

دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق

درس مبانی سیستم‌های هوشمند

استاد: دکتر مهدی علیاری

پروژه پایانی بخش اول

این پروژه با تایید جناب آقای دکتر علیاری انجام شده است

محمدامین محمدیون شبستری	نام و نام خانوادگی
۴۰۱۲۲۵۰۳	شماره دانشجویی
محمد سبحان سخایی	نام و نام خانوادگی
۴۰۱۱۹۲۵۳	شماره دانشجویی
۱۴۰۴ بهمن	تاریخ
لینک‌های مربوط به نتایج و کد :	
گوگل درایو لینک گیت‌هاب	



فهرست مطالب

۶	۱ معرفی مسئله و داده‌ها
۶	۱.۱ معرفی مجموعه داده Database Defect Surface NEU
۶	۲.۱ ماهیت داده‌ها
۶	۳.۱ کلاس‌های خرابی سطحی
۷	۴.۱ ساختار و توزیع داده‌ها
۷	۵.۱ چالش‌های مجموعه داده
۷	۶.۱ اهداف پژوهش
۹	۲ پیش‌پردازش داده‌ها و تحلیل تأثیر آن بر آموزش مدل YOLO
۹	۱.۲ اهمیت پیش‌پردازش در مسائل بینایی ماشین صنعتی
۹	۲.۲ بهبود کنترast تصاویر با استفاده از CLAHE
۹	۳.۲ شبیه‌سازی تاری حرکتی (Motion Blur)
۱۰	۴.۲ افزودن نویز گاوسی کنترل شده
۱۰	۵.۲ تفاوت استراتژی پیش‌پردازش در داده‌های آموزش و اعتبارسنجی
۱۰	۶.۲ تأثیر پیش‌پردازش بر آموزش مدل YOLO
۱۴	۷.۲ تبدیل برچسب‌ها از فرمت VOC به فرمت YOLO
۱۴	۱.۷.۲ ساختار برچسب در قالب YOLO
۱۵	۲.۷.۲ فرآیند استخراج اطلاعات از فایل XML
۱۵	۳.۷.۲ نرم‌السازی مختصات جعبه مرزی
۱۵	۴.۷.۲ تولید فایل‌های برچسب YOLO
۱۵	۸.۲ ذخیره‌سازی دیتاست بعد از اعمال انجام شده
۱۷	۳ آموزش سه مدل YOLOV
۱۷	۱.۳ آموزش مدل YOLOV8-GSEBlock
۱۷	۲.۳ معماری کلی مدل YOLOV8
۱۸	۳.۳ انگیزه استفاده از مژاول GSE
۱۸	۴.۳ ساختار مژاول GSE بهبودیافته
۱۸	۱.۴.۳ فشرده‌سازی سراسری (Global Squeeze)
۱۹	۲.۴.۳ یادگیری وزن‌های کانالی
۱۹	۳.۴.۳ بازکالیرسیون ویژگی‌ها
۱۹	۵.۳ ادغام مژاول GSE با SPPF در YOLOV8
۱۹	۶.۳ استراتژی آموزش مدل
۲۰	۷.۳ کنترل بیش‌برازش و توقف زودهنگام
۲۵	۸.۳ پیاده‌سازی مدل YOLOV8n-Improvement
۲۵	۹.۳ معماری کلی شبکه



۲۶	ماژول استخراج ویژگی CSP-ABAN	۱۰.۳
۲۶	mekanizm Tوجه سراسری (GAM)	۱۱.۳
۲۶	ماژول کانولوشن GSConv	۱۲.۳
۲۷	بلوک توجه GSE بهبودیافته در SPPF	۱۳.۳
۲۷	تابع هزینه PIoUv2	۱۴.۳
۲۸	mekanizm هم‌افزایی ماژول‌ها	۱۵.۳



فهرست تصاویر

۱۱	تاثیر پیش‌پردازش CLAHE	۱
۱۱	تاثیر پیش‌پردازش Motion Blur	۲
۲۲	نمودارهای مربوط به روند آموزش مدل	۳
۲۳	. precision-Recall و Confidence	نمودارهای precision، Recall و همچنین نمودار	۴
۲۴	ماتریس درهم‌یختگی	۵
۲۵	نمونه‌هایی از تشخیص مدل	۶
۳۳	نمودارهای مربوط به روند آموزش مدل	۷



فهرست جداول



فهرست برنامه‌ها

۸	بالا آوردن دیتاست روی کولب	۱
۸	مشخص کردن کلاس‌ها و تقسیم دیتا به دو بخش آموزش و آزمون	۲
۱۱	پیش‌پردازش روی مجموعه داده‌های آموزشی و اعتبارسنجی	۳
۱۳	تبديل فرمت لیبل‌ها به فرمت مناسب برای آموزش YOLO	۴
۱۶	ذخیره‌سازی دیتاست و ایجاد فایل data.yaml	۵
۲۰	ساختار و استراتژی آموزش مدل YOLOV8n-GSE	۶
۲۸	ساختار و استراتژی آموزش مدل YOLOV8n-Improvement	۷



۱ معرفی مسئله و داده‌ها

در صنایع فولاد، بهویژه در فرآیند نورد گرم (Hot-Rolling)، کیفیت سطحی محصولات نقش کلیدی در عملکرد مکانیکی، قابلیت استفاده و ارزش اقتصادی ورق‌های فولادی دارد. بروز عیوب سطحی در حین تولید می‌تواند منجر به کاهش کیفیت، افزایش ضایعات، توقف خط تولید و تحمیل هزینه‌های قابل توجه به واحدهای صنعتی شود.

روش‌های سنتی بازرسی کیفیت در خطوط نورد عمدهاً به صورت پایش غیرفعال (Passive Monitoring) و ممکن بر اپراتور انسانی انجام می‌شوند. این روش‌ها علاوه بر وابستگی شدید به تجربه فردی، در سرعت‌های بالای تولید با کاهش دقت، خستگی اپراتور و عدم یکنواختی تصمیم‌گیری مواجه هستند. از سوی دیگر، بسیاری از سیستم‌های بینایی ماشین موجود صرفاً به تشخیص عیب بسنده کرده و قادر توانایی تصمیم‌گیری کنترلی در پاسخ به شرایط سطح محصول می‌باشند.

هدف این پژوهه، گذار از یک سیستم پایش غیرفعال به یک سیستم تصمیم‌گیر فعال (Active Decision-Making System) است؛ به گونه‌ای که سیستم علاوه بر شناسایی عیوب سطحی، با در نظر گرفتن شدت خرابی و میزان اطمینان تشخیص، به صورت هوشمند بر پارامترهای عملیاتی خط تولید، از جمله سرعت نوار نقاله، تأثیر بگذارد.

۱.۱ معرفی مجموعه داده Database Defect Surface NEU

در این پژوهه، به عنوان زیربنای اصلی آموزش و ارزیابی مدل بینایی ماشین، از مجموعه داده استاندارد Northeastern University Surface Defect Database (NEU-DET) استفاده شده است. این مجموعه داده توسط Yunhui Yan و Kechen Song ارائه شده و یکی از منابع

مرجع در حوزه تشخیص عیوب سطحی ورق‌های فولادی نورد گرم محسوب می‌شود.

این دیتابست شامل شش نوع عیب متداول سطحی در صنایع فولاد است که عبارت‌اند از: Patches (Pa), Rolled-in Scale (RS), Scratches (Sc) و Inclusion (In), Pitted Surface (PS), Crazing (Cr)

۲.۱ ماهیت داده‌ها

تصاویر موجود در مجموعه داده NEU-DET از نوع تصاویر خاکستری (Grayscale) بوده و از سطوح فولادی نورد گرم در شرایط صنعتی واقعی تهیه شده‌اند. این تصاویر شامل ویژگی‌های بصری متنوعی نظیر بازتاب نور از سطح فلز، تغییرات روشنایی و ناهمگونی بافت سطح هستند که شبیه‌سازی مناسبی از محیط واقعی کارخانه محسوب می‌شوند.

۳.۱ کلاس‌های خرابی سطحی

مجموعه داده NEU-DET شامل شش کلاس خرابی سطحی به شرح زیر است:

Crazing (ترک خودگی): ترک‌های ریز و نامنظم سطحی که می‌توانند موجب کاهش استحکام مکانیکی محصول شوند. •

Inclusion (ناخالصی): وجود ذرات غیر فلزی یا مواد خارجی در ساختار سطح که معمولاً ناشی از آلودگی فرآیند تولید است. •

Patches (لکه): نواحی غیر یکنواخت سطح که اغلب در اثر مشکلات حرارتی یا شیمیایی ایجاد می‌شوند. •

Surface Pitted (سطح حفره‌دار): وجود حفره‌ها یا فرورفتگی‌های موضعی که می‌توانند منشأ ایجاد ترک‌های عمیق‌تر باشند. •



Scale Rolled-in •
می‌دهند.

Scratches •
خطوط یا خراش‌های سطحی که معمولاً شدت کمتری نسبت به سایر عیوب دارند.

۴.۱ ساختار و توزیع داده‌ها

این مجموعه داده شامل مجموعاً ۱۸۰۰ تصویر است که برای هر یک از شش کلاس خرابی، ۳۰۰ تصویر در نظر گرفته شده است. توزیع داده‌ها در بین کلاس‌ها کاملاً متوازن بوده و این موضوع از بروز سوگیری در فرآیند آموزش مدل جلوگیری می‌کند. هر تصویر به همراه یک فایل برچسب‌گذاری با فرمت XML ارائه شده است که شامل مختصات دقیق ناحیه خرابی به صورت Bound-ing Box می‌باشد. این ساختار امکان آموزش مدل‌های تشخیص شیء (Object Detection) را با دقت بالا فراهم می‌سازد.

۵.۱ چالش‌های مجموعه داده

از مهم‌ترین چالش‌های موجود در مجموعه داده NEU-DET می‌توان به موارد زیر اشاره کرد:

- تغییرات شدید روشنایی و کنتراست
- بازتاب نور از سطح فلزی
- شاهت ظاهری برخی عیوب به بافت طبیعی سطح فولاد
- تنوع اندازه و شکل نواحی خرابی

این چالش‌ها سبب می‌شود که مجموعه داده NEU-DET معیار مناسبی برای ارزیابی توانایی تعمیم‌پذیری مدل‌های بینایی ماشین در شرایط صنعتی واقعی باشد.

۶.۱ اهداف پژوهه

هدف اصلی این پژوهه، طراحی و پیاده‌سازی یک سیستم هوشمند تصمیم‌گیر فعال برای خطوط تولید فولاد است که بتواند به صورت خودکار تعادل مناسبی میان کیفیت محصول و بهره‌وری تولید برقار کند. اهداف جزئی پژوهه به شرح زیر تعریف می‌شوند:

۱. تشخیص دقیق عیوب سطحی: شناسایی نوع و مکان عیوب سطحی با استفاده از سه مدل آموزش دیده که به تفکیک Yolov8n، Yolov11M و Yolov8n-Improvement، GSEBlock می‌باشند.
۲. مدلسازی عدم قطعیت: به کارگیری منطق فازی برای شبیه‌سازی قضاوت انسانی در ارزیابی شدت خرابی و تفکیک عیوب جزئی از خرابی‌های بحرانی.
۳. بهینه‌سازی نرخ تولید: استفاده از یادگیری تقویتی (Reinforcement Learning) به منظور ایجاد تعادل دینامیک میان سرعت خط تولید و دقت فرآیند بازرگانی.



در بخش زیر کد مربوط به بالا آوردن دیتا بر روی کولب آورده شده است. این دیتاست خود به دو بخش آموزش و اعتبارسنجی تقسیم شده است. بنابراین ما نیز مطابق با همان طبقه‌بندی مدل را آموزش دادیم و در حین آموزش اعتبارسنجی نمودیم. ۱۴۴۰ تصویر به همراه ناحیه مشخص شده به عنوان Bounding Box برای آموزش مدل استفاده شد. همچنین از ۳۶۰ داده اعتبارسنجی برای ارزیابی نحوه عملکرد مدل استفاده شد. دیتاست ۶ کلاس دارد که در داده‌های آموزش، هرکلاس دارای ۲۴۰ تصویر می‌باشد و در داده‌های اعتبارسنجی، هرکلاس دارای ۶۰ تصویر است.

```
1 import kagglehub
2
3 path = kagglehub.dataset_download(
4     "kaustubhdikshit/neu-surface-defect-database"
5 )
6
7 print("Path to dataset files:", path)
```

۱: بالا آوردن دیتاست روی کولب

```
1 import os
2 NEU_ROOT = path+"/NEU-DET"
3
4
5 YOLO_ROOT = "dataset_yolo"
6
7
8 CLASSES = [
9     "crazing",
10    "inclusion",
11    "patches",
12    "pitted_surface",
13    "rolled-in_scale",
14    "scratches"
15 ]
16
17 CLASS_MAP = {name: idx for idx, name in enumerate(CLASSES)}
18
19 for split in ["train", "val"]:
20     os.makedirs(f"{YOLO_ROOT}/images/{split}", exist_ok=True)
21     os.makedirs(f"{YOLO_ROOT}/labels/{split}", exist_ok=True)
```

۲: مشخص کردن کلاس‌ها و تقسیم دیتا به دو بخش آموزش و آزمون



۲ پیش‌پردازش داده‌ها و تحلیل تأثیر آن بر آموزش مدل YOLO

۱.۲ اهمیت پیش‌پردازش در مسائل بینایی ماشین صنعتی

در مسائل بینایی ماشین صنعتی، بهویژه در حوزه بازرسی سطوح فلزی، کیفیت داده‌های ورودی نقش تعیین‌کننده‌ای در عملکرد نهایی مدل‌های یادگیری عمیق ایفا می‌کند. تصاویر ثبت شده از خطوط نورد گرم فولاد معمولاً تحت تأثیر عواملی نظیر تغییرات شدید روشنایی، بازتاب نور از سطح فلز، لرزش دوربین، حرکت نوار نقاله و نویز سنسور قرار دارند. این عوامل سبب می‌شوند که توزیع داده‌های آموزشی با شرایط ایده‌آل آزمایشگاهی فاصله قابل توجهی داشته باشد.

