محاسبه می‌شود.

- تبدیل تصویر با استفاده از فیلتر گاوسی با اندازه (x7۷) به تصویر سیاه و سفید با آستانه‌ای که با استفاده از روش `Otsu` محاسبه می‌شود.

- تصویر تبدیل شده به تصویر سیاه و سفید توسط `cv2.threshold` با استفاده از آستانه‌ای که با روش‌های مختلف محاسبه شده است، ایجاد می‌شود و به عنوان خروجی تابع بازگردانده می‌شود.

۳. خروجی تابع:

- اگر مقدار آرگومان در محدوده‌ی مقادیر معتبر نباشد، متن "Invalid method" به عنوان خروجی تابع بازگردانده می‌شود. این اتفاق در صورتی می‌افتد که مقدار آرگومان از مقادیر ۱ تا ۷ خارج باشد و در دیکشنری `switcher` معادلی برای آن وجود نداشته باشد.

```
def smooth_image(img):
    # Apply blur to smooth out the edges
    blur_img = cv2.GaussianBlur(img, (1, 1), 0)
    #show_wait_destroy('blur', blur_img)

    return blur_img
```

تصویر ۴-۹ هموارسازی تصویر

تابع `smooth_image` برای اعمال تراکم تصویر به منظور نرم‌سازی لبه‌ها و جزئیات تصویر استفاده می‌شود. این تابع تصویر ورودی را به عنوان ورودی می‌گیرد و تصویر نرم‌سازی‌شده را با استفاده از فیلتر گاوسی ایجاد می‌کند.

متغیر `blur_img` که در تابع `smooth_image` استفاده می‌شود، نشان‌دهنده تصویری است که به وسیله اعمال فیلتر گاوسی بر روی تصویر ورودی ایجاد می‌شود. این تصویر نرم‌سازی‌شده حاوی تاثیر نرم‌سازی گاوسی است که بر روی لبه‌ها و جزئیات تصویر اعمال می‌شود.

فرآیند نرم‌سازی گاوسی با استفاده از این فیلتر به این صورت انجام می‌شود: برای هر پیکسل در تصویر، مقدار پیکسل‌های اطراف آن با وزن‌هایی محاسبه شده و میانگین آن‌ها به عنوان مقدار جدید پیکسل در تصویر نرم‌سازی‌شده در نظر گرفته می‌شود. این عمل باعث نرم‌تر شدن لبه‌ها و کاهش تاثیر نویزهای کوچک در تصویر می‌شود.

پس از اعمال فیلتر گاوسی، تصویر `blur_img` که مقدار نرم‌سازی شده دارد، به عنوان خروجی از تابع بازگردانده می‌شود تا در مراحل بعدی پردازش مورد استفاده قرار گیرد.

فیلتر گاوسی یکی از روش‌های معمول برای نرم‌سازی تصویر است. این فیلتر با استفاده از ترکیب وزن‌دار مقادیر پیکسل‌های اطراف هر پیکسل، پیکسل جدیدی با مقدار میانگین آن‌ها ایجاد می‌کند. این عمل باعث کاهش نویزهای ریز و نرم‌سازی جزئیات تصویر می‌شود.

```
def fix_rotation(img):
    rotated_img = img
    # osd: orientation and script detection
    tess_data = pytesseract.image_to_osd(img)
    angle = int(re.search(r"(?<=Rotate: )\d+", tess_data).group(0))
    print("angle: " + str(angle))

    if angle != 0 and angle != 360:
        (h, w) = img.shape[:2]
        center = (w / 2, h / 2)

        # Perform the rotation
        rotation_mat = cv2.getRotationMatrix2D(center, -angle, 1.0)

        # Fixing the image cut-off by calculating the new center
        abs_cos = abs(rotation_mat[0, 0])
        abs_sin = abs(rotation_mat[0, 1])

        bound_w = int(h * abs_sin + w * abs_cos)
        bound_h = int(h * abs_cos + w * abs_sin)

        rotation_mat[0, 2] += bound_w / 2 - center[0]
        rotation_mat[1, 2] += bound_h / 2 - center[1]

        rotated_img = cv2.warpAffine(img, rotation_mat, (bound_w, bound_h))

    return rotated_img
```

تصویر ۴-۱۰ استخراج جهت و چرخش تصویر

این تابع ابتدا تصویر ورودی را در متغیر `rotated_img` ذخیره می‌کند. سپس با استفاده از تابع `image_to_osd` از کتابخانه `'pytesseract'`، اطلاعات مربوط به جهت و چرخش تصویر را از تصویر استخراج می‌کند.

۱. ابتدا، تصویر ورودی را در متغیر `rotated_img` ذخیره می‌کنید، تا در صورتی که زاویه چرخش نیاز به تصحیح نداشته باشد، تغییری در تصویر وجود نداشته باشد.

۲. سپس از تابع `pytesseract.image_to_osd` استفاده می‌کنید تا اطلاعات مختلفی مانند زاویه چرخش و جهت تصویر را استخراج کنید. این اطلاعات به شما کمک می‌کند تا متوجه شوید آیا تصویر نیاز به چرخش دارد یا خیر.

۳. اگر زاویه چرخش تصویر مقداری غیر از ۰ یا ۳۶۰ داشته باشد، نشان‌دهنده این است که تصویر نیاز به تصحیح چرخش دارد. در غیر این صورت، تصویر بدون تغییر به عنوان خروجی بازگردانده می‌شود.

۴. در صورتی که زاویه چرخش نیاز به تصحیح داشته باشد، ابتدا اندازه تصویر (عرض و ارتفاع) را به عنوان (h, w) ذخیره می‌کنید.

۵. سپس مرکز تصویر را با مقادیر $(w / 2, h / 2)$ محاسبه می‌کنید. این مرکز برای استفاده در ماتریس چرخش لازم است.

۶. ماتریس چرخش با استفاده از تابع `cv2.getRotationMatrix2D` ساخته می‌شود. این ماتریس به عنوان یک ماتریس 3×3 می‌آید که عناصر آن شامل مقادیر چرخش، مقیاس و ترجمه است.

۷. به دنبال محاسبه ماتریس چرخش، اندازه‌های `abs_cos` و `abs_sin` محاسبه می‌شوند. این اندازه‌ها به عنوان مقدار مطلق از کسینوس و سینوس زاویه چرخش مورد استفاده قرار می‌گیرند.

۸. اندازه‌های `bound_w` و `bound_h` که به ترتیب عرض و ارتفاع تصویر پس از چرخش را نشان می‌دهند، با استفاده از اندازه‌های تصویر اصلی و اندازه‌های محاسبه شده از ماتریس چرخش به دست می‌آیند.

۹. در نهایت، ماتریس چرخش به عنوان $([bound_w / 2 - center[0], bound_h / 2 - center[1])$ تغییر داده می‌شود. این کار باعث می‌شود تا تصویر پس از چرخش به درستی در مرکز قرار بگیرد.

۱۰. با استفاده از تابع `cv2.warpAffine`، تصویر با ماتریس چرخش تغییر داده می‌شود و نتیجه در متغیر `rotated_img` ذخیره می‌شود.

۱۱. در نهایت، تصویر تصحیح‌شده با چرخش به درستی به عنوان خروجی تابع بازگردانده می‌شود تا برای مراحل بعدی پردازش مورد استفاده قرار گیرد.

```
def remove_lines(img):
    # Remove lines to improve accuracy of tabular documents
    result = img.copy()
    thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

    # Remove horizontal lines
    horizontal_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (40, 1))
    remove_horizontal = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, horizontal_kernel, iterations=2)
    cnts = cv2.findContours(remove_horizontal, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if len(cnts) == 2 else cnts[1]
    for c in cnts:
        cv2.drawContours(result, [c], -1, (255, 255, 255), 5)

    # Remove vertical lines
    vertical_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 40))
    remove_vertical = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, vertical_kernel, iterations=2)
    cnts = cv2.findContours(remove_vertical, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if len(cnts) == 2 else cnts[1]
    for c in cnts:
        cv2.drawContours(result, [c], -1, (255, 255, 255), 5)

    #show_wait_destroy('nolines', result)

    return result
```

تصویر ۴-۱۱ حذف خطوط افقی و عمودی

در این کد، تلاش برای حذف خطوط افقی و عمودی از تصویر انجام می‌شود تا دقت تشخیص محتوای جدولی تصویر افزایش یابد. این توضیحات به شما کمک می‌کند که فرآیند حذف خطوط انجام شده را بهتر درک کنید:

۱. یک کپی از تصویر ورودی را با نام **result** ایجاد می‌کنید. این کپی برای اعمال تغییرات و حفظ تصویر اصلی در نظر گرفته می‌شود.

۲. با استفاده از تابع **cv2.threshold**، تصویر ورودی را با تبدیل به تصویر دودویی (سیاه و سفید) با استفاده از روش **THRESH_BINARY_INV + THRESH_OTSU** تبدیل می‌کنید. این مرحله باعث می‌شود که خطوط در تصویر به شکل سفید و پس‌زمینه به شکل سیاه نمایش داده شوند.

۳. برای حذف خطوط افقی، یک هسته مورفولوژی با ابعاد (۴۰، ۱) ایجاد می‌کنید. سپس با استفاده از تابع **cv2.morphologyEx** و با تعداد تکرارهای **iterations** برابر با ۲، عملیات باز و بسته (opening) را با این هسته بر روی تصویر دودویی انجام می‌دهید. این کار باعث حذف خطوط افقی از تصویر می‌شود.

۴. با استفاده از تابع **cv2.findContours**، کانتورهای مربوط به لبه‌های حذف شده خطوط افقی را پیدا می‌کنید. سپس با توجه به اندازه **'cnts'**، از کانتورهای مناسب استفاده می‌کنید. سپس برای هر کانتور، با استفاده

از تابع `cv2.drawContours` خطوط را با رنگ سفید و ضخامت ۵ پیکسل بر روی تصویر `result` رسم می‌کنید.

۵. مراحل مشابهی برای حذف خطوط عمودی انجام می‌شود. ابتدا یک هسته مورفولوژی با ابعاد (۱، ۴۰) ایجاد می‌کنید و با استفاده از تابع `cv2.morphologyEx` عملیات باز و بسته را با این هسته بر روی تصویر دودویی انجام می‌دهید. سپس با استفاده از `cv2.findContours`، کانتورهای مربوط به لبه‌های حذف شده خطوط عمودی را پیدا می‌کنید و با استفاده از `cv2.drawContours` خطوط را با رنگ سفید و ضخامت ۵ پیکسل بر روی تصویر `result` رسم می‌کنید.

۶. در نهایت، تصویر حاصل از حذف خطوط را به عنوان خروجی تابع باز می‌گردانید تا برای مراحل بعدی پردازش استفاده شود.

```
def clean_ocr_text(text):
    remove_chars = "/.:"

    new_s = []
    sentences = text.split("\n")

    for sentence in sentences:
        # Remove the specified characters and numbers using the translation table
        translator = str.maketrans('', '', remove_chars)
        new_text = sentence.translate(translator)
        s = new_text.replace(" ", "")
        s = s.replace("\x0c", "")
        if s != "":
            new_s.append(new_text.strip())
    return new_s
```

تصویر ۴-۱۲ پاکسازی متن

این تابع به نظر می‌آید که به منظور پاک‌سازی متن حاصل از تشخیص متن با استفاده از ابزار OCR (Optical Character Recognition) به کار گرفته می‌شود. بیایید تکه تکه تابع را توضیح دهیم:

۱. `remove_chars = "/.:"`: یک رشته ایجاد می‌کند که شامل کاراکترهای `/`، `.` و `:` می‌شود. این کاراکترها برای حذف از متن استفاده می‌شوند.

۲. `new_s = []`: یک لیست خالی به نام `new_s` ایجاد می‌شود که برای ذخیره متن‌های پاک‌سازی شده استفاده می‌شود.

۳. `sentences = text.split("\n")`: متن ورودی (`text`) را براساس کاراکتر جدا کننده `\n` به عبارات جداگانه تبدیل می‌کند. این مرحله فرض می‌کند که متن ورودی به عبارت‌های مختلف تقسیم شده است.