هدف از پیش‌پردازش داده‌ها در این پژوهه، صرفاً بهبود ظاهری تصاویر نیست، بلکه افزایش قابلیت تعمیم (Generalization) مدل YOLO و نزدیکسازی داده‌های آموزشی به شرایط واقعی محیط کارخانه می‌باشد. بدین منظور، مجموعه‌ای از تکنیک‌های پیش‌پردازش و افزایش داده به صورت کنترل شده به کار گرفته شده است.

۲.۲ بهبود کنتراست تصاویر با استفاده از CLAHE

در نخستین مرحله، بر روی تمامی تصاویر آموزشی و اعتبارسنجی از الگوریتم Contrast Limited Adaptive Histogram Equalization (CLAHE) استفاده شده است. این روش برخلاف همسان‌سازی هیستوگرام سراسری، به صورت محلی عمل کرده و از تقویت بیش از حد نویز جلوگیری می‌کند.

در تصاویر سطوح فولادی، بسیاری از عیوب نظیر ترک‌خوردگی‌ها و خراش‌ها دارای اختلاف کنتراست اندکی نسبت به پس‌زمینه هستند. علاوه بر این، بازتاب نور از سطح فلز موجب ایجاد نواحی بسیار روشن یا تیره می‌شود که تشخیص عیوب را دشوار می‌سازد. استفاده از CLAHE باعث افزایش کنتراست موضعی تصویر شده و مرز میان ناحیه معیوب و بافت سالم را برجسته‌تر می‌کند.

به کارگیری این روش موجب می‌شود شبکه YOLO بتواند ویژگی‌های مبتنی بر لبه، بافت و تغییرات شدت روشنایی را با دقت بالاتری استخراج کند. در نتیجه، همگرایی سریع‌تر مدل در فاز آموزش و بهبود دقت مکان‌یابی (Localization Accuracy) عیوب حاصل می‌شود.

۳.۲ شبیه‌سازی تاری حرکتی (Motion Blur)

در مرحله بعد، به منظور شبیه‌سازی شرایط واقعی خط تولید، تاری حرکتی به صورت تصادفی بر روی حدود 30° درصد از تصاویر آموزشی اعمال شده است. در خطوط نورد واقعی، به دلیل حرکت پیوسته نوار فولادی و محدودیت سرعت شاتر دوربین، تصاویر اغلب دچار تاری جهت‌دار می‌شوند.

در صورت آموزش مدل تنها بر روی تصاویر کاملاً شفاف، شبکه عصبی به ویژگی‌های بسیار تیز و ایده‌آل وابسته شده و در مواجهه با تصاویر واقعی دچار افت عملکرد قابل توجه خواهد شد. افزودن تاری حرکتی باعث می‌شود مدل YOLO یاد بگیرد ویژگی‌های پایدارتری را استخراج کند که نسبت به تغییرات ناشی از حرکت حساسیت کمتری دارند.

این مرحله را می‌توان نوعی تطبیق دامنه (Domain Adaptation) دانست که موجب افزایش پایداری مدل در سرعت‌های مختلف خط تولید و شرایط تصویربرداری متفاوت می‌شود.



۴.۲ افزودن نویز گاوی کنترل شده

به منظور شبیه‌سازی نویزهای موجود در شرایط صنعتی واقعی، نویز گاوی با واریانس تصادفی به درصد محدودی (حدود ۱۰ درصد) از تصاویر آموزشی افزوده شده است. این نویز می‌تواند نماینده نویز سنسور، نویز الکترونیکی یا اختلالات محیطی نظری گرد و غبار و لرزش باشد.

افزودن نویز به صورت کنترل شده موجب جلوگیری از بیش‌برازش (Overfitting) شده و مدل را قادر می‌کند به جای واپس‌گردی به مقادیر دقیق پیکسلی، الگوهای ساختاری و معنادار تصویر را یاد بگیرد. این امر به طور مستقیم موجب افزایش قابلیت تعمیم مدل در داده‌های دیده‌نشده می‌شود.

۵.۲ تفاوت استراتژی پیش‌پردازش در داده‌های آموزش و اعتبارسنجی

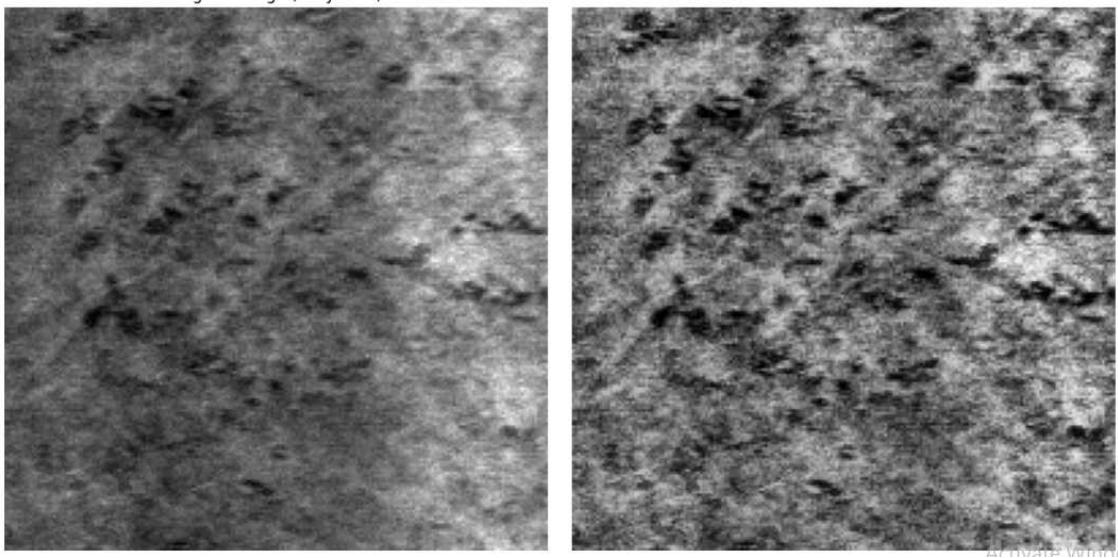
در این پژوهه، استراتژی متفاوتی برای داده‌های آموزشی و اعتبارسنجی اتخاذ شده است. در مجموعه آموزش، تمامی مراحل شامل بهبود کنتراس، تاری حرکتی و نویز اعمال شده‌اند، در حالی که در مجموعه اعتبارسنجی تنها از CLAHE استفاده شده است. دلیل این انتخاب آن است که داده‌های اعتبارسنجی باید نماینده شرایط واقعی اما پایدار باشند و اعمال نویز یا تاری مصنوعی می‌تواند منجر به برآورد نادرست عملکرد مدل شود. این رویکرد باعث می‌شود معیارهای ارزیابی مدل، بازتاب‌دهنده توان واقعی آن در محیط عملیاتی باشند.

۶.۲ تأثیر پیش‌پردازش بر آموزش مدل YOLO

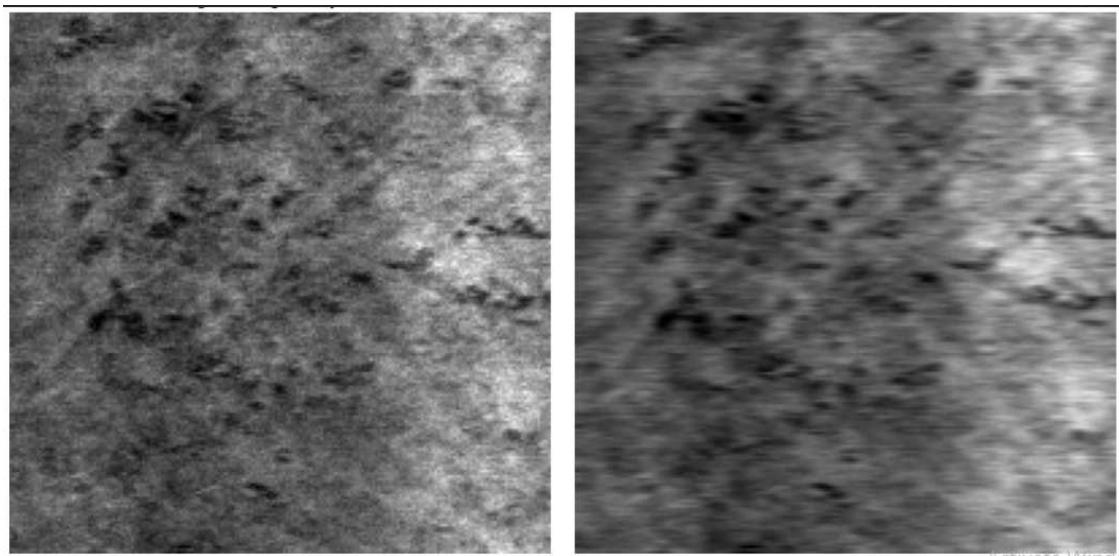
اعمال این مراحل پیش‌پردازش تأثیر مستقیمی بر فرآیند آموزش مدل YOLO دارد. افزایش کنتراس موضعی باعث بهبود تشخیص عیوب ظریف نظری خراش‌ها و ترک‌های سطحی شده و نرخ بازخوانی (Recall) را افزایش می‌دهد. شبیه‌سازی تاری حرکتی موجب افزایش پایداری مدل در شرایط دینامیک خط تولید می‌شود و افزودن نویز، از بیش‌برازش جلوگیری کرده و دقت مدل را در داده‌های واقعی افزایش می‌دهد.

در صورت حذف این مراحل، مدل YOLO اگرچه ممکن است بر روی داده‌های آموزشی به دقت بالایی دست یابد، اما در محیط واقعی کارخانه با افت شدید عملکرد مواجه خواهد شد. بهویژه، تشخیص عیوب کم کنتراس دشوار شده و حساسیت مدل به تغییرات نوری و نویز افزایش می‌یابد.

در زیر دو نمونه از تصاویر آورده شده است که بر روی اولی پیش‌پردازش CLAHE و بر روی دومی پیش‌پردازش Motion Blur انجام شده است. همانطور که مشخص است پیش‌پردازش CLAHE باعث می‌شود تا نواحی خراشی به شکل مشخص‌تری مشاهده شوند و ناحیه خراشی از ناحیه پس‌زمینه به صورت مناسبی تفکیک شود. این کار دقیقاً هم‌راستا با بالابردن دقت مدل است. در تصویر پایین‌تر نیز می‌بینیم که اضافه کردن تاری روی بعضی از تصاویر باعث می‌شود تا به شکل بهتری بتوانیم در واقعیت یک خط تولید را شبیه‌سازی نماییم و تعمیم‌پذیری مدلمان را بالا ببریم.



شکل ۱: تاثیر پیش‌پردازش CLAHE



شکل ۲: تاثیر پیش‌پردازش Motion Blur

تصاویر سمت راست نمایانگر تصاویری هستند که بر روی آنها پیش‌پردازش‌های مذکور اعمال شده است و تصاویر سمت چپ، تصاویر خام هستند.
کدهای زده شده در این بخش به صورت زیر می‌باشند:

```
1 import cv2
2 import numpy as np
3 import xml.etree.ElementTree as ET
4 from tqdm import tqdm
5 import glob
6 import random
```



```
v  import matplotlib.pyplot as plt
^  def apply_clahe():
^    clahe = cv2.createCLAHE(clipLimit=2.5, tileGridSize=(8, 8))
10   return clahe.apply(img)

11
12 def motion_blur(img, k=np.random.choice([3, 5, 7])):
13   kernel = np.zeros((k, k))
14   kernel[k // 2, :] = np.ones(k)
15   kernel /= k
16   return cv2.filter2D(img, -1, kernel)
17 def process_train():
18   img_root = f"{NEU_ROOT}/train/images"
19   ann_root = f"{NEU_ROOT}/train/annotations"
20
21   for cls in CLASSES:
22     img_dir = f"{img_root}/{cls}"
23     if not os.path.exists(img_dir): continue
24
25     for img_name in tqdm(os.listdir(img_dir), desc=f"Processing {cls}"):
26       img_path = os.path.join(img_dir, img_name)
27
28
29     base_name = os.path.splitext(img_name)[0]
30     xml_path = os.path.join(ann_root, f"{base_name}.xml")
31
32     if not os.path.exists(xml_path):
33       continue
34
35     img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
36     if img is None: continue
37     h, w = img.shape
38
39     img = apply_clahe(img)
40
41
42     if np.random.rand() < 0.3:
43       img = motion_blur(img, k=np.random.choice([3, 5, 7]))
44
45     if np.random.rand() < 0.1:
46       sigma = np.random.uniform(2, 10)
47       noise = np.random.normal(0, sigma, (h, w)).astype(np.int16)
48       img = np.clip(img.astype(np.int16) + noise, 0, 255).astype(np.uint8)
49
50     cv2.imwrite(f"{YOLO_ROOT}/images/train/{base_name}.jpg", img)
51
```



```

52     labels = voc_to_yolo(xml_path, w, h)
53     with open(f"{YOLO_ROOT}/labels/train/{base_name}.txt", "w") as f:
54         f.write("\n".join(labels))
55
56 def process_val():
57     img_root = f"{NEU_ROOT}/validation/images"
58     ann_root = f"{NEU_ROOT}/validation/annotations"
59
60     for cls in CLASSES:
61         img_dir = f"{img_root}/{cls}"
62         if not os.path.exists(img_dir): continue
63
64         for img_name in tqdm(os.listdir(img_dir), desc=f"Val {cls}"):
65             img_path = os.path.join(img_dir, img_name)
66
67             base_name = os.path.splitext(img_name)[0]
68             xml_path = os.path.join(ann_root, f"{base_name}.xml")
69
70             if not os.path.exists(xml_path):
71                 continue
72
73             img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
74             if img is None: continue
75             h, w = img.shape
76
77             img = apply_clahe(img)
78
79             cv2.imwrite(f"{YOLO_ROOT}/images/val/{base_name}.jpg", img)
80
81             labels = voc_to_yolo(xml_path, w, h)
82             with open(f"{YOLO_ROOT}/labels/val/{base_name}.txt", "w") as f:
83                 f.write("\n".join(labels))

```

۳: پیش‌پردازش روی مجموعه داده‌های آموزشی و اعتبارسنجی

کد زیر نیز برای آمده‌سازی Annotations زده شده است. می‌بایست لیل‌ها از فرمت XML خارج شوند و به فرمتی تبدیل شوند که بتوانند با آن آموزش بینند:

```

1 def voc_to_yolo(xml_path, img_w, img_h):
2     tree = ET.parse(xml_path)
3     root = tree.getroot()
4     labels = []
5
6     for obj in root.findall("object"):
7         cls_name = obj.find("name").text
8         cls_id = CLASS_MAP[cls_name]
9

```



```

10     b = obj.find("bndbox")
11     xmin, ymin = float(b.find("xmin").text), float(b.find("ymin").text)
12     xmax, ymax = float(b.find("xmax").text), float(b.find("ymax").text)
13
14     xc = ((xmin + xmax) / 2) / img_w
15     yc = ((ymin + ymax) / 2) / img_h
16     w = (xmax - xmin) / img_w
17     h = (ymax - ymin) / img_h
18
19     labels.append(f"{cls_id} {xc:.6f} {yc:.6f} {w:.6f} {h:.6f}")
20
21     return labels

```

۴: تبدیل فرمت لیبل‌ها به فرمت مناسب برای آموزش YOLO

۷.۲ تبدیل برچسب‌ها از فرمت VOC به فرمت YOLO

مجموعه داده NEU-DET به صورت پیش‌فرض از قالب برچسب‌گذاری Pascal VOC استفاده می‌کند که در آن، اطلاعات هر شیء شامل نام کلاس و مختصات ناحیه خرابی به صورت مختصات مطلق پیکسلی ($xmin, ymin, xmax, ymax$) در فایل‌های XML ذخیره می‌شود. در مقابل، مدل‌های مبتنی بر YOLO از قالب برچسب‌گذاری متفاوتی استفاده می‌کنند که در آن، موقعیت و ابعاد جعبه مرزی به صورت نرمال‌شده و نسبی نسبت به ابعاد تصویر بیان می‌شود. به منظور سازگارسازی دیتابیس با الزامات مدل YOLO، یکتابع تبدیل طراحی شده است که وظیفه آن استخراج اطلاعات برچسب‌ها از فایل XML و تبدیل آن‌ها به فرمت مورد نیاز YOLO می‌باشد.

۱.۷.۲ ساختار برچسب در قالب YOLO

در قالب YOLO، هر شیء موجود در تصویر با یک خط شامل پنج مقدار نمایش داده می‌شود:

$$(class_id, x_c, y_c, w, h)$$

که در آن:

• $class_id$: شناسه عددی کلاس خرابی

• x_c, y_c : مختصات مرکز جعبه مرزی، نرمال‌شده نسبت به عرض و ارتفاع تصویر

• w, h : عرض و ارتفاع جعبه مرزی، نرمال‌شده نسبت به ابعاد تصویر

تمامی این مقادیر به صورت عددی در بازه $[0, 1]$ تعریف می‌شوند.



۲.۷.۲ فرآیند استخراج اطلاعات از فایل XML

در مرحله نخست، فایل XML مربوط به هر تصویر با استفاده از ساختار درختی (XML Tree) خوانده می‌شود. سپس برای هر شیء (Object) موجود در تصویر، نام کلاس خرابی استخراج شده و با استفاده از یک نگاشت از پیش‌تعریف‌شده (CLASS_MAP) به شناسه عددی متضاد تبدیل می‌شود. این نگاشت تضمین می‌کند که ترتیب کلاس‌ها در تمامی تصاویر یکسان باقی بماند. پس از آن، مختصات گوشش‌های جعبه مرزی شامل x_{min} , y_{min} , x_{max} و y_{max} استخراج می‌شوند که بیانگر موقعیت مطلق ناحیه خرابی در تصویر هستند.

۳.۷.۲ نرمال‌سازی مختصات جعبه مرزی

برای تبدیل مختصات از قالب VOC به YOLO ابتدا مختصات مرکز جعبه مرزی محاسبه می‌شود:

$$x_c = \frac{x_{min} + x_{max}}{2 \times W}$$

$$y_c = \frac{y_{min} + y_{max}}{2 \times H}$$

که در آن W و H به ترتیب عرض و ارتفاع تصویر هستند.

سپس عرض و ارتفاع جعبه مرزی به صورت زیر محاسبه و نرمال‌سازی می‌شوند:

$$w = \frac{x_{max} - x_{min}}{W}$$

$$h = \frac{y_{max} - y_{min}}{H}$$

این نرمال‌سازی باعث می‌شود مدل YOLO مستقل از رزولوشن تصویر عمل کرده و قابلیت تعمیم به تصاویر با ابعاد متفاوت را داشته باشد.

۴.۷.۲ تولید فایل‌های برچسب YOLO

پس از محاسبه مقادیر نهایی، اطلاعات هر شیء در قالب یک رشته متّی با دقت عددی مناسب ذخیره می‌شود. هر تصویر دارای یک فایل متّی مستقل با پسوند .txt است که در آن، هر خط متضاد با یک ناحیه خرابی در تصویر می‌باشد. این ساختار ساده و استاندارد، امکان بارگذاری سریع داده‌ها و آموزش کارآمد مدل YOLO را فراهم می‌سازد.

۸.۲ ذخیره‌سازی دیتاست بعد از اعمال انجام شده

پس از محاسبه مقادیر نهایی، اطلاعات هر شیء در قالب یک رشته متّی با دقت عددی مناسب ذخیره می‌شود. هر تصویر دارای یک فایل متّی مستقل با پسوند .txt است که در آن، هر خط متضاد با یک ناحیه خرابی در تصویر می‌باشد. این ساختار ساده و استاندارد، امکان بارگذاری سریع داده‌ها و آموزش کارآمد مدل YOLO را فراهم می‌سازد.



در ادامه نیز یکسری کد زده شده است تا اعمالی که روی دیتا اعمال نمودیم را ذخیره نماییم. تمامی این دیتاهای که بر روی آنها این اعمال انجام شده است به همراه فایل گزارش ارسال می‌گردد.
در نهایت یک دیتاست به صورت فایلی به نام data.yaml ایجاد می‌گردد. در کد زیر این روند انجام شده است:

```
 1 import shutil
 2
 3 zip_file_name = "NEU_DET_YOLO_Ready"
 4
 5 shutil.make_archive(zip_file_name, 'zip', YOLO_ROOT)
 6
 7
 8 print(f" Saved {zip_file_name}.zip")
 9 print(f" {os.path.getsize(zip_file_name + '.zip')} / (1024*1024):.2f} MB")
10 import os
11 from google.colab import drive
12
13 drive.mount('/content/drive')
14
15
16 destination_path = '/content/drive/MyDrive/FINAL_Project/Dataset'
17 zip_file_name = "NEU_DET_YOLO_Ready.zip"
18
19 if not os.path.exists(destination_path):
20     os.makedirs(destination_path)
21 else:
22     print(f"Done")
23 source_file = f"{zip_file_name}"
24 final_destination = os.path.join(destination_path, zip_file_name)
25
26 !cp {source_file} "{final_destination}"
27
28 !cp -r {YOLO_ROOT} "{destination_path}"
29 !cp -r {NEU_ROOT} "{destination_path}"
30
31 if os.path.exists(final_destination):
32     print(f"Coppied into: {final_destination}")
33 else:
34     print(f"Error")
35 import os
36 import zipfile
37
38 drive_zip_path = '/content/drive/MyDrive/FINAL_Project/Dataset/NEU_DET_YOLO_Ready.zip'
39 local_zip_path = '/content/dataset_yolo.zip'
40 extract_to = YOLO_ROOT
41
```



```
۴۲     if os.path.exists(drive_zip_path):
۴۳         !cp "{drive_zip_path}" {local_zip_path}
۴۴
۴۵     else:
۴۶         print("Not found")
۴۷
۴۸     if not os.path.exists(extract_to):
۴۹         os.makedirs(extract_to)
۵۰
۵۱     !unzip -q {local_zip_path} -d {extract_to}
۵۲
۵۳     !ls {extract_to}
۵۴     with open("data.yaml", "w") as f:
۵۵         f.write("""
۵۶     path: dataset_yolo
۵۷     train: images/train
۵۸     val: images/val
۵۹
۶۰     nc: 6
۶۱     names:
۶۲         - crazing
۶۳         - inclusion
۶۴         - patches
۶۵         - pitted_surface
۶۶         - rolled-in_scale
۶۷         - scratches
۶۸     """)
```

Code ۵: ذخیره‌سازی دیتاست و ایجاد فایل data.yaml

۳ آموزش سه مدل YOLOV

۱.۳ آموزش مدل YOLOV8-GSEBlock

۲.۳ معماری کلی مدل YOLOV8

در این پژوهه، از مدل YOLOV8 به عنوان هسته اصلی تشخیص عیوب سطحی استفاده شده است. YOLOV8 یکی از مدل‌های تک مرحله‌ای (One-Stage Detector) است که به دلیل سرعت بالا و دقت مناسب، گزینه‌ای ایده‌آل برای کاربردهای صنعتی و بلاذرنگ محسوب می‌شود. ساختار کلی YOLOV8 شامل سه بخش اصلی است:

• Backbone: استخراج ویژگی‌های سطح پایین و میانی از تصویر

• Neck: تجمعی چندمقیاسی ویژگی‌ها (Feature Fusion)



• Head: پیش‌بینی کلاس و مختصات جعبه‌های مرزی

در معماری YOLOv8، مازول (Neck SPPF) در بخش Spatial Pyramid Pooling - Fast نقش مهمی در افزایش میدان دید مؤثر (Receptive Field) و استخراج اطلاعات چندمقیاسی ایفا می‌کند.

۳.۳ انگیزه استفاده از مازول GSE

عيوب سطحی فولاد معمولاً:

- کوچک و کم‌کنترast هستند
- در بافت‌های پیچیده فلزی پنهان می‌شوند
- تفاوت ظریفی با زمینه دارند

در چنین شرایطی، تمام کanal‌های ویژگی استخراج شده اهمیت یکسانی ندارند. برخی کanal‌ها حاوی اطلاعات حیاتی مرتبط با خرابی هستند، در حالی که سایر کanal‌ها عمدتاً نویز یا بافت زمینه را نمایش می‌دهند. به منظور تقویت کanal‌های مهم و تضعیف کanal‌های غیر مؤثر، از یک مازول توجه کanalی بهبودیافته تحت عنوان Global Squeeze Enhancement است.

۴.۳ ساختار مازول GSE بهبودیافته

مازول GSE پیشنهادی از ترکیب همزمان اطلاعات آماری میانگین و بیشینه کanal‌ها استفاده می‌کند. ساختار این مازول شامل مراحل زیر است:

۱.۴.۳ فشرده‌سازی سراسری (Global Squeeze)

ابتدا دو نوع pooling سراسری بر روی نقشه ویژگی ورودی $X \in \mathbb{R}^{C \times H \times W}$ اعمال می‌شود:

$$\text{GAP}(X) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W X_{c,i,j}$$

$$\text{GMP}(X) = \max_{i,j}(X_{c,i,j})$$

که به ترتیب نمایانگر میانگین و بیشینه اطلاعات هر کanal هستند. ترکیب این دو دیدگاه باعث می‌شود هم اطلاعات کلی و هم ویژگی‌های برجسته حفظ شوند.



۲.۴.۳ یادگیری وزن‌های کanalی

خروجی‌های pooling با یکدیگر جمع شده وارد یک شبکه کوچک کانولوشنی شامل دو لایه 1×1 می‌شوند:

$$S = \sigma(W_2 \cdot \delta(W_1 \cdot (GAP + GMP)))$$

که در آن:

- W_1 : وزن‌های کانولوشنی

- δ : تابع فعال‌ساز ReLU

- σ : تابع سیگموید

خروجی S یک بردار وزن کanalی در بازه $[0, 1]$ است.

۳.۴.۳ بازکالیبراسیون ویژگی‌ها

در نهایت، نقشه ویژگی ورودی در وزن‌های کanalی ضرب می‌شود:

$$X' = X \odot S$$

این عملیات موجب تقویت کanalهای مرتبط با عیوب سطحی و تضعیف کanalهای غیرضروری می‌شود.