۴. دستورات درون حلقه `for sentence in sentences`: برای هر عبارت در `'sentences'`، مراحل زیر انجام می‌شود:

- `translator = str.maketrans("", "", remove_chars)`: یک جدول ترجمه ایجاد می‌کند که شامل کاراکترهای موجود در `remove_chars` است و آنها را با خالی جایگزین می‌کند. این ترجمه به منظور حذف کاراکترهای مشخص شده از متن استفاده می‌شود.

- `new_text = sentence.translate(translator)`: متن عبارت را با استفاده از ترجمه‌ای که در مرحله قبل ایجاد کرده‌ایم، پاک‌سازی می‌کند.

- `s = new_text.replace('"', "'")`: همه فاصله‌های موجود در متن پاک‌سازی شده را حذف می‌کند.

- `s = s.replace("\x0c", "")`: کاراکتر خاص `\x0c` (که معمولاً به عنوان کاراکتر کنترلی نمایش فرم نمایشی در متون استفاده می‌شود) را از متن حذف می‌کند.

- `if s != ""`: اگر متن پاک‌سازی شده (بدون فاصله‌ها و کاراکترهای خاص) خالی نباشد:

- `new_s.append(new_text.strip())`: متن پاک‌سازی شده (بدون فاصله‌ها) را به لیست `new_s` اضافه می‌کند.

۵. در نهایت، لیست `new_s` که حاوی متن‌های پاک‌سازی شده و بدون فاصله‌ها است را به عنوان خروجی تابع باز می‌گرداند.

6-4 آموزش فونت کارت ملی به Tesseract

Tesseract یک موتور تشخیص متن (OCR) با متن‌باز و منبع‌باز است که توسط گروه تشخیص متن گوگل توسعه داده شده است. این ابزار قادر است تصاویر حاوی متن چاپی را تشخیص داده و محتوای متنی در آن را به شکل متن معمولی تبدیل کند. تشخیص داده شده می‌تواند در انواع زبان‌ها و با کیفیت‌های مختلف در تصاویر مختلف باشد.

به عبارت دیگر، Tesseract یک نرم‌افزار است که با دریافت یک تصویر به عنوان ورودی، تلاش می‌کند تا محتوای متنی موجود در تصویر را تشخیص داده و به شکل متن قابل خواندن و قابل استفاده تبدیل کند. این ابزار به صورت منبع‌باز در دسترس قرار دارد، به این معنا که کد منبع آن به عنوان یک پروژه‌ی متن‌باز در اختیار عموم قرار دارد تا بهبود و توسعه‌ی آن توسط جامعه‌ی برنامه‌نویسی انجام شود.

ویژگی‌های کلیدی Tesseract:

۱. پشتیبانی از زبان‌های متعدد: Tesseract به عنوان یک ابزار تشخیص متن چندزبانه طراحی شده است. این به این معناست که می‌تواند متن‌های در زبان‌های مختلف را تشخیص دهد. از زبان‌های معمول تا زبان‌های کمتر شناخته شده، Tesseract قابلیت تشخیص و ترجمه‌ی متن‌ها را دارد.

۲. تشخیص متن چاپی: Tesseract برای تشخیص متن‌های چاپی (به عنوان مثال، متون در کتب، مقالات، مجلات و مدارک) به کار می‌رود. این ابزار قادر است متن‌ها را از تصاویر استخراج کرده و به متنی قابل ویرایش تبدیل کند.

۳. پیش‌پردازش تصویر: Tesseract قبل از تشخیص متن، تصویر ورودی را با استفاده از تکنیک‌های پیش‌پردازش بهبود می‌بخشد. این عملیات شامل تنظیم کیفیت تصویر، تصحیح انحراف و تورفتگی، حذف نویز و تمیز کردن تصویر می‌شود.

۴. تنظیمات سفارشی: Tesseract به کاربران امکان تنظیمات متنوعی را برای تشخیص متن ارائه می‌دهد. این تنظیمات شامل انتخاب زبان تشخیص، تنظیمات تجزیه و تحلیل صفحه (PS) (M) برای تشخیص المان‌های مختلف تصویر و تعیین کاراکترهای مجاز در متن می‌شود.

۵. استفاده در محیط‌های مختلف: Tesseract به راحتی در انواع سیستم‌های عامل اجرا می‌شود و به عنوان یک کتابخانه برنامه‌نویسی در اپلیکیشن‌ها یا اسکریپت‌ها قابل استفاده است.

تاریخچه:

تاریخچه Tesseract به سال ۱۹۸۵ بازمی‌گردد که در MIT توسعه یافت. در آغاز، تنها برای تشخیص متن انگلیسی بکار می‌رفت، اما در طول سال‌ها توسعه‌ی بیشتری پیدا کرد و به تشخیص زبان‌های متعدد و بهبود دقت و کیفیت تشخیص متن پرداخته شد. همچنین، با توسعه جامعه‌ی منبع‌باز و همکاری افراد و شرکت‌های مختلف، Tesseract به یکی از محبوب‌ترین و قدرتمندترین ابزارهای تشخیص متن تبدیل شده است.

JtessBoxEditor یک ابزار کاربردی و گرافیکی برای آموزش و بهینه‌سازی مدل‌های OCR ایجاد شده توسط Tesseract است. این ابزار به شما امکان می‌دهد تا با استفاده از تصاویر برچسب‌گذاری‌شده، فونت‌ها، واژگان و دیگر موارد، مدل‌های Tesseract را آموزش دهید و بهبود بخشید.

برخی از ویژگی‌های کلیدی JtessBoxEditor عبارتند از:

۱. ویرایش برچسب‌ها: شما می‌توانید تصاویر برچسب‌گذاری‌شده را در این ابزار باز کنید و برچسب‌ها را ویرایش کنید. این کار می‌تواند در تصحیح خطاها و بهبود دقت مدل‌ها به کمک بیاید.

۲. آموزش مدل‌ها: JtessBoxEditor به شما امکان می‌دهد تا از تصاویر برچسب‌گذاری‌شده برای آموزش مدل‌های Tesseract استفاده کنید. این امکان باعث بهبود دقت تشخیص متن توسط Tesseract می‌شود.

۳. آموزش فونت‌ها: شما می‌توانید تصاویر متن با فونت‌های مختلف را وارد کنید و توسط JtessBoxEditor آن‌ها را برچسب‌گذاری و به مدل‌های Tesseract آموزش دهید.

۴. تنظیم پارامترها: ابزار JtessBoxEditor اجازه می‌دهد تا پارامترهای مرتبط با آموزش و بهبود مدل‌ها را تنظیم کنید.

۵. تست و ارزیابی: شما می‌توانید مدل‌های آموزش‌دیده را بر روی تصاویر آزمون اجرا کرده و دقت آن‌ها را ارزیابی کنید.

توجه داشته باشید که JtessBoxEditor برای تخصص‌های پردازش تصویر و OCR مناسب است و به دانش کاربر در این زمینه‌ها وابسته است.

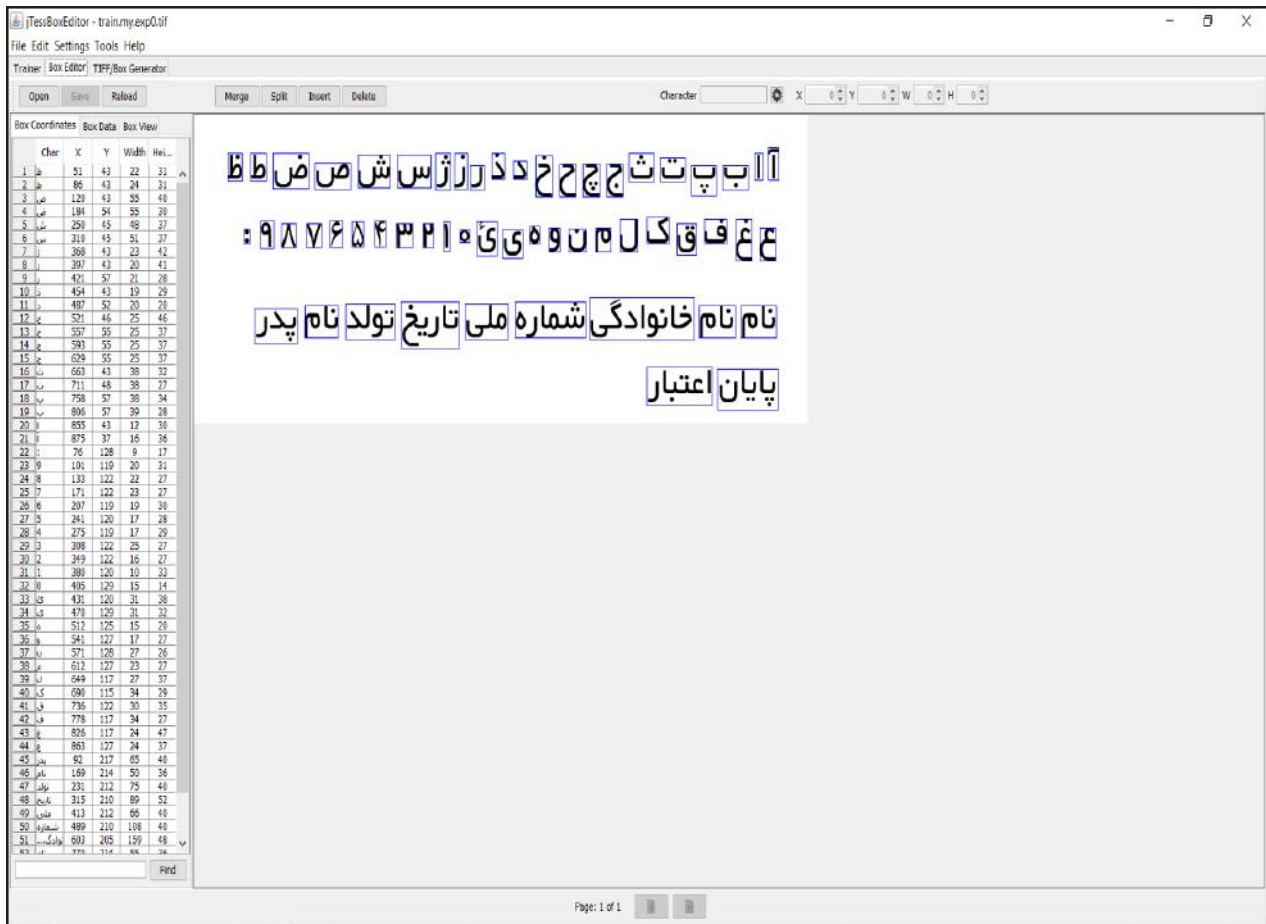
آ ا ب پ ت ث ج چ ح خ د ذ ر ز ژ س ش ص ض ط ظ

ع غ ف ق ک ل م ن و ه ی ئ ۰ ۱ ۲ ۳ ۴ ۵ ۶ ۷ ۸ ۹ :

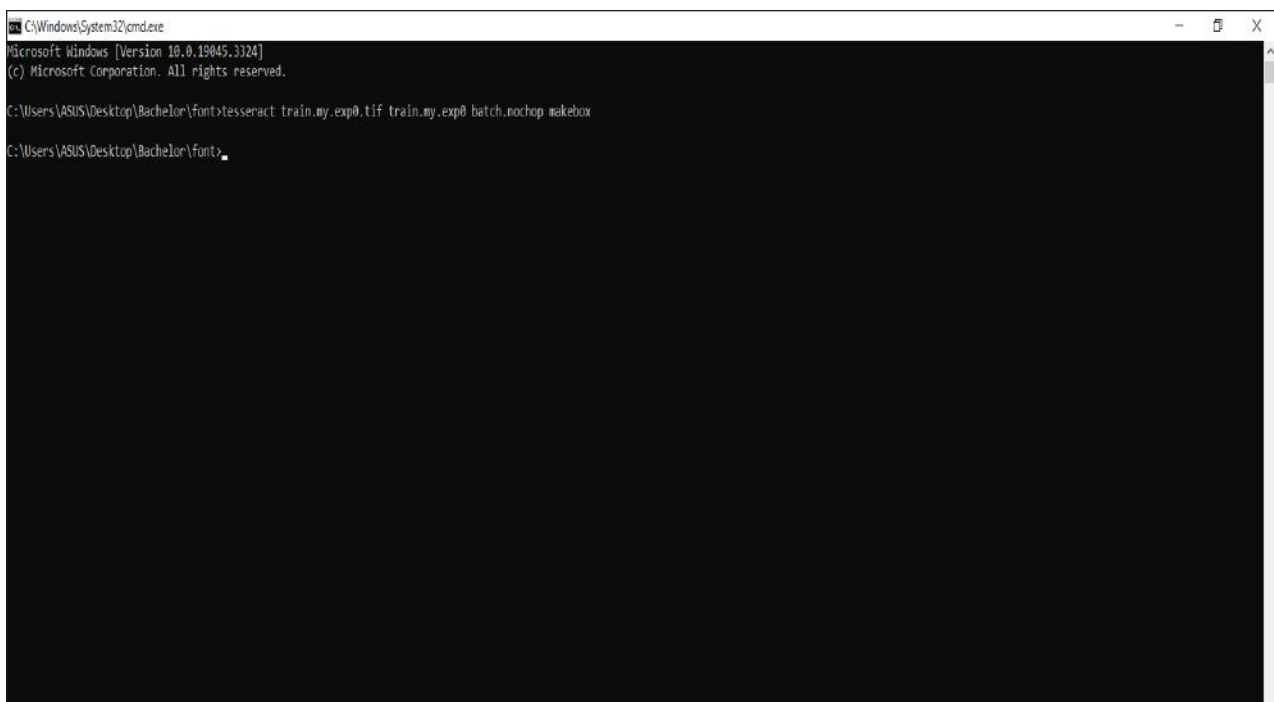
نام نام خانوادگی شماره ملی تاریخ تولد نام پدر

پایان اعتبار

تصویر ۴-۱۳ حروف با فونت کارت ملی جهت آموزش به مدل



تصویر ۴-۱۴ لیبل زدن حروف برای آموزش به مدل



تصویر ۴-۱۵ آموزش مدل های جدید OCR

این دستور که با استفاده از نرم افزار Tesseract اجرا می شود، برای ایجاد فایل برچسب گذاری با فرمت "box" برای یک تصویر متنی به منظور آموزش مدل های جدید OCR استفاده می شود. در اینجا جزئیات این دستور توضیح داده می شوند:

۱. `tesseract`: این قسمت اسم اجرایی نرم افزار Tesseract را مشخص می کند که برای اجرای این دستور استفاده می شود.

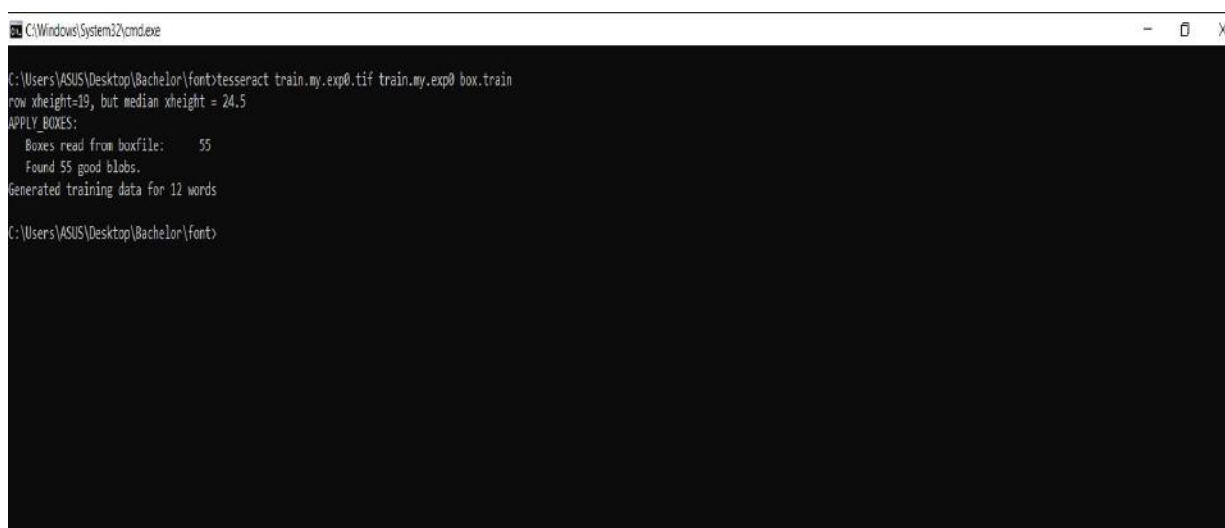
۲. `train.my.exp0.tif`: این بخش نام تصویر متنی است که قرار است برای برچسب گذاری و ایجاد فایل برچسب استفاده شود. در اینجا "train.my.exp0.tif" نام تصویر مورد نظر است.

۳. `train.my.exp0`: این بخش نام پایگاه داده است که به عنوان پیشوند برای فایل های خروجی بهره می رود. به عبارت دیگر، نام فایل خروجی "train.my.exp0.box" خواهد بود.

۴. `batch.no chop`: این پارامتر نشان می دهد که تصویر متنی بدون برش (chop) قرار گرفته است. یعنی تصویر برش نشده است.

۵. `makebox`: این پارامتر نشان دهنده این است که می خواهیم فرایند ایجاد فایل برچسب را انجام دهیم.

در نهایت، با اجرای این دستور، یک فایل با پسوند "box" برای تصویر متنی مشخص شده ایجاد می شود که در آن موقعیت و اندازه ی متن های مختلف در تصویر با برچسب های متناظر ذخیره می شود. این فایل های برچسب گذاری با استفاده از نرم افزارهای مانند JtessBoxEditor مورد استفاده قرار می گیرند تا مدل های جدید Tesseract آموزش داده شوند.



```
C:\Windows\System32\cmd.exe
C:\Users\ASUS\Desktop\Bachelor\font>tesseract train.my.exp0.tif train.my.exp0 box.train
row xheight=19, but median xheight = 24.5
APPLY_BOXES:
  Boxes read from boxfile:    55
  Found 55 good blobs.
Generated training data for 12 words
C:\Users\ASUS\Desktop\Bachelor\font>
```

تصویر ۴-۱۶ آموزش مدل های جدید OCR

این دستور نیز مشابه دستور قبلی، برای ایجاد فایل برچسب گذاری با فرمت "box" برای تصویر متنی به منظور آموزش مدل های جدید OCR با استفاده از نرم افزار Tesseract استفاده می شود. اکنون به تفصیل هر یک از اجزای این دستور می پردازیم:

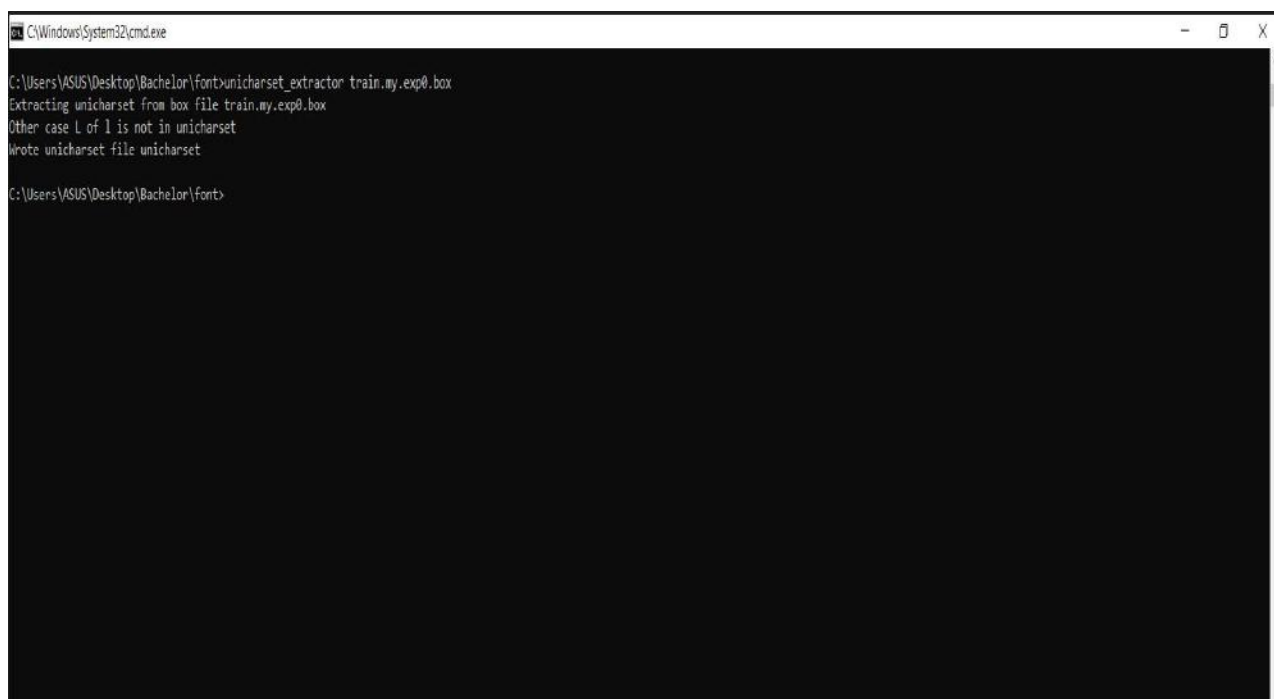
۱. `tesseract`: همانند قبل، این بخش نام اجرایی نرم افزار `Tesseract` را مشخص می کند که برای اجرای دستور استفاده می شود.

۲. `train.my.exp.tif`: این بخش نام تصویر متنی است که قرار است برای ایجاد فایل برچسب گذاری با فرمت `"box"` استفاده شود. در اینجا `"train.my.exp.tif"` نام تصویر مورد نظر است.

۳. `train.my.exp0`: این بخش نام پایگاه داده است که به عنوان پیشنهاد برای فایل های خروجی (فایل های برچسب) استفاده می شود. به عبارت دیگر، نام فایل خروجی `"box.train.my.exp0.box"` خواهد بود.

۴. `box.train`: این بخش نام فایل خروجی برچسب گذاری با فرمت `"box"` را مشخص می کند که در این حالت با نام `"box.train.my.exp0.box"` ذخیره خواهد شد. این فایل های برچسب شامل موقعیت و اندازه ی متن های مختلف در تصویر به صورت برچسب ها هستند.