۵.۳ ادغام ماذول SPPF با GSE در YOLOV8

در این پژوهه، ماذول GSE به صورت هوشمندانه پس از خروجی ماذول SPPF در بخش Neck YOLOV8 مدل افزوده شده است. دلیل این انتخاب آن است که خروجی SPPF شامل ویژگی‌های چندمقیاسی و غنی شده است و اعمال توجه کanalی در این مرحله بیشترین تأثیر را بر کیفیت ویژگی‌های منتقل شده به Head خواهد داشت.

به منظور حفظ سازگاری با معماری اصلی و جلوگیری از تغییرات گسترده، تابع forward ماذول SPPF به صورت پویا بازنویسی شده و ماذول GSE تنها یکبار در زمان اجرا ایجاد می‌شود. این رویکرد:

- سربار محاسباتی ناچیزی ایجاد می‌کند

- امکان استفاده از وزن‌های از پیش آموزش دیده YOLOV8 را حفظ می‌کند

- به سادگی قابل تعمیم به نسخه‌های دیگر YOLOv است

۶.۳ استراتژی آموزش مدل

آموزش مدل با استفاده از وزن‌های از پیش آموزش دیده YOLOV8n آغاز شده است که موجب تسريع همگرایی و کاهش نیاز به داده‌های بسیار زیاد می‌شود. تنظیمات آموزش به شرح زیر است:



(overfitting به منظور کنترل بهتر AdamW :Optimizer •

 10^{-3} اولیه: Learning Rate •

Cosine Annealing :Scheduler •

epoch ۵ :Warm-up •

۱۶ :Batch Size •

640 × 640 :Image Size •

۳۰۰ :Epochs •

برای جلوگیری از یادگیری الگوهای غیرواقعی، تکنیک‌های افزایش داده شدید نظیر، CutMix و MixUp Mosaic غیرفعال شده‌اند و تنها تبدیلات هندسی ملایم مانند چرخش، مقیاس‌دهی و وارونگی اعمال شده است. این انتخاب با هدف حفظ ماهیت صنعتی داده‌ها انجام شده است.

۷.۳ کنترل بیش‌برازش و توقف زودهنگام

به منظور جلوگیری از بیش‌برازش، از مکانیزم Early Stopping با مقدار patience برابر با ۵۰ استفاده شده است. بدین معنا که در صورت عدم بهبود معیارهای اعتبارسنجی، فرآیند آموزش به صورت خودکار متوقف می‌شود. کل زده شده در این بخش به صورت زیر است:

```

1 import torch
2 import torch.nn as nn
3
4 class GSEBlock(nn.Module):
5     def __init__(self, channels, reduction=16):
6         super().__init__()
7
8         self.f = None
9         self.i = None
10        self.type = "GSE"
11
12        self.gap = nn.AdaptiveAvgPool2d(1)
13        self.fc = nn.Sequential(
14            nn.Conv2d(channels, channels // reduction, 1, bias=False),
15            nn.ReLU(inplace=True),
16            nn.Conv2d(channels // reduction, channels, 1, bias=False),
17            nn.Sigmoid()
18        )
19
20    def forward(self, x):
21        return x * self.fc(self.gap(x))

```



```
۲۱
۲۲     class GSEBlock_Improved(nn.Module):
۲۳         def __init__(self, channels, reduction=16):
۲۴             super().__init__()
۲۵             self.gap = nn.AdaptiveAvgPool2d(1)
۲۶             self.gmp = nn.AdaptiveMaxPool2d(1)
۲۷
۲۸             self.fc = nn.Sequential(
۲۹                 nn.Conv2d(channels, channels // reduction, 1, bias=False),
۳۰                 nn.ReLU(inplace=True),
۳۱                 nn.Conv2d(channels // reduction, channels, 1, bias=False),
۳۲                 nn.Sigmoid()
۳۳             )
۳۴
۳۵         def forward(self, x):
۳۶             avg_out = self.gap(x)
۳۷             max_out = self.gmp(x)
۳۸             scale = self.fc(avg_out + max_out)
۳۹             return x * scale
۴۰     from ultralytics.nn.modules import SPPF
۴۱
۴۲     if not hasattr(SPPF, "_gse_patched"):
۴۳
۴۴         SPPF._orig_forward = SPPF.forward
۴۵
۴۶         def sppf_forward_with_gse(self, x):
۴۷             y = SPPF._orig_forward(self, x)
۴۸
۴۹             if not hasattr(self, "gse"):
۵۰                 self.gse = GSEBlock_Improved(y.shape[1]).to(y.device)
۵۱
۵۲             return self.gse(y)
۵۳
۵۴         SPPF.forward = sppf_forward_with_gse
۵۵         SPPF._gse_patched = True
۵۶     from ultralytics import YOLO
۵۷     model = YOLO("yolov8n.pt")
۵۸     model.train(
۵۹         data="data.yaml",
۶۰         epochs=300,
۶۱         imgsz=640,
۶۲         batch=16,
۶۳         optimizer="AdamW",
۶۴         lr0=1e-3,
۶۵         lrf=0.01,
۶۶         cos_lr=True,
```

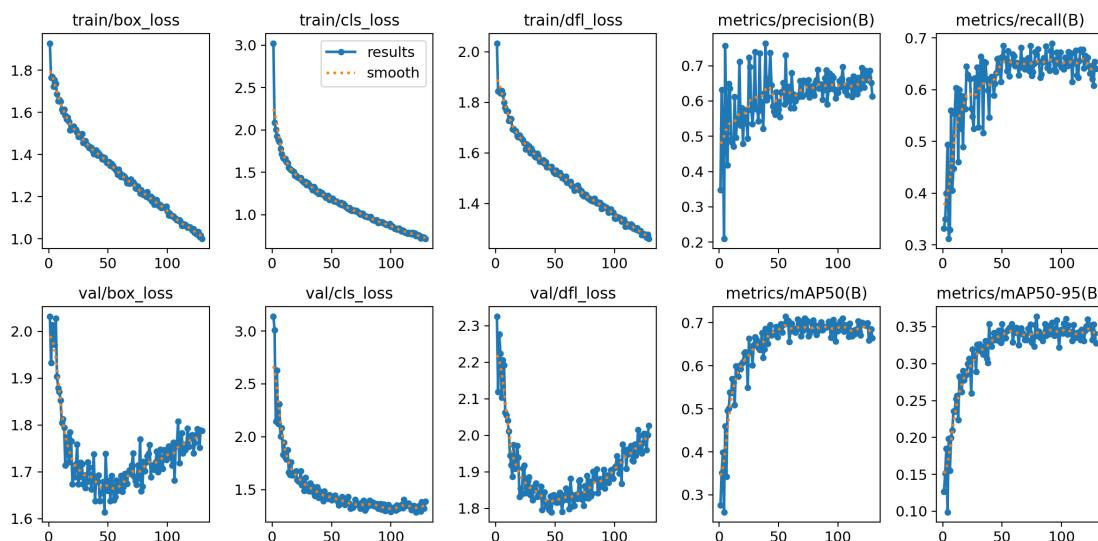
```

    ۶۷     warmup_epochs=5,
    ۶۸
    ۶۹     mosaic=0.0,
    ۷۰     mixup=0.0,
    ۷۱     cutmix=0.0,
    ۷۲     flipplr=0.5,
    ۷۳     flipud=0.5,
    ۷۴     scale=0.3,
    ۷۵
    ۷۶     hsv_h=0.0, hsv_s=0.0, hsv_v=0.0,
    ۷۷     patience=50,
    ۷۸     device=0,
    ۷۹
    ۸۰     project="/content/drive/MyDrive/FINAL_Project/Training_Results",
    ۸۱     name="YOLO_GSE_v1"
    ۸۲ )

```

۶: ساختار و استراتژی آموزش مدل YOLOV8n-GSE

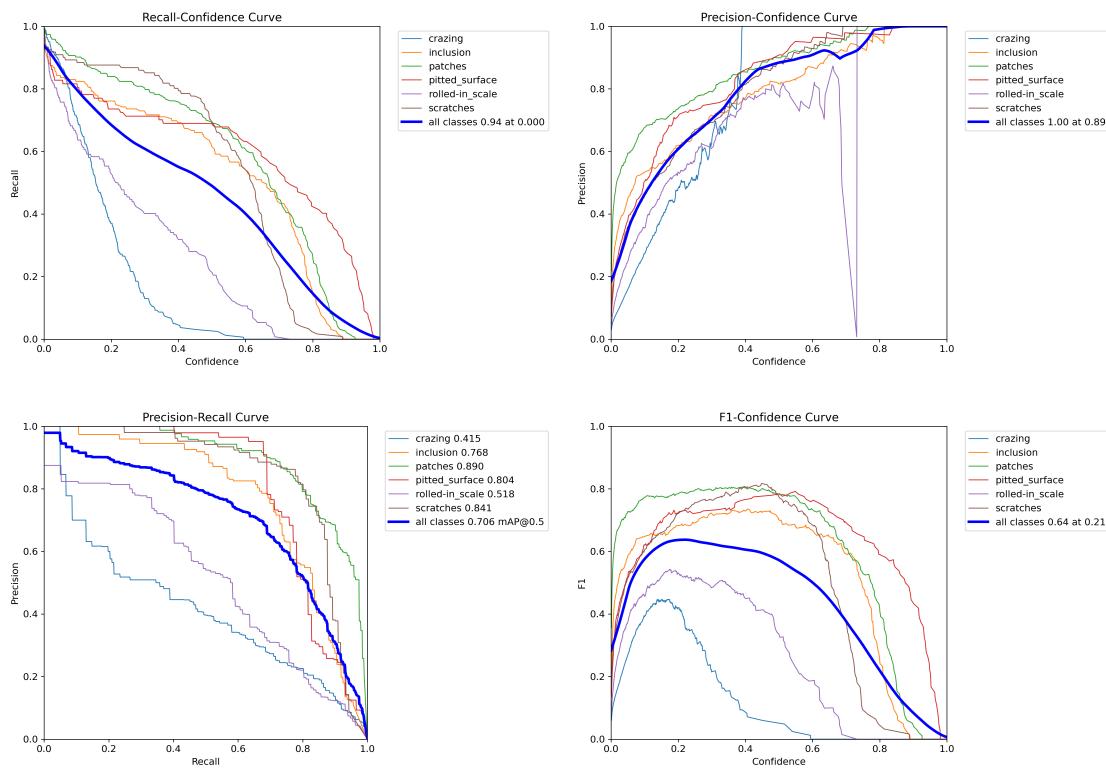
نتایج آموزش این مدل به صورت زیر می‌باشد:



شکل ۳: نمودارهای مربوط به روند آموزش مدل

همانطور که در شکل سه مشخص است، مدل در بخش طبقه‌بندی به خوبی آموزش دیده و همواره مقدار ارور آن بر روی داده‌های اعتبارسنجی رو به پایین بوده است اما در کشیدن Bounding Box دچار اورفیت شده است و از اپوک ۵۰ به بعد مقدار ارور آن رو به افزایش رفته است.

در تمامی متريک‌های ديگر نيز همانطور که از نمودار شکل سه مشخص است همواره در حال افزایش بوديم هرچند که به مقادير آنچنان بالاي دست نياfتم. نمودارها نشان مي دهند که روند آموزش مدل به شيوه درستی صورت پذيرفته است.



شکل ۴: نمودارهای precision-Recall و F1score بر حسب مقدار Confidence و همچنین نمودار precision-Confidence بر حسب مقدار Recall

نمودار Precision بر حسب Confidence این نمودار تغییرات معیار Precision را نسبت به آستانه اطمینان (Confidence) نشان می‌دهد. بیانگر نسبت پیش‌بینی‌های صحیح مثبت به کل پیش‌بینی‌های مثبت مدل است. با افزایش مقدار Confidence، مدل تنها نمونه‌هایی را به عنوان مثبت در نظر می‌گیرد که از اطمینان بالاتری برخوردارند؛ در نتیجه معمولاً مقدار Precision افزایش می‌یابد. این نمودار نشان می‌دهد که مدل تا چه حد در تشخیص نمونه‌های مثبت، از خطای پیش‌بینی مثبت کاذب جلوگیری می‌کند.

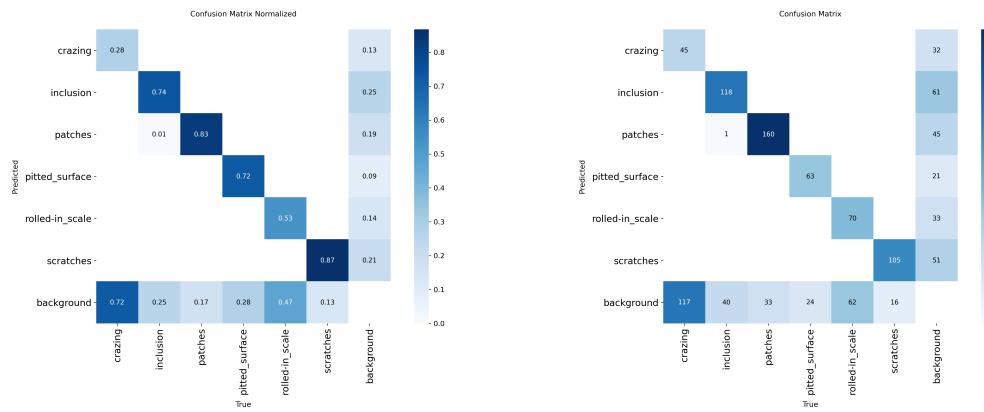
نمودار Recall بر حسب Confidence این نمودار تغییرات معیار Recall را بر حسب مقدار Confidence نمایش می‌دهد. نشان‌دهنده نسبت نمونه‌های مثبت شناسایی شده توسط مدل به کل نمونه‌های مثبت واقعی است. با افزایش آستانه Confidence، مدل سخت‌گیرانه‌تر عمل کرده و ممکن است برخی نمونه‌های مثبت واقعی را شناسایی نکند؛ در نتیجه مقدار Recall معمولاً کاهش می‌یابد. این نمودار توانایی مدل در پوشش دادن تمام نمونه‌های مثبت را نشان می‌دهد.