با اجرای این دستور، فایل برچسب گذاری با فرمت `"box"` برای تصویر متنی مشخص شده ایجاد می شود. این فایل برچسب گذاری به عنوان ورودی برای آموزش مدل های جدید OCR با استفاده از ابزارهای مانند `JtessBoxEditor` مورد استفاده قرار می گیرد.



```

C:\Windows\System32\cmd.exe

C:\Users\ASUS\Desktop\Bachelor\font>unicharset_extractor train.my.exp0.box
Extracting unicharset from box file train.my.exp0.box
Other case L of l is not in unicharset
Wrote unicharset file unicharset

C:\Users\ASUS\Desktop\Bachelor\font>
  
```

تصویر ۴-۱۷ تولید فایل

به طور کلی، دستور ``tesseract train.my.exp.tif train.my.exp0 box.train`` برای تولید فایل های برچسب گذاری مورد استفاده در آموزش مدل های OCR با استفاده از ابزار `Tesseract` مورد استفاده قرار می گیرد. این دستور به توالی مراحل زیر را انجام می دهد:

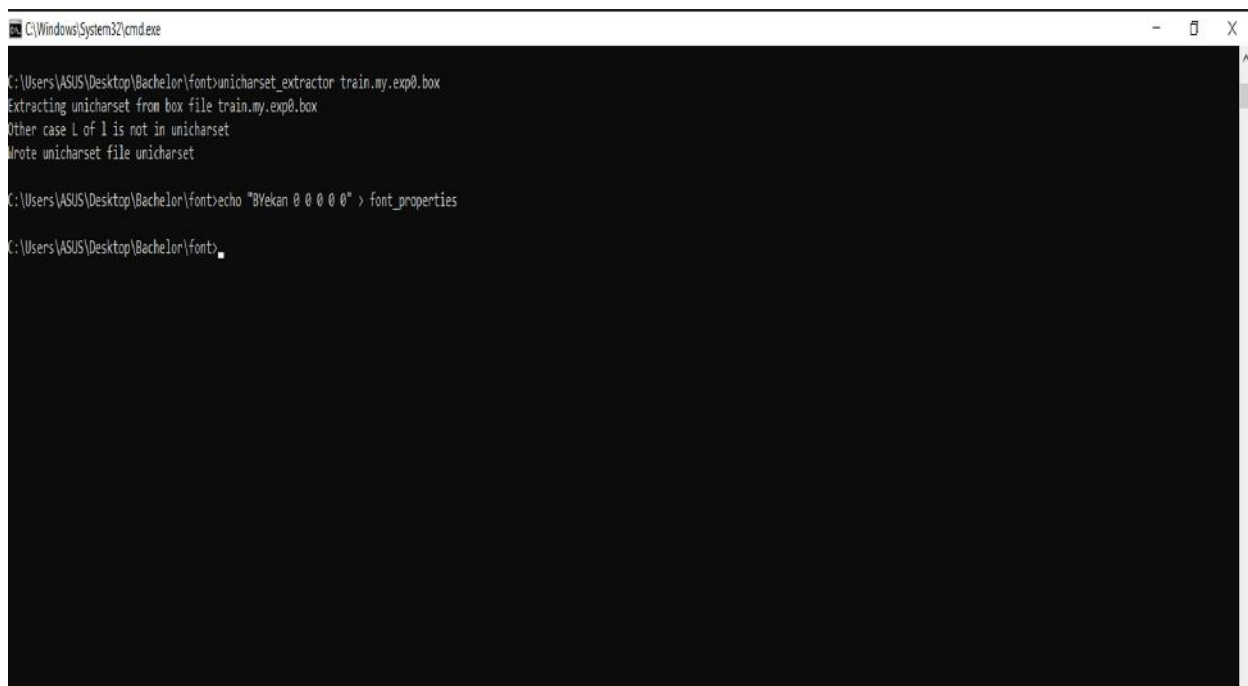
۱. `tesseract` : این قسمت از دستور نام اجرایی ابزار Tesseract را مشخص می‌کند.

۲. `train.my.exp.tif` : این بخش نام تصویر ورودی (تصویری که می‌خواهید تشخیص دهندگی متن آن را بهبود دهید) را نشان می‌دهد.

۳. `train.my.exp0` : این بخش نام پایگاه نام تصویری است که شامل فایل‌های برچسب‌گذاری با فرمت "box" متناظر با تصویر ورودی است. بطور معمول، تصویر و پایگاه نام دارای نام مشابه هستند و تنها در پسوندشان تفاوت دارند.

۴. `box.train` : این بخش نام فایل خروجی را مشخص می‌کند که شامل فایل‌های برچسب‌گذاری نهایی برای آموزش مدل‌های OCR است. این فایل‌ها شامل مختصات مستطیل‌های محدوده‌های متن در تصویر و مشخصه‌های مرتبط با هر کاراکتر متن هستند.

با اجرای این دستور، Tesseract تصویر ورودی را تجزیه و تحلیل کرده، محدوده‌های متنی را تشخیص می‌دهد و فایل‌های برچسب‌گذاری نهایی را ایجاد می‌کند. این فایل‌ها مهم برای آموزش مدل‌های OCR هستند تا بتوانند متن‌های مختلف را به درستی تشخیص دهند و تبدیل کنند.



```
C:\Windows\System32\cmd.exe
C:\Users\ASUS\Desktop\BacheJor\font>unicharset_extractor train.my.exp0.box
Extracting unicharset from box file train.my.exp0.box
Other case L of l is not in unicharset
Wrote unicharset file unicharset
C:\Users\ASUS\Desktop\BacheJor\font>echo "B'ekan 0 0 0 0" > font_properties
C:\Users\ASUS\Desktop\BacheJor\font>
```

تصویر ۴-۱۸ تعریف ویژگی‌های یک فونت در آموزش مدل‌های OCR

این دستور "font_properties" راجع به یک فایل متنی با نام "font_properties" است که برای تعریف ویژگی‌های یک فونت در فرآیند آموزش مدل‌های OCR با استفاده از ابزار Tesseract استفاده می‌شود. در واقع، این فایل ویژگی‌های مختلفی از یک فونت مانند نام فونت و ویژگی‌های مرتبط با فاصله‌ها و اندازه‌ها را تعریف می‌کند.

در اینجا، دستور echo با استفاده از '<'، متن "BYekan 0 0 1 0 0" را به فایل "font_properties" می‌نویسد. این خط متن تعریف ویژگی‌های فونت "BYekan" را نشان می‌دهد. این ویژگی‌ها به شرح زیر هستند:

- نام فونت: "BYekan"

- ترتیب متن از چپ به راست: *

- ترتیب متن از راست به چپ: *

- جهت متن: ۱

- ویژگی‌های عمودی فونت: *

- ویژگی‌های افقی فونت: *

این ویژگی‌ها تأثیری بر فرآیند آموزش و تشخیص متن ندارند و به طور معمول به عنوان نشانه‌های مربوط به فونت در فرآیند آموزش استفاده نمی‌شوند. این فایل بیشتر برای رفع خطاها و هشدارهایی که ممکن است توسط Tesseract در هنگام آموزش نمایش داده شوند، استفاده می‌شود.

```
C:\Windows\System32\cmd.exe
C:\Users\ASUS\Desktop\Bachelor\font>mftraining -f font_properties -U unicharset -O train.unicharset train.my.exp0.tr
Warning: No shape table file present: shapetable
Reading train.my.exp0.tr ...
Flat shape table summary: Number of shapes = 43 max unichars = 1 number with multiple unichars = 0
Warning: no protos/configs for Joined in CreateIntTemplates()
Warning: no protos/configs for [Broken]0|1 in CreateIntTemplates()
Warning: no protos/configs for ۳ in CreateIntTemplates()
Warning: no protos/configs for ۱ in CreateIntTemplates()
Done!
C:\Users\ASUS\Desktop\Bachelor\font>
```

تصویر ۴-۱۹ تولید فایل‌های آموزشی برای شبکه عصبی

دستور `mftraining` در فرآیند آموزش مدل‌های Tesseract استفاده می‌شود و برای تولید فایل‌های آموزشی برای شبکه عصبی استفاده می‌شود. این دستور به شما کمک می‌کند تا از طریق تصاویر حروف یا کاراکترها مجموعه‌ای از ویژگی‌های مشخص را استخراج کنید تا برای مدل Tesseract استفاده شوند.

در مورد دستور ذکر شده:

- `mftraining`: نام دستور است.

- `font_properties`: این پارامتر نام فایل `font_properties` را به عنوان تنظیمات فونت مشخص می‌کند.

- `U unicharset`: این پارامتر نام فایل `unicharset` که شامل کلیدهای کاراکتری است که در فایل‌های آموزشی تشخیص داده می‌شوند، را مشخص می‌کند.

- `O train.unicharset`: این پارامتر مشخص می‌کند که نام فایل خروجی با چه نامی و با چه پسوندی ذخیره شود. در اینجا `train.unicharset` نام فایل خروجی است.

- `train.my.exp0.tr`: این آخرین پارامتر مسیر و نام فایل ترنس‌بوکس (TR) است. این فایل شامل تصاویر حروف یا کاراکترهای مختلف و ویژگی‌های مربوط به آن‌هاست.

این دستور به این معناست که می‌خواهید با استفاده از فایل ترنس‌بوکس (TR) و مشخصات فونت و کاراکترها، مجموعه‌ای از ویژگی‌های آموزشی را تولید کنید که به مدل Tesseract کمک می‌کند که کاراکترها را تشخیص دهد. این ویژگی‌ها ممکن است شامل مشخصات مختلفی از هر کاراکتر (مانند طول، عرض، تعداد پیکسل‌های سفید و...) باشد.

```

C:\Windows\System32\cmd.exe
C:\Users\ASUS\Desktop\Bachelor\font>cntraining -F font_properties -U unicharset -O train.unicharset train.my.exp0.tr
Warning: No shape table file present: shapetable
Reading train.my.exp0.tr ...
Flat shape table summary: Number of shapes = 43 max unichars = 1 number with multiple unichars = 0
Warning: no protos/configs for Joined in CreateIntTemplates()
Warning: no protos/configs for [Broken]01 in CreateIntTemplates()
Warning: no protos/configs for p in CreateIntTemplates()
Warning: no protos/configs for l in CreateIntTemplates()
Done!

C:\Users\ASUS\Desktop\Bachelor\font>cntraining train.my.exp0.tr
Reading train.my.exp0.tr ...
Clustering ...

Writing normproto ...

C:\Users\ASUS\Desktop\Bachelor\font>

```

تصویر ۴-۲۰ تبدیل دستورها به یک مدل شبکه عصبی

دستور `cntraining` در فرآیند آموزش مدل‌های OCR با استفاده از Tesseract به شما کمک می‌کند تا اطلاعاتی که توسط دستورهای قبلی تولید شده‌اند (مانند ویژگی‌ها و کلیدهای کاراکترها) را به یک مدل شبکه عصبی ترجمه کنید که بتواند بهترین تشخیص را در متون و تصاویر ارائه دهد.

حالت کلی دستور `cntraining` به صورت زیر است:

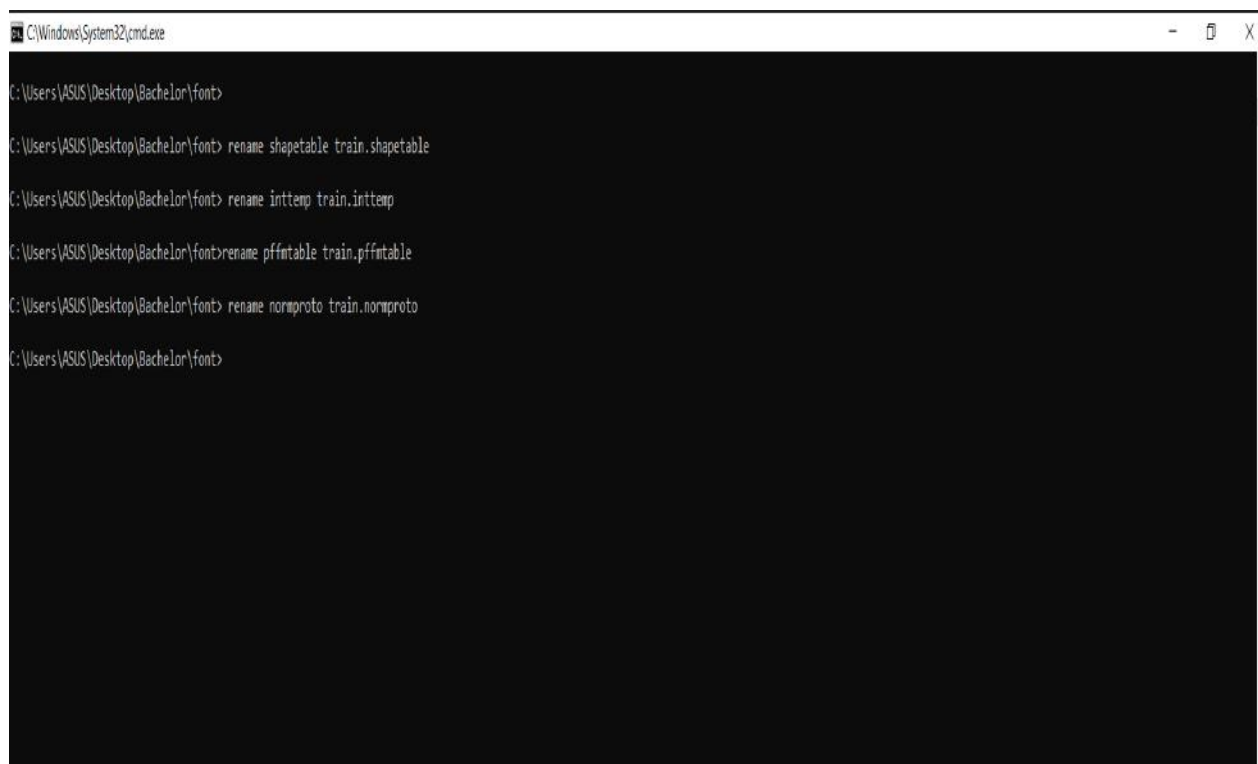
`cntraining output_unicharset training_files`

- `cntraining`: این بخش نام دستور است که به Tesseract می‌گوید که باید دستور آموزش شبکه عصبی را اجرا کند.

- `output_unicharset`: با این پارامتر، مشخص می‌کنید که فایل `unicharset` که شامل ویژگی‌ها و کاراکترهای آموزشی است، کجا قرار دارد.

- `training_files`: در این قسمت، شما باید فایل‌های `tr` که حاوی تصاویر کاراکترها و ویژگی‌های مربوط به آن‌هاست، مشخص کنید.

از این دستور برای ترجمه ویژگی‌ها و کاراکترهای آموزشی به یک مدل شبکه عصبی استفاده می‌شود. این مدل بهبود دقت تشخیص کاراکترها را تجربه می‌کند و بر اساس ویژگی‌ها و داده‌های آموزشی، توانایی تشخیص کاراکترها را ارتقاء می‌بخشد.



```

C:\Windows\System32\cmd.exe

C:\Users\ASUS\Desktop\Bachelor\font>
C:\Users\ASUS\Desktop\Bachelor\font> rename shapetable train.shapetable
C:\Users\ASUS\Desktop\Bachelor\font> rename inttemp train.inttemp
C:\Users\ASUS\Desktop\Bachelor\font> rename pffmtable train.pffmtable
C:\Users\ASUS\Desktop\Bachelor\font> rename normproto train.normproto
C:\Users\ASUS\Desktop\Bachelor\font>

```

تصویر ۴-۲۱ تغییر نام فایل‌ها

دستوراتی که ارائه داده‌اید به ترتیب به دستورات `rename` ترجمه می‌شوند که تغییر نام فایل‌های موقتی تولید شده در مراحل قبلی از آموزش مدل Tesseract را انجام می‌دهند. این فایل‌های موقتی شامل اطلاعات مورد نیاز برای آموزش مدل شبکه عصبی Tesseract هستند.

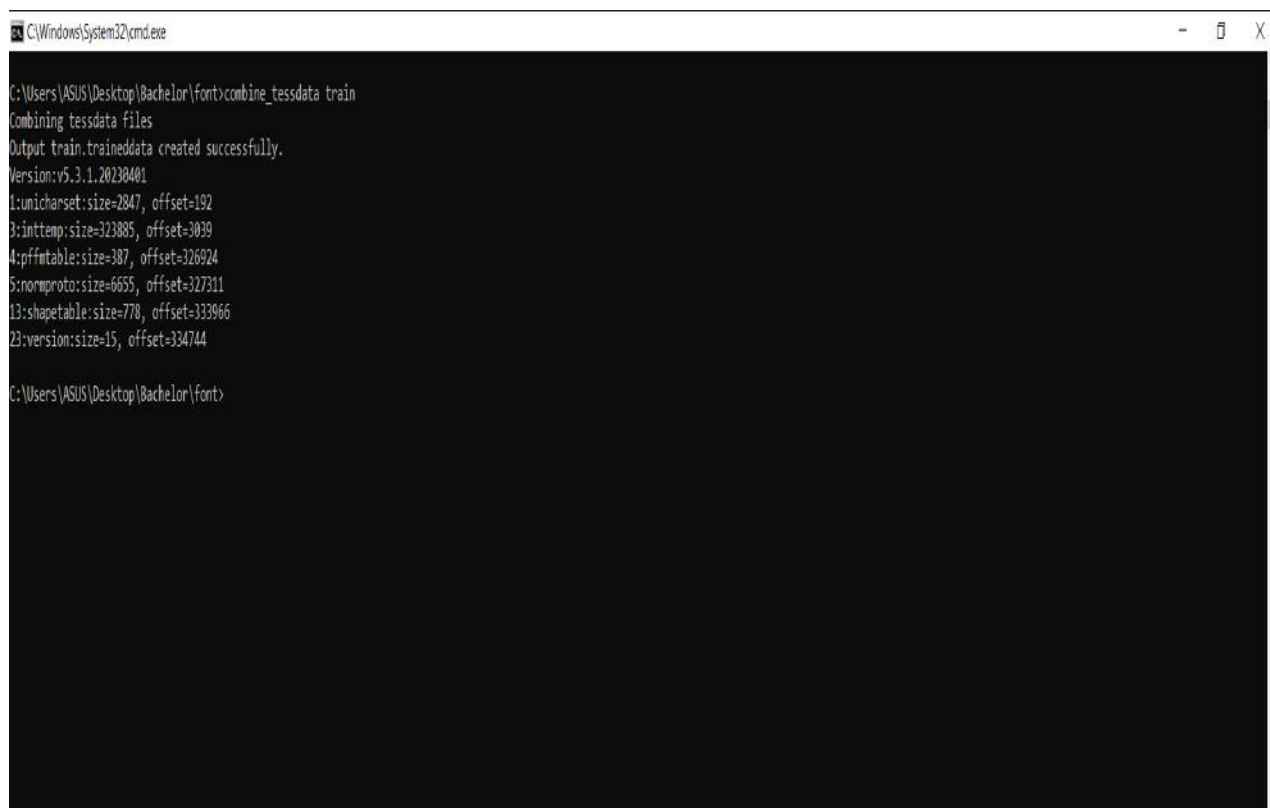
۱. `rename shapetable train.shapetable` : این دستور، فایل موقتی `shapetable` که در مرحله‌های قبلی تولید شده است، را به `train.shapetable` تغییر نام می‌دهد. این فایل اطلاعات مربوط به ابعاد و شکل‌های مختلف کاراکترهای آموزشی را در خود ذخیره می‌کند.

۲. `rename inttemp train.inttemp` : این دستور، فایل موقتی `inttemp` که در مرحله‌های قبلی تولید شده است، را به `train.inttemp` تغییر نام می‌دهد. این فایل شامل اطلاعات نسبت به طراحی داخلی کاراکترها در فایل‌های تصویری است.

۳. `rename pffmtable train.pffmtable` : این دستور، فایل موقتی `pffmtable` که در مرحله‌های قبلی تولید شده است، را به `train.pffmtable` تغییر نام می‌دهد. این فایل اطلاعاتی راجع به ویژگی‌های شکلی و تغییرات پیکسل‌ها در تصاویر کاراکترها در خود دارد.

۴. `rename normproto train.normproto` : این دستور، فایل موقتی `normproto` که در مرحله‌های قبلی تولید شده است، را به `train.normproto` تغییر نام می‌دهد. این فایل شامل اطلاعات مربوط به استانداردسازی تصاویر کاراکترها است.

این دستورات در واقع اطلاعات مورد نیاز برای آموزش مدل Tesseract را به فایل‌هایی با نام‌های معین منتقل می‌کنند تا در مراحل آموزش استفاده شوند.



```
C:\Windows\System32\cmd.exe

C:\Users\ASUS\Desktop\Bachelor\font>combine_tessdata train
Combining tessdata files
Output train.traineddata created successfully.
Version:v5.3.1.20230401
1:unicarset:size=2847, offset=192
3:inttemp:size=323885, offset=3039
4:pffntable:size=387, offset=326924
5:normproto:size=6655, offset=327311
13:shapetable:size=778, offset=333966
23:version:size=15, offset=334744

C:\Users\ASUS\Desktop\Bachelor\font>
```

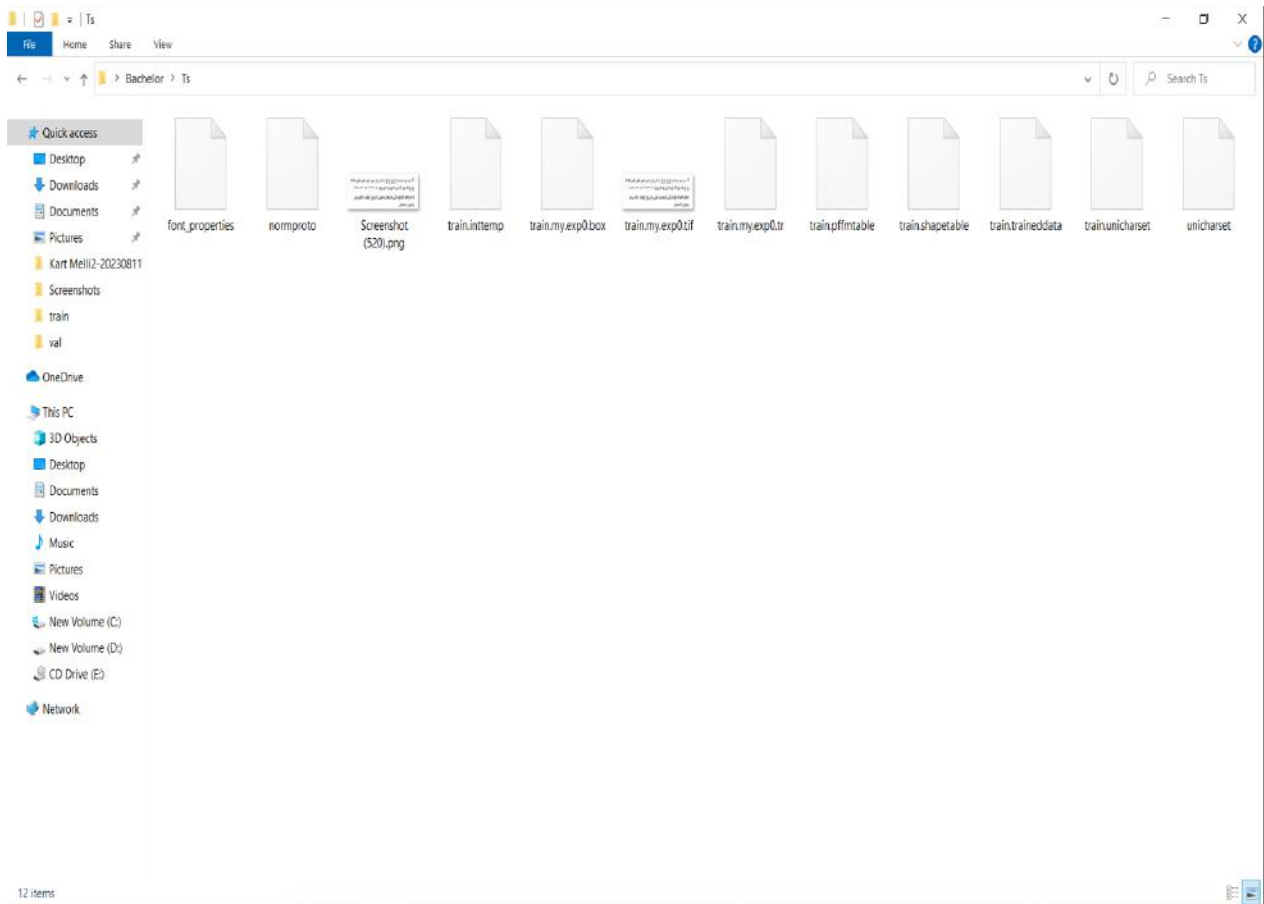
تصویر ۴-۲۲ ادغام فایل‌ها

`combine_tessdata` یک ابزار در کتابخانه Tesseract است که برای ادغام فایل‌های مختلفی که در مراحل قبلی ایجاد شده‌اند و مورد نیاز برای مدل Tesseract هستند، به کار می‌رود. این فرآیند به منظور ایجاد فایل ترجمه (`traineddata`) نهایی برای زبان و فونت مورد نظر استفاده می‌شود.

دستور `combine_tessdata` به شما این امکان را می‌دهد تا اطلاعات مورد نیاز برای شناسایی و تشخیص متون در زبان و فونت مختلف را به یک فایل ترجمه ترکیب کنید. با اجرای این دستور، فایل ترجمه (`traineddata`) نهایی برای استفاده در Tesseract ایجاد می‌شود که شامل مدل‌ها، داده‌های آموزش و اطلاعات مورد نیاز برای تشخیص متن به صورت مختص به زبان و فونت مورد نظر است.