نمودار F1-score بر حسب Confidence نمودار F1-score میانگین هارمونیک معیارهای Precision و Recall را نسبت به مقدار Confidence نشان می‌دهد. این معیار تعادلی بین دقت و فراخوانی ایجاد می‌کند و زمانی مقدار بالاتری دارد که هر دو معیار Precision و Recall مناسب باشند. این نمودار برای انتخاب آستانه Confidence بهینه، که در آن عملکرد کلی مدل متوازن است، بسیار کاربردی می‌باشد.

نمودار Precision-Recall نمودار Precision-Recall رابطه بین دو معیار Precision و Recall را برای آستانه‌های مختلف Confidence نمایش می‌دهد. این نمودار نشان می‌دهد که با افزایش Recall (شناسایی نمونه‌های مثبت بیشتر)، معمولاً مقدار Precision کاهش می‌یابد.

نمودار Precision-Recall بهویژه در مجموعه داده‌های نامتوابن اهمیت زیادی دارد و عملکرد مدل را در تشخیص کلاس مثبت به صورت جامع ارزیابی می‌کند. هرچه منحنی به ناحیه بالا-راست نزدیک‌تر باشد، عملکرد مدل مطلوب‌تر است. از این نمودارها نتیجه می‌گیریم که روی دو کلاس Crazing و rolled in scale نتایج چندان خوب نیست و پایین است. روی کلاس‌های Pitted surface و Patches نتایج بالاتر و بهتر است.

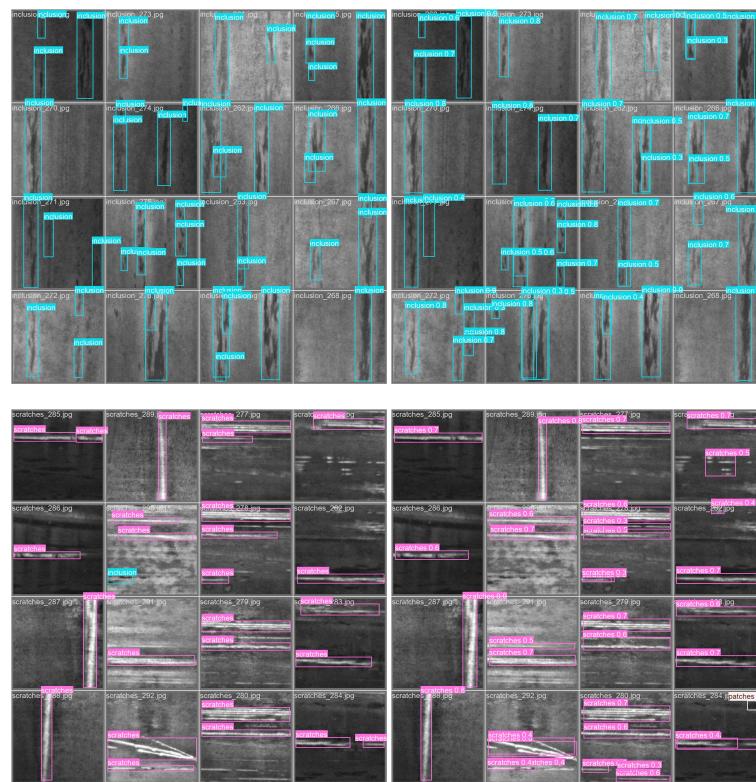
در بخش طبقه‌بندی نیز دو ماتریس درهم‌یختگی ترسیم شده است که در شکل‌های زیر قابل مشاهده است: همانطور که مشخص



شکل ۵: ماتریس درهم‌یختگی

است مدل در تشخیص قسمت پس‌زمینه خیلی خطای کند و جایایی رو به عنوان خرابی اعلام می‌کند که واقعاً خرابی در آنجا نیست. نکته قابل تأمل این است که مدل در تشخیص بین دو کلاس خطایی نداشته و فقط یک مورد خطای داشته است. این یعنی ما کلاسی نداشتمیم که به اشتباه چیز دیگری تشخیص دهیم و هرجا هم اشتباهی رخ داده فقط این بوده که اونجا خرابی نبوده است و مدل آن را خراب تشخیص داده است. بنظر می‌آید که اگر پیش‌پردازش را بهتر کنیم و بتوانیم بهتر ناحیه پس‌زمینه را از کلاس‌ها جدا کنیم، نتایج بهتری به دست می‌آوریم. همچنین می‌توانیم ساختار مدل را بهبود بدیم و یا از مدل‌های YOLO با ورژن بالاتر استفاده نماییم که در ادامه این امر تحقیق پیدا کرده است.

در شکل زیر نمونه از Validation Batch‌های آورده شده است که تشخیص مدل را نشان می‌دهد: همانطور که مشاهده می‌شود از نمونه‌ها هم معلوم است که مدل در تشخیص پس‌زمینه عملکرد مناسبی ندارد و بیشتر در آنجا دچار خطای شود.



شکل ۶: نمونه‌هایی از تشخیص مدل

۸.۳ پیاده‌سازی مدل YOLOv8n-Improvement

در این پژوهه، نسخه‌ای بهبودیافته و سبک از مدل YOLOv8n با عنوان YOLOv8n-improvement ارائه شده است. این معماری با اعمال تغییرات ساختاری و بهینه‌سازی در سطوح مختلف شبکه، شامل مازول استخراج ویژگی CSP-ABAN، مکانیزم توجه سراسری GAM، مازول کانولوشن GSE، تابع هزینه PIoUv2 GSConv و همچنین بلوک توجه در مازول SPPF طراحی شده است. ترکیب این اجزا یک ساختار هم‌افزا (Synergistic) از مرحله استخراج ویژگی تا بهینه‌سازی نهایی ایجاد می‌کند که منجر به افزایش دقیق و پایداری مدل در عین حفظ سبکی آن می‌شود.

۹.۳ معماری کلی شبکه

در مدل YOLOv8n-Improvement تغییرات زیر نسبت به YOLOv8n پایه اعمال شده است:

- جایگزینی کانولوشن‌های استاندارد با GSConv در بخش‌های کلیدی Neck و Backbone
- بازطراحی مازول‌های C2f به صورت CSP-ABAN
- ادغام مکانیزم توجه سراسری GAM
- جایگزینی تابع هزینه CIOU با PIoUv2
- افزودن بلوک توجه GSE بهبودیافته در SPPF



این تغییرات به صورت یک ساختار انتها به انتهای (End-to-End) عمل کرده و جریان ویژگی و گرادیان را به شکل هماهنگ بینه می‌کند.

۱۰.۳ مازول استخراج ویژگی CSP-ABAN

مازول C2f در YOLOv8 اگرچه قدرت استخراج ویژگی بالایی دارد، اما باعث افزایش پیچیدگی محاسباتی می‌شود. به همین منظور، مازول CSP-ABAN طراحی شده است که ترکیبی از ساختار CSPNet و کانولوشن دو مسیره (DualConv) می‌باشد. هر بلوک DualConv شامل:

- کانولوشن گروهی 3×3 با ۴ گروه

- کانولوشن 1×1 برای فشرده‌سازی کanalی

کانولوشن گروهی موجب استخراج کارآمد اطلاعات فضایی شده و کانولوشن 1×1 تبدیل اطلاعات بین کانال‌ها را تسهیل می‌کند. همچنین ساختار CSP از محاسبات تکراری جلوگیری کرده و کارایی شبکه را افزایش می‌دهد. این طراحی تعادل مناسبی بین توان بازنمایی ویژگی و هزینه محاسباتی برقرار می‌کند.

۱۱.۳ مکانیزم توجه سراسری (GAM)

برای افزایش توان تمایز ویژگی‌ها، مکانیزم توجه سراسری GAM به مدل افزوده شده است. برای نگاشت ورودی F_1 ، ابتدا توجه کanalی اعمال می‌شود:

$$F_2 = F_1 \cdot \sigma(\text{ChannelAttention}(F_1)) \quad (1)$$

سپس توجه فضایی اعمال می‌شود:

$$F_3 = F_2 \cdot \sigma(\text{SpatialAttention}(F_2)) \quad (2)$$

که در آن σ تابع سیگموید است.

زیرمازول توجه کanalی از یک ساختار MLP دو لایه با نسبت فشرده‌سازی r استفاده می‌کند. زیرمازول توجه فضایی نیز شامل دو لایه کانولوشن بدون استفاده از Max-Pooling است تا از اتلاف اطلاعات جلوگیری شود. این مکانیزم موجب سرکوب نواحی غیرمهم و تقویت نواحی حاوی ویژگی‌های هدف می‌شود.

۱۲.۳ مازول کانولوشن GSConv

به منظور حفظ سبکی مدل همراه با افزایش تبادل اطلاعات کanalی، کانولوشن‌های استاندارد با GSConv جایگزین شده‌اند. ساختار GSConv شامل:



- کانولوشن استاندارد
- کانولوشن عمقی تفکیک‌پذیر (Depthwise Separable)
- الحاق کanalی
- عملیات Shuffle Channel

ترکیب کانولوشن استاندارد و عمقی موجب افزایش توان استخراج ویژگی شده و عملیات Shuffle تبدل اطلاعات بین کanalها را تقویت می‌کند. این مازول نسبت به DWConv کارایی بازنمایی بالاتری ارائه می‌دهد.

۱۳.۳ بلوک توجه GSE بهبودیافته در SPPF

برای تقویت بازنظمی کanalی ویژگی‌ها، یک بلوک GSE بهبودیافته در خروجی SPPF افزوده شده است. مقیاس توجه به صورت زیر محاسبه می‌شود:

$$S = \sigma(\text{MLP}(\text{GAP}(X) + \text{GMP}(X))) \quad (3)$$

$$Y = X \cdot S \quad (4)$$

در اینجا GAP و GMP به ترتیب بیانگر میانگین‌گیری و بیشینه‌گیری سراسری هستند. استفاده همزمان از این دو نوع Pooling موجب حفظ اطلاعات بحرانی و تقویت کanalهای مهم می‌شود.

۱۴.۳ تابع هزینه PIoUv2

در YOLOv8 پایه از CIOU استفاده می‌شود. در این پژوهه، تابع PIoUv2 جایگزین شده است. ابتدا عامل جریمه تطبیقی تعریف می‌شود:

$$P = \frac{1}{4} \left(\frac{dw_1}{w_{gt}} + \frac{dw_2}{w_{gt}} + \frac{dh_1}{h_{gt}} + \frac{dh_2}{h_{gt}} \right) \quad (5)$$

$$f(P) = 1 - e^{-P^2} \quad (6)$$

$$L_{PIoU} = 1 - IoU + f(P) \quad (7)$$

در نسخه PIoUv2 یک تابع توجه غیر یکنواخت افزوده شده است:



$$q = e^{-P} \quad (8)$$

$$u(x) = 3xe^{-x^2} \quad (9)$$

$$L_{PIoUv2} = u(\lambda q) \cdot L_{PIoU} \quad (10)$$

این تابع تمرکز بیشتری بر جعبه‌های با کیفیت متوسط ایجاد کرده و همگرا بی سریع‌تری نسبت به CIoU فراهم می‌کند.

۱۵.۳ مکانیزم هم‌افزایی مازول‌ها

در معماری YOLOv8n-Improvement، مازول‌ها به صورت زنجیره‌ای و هم‌افزا عمل می‌کنند:

• استخراج سلسله‌مراتبی و حذف افزونگی را انجام می‌دهد.

• GAM وزن‌دهی پویا در ابعاد فضایی و کانالی را اعمال می‌کند.

• PIoUv2 نظارت دقیق بر رگرسیون جعبه‌ها اعمال می‌کند.

این طراحی هماهنگ موجب بهبد دقت، افزایش پایداری آموزش و ارتقای قابلیت تعمیم‌پذیری مدل در سناریوهای پیچیده می‌شود، در حالی که سبکی مدل حفظ می‌گردد.