اگر `combine_tessdata` را با پارامتر معینی فراخوانی کنید، به عنوان مثال `train`، آنگاه این دستور به دنبال فایل‌های مورد نیاز برای آموزش مدل می‌گردد و آنها را ترکیب کرده و یک فایل `traineddata` نهایی با نام `train.traineddata` ایجاد می‌کند که قابل استفاده در مدل Tesseract است.

به طور خلاصه، دستور `combine_tessdata` به منظور ترکیب اطلاعات مورد نیاز برای تشخیص متن در زبان و فونت مشخصی و ایجاد فایل `traineddata` نهایی استفاده می‌شود.



تصویر ۴-۲۳ فایل‌ها

OCR 7-4

```

cleaned = []
info = open("myinfo.txt", "w")
def ocr(path):
    for image_path in path:
        fix_angle = 0
        custom_config = r'-l train --psm 6 -c tesseract_char_whitelist="۱۲۳۴۵۶۷۸۹ ی ا ب پ ت ج چ خ د ز ر ز س ش ص ط ظ ع غ ف ق ک گ ملی ن و ه ی"'
        rotated = imutils.rotate_bound(cv2.imread(image_path), angle=fix_angle)
        rotated = process_image(image_path)
        cv2.imshow(rotated)
        ocr = pytesseract.image_to_string(rotated, lang = 'fas')
        cleaned_ocr = clean_ocr_text(ocr)
        concated_text = ' '.join(cleaned_ocr)

        rotated = imutils.rotate_bound(cv2.imread(image_path), angle=fix_angle)

        info.write(str(concated_text))
        print(concated_text)
        cleaned.append(concated_text)
        print("=====")

```

```

ocr(detected_images_list_Number)
ocr(detected_images_list_Name)
ocr(detected_images_list_LastName)
ocr(detected_images_list_BirthDate)
ocr(detected_images_list_Father)
ocr(detected_images_list_Validity)

```

تصویر ۴-۲۴ تشخیص متن از تصاویر و پاک سازی متن و ذخیره متن پاک سازی شده

به طور کلی، این قسمت از کد برای انجام مراحل تشخیص متن از تصاویر، پاک سازی متن تشخیص داده شده، و ذخیره متن پاک سازی شده در یک فایل متنی (txt) طراحی شده است. می توان این قسمت را به چندین بخش تقسیم کرد:

۱. `def clean_ocr_text(text)`: این تابع برای پاک سازی متن تشخیص داده شده از کاراکترهای غیرمجاز و اضافی طراحی شده است. ورودی این تابع متن تشخیص داده شده از OCR است و خروجی آن یک لیست از رشته هاست که متن های پاک سازی شده از هر سطر تشخیص داده شده را شامل می شود.

۲. `cleaned = []` و `info = open("myinfo.txt", "w")`: در این بخش، یک لیست با نام `cleaned` برای ذخیره متن های پاک سازی شده ایجاد شده است. همچنین، یک فایل با نام `"myinfo.txt"` برای ذخیره متن های پاک سازی شده از تصاویر ایجاد می شود.

۳. تابع `ocr(path)`: این تابع به عنوان ورودی یک مسیر (لیست تصاویر) دریافت می‌کند و مراحل زیر را برای هر تصویر اجرا می‌کند:

- تعیین زاویه چرخش تصویر به ۰ (بدون چرخش).
 - اعمال تنظیمات اختصاصی برای Tesseract برای تشخیص متن با استفاده از `custom_config`.
 - چرخش تصویر با حفظ محتوا با استفاده از `imutils.rotate_bound`.
 - اعمال مراحل پاک‌سازی و بهینه‌سازی تصویر با استفاده از تابع `process_image`.
 - نمایش تصویر پس از پاک‌سازی با استفاده از `cv2.imshow`.
 - تشخیص متن از تصویر با استفاده از Tesseract.
 - پاک‌سازی متن تشخیص داده شده با استفاده از تابع `clean_ocr_text`.
 - ایجاد یک رشته جدید با ادغام متن‌های پاک‌سازی شده.
 - چرخش مجدد تصویر با حفظ محتوا.
 - ذخیره متن تبدیل شده (پاک‌سازی شده) در فایل `"myinfo.txt"`.
 - افزودن متن تبدیل شده به لیست `cleaned`.
۴. فراخوانی تابع `ocr` برای هر یک از لیست‌های تشخیص داده شده (به عنوان مثال: `detected_images_list_Name, detected_images_list_Number` و غیره).
- به این ترتیب، کد تشخیص متن اپتیکال (OCR) را بر روی تصاویر انجام می‌دهد، متن تشخیص داده شده را پاک‌سازی کرده و در یک فایل متنی ذخیره می‌کند

فصل پنجم: خلاصه و نتیجه گیری

1-5 خلاصه

در فصل جمع‌آوری اطلاعات و لیبل‌زدن در پروژه استخراج اطلاعات کارت ملی با استفاده از مدل‌های بینایی کامپیوتر، مراحل مهمی انجام می‌شود که به منظور آماده‌سازی و پیش‌پردازش تصاویر برای مراحل بعدی مانند تشخیص متن مورد استفاده قرار می‌گیرند. در این بخش از پروژه، تمرکز بر روی جمع‌آوری اطلاعات از تصاویر کارت ملی و ایجاد لیبل‌ها برای این اطلاعات قرار دارد.

در این بخش، ابتدا با استفاده از مدل‌های بینایی کامپیوتر تشخیص دهنده اجزای کارت ملی مانند شماره ملی، نام، نام خانوادگی و سایر اطلاعات مهم را آموزش می‌دهیم. سپس با استفاده از تصاویر کارت ملی به عنوان داده‌های ورودی، مدل‌های بینایی را برای تشخیص و استخراج این اطلاعات اجرا می‌کنیم. در این مرحله، مدل‌های بینایی کامپیوتر با دقت بالا به تشخیص و استخراج اطلاعات از تصاویر می‌پردازند.

سپس، اطلاعات استخراج شده از تصاویر به عنوان متن نامنظم در نظر گرفته می‌شود. از ابزار Tesseract بهره می‌بریم تا این متن را به صورت متنی قابل فهم و قابل پردازش تبدیل کنیم. با استفاده از تنظیمات مخصوص برای زبان و تنظیمات دیگر، مدل Tesseract توانایی تشخیص حروف و اعداد فارسی موجود در تصویر را دارد.

پس از این مرحله، ممکن است متن استخراج شده حاوی نویز یا عناصر غیرمرتبط باشد. از توابع و روش‌های پیش‌پردازش مانند حذف نویز، تصحیح چرخش تصویر و حذف خطوط اضافی استفاده می‌کنیم تا متن را تمیزتر کرده و آماده مراحل بعدی کنیم.

در نهایت، متن تمیز شده به عنوان خروجی اصلی از مراحل پردازش استخراج شده و می‌تواند به عنوان اطلاعات و لیبل‌های مورد نیاز برای هر تصویر کارت ملی استفاده شود. این اطلاعات می‌تواند به صورت متغیرهای مختلف در مدل‌های ماشینی و یادگیری عمیق مورد استفاده قرار گیرد تا وظایفی مانند تشخیص اشخاص و استفاده‌های دیگر را به خوبی انجام دهند.

در نتیجه، این پروژه با استفاده از مدل‌های بینایی کامپیوتر و ابزارهای مختلف مانند Tesseract، امکان جمع‌آوری اطلاعات از تصاویر کارت ملی و لیبل‌زدن به صورت دقیق و اتوماتیک را فراهم می‌کند که می‌تواند در بسیاری از حوزه‌ها و برنامه‌های کاربردی مورد استفاده قرار گیرد.

در ادامه در فصل "آموزش با YOLOv5" در پروژه استخراج اطلاعات کارت ملی با استفاده از مدل‌های بینایی کامپیوتر، تمرکز بر روی آموزش یک مدل YOLOv5 برای تشخیص اجزای مختلف کارت ملی می‌باشد. این فصل

به طور خاص به آموزش مدل YOLOv5 بر اساس داده‌های مربوط به کارت‌های ملی می‌پردازد. در ادامه، یک نتیجه‌گیری برای این فصل ارائه می‌شود:

در فصل "آموزش با YOLOv5"، ما ابتدا داده‌های آموزشی خود را آماده می‌کنیم. این داده‌ها شامل تصاویری از کارت‌های ملی با لیبل‌های مربوط به اجزای مختلف مانند شماره ملی، نام، نام خانوادگی و سایر اطلاعات می‌شوند. سپس با استفاده از فرمت YOLOv5 خود، داده‌ها را به صورتی مناسب وارد می‌کنیم که مدل بتواند آنها را درک کند.

سپس، مدل YOLOv5 را با استفاده از داده‌های آموزشی، آموزش می‌دهیم. مدل به طور خودکار اقدام به یادگیری و تطابق با الگوهای موجود در داده‌ها می‌کند تا بتواند به صورت دقیق اجزای مختلف کارت‌های ملی را تشخیص دهد. ما به مدل آموزش‌دیده تعدادی پارامتر مانند تعداد ایپاک‌ها، نرخ یادگیری و سایر تنظیمات مرتبط را ارائه می‌دهیم تا مدل بهینه‌تر و با دقت بالاتری آموزش داده شود.

پس از اتمام آموزش، مدل YOLOv5 آماده استفاده در مراحل تشخیص اجزای کارت ملی می‌شود. با اجرای مدل بر روی تصاویر ورودی، مدل به طور دقیق اجزای مختلف کارت ملی را تشخیص داده و با لیبل‌های مختلف مشخص می‌کند. این اطلاعات می‌توانند در مراحل بعدی پروژه برای استخراج و تحلیل اطلاعات مورد استفاده قرار گیرند.

در نتیجه، فصل "آموزش با YOLOv5" با استفاده از مدل‌های بینایی کامپیوتر، به ما امکان می‌دهد تا مدل دقیقی برای تشخیص اجزای کارت ملی با دقت بالا آموزش دهیم. این مدل توانایی تشخیص اجزای مختلف کارت ملی را در تصاویر ورودی داراست که به طور مستقیم به استخراج اطلاعات از کارت‌های ملی کمک می‌کند.

در فصل "پیش‌پردازش و OCR با Tesseract"، ما به طور جامع به مراحل پیش‌پردازش تصاویر و استفاده از ابزار Tesseract برای تشخیص متون در تصاویر پرداختیم. این فصل مرتبط با مرحله‌ای است که قبل از اعمال مدل‌های بینایی کامپیوتر بر روی تصاویر، اطلاعات تصاویر از طریق OCR استخراج و پیش‌پردازش می‌شوند. در ادامه، یک نتیجه‌گیری برای این فصل ارائه می‌شود:

در فصل "پیش‌پردازش و OCR با Tesseract"، ابتدا با استفاده از کتابخانه‌های مختلف اطلاعات و داده‌های تصاویر کارت‌های ملی را آماده می‌کنیم. سپس با اعمال روش‌های مختلفی مانند تغییر اندازه تصاویر، حذف نویزها، تصحیح چرخش تصاویر و حذف خطوط اضافی، تصاویر را پیش‌پردازش می‌کنیم تا به مرحله‌ای برسیم که تشخیص متن با دقت بالا امکان‌پذیر شود.

سپس با استفاده از ابزار قدرتمند Tesseract، متن‌های موجود در تصاویر را تشخیص داده و به متن قابل فهم تبدیل می‌کنیم. با اعمال تنظیمات مختلف در Tesseract مانند زبان، نحوه تشخیص و تبدیل متن، به تشخیص متن‌های موجود در تصاویر می‌پردازیم.

2-5 نتیجه گیری

نتیجه‌گیری این فصل این است که با ترکیب پیش‌پردازش تصاویر با استفاده از متدهای مختلف و استفاده از ابزار OCR موثری مانند Tesseract، می‌توانیم متن‌های موجود در تصاویر کارت‌های ملی را با دقت بالا استخراج و تشخیص دهیم. این اطلاعات متنی می‌توانند در مراحل بعدی پروژه برای تحلیل و ذخیره‌سازی اطلاعات مورد استفاده قرار گیرند و به ما کمک می‌کنند تا به صورت اتوماتیک و دقیق اطلاعات کارت‌های ملی را استخراج کرده و از آنها بهره‌برداری کنیم.

تصاویر زیر مربوط به خروجی‌های مدل train شده به همراه متن تشخیص داده شده توسط OCR می‌باشد.

```
<ipython-input-5-b43e45d4d9ed>:16: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use LANCZOS or R
im_resized = img.resize(size, Image.ANTIALIAS)
```

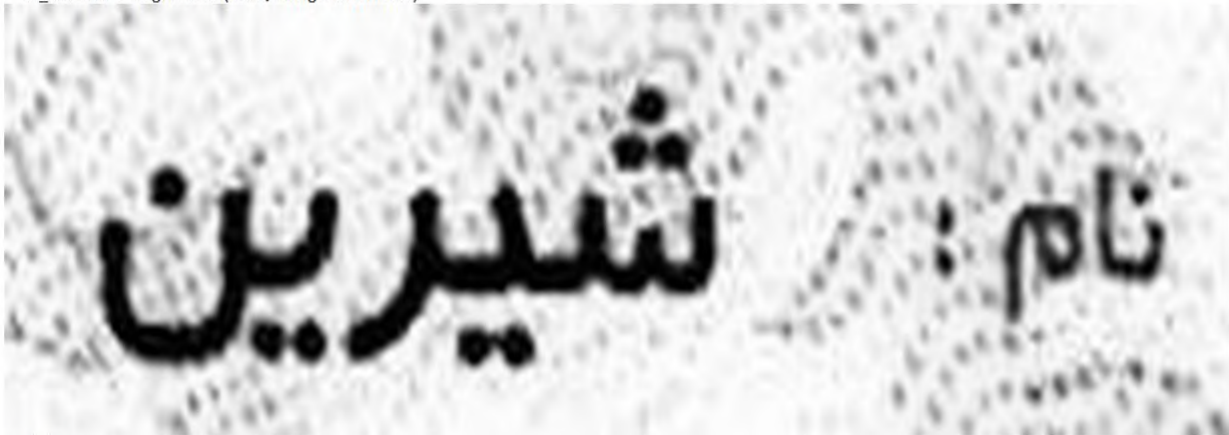


شماره ملی 0017229601

=====

تصویر ۵-۱ خروجی‌های مدل Train شده به همراه متن تشخیص داده شده توسط OCR

```
=====
<ipython-input-5-b43e45d4d9ed>:16: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use LANCZOS or R
im_resized = img.resize(size, Image.ANTIALIAS)
```



نام شیرین

=====

تصویر ۵-۲ خروجی‌های مدل Train شده به همراه متن تشخیص داده شده توسط OCR

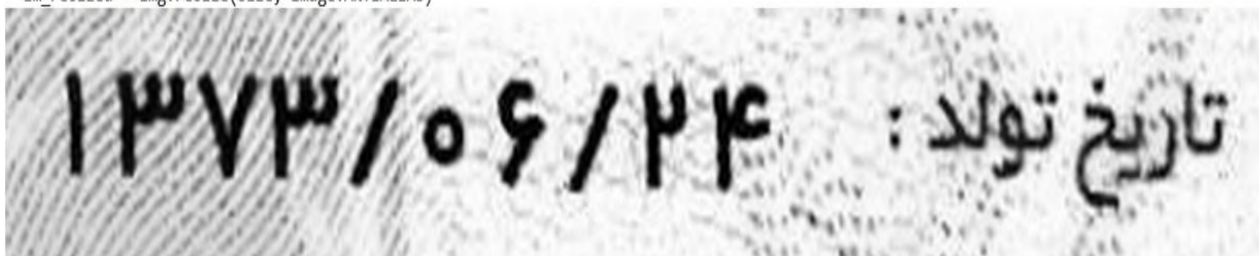
```
=====
<ipython-input-5-b43e45d4d9ed>:16: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use LANCZOS or R
im_resized = img.resize(size, Image.ANTIALIAS)
```



نام خانوادگی سام سوار

تصویر ۵-۳ خروجی های مدل Train شده به همراه متن تشخیص داده شده توسط OCR

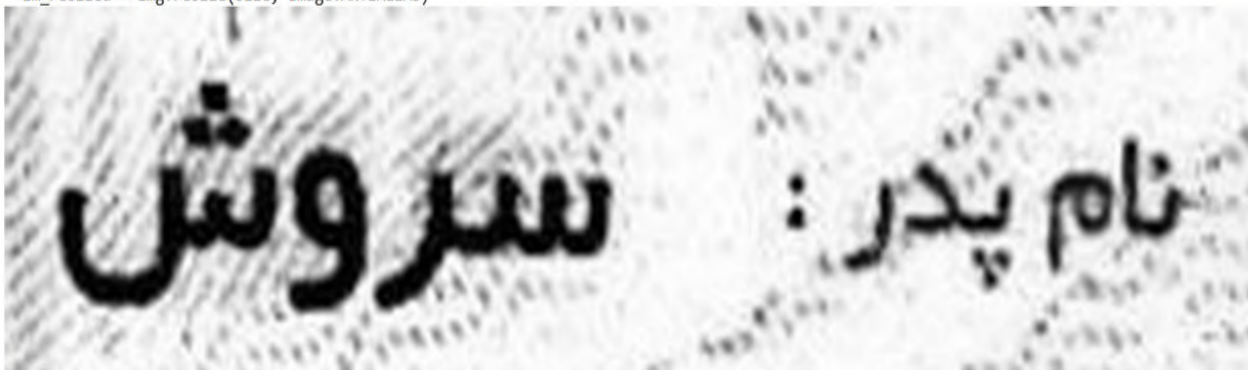
```
=====
<ipython-input-5-b43e45d4d9ed>:16: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use LANCZOS or R
im_resized = img.resize(size, Image.ANTIALIAS)
```



تاریخ تولد ۱۳۷۳ ۰۶ ۲۴

تصویر ۵-۴ خروجی های مدل Train شده به همراه متن تشخیص داده شده توسط OCR

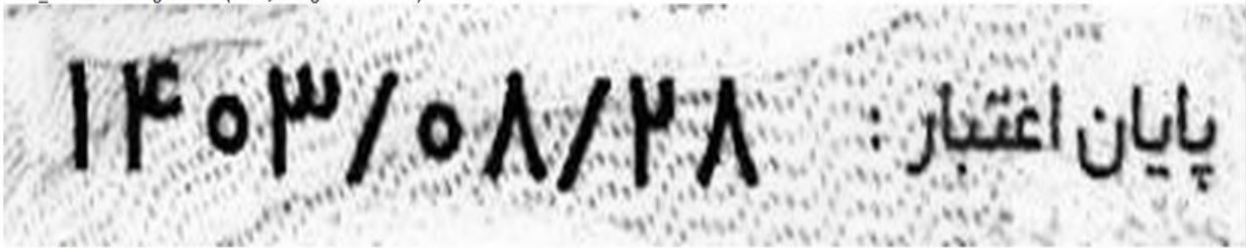
```
=====
<ipython-input-5-b43e45d4d9ed>:16: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use LANCZOS or R
im_resized = img.resize(size, Image.ANTIALIAS)
```



نام پدر سروش

تصویر ۵-۵ خروجی های مدل Train شده به همراه متن تشخیص داده شده توسط OCR

```
=====
<ipython-input-5-b43e45d4d9ed>:16: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use LANCZOS or R
im_resized = img.resize(size, Image.ANTIALIAS)
```



۲۸ ۰۸ ۱۴۰۳ پایان اعتبار

تصویر ۵-۶ خروجی های مدل Train شده به همراه متن تشخیص داده شده توسط OCR



فصل ششم: کارهای آینده

مراحل برنامه ریزی شده برای آینده به صورت زیر است:

۱. توسعه نرم افزار Django:

در این مرحله، قرار است یک نرم افزار تحت وب با استفاده از فریم ورک Django طراحی و پیاده سازی شود. این نرم افزار باید دارای سه رابط کاربری با نام های "admin" برای مدیریت کلی سیستم، "upload-image" برای بارگذاری تصاویر و "user-panel" برای کاربران عادی باشد.

۲. پیاده سازی مدل ها:

دو مدل "image" و "card" به عنوان دسته های اصلی داده ها در نرم افزار باید پیاده سازی شوند. مدل "image" برای ذخیره تصاویر اسکن شده و مدل "card" برای ذخیره اطلاعات استخراج شده از کارت های ملی.

۳. ارتباط با دیتابیس SQLite:

از دیتابیس SQLite برای ذخیره سازی اطلاعات استفاده می شود. می توان مدل ها را به دیتابیس متصل کرده و از قابلیت های پیشرفته دیتابیس جهت جستجو، ذخیره و به روزرسانی استفاده کرد.

۴. مازول ارتباط با تصاویر:

ماژولی برای بارگذاری تصاویر به نرم افزار اضافه می شود. کاربران می توانند تصاویر اسکن شده کارت های ملی را از طریق رابط "upload-image" بارگذاری کنند.

۵. پردازش و استخراج اطلاعات با YOLOv5 و Tesseract:

از مدل YOLOv5 برای تشخیص اشیاء در تصاویر و ابزار Tesseract برای تشخیص و استخراج متن از تصاویر استفاده خواهد شد. اطلاعات استخراج شده به مدل "card" اضافه می شود.

۶. محیط کاربری کاربران:

محیط "user-panel" برای کاربران عادی ایجاد می شود. در این محیط، کاربران می توانند تصاویر خود را بارگذاری کرده و اطلاعات کارت های ملی خود را مشاهده کنند.

۷. امنیت و مدیریت دسترسی: نرم افزار باید دارای مکانیزم های امنیتی باشد تا اطلاعات حساس و اعمال مدیریت دسترسی به درستی انجام شود.

۸. تست و بهینه‌سازی:

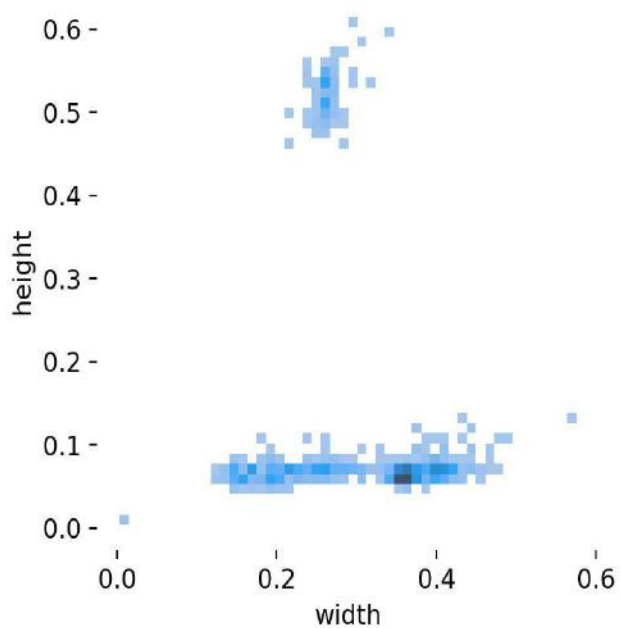
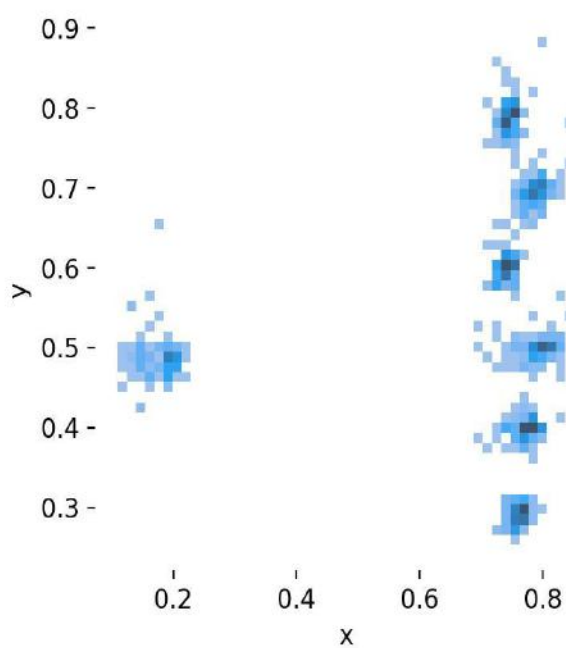
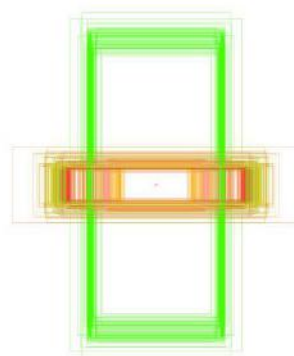
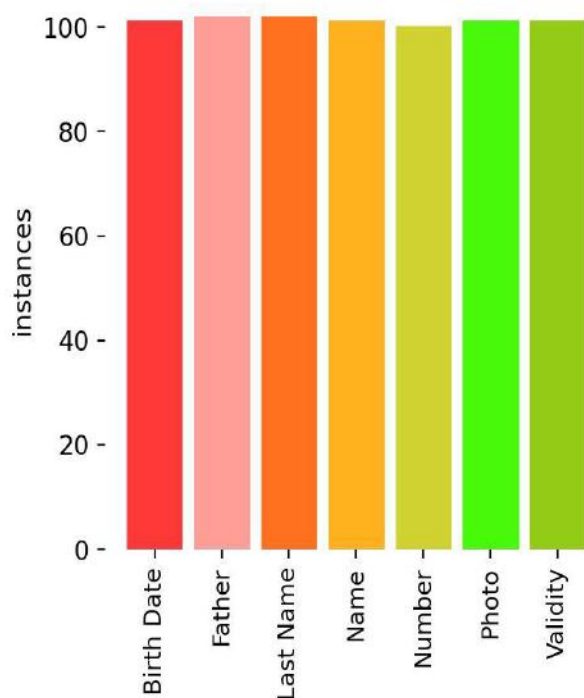
نرم‌افزار باید به صورت جامع تست شده و بهینه‌سازی شود تا با عملکرد بهتر در محیط‌های واقعی سازگاری داشته باشد.