در بخش زیر کد مربوط به این مدل نیز زده شده است:

```
 1 import torch
 2 import torch.nn as nn
 3 import torch.nn.functional as F
 4 from ultralytics.nn.tasks import DetectionModel
 5 from ultralytics.nn.modules import Conv
 6 from ultralytics.utils.loss import BboxLoss
 7 from ultralytics import YOLO
 8
 9 class GSConv(nn.Module):
10     def __init__(self, c1, c2, k=3, s=1):
11         super().__init__()
12         self.cv1 = nn.Conv2d(c1, c2 // 2, k, s, k//2, bias=False)
13         self.bn1 = nn.BatchNorm2d(c2 // 2)
14
15         self.dw = nn.Conv2d(c2//2, c2//2, 3, 1, 1,
16                            groups=c2//2, bias=False)
17         self.pw = nn.Conv2d(c2//2, c2//2, 1, 1, 0, bias=False)
18         self.bn2 = nn.BatchNorm2d(c2//2)
```



```
۱۹
۲۰     def shuffle(self, x):
۲۱         b, c, h, w = x.size()
۲۲         x = x.view(b, 2, c//2, h, w)
۲۳         x = x.transpose(1,2).contiguous()
۲۴         return x.view(b,c,h,w)
۲۵
۲۶     def forward(self,x):
۲۷         x1 = F.relu(self.bn1(self.cv1(x)))
۲۸         x2 = F.relu(self.bn2(self.pw(self.dw(x1))))
۲۹         return self.shuffle(torch.cat([x1,x2],1))
۳۰
۳۱
۳۲     class DualConv(nn.Module):
۳۳         def __init__(self, c):
۳۴             super().__init__()
۳۵             self.conv3 = nn.Conv2d(c,c,3,1,1,groups=4,bias=False)
۳۶             self.conv1 = nn.Conv2d(c,c,1,1,0,bias=False)
۳۷             self.bn = nn.BatchNorm2d(c)
۳۸
۳۹         def forward(self,x):
۴۰             return F.relu(self.bn(self.conv1(self.conv3(x))))
۴۱
۴۲
۴۳     class CSP_ABAN(nn.Module):
۴۴         def __init__(self,c,n=2):
۴۵             super().__init__()
۴۶             self.c_ = c//2
۴۷             self.blocks = nn.Sequential(*[DualConv(self.c_) for _ in range(n)])
۴۸             self.cv = nn.Conv2d(c,c,1,1,0,bias=False)
۴۹             self.bn = nn.BatchNorm2d(c)
۵۰
۵۱         def forward(self,x):
۵۲             x1,x2 = torch.split(x,[self.c_,self.c_],1)
۵۳             y = self.blocks(x2)
۵۴             return F.relu(self.bn(self.cv(torch.cat([x1,y],1))))
۵۵
۵۶
۵۷     class GAM(nn.Module):
۵۸         def __init__(self,c,r=16):
۵۹             super().__init__()
۶۰             self.channel = nn.Sequential(
۶۱                 nn.Conv2d(c,c//r,1),
۶۲                 nn.ReLU(),
۶۳                 nn.Conv2d(c//r,c,1)
```



```
94         )
95         self.spatial = nn.Sequential(
96             nn.Conv2d(c,c//r,3,padding=1),
97             nn.ReLU(),
98             nn.Conv2d(c//r,c,3,padding=1)
99         )
100        self.sigmoid = nn.Sigmoid()
101
102    def forward(self,x):
103        x = x*self.sigmoid(self.channel(x))
104        x = x*self.sigmoid(self.spatial(x))
105        return x
106
107
108    from ultralytics.utils.loss import BboxLoss
109    import torch
110
111
112    def piouv2_iou(pred, target, lambda_=1.7):
113
114        x1 = torch.max(pred[:,0], target[:,0])
115        y1 = torch.max(pred[:,1], target[:,1])
116        x2 = torch.min(pred[:,2], target[:,2])
117        y2 = torch.min(pred[:,3], target[:,3])
118
119        inter = (x2-x1).clamp(0)*(y2-y1).clamp(0)
120
121        area1 = (pred[:,2]-pred[:,0])*(pred[:,3]-pred[:,1])
122        area2 = (target[:,2]-target[:,0])*(target[:,3]-target[:,1])
123
124        union = area1+area2-inter
125        iou = inter/(union+1e-6)
126
127
128        dw1 = torch.abs(pred[:,0]-target[:,0])
129        dw2 = torch.abs(pred[:,2]-target[:,2])
130        dh1 = torch.abs(pred[:,1]-target[:,1])
131        dh2 = torch.abs(pred[:,3]-target[:,3])
132
133
134        wgt = target[:,2]-target[:,0]
135        hgt = target[:,3]-target[:,1]
136
137
138        P = (dw1/wgt+dw2/wgt+dh1/hgt+dh2/hgt)/4
139        q = torch.exp(-P)
140
141
142        L_piou = 1-iou+(1-torch.exp(-P**2))
143        attention = 3*(lambda_*q)*torch.exp(-(lambda_*q)**2)
144
```



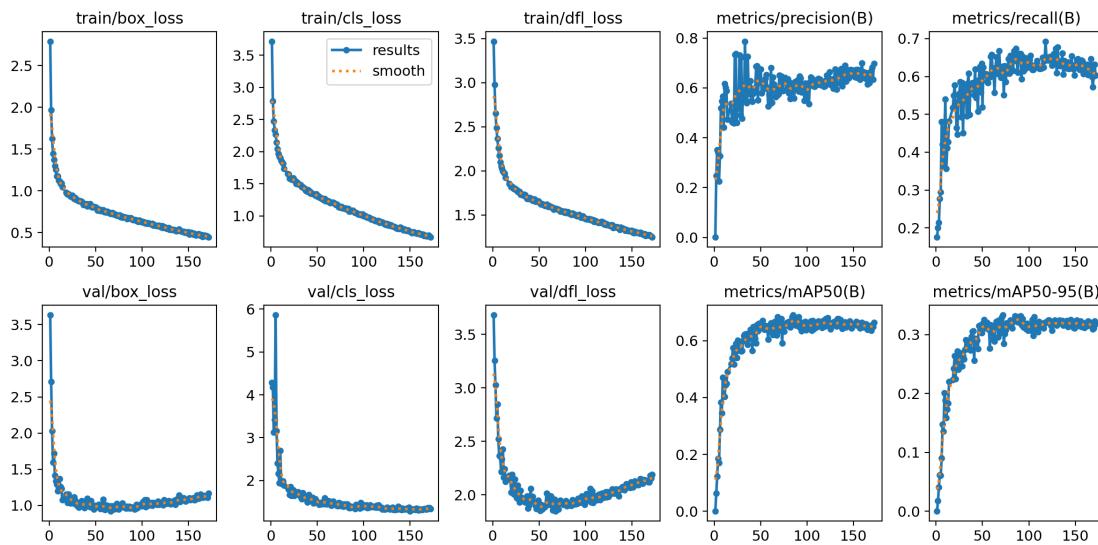
```
109     return attention*L_piou
110
111
112 from ultralytics.utils.loss import BboxLoss
113 import torch
114
115 _orig_forward = BboxLoss.forward
116
117 def new_forward(self,
118             pred_dist,
119             pred_bboxes,
120             anchor_points,
121             target_bboxes,
122             target_scores,
123             target_scores_sum,
124             fg_mask,
125             *args,
126             **kwargs):
127
128     loss_iou, loss_dfl = _orig_forward(
129         self,
130         pred_dist,
131         pred_bboxes,
132         anchor_points,
133         target_bboxes,
134         target_scores,
135         target_scores_sum,
136         fg_mask,
137         *args,
138         **kwargs
139     )
140
141     if fg_mask.sum() > 0:
142         pred_pos = pred_bboxes[fg_mask]
143         target_pos = target_bboxes[fg_mask]
144
145         piouv2_iou = piouv2_iou(pred_pos, target_pos).mean()
146         loss_iou = piouv2_iou
147
148     return loss_iou, loss_dfl
149
150
151 BboxLoss.forward = new_forward
152
153 print (" PIoUv2 Successfully Patched for YOLOv8.4.14")
```



```
104 from ultralytics.nn.modules.head import Detect
105 model = YOLO("yolov8n.yaml")
106
107 for i,m in enumerate(model.model.model):
108     if m.__class__.__name__ == "Conv":
109         c1 = m.conv.in_channels
110         c2 = m.conv.out_channels
111         s   = m.conv.stride[0]
112         k   = m.conv.kernel_size[0]
113         model.model.model[i] = GSConv(c1, c2, k=k, s=s)
114
115
116
117     if m.__class__.__name__ == "C2f":
118         c = m.cv2.conv.out_channels
119         model.model.model[i] = CSP_ABAN(c)
120
121
122
123     print("YOLOv8n-GSE Loaded Successfully ")
124
125
126 model.train(
127     data="data.yaml",
128     epochs=300,
129     imgsz=640,
130     batch=16,
131     optimizer="AdamW",
132     lr0=1e-3,
133     lrf=0.01,
134     cos_lr=True,
135     warmup_epochs=5,
136
137     mosaic=0.0,
138     mixup=0.0,
139     cutmix=0.0,
140     fliplr=0.5,
141     flipud=0.5,
142     scale=0.3,
143
144     hsv_h=0.0, hsv_s=0.0, hsv_v=0.0,
145     patience=100,
146     device=0,
147
148     project="/content/drive/MyDrive/FINAL_Project/Training_Results",
149     name="YOLO_final"
150 )
151 )
```

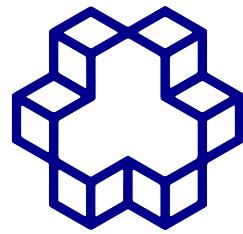
۷: ساختار و استراتژی آموزش مدل YOLOV8n-Improvement Code

این ساختار و این نحوه آموزش مدل از یک مقاله IEEE استخراج شده است که در سال ۲۰۲۵ چاپ گردیده است.
نتایج مربوط به آموزش این مدل به صورت زیر است:



شکل ۷: نمودارهای مربوط به روند آموزش مدل

همانطور که از نمودارها نیز مشخص است، مقدار ارور بر روی مجموعه داده اعتبارسنجی کاهش یافته است و با اورفیتی که در مدل قبلی روبه روبودیم، اینجا روبه رو نیستیم. مقدار Recall و Precision تا حدودی بهبود یافته و یا تغییری نکرده است.



دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق

درس مبانی سیستم‌های هوشمند

استاد: دکتر مهدی علیاری

پروژه پایانی بخش دوم

این پروژه با تایید جناب آقای دکتر علیاری انجام شده است

محمدامین محمدیون شبستری	نام و نام خانوادگی
۴۰۱۲۲۵۰۳	شماره دانشجویی
محمد سبحان سخایی	نام و نام خانوادگی
۴۰۱۱۹۲۵۳	شماره دانشجویی
۱۴۰۴ بهمن	تاریخ
لینک‌های مربوط به نتایج و کد :	
گوگل درایو لینک گیت‌هاب	



فهرست مطالب

۵	معیارهای ارزیابی عملکرد مدل	۱
۵	Recall و Precision	۱.۱
۵	(AP) Precision Average	۲.۱
۵	Mean Average Precision (mAP)	۳.۱
۶	mAP@0.5 (mAP50)	۴.۱
۶	mAP50 و mAP	۵.۱
۹	پیاده‌سازی مدل YOLOv11m	۶.۱
۹	تنظیمات آموزش	۷.۱
۹	مقایسه با YOLOv8n-GSE	۸.۱
۱۰	مقایسه با YOLOv8n-Improvement	۹.۱
۱۰	جمع‌بندی مقایسه‌ای	۱۰.۱
۱۴	پیاده‌سازی فازی	۲
۱۴	طراحی و پیاده‌سازی سیستم کنترل فازی مبتنی بر خروجی YOLO	۱.۲
۱۴	استخراج اطلاعات از خروجی YOLO	۱.۱.۲
۱۵	ساختار سیستم استنتاج فازی	۲.۱.۲
۱۵	انتخاب توابع عضویت گاوسی	۳.۱.۲
۱۶	قوانين فازی طراحی شده	۴.۱.۲
۱۶	فرآیند استنتاج	۵.۱.۲
۲۲	خروجی فازی مربوط به مدل YOLO8n-GSE	۲.۲
۲۴	خروجی فازی مربوط به مدل YOLOV11m	۳.۲
۲۵	خروجی فازی مربوط به مدل YOLOV8n-Improvement	۴.۲



فهرست تصاویر

۷	نمودارهای precision-Recall و همچنین نمودار F1score بر حسب مقدار Confidence	۱
۸	ماتریس درهمریختگی	۲
۸	نمونه‌هایی از تشخیص مدل	۳
۱۲	نمودارهای مربوط به روند آموزش مدل	۴
۱۲	نمودارهای precision-Recall و همچنین نمودار F1score بر حسب مقدار Confidence	۵
۱۳	ماتریس درهمریختگی	۶
۱۳	نمونه‌هایی از تشخیص مدل	۷
۲۲	نمایش نمونه داده‌های داده شده به مدل فازی و نمایش خروجی آن	۸
۲۳	خروجی‌های فازی مدل YOLOV8n-GSE (نمونه اول)	۹
۲۳	خروجی‌های فازی مدل YOLOV8n-GSE (نمونه دوم)	۱۰
۲۴	خروجی‌های فازی مدل YOLOV11m (نمونه اول)	۱۱
۲۴	خروجی‌های فازی مدل YOLOV11m (نمونه دوم)	۱۲
۲۵	خروجی‌های مربوط به فازی برای مدل YOLOV8n-Improvement	۱۳



فهرست جداول



فهرست برنامه‌ها

۱۰ ساختار و استراتژی آموزش مدل YOLOV11m	۱
۱۷ کدهای مربوط به بخش پیاده‌سازی فازی	۲



۱ معیارهای ارزیابی عملکرد مدل

برای ارزیابی عملکرد مدل آشکارسازی اشیا از معیار میانگین دقت متوسط (Mean Average Precision - mAP) استفاده می‌شود. این معیار یکی از استاندارددترین شاخص‌ها در حوزه تشخیص اشیا بوده و عملکرد مدل را از نظر دقت طبقه‌بندی و کیفیت مکان‌یابی جعبه‌های مرزی به صورت همزمان ارزیابی می‌کند.

۱.۱ Recall و Precision

ابتدا دو معیار پایه تعریف می‌شوند:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

که در آن:

• تعداد نمونه‌های مثبت درست (*True Positive*) *TP*

• تعداد نمونه‌های مثبت نادرست (*False Positive*) *FP*

• تعداد نمونه‌های منفی نادرست (*False Negative*) *FN*

Precision میزان صحت پیش‌بینی‌های مثبت مدل و Recall میزان توانایی مدل در بازیابی تمامی نمونه‌های واقعی را اندازه‌گیری می‌کند.