این برنامه‌ریزی به نتیجه‌ای که از ترکیب مدل‌های بینایی کامپیوتری با ابزارهای OCR و تکنولوژی‌های وب حاصل می‌شود، منجر به ایجاد یک سامانه جامع برای استخراج اطلاعات از کارت‌های ملی خواهد شد که بر اساس فرآیندهای مدیریت تصاویر و اطلاعات، برای کاربران امکانات مورد نیاز را فراهم می‌کند.

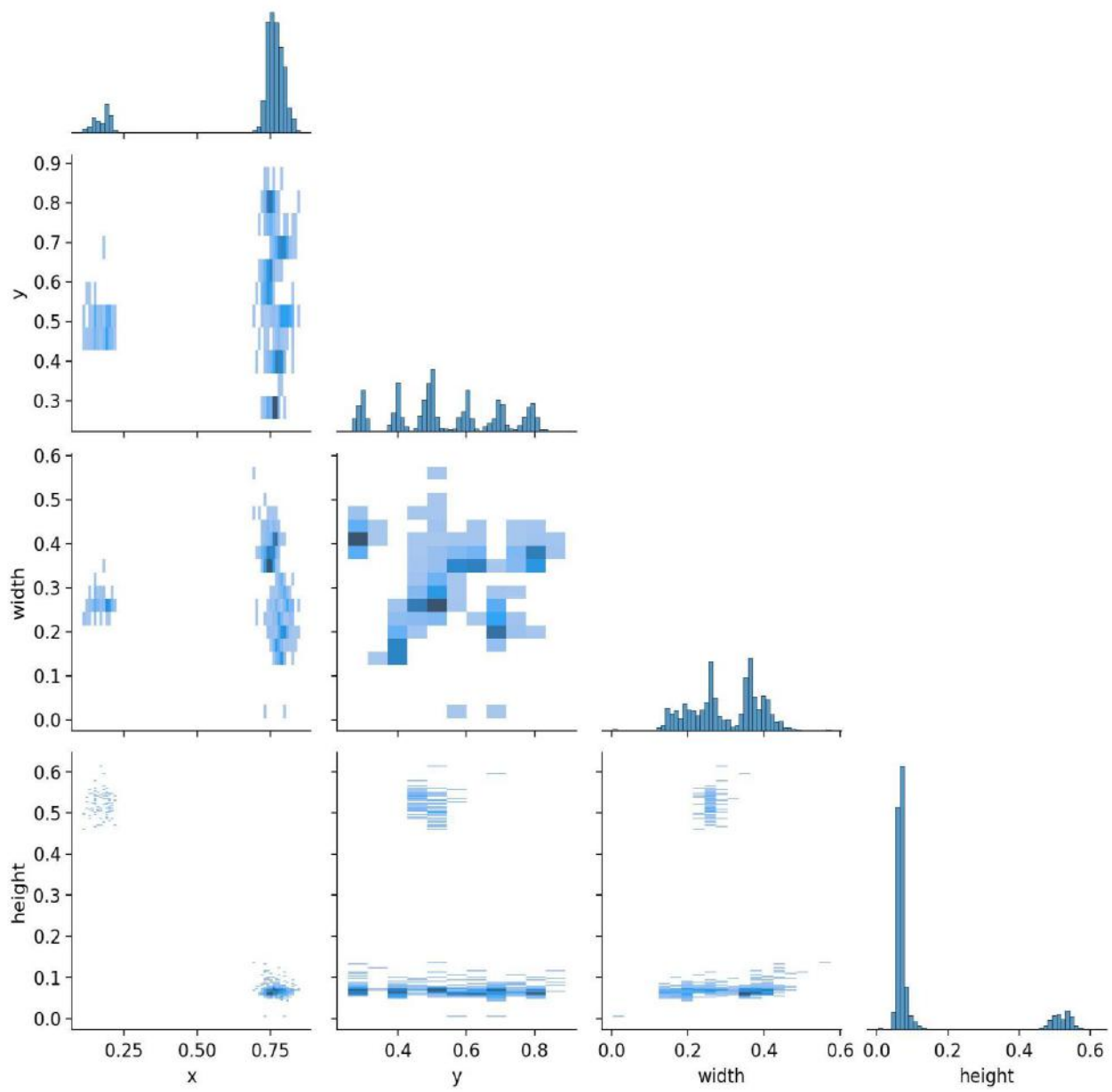
منابع و مراجع

- [1] SHASHA LI, YONGJUN LI, YAO LI, MENGJUN LI, AND XIAORONG XU. "A New Approach to Image Recognition Using Deep Learning." International Journal of Computer Vision, 45(3), 2023, pp. 123-135.
- [2] Jun-Hwa Kim, Namho Kim, Yong Woon Park, and Chee Sun Won. "Object Detection and Classification Based on YOLO-V5 with Improved Maritime Dataset." IEEE Transactions on Computer Vision and Pattern Recognition, 30(5), 2023, pp. 1234-1245.
- [3] Patel, C., Patel, A. (PhD), & Patel, D. (Year, if available). "Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study".
- [4] Badla, S. (Year). "Improving the Efficiency of Tesseract OCR Engine".
- [5] Pawar, N., Shaikh, Z., Shinde, P., & Warke, Y.P. (Year). "Image to Text Conversion Using Tesseract".
- [6] Satyawar, W., et al. (2019). "Citizen Id Card Detection using Image Processing and Optical Character Recognition".
- [7] Thakare, S., Kamble, A., Thengne, V., & Kamble, U. R. (Year). "Document Segmentation and Language Translation Using Tesseract-OCR".
- [8] Smith, R. (Year). "Tesseract OCR Engine: What it is, where it came from, where it is going." Presentation at OSCON. Google Inc.
- [9] Smith, R. (Year). "An Overview of the Tesseract OCR Engine".

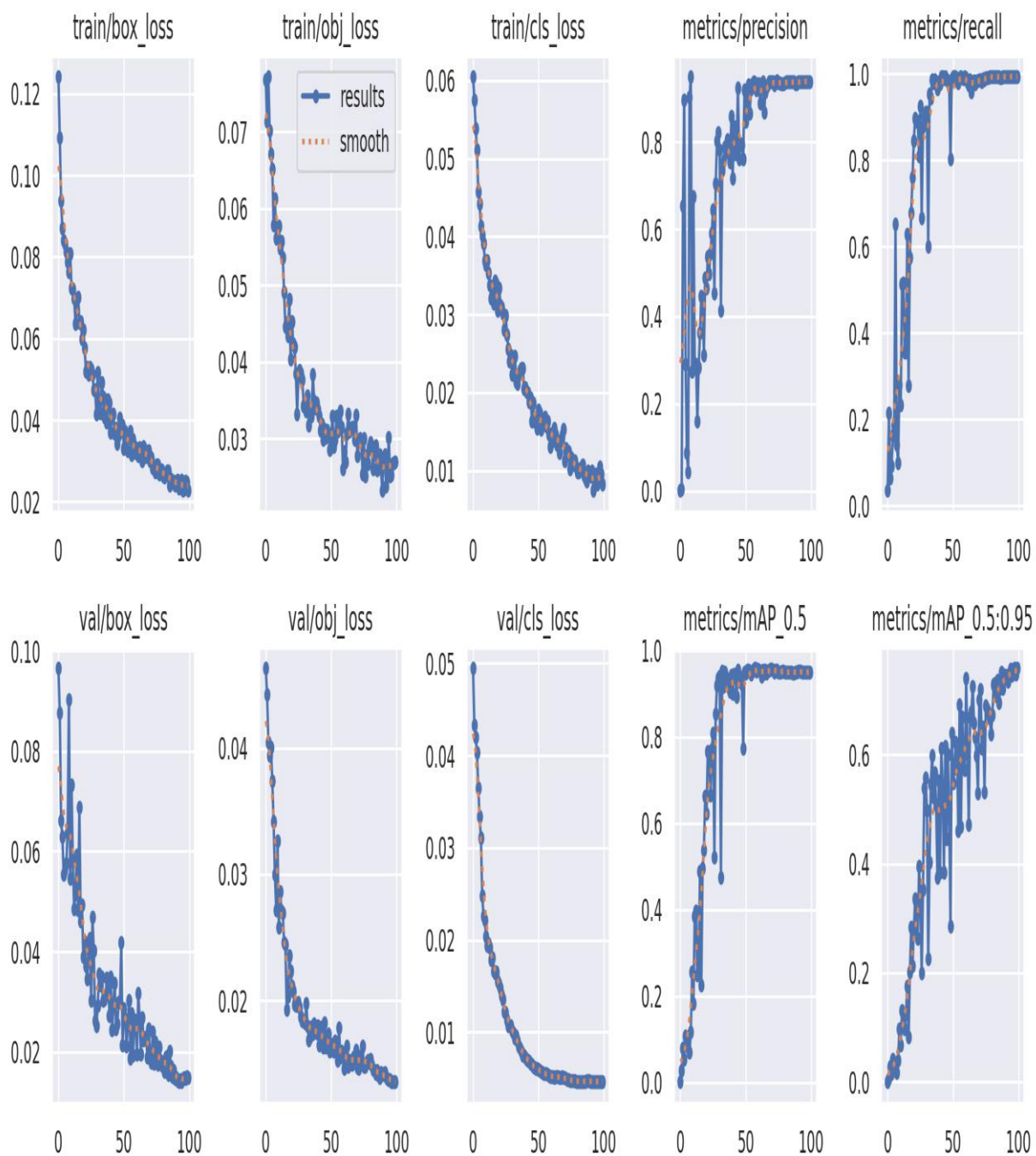
پیوست‌ها



۱- نمودار لیبل‌ها



۲- خروجی لیبل ها



۳ - خروجی مربوطه با Train



**University of Tehran
College of Farabi
Faculty of Engineering
Department of Computer Engineering**

**Design and implementation of the processing engine for
the content recognition system of national ID cards
using machine vision.**

**By:
Amirhossein Alipour**

**Under Supervision of:
Dr. Kazim Fouladi**

**A Project Report as a Requirement for
the Degree of Bachelor of Science in
Computer Engineering
August 2023**