۲.۱ (AP) Precision Average

برای هر کلاس، با تغییر آستانه اطمینان (Confidence Threshold)، منحنی Precision-Recall (Precision-Recall Curve) ترسیم می‌شود. مساحت زیر این منحنی، معیار (AP) را تشکیل می‌دهد:

$$AP = \int_0^1 Precision(Recall) d(Recall) \quad (3)$$

این مقدار بیانگر عملکرد مدل در یک کلاس مشخص است.

۳.۱ Mean Average Precision (mAP)

میانگین AP در تمام کلاس‌ها به صورت زیر تعریف می‌شود:



$$mAP = \frac{1}{C} \sum_{i=1}^C AP_i \quad (4)$$

که در آن C تعداد کلاس‌ها است.

۴.۱ معیار mAP@0.5 (mAP50)

در آشکارسازی اشیا، برای تعیین صحیح بودن یک پیش‌بینی از معیار همپوشانی (Intersection over Union) IoU استفاده می‌شود:

$$IoU = \frac{Area_{intersection}}{Area_{union}} \quad (5)$$

در معیار mAP@0.5، یک پیش‌بینی زمانی صحیح در نظر گرفته می‌شود که:

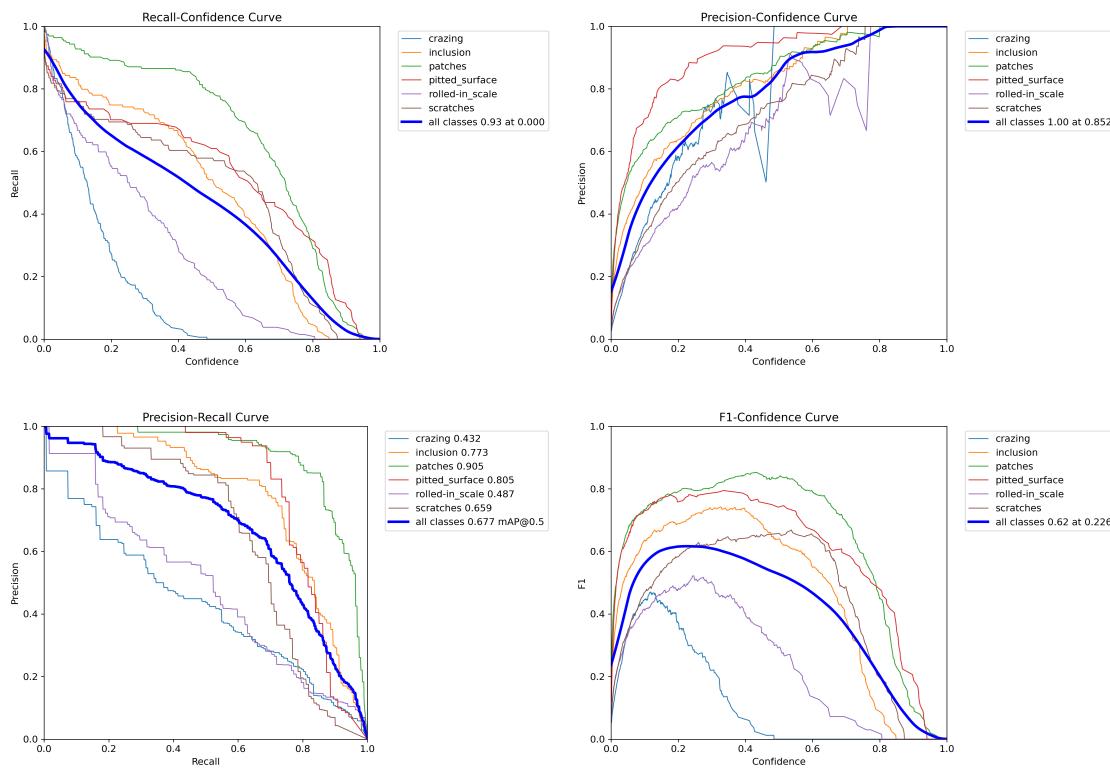
$$IoU \geq 0.5 \quad (6)$$

به عبارت دیگر، اگر همپوشانی بین جعبه پیش‌بینی شده و جعبه واقعی حداقل ۵۰ درصد باشد، آن پیش‌بینی به عنوان محسوب می‌شود.

۵.۱ مقاوت mAP50 و mAP

- mAP50 تنها در آستانه IoU برابر با 0.5 محاسبه می‌شود.
- در نسخه‌های جدید YOLO، معیار mAP@0.5:0.95 نیز گزارش می‌شود که میانگین mAP در بازه آستانه‌های IoU از 0.5 تا 0.95 با گام 0.05 است.

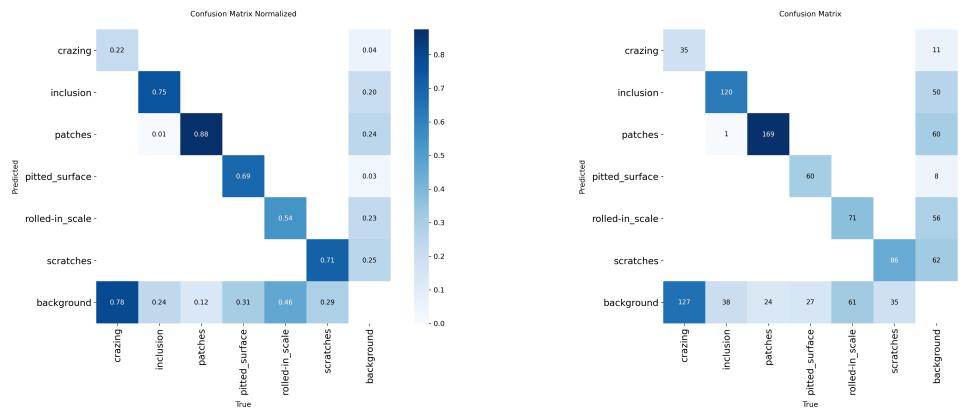
معیار mAP@0.5 معمولاً مقدار بالاتری دارد و نشان‌دهنده توانایی کلی مدل در تشخیص اهداف است، در حالی که mAP@0.5:0.95 معیار سخت‌گیرانه‌تری بوده و کیفیت دقیق مکان‌یابی جعبه‌های مرزی را بهتر ارزیابی می‌کند.



شکل ۱: نمودارهای precision-Recall و F1score بر حسب مقدار Confidence و همچنین نمودار precision-Recall

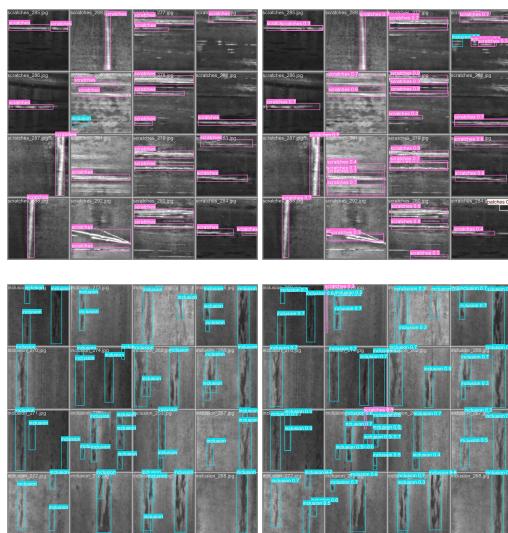
در بالا این معیارها را توضیح دادیم. همانطور که مشخص است در اینجا در برخی از کلاس‌ها بهبود داشتیم و در برخی افت. مثلا در کلاس scratches و rolled in scale افت در عملکرد داشتیم اما در مابقی کلاس‌ها عملکردهای بهبود یافته است. این موضوع نشان می‌دهد که بلوک‌هایی که باعث افزایش توجه در برخی نواحی می‌شوند دقیقاً روی کلاس‌ها می‌اندازند مثلاً خراش‌ها که در طول قطعه پیوسته وجود دارند، توسط این بلوک‌ها قابل تشخیص نمی‌شوند و یا دقیقاً پایین‌تر می‌آید اما به طور کلی این روش تا حدودی باعث بهبود دقیق مدل شده است.

در طبقه‌بندی کلاس‌ها نیز ماتریس درهم‌یختگی به صورت شکل زیر می‌شود. همانطور که مشخص است تا حد زیادی بهبود در نتایج این ماتریس دیده می‌شود. یعنی تا حد خوبی توانستیم تشخیص اشتباه پس زمینه با کلاس‌های دیگر را توسط مدل کاهش دهیم.



شکل ۲: ماتریس درهم رینتگی

نمونه‌ای از تشخیص مدل به صورت زیر می‌باشد:



شکل ۳: نمونه‌هایی از تشخیص مدل

همانطور که مشخص است تا حدی توانسته‌ایم با این روش مدل را بهبود دهیم و truth ground را تا حد امکان دنبال نماییم اما این روش نیز باز می‌تواند بهتر شود. این بار به سراغ پیاده‌سازی سومین مدل رفتیم که در آن یک مدل YOLOv11m را آموخته دادیم.



۶.۱ پیاده‌سازی مدل YOLOv11m

در این بخش، مدل YOLO11m به عنوان یک مدل میان‌رده (Medium) از خانواده YOLO11 مورد استفاده قرار گرفت. این مدل نسبت به نسخه‌های Nano و Small دارای عمق و عرض شبکه بیشتر بوده و ظرفیت یادگیری بالاتری دارد. YOLO11m با بهره‌گیری از معماری بهینه‌شده، ساختار چندمقیاسی در بخش Neck و سر آشکارساز تفکیک شده (Decoupled Head)، قادر به استخراج ویژگی‌های پیچیده‌تر و مدل‌سازی دقیق‌تر اهداف در مقیاس‌های مختلف است. افزایش تعداد پارامترها در این مدل موجب بهبود دقت تشخیص می‌شود، هرچند هزینه محاسباتی و مصرف حافظه نیز افزایش می‌یابد.

۷.۱ تنظیمات آموزش

مدل با تنظیمات زیر آموزش داده شد:

- اندازه ورودی تصاویر: 640×640
- تعداد دوره آموزش: ۲۰۰
- اندازه دسته (Batch Size): ۱۶
- بهینه‌ساز: AdamW
- نرخ یادگیری اولیه: 10^{-3}
- زمان گرمسازی: ۵ دوره
- غیرفعال‌سازی MixUp و Mosaic
- استفاده از افزایش داده رنگی (HSV augmentation)

استفاده از AdamW موجب بهبود تنظیم وزن‌ها و کاهش بیش‌برازش شده و غیرفعال‌سازی Mosaic باعث حفظ ساختار طبیعی نمونه‌ها گردید.

۸.۱ مقایسه با YOLOv8n-GSE

در نسخه اولیه توسعه‌یافته بر پایه YOLOv8n، شامل اضافه کردن تنها یک بلوک GSE بود. در مقایسه:

- YOLO11m دارای پارامترهای بیشتر و ظرفیت یادگیری بالاتر است.
- YOLOv8n اولیه سبک‌تر بوده و هزینه محاسباتی کمتری دارد.
- YOLO11m در سناریوهای پیچیده دقت بالاتری ارائه می‌دهد، اما زمان آموزش و استنتاج بیشتری نیاز دارد.



۹.۱ مقایسه با YOLOv8n-Improvement

مدل YOLOv8n-Improvement نسخه‌ای بهینه‌سازی شده با تمرکز بر مکانیزم‌های توجه و بهبود تابع هزینه بود. این مدل شامل-CSP، ABAN، GAM، ESCD، PIoUv2 و ساختار آن به صورت هم‌افرا طراحی شده است. در مقایسه با YOLO11m :

- YOLO11m از نظر ظرفیت شبکه بزرگ‌تر است، اما فاقد مأذول‌های توجه اختصاصی مانند GAM و GSE می‌باشد.
- YOLOv8n-GSE با وجود پارامتر کمتر، از طریق بهینه‌سازی ساختاری عملکرد رقابتی ارائه می‌دهد.
- YOLOv8n-GSE در PIoUv2 موجب همگرایی سریع‌تر در رگرسیون جعبه‌های مرزی می‌شود.
- YOLO11m بیشتر بر افزایش عمق و عرض شبکه متکی است، در حالی که YOLOv8n-GSE بر بهبود کیفیت استخراج و تلفیق ویژگی تمرکز دارد.

۱۰.۱ جمع‌بندی مقایسه‌ای

به طور کلی می‌توان بیان کرد:

- YOLOv8n اولیه: سبک، سریع و دارای بهبودهای ساختاری پایه.
- YOLOv8n-Improvement: نسخه بهینه‌شده با مکانیزم‌های توجه و تابع هزینه پیشرفته، دارای تعادل مناسب بین دقت و کارایی.
- YOLO11m: مدل بزرگ‌تر با ظرفیت یادگیری بالاتر و دقت بالقوه بیشتر، اما با هزینه محاسباتی بالاتر.

بنابراین، انتخاب مدل مناسب وابسته به محدودیت منابع محاسباتی و سطح دقت مورد نیاز است. در کاربردهای صنعتی با محدودیت سخت‌افزاری، YOLOv8n-GSE گزینه‌ای کارآمد محسوب می‌شود، در حالی که در صورت عدم محدودیت منابع، YOLO11m می‌تواند عملکرد دقیق‌تری ارائه دهد.

هرچند که با تمامی این تقاضای باز هم خیلی مدل YOLO11m تغییرات آنچنانی در نتایج ایجاد نکرده است. با اینکه مقداری نتایج را

بهبود داده اما این مقدار بهبود نسبت به بار محاسباتی اعمال شده، چندان منطقی نیست.

کد زیر برای آموزش این مدل زده شده است:

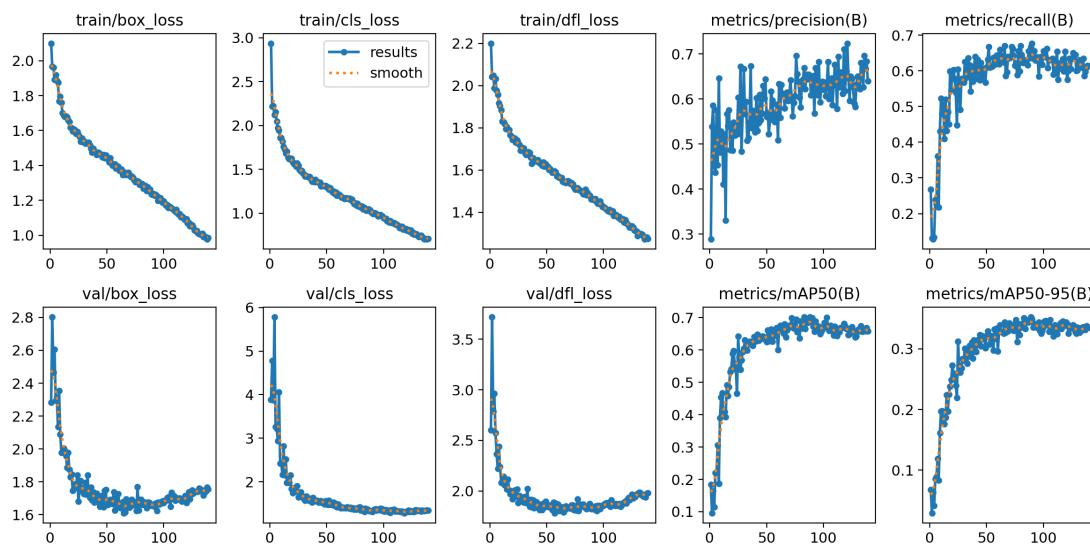
```
 1 import torch
 2 import torch.nn as nn
 3 from ultralytics import YOLO
 4 from ultralytics.nn.modules import SPPF
 5
 6 model = YOLO("yolo11m.pt")
 7
 8
 9
10 model.model.to('cuda')
11
12 model.train()
```



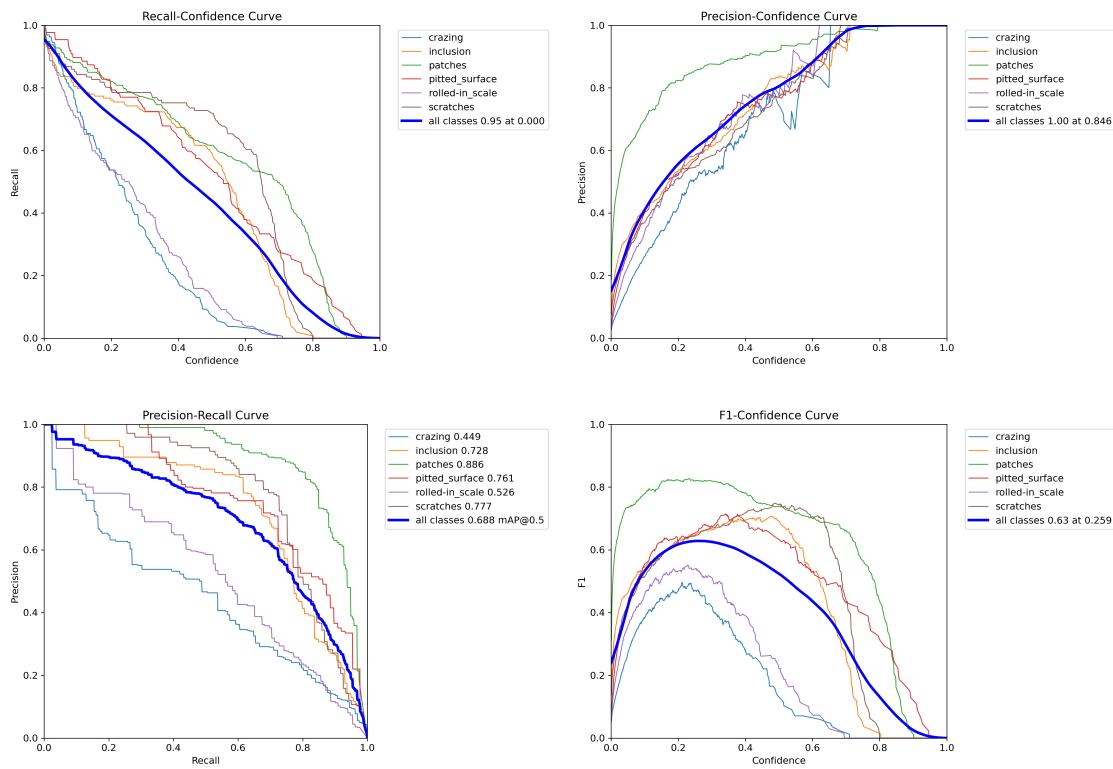
```
۱۳     data="data.yaml",
۱۴     epochs=200,
۱۵     imgsz=640,
۱۶     batch=16,
۱۷     optimizer="AdamW",
۱۸     lr0=1e-3,
۱۹     lrf=0.01,
۲۰     cos_lr=True,
۲۱     warmup_epochs=5,
۲۲
۲۳     mosaic=0.0,
۲۴     mixup=0.0,
۲۵     cutmix=0.0,
۲۶     fliplr=0.5,
۲۷     flipud=0.5,
۲۸     scale=0.3,
۲۹
۳۰     hsv_h=0.015,
۳۱     hsv_s=0.4,
۳۲     hsv_v=0.4,
۳۳
۳۴     patience=50,
۳۵     device=0,
۳۶
۳۷     project="/content/drive/MyDrive/FINAL_Project/Training_Results2",
۳۸     name="YOLO11m"
۳۹ )
```

۱: ساختار و استراتژی آموزش مدل YOLOV11m Code

نتایج مربوط به آموزش مدل به صورت زیر است:



شکل ۴: نمودارهای مربوط به روند آموزش مدل



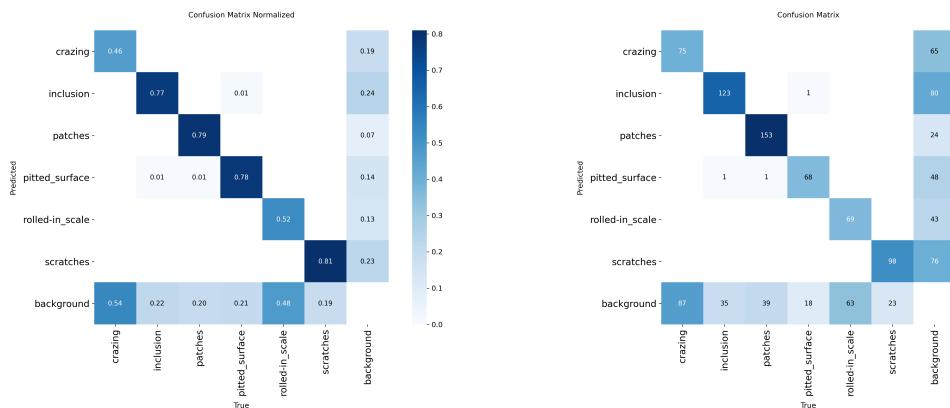
شکل ۵: نمودارهای precision-Recall و precision-Confidence و همچنین نمودار F1score بر حسب مقدار Recall و Confidence

همانطور که مشاهده می‌شود در کلاس crazing و scratches نتایج بهبود داشته است. در برخی کلاس‌ها مقداری نتایج افت داشته است ولی به صورت برآیندی نتایج بهتر شده است. در نمودارهای روند آموزش اورفیت مشاهده نمی‌شود و مدل به شکل مناسبی آموزش دیده است.

اندک بهبودی که مدل ایجاد کرده قابل تقدیر است اما بار محاسباتی را شدیداً بالا برده است بطوریکه تعداد پارامتر آموزش دیده ۷ برابر شده است.

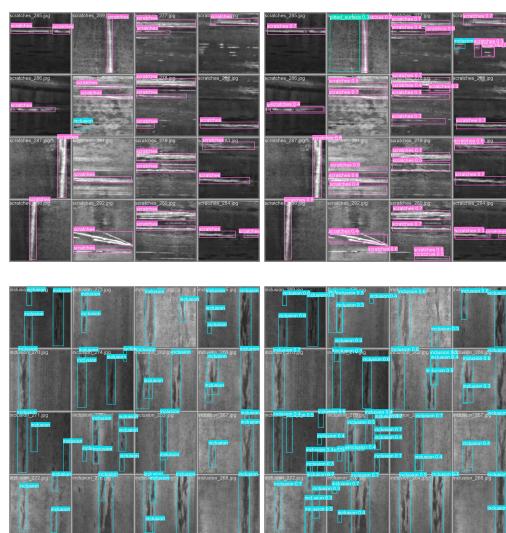
در کل عملکرد این مدل بهتر از مدل‌های قبلی است و بین دو مدل قبل آنچنان تفاوتی نیست و بنظر می‌آید که مدل اولی در یکسری از کلاس‌ها عملکرد بهتری داشته که به چشم می‌آید اما مدل دوم در آنها خوب نبوده بر عکس مدل دوم در بعضی کلاس‌ها مقداری بهبود داشته است.

ماتریس درهمریختگی مربوط به این مدل نیز به صورت زیر است:



شکل ۶: ماتریس درهمریختگی

همانطور که مشاهده می‌شود تشخیص مدل در کلاس‌ها بهتر شده و کمتر تشخیص غلط بر روی پس زمینه دارد. نمونه خروجی گرفته شده از این مدل نیز به صورت زیر است:



شکل ۷: نمونه‌هایی از تشخیص مدل



۲ پیاده‌سازی فازی

۱.۲ طراحی و پیاده‌سازی سیستم کنترل فازی مبتنی بر خروجی YOLO

در این بخش، به منظور تبدیل خروجی مدل آشکارساز شیء به یک تصمیم کنترلی سطح بالا، از یک سیستم استنتاج فازی (Fuzzy Inference System) استفاده شده است. هدف این سیستم، تبدیل اطلاعات حاصل از تشخیص عیوب سطحی به یک فرمان کنترلی پیوسته برای تنظیم سرعت خط تولید می‌باشد.

۱.۱.۲ استخراج اطلاعات از خروجی YOLO

مدل YOLO پس از انجام آشکارسازی، برای هر شیء شناسایی شده اطلاعات زیر را تولید می‌کند:

- مختصات جعبه مرزی (Bounding Box)
- شناسه کلاس
- میزان اطمینان (Confidence Score)

برای استفاده از این خروجی در سیستم فازی، ابتدا یک معیار ریسک وزندار تعریف شده است. برای هر کلاس، یک ضریب ریسک از پیش تعیین شده در نظر گرفته شده است:

$$\text{Risk Weighted} = \sum_{i=1}^N \left(\frac{\text{Area}_i}{\text{ImageArea}} \times \text{RiskFactor}_i \right) \times 10$$

که در آن:

- مساحت جعبه مرزی شیء i : Area_i
- مساحت کل تصویر: ImageArea
- ضریب ریسک متناظر با کلاس RiskFactor_i

این مقدار در بازه $[0, 1]$ نرمال‌سازی می‌شود. همچنین میانگین اطمینان تشخیص‌ها به صورت زیر محاسبه می‌شود:

$$\text{Confidence Mean} = \frac{1}{N} \sum_{i=1}^N \text{Conf}_i$$

در صورت عدم وجود تشخیص، ریسک برابر صفر و اطمینان برابر یک در نظر گرفته شده است (به معنای نبود تهدید). بنابراین خروجی YOLO به دو ورودی عددی برای سیستم فازی تبدیل می‌شود:

- امتیاز ریسک نرمال‌شده
- میانگین میزان اطمینان مدل



۲.۱.۲ ساختار سیستم استنتاج فازی

سیستم طراحی شده از نوع Mamdani Fuzzy Inference System می‌باشد و شامل:

- دو متغیر ورودی (Antecedent)
- یک متغیر خروجی (Consequent)
- مجموعه‌ای از قوانین فازی

ورودی‌ها

[0, 1] در بازه Score Risk .۱

[0, 1] در بازه Confidence .۲

هر ورودی دارای سه مجموعه فازی می‌باشد:

{Low, Medium, High}

خروجی متغیر خروجی با عنوان Speed Change در بازه [1, -1] تعریف شده است که شامل سه مجموعه فازی است:

{Brake, Maintain, Accelerate}

۳.۱.۲ انتخاب توابع عضویت گاوسی

برای تمامی متغیرها از تابع عضویت گاوسی (Gaussian Membership Function) استفاده شده است:

$$\mu(x) = e^{-\frac{(x-c)^2}{2\sigma^2}}$$

که در آن c مرکز تابع و σ انحراف معیار است.

دلایل انتخاب تابع گاوسی:

- پیوستگی و مشتق‌پذیری کامل (رفتار نرم و بدون شکست ناگهانی)
- مناسب برای مدل‌سازی عدم قطعیت تدریجی
- جلوگیری از تغییرات ناگهانی در خروجی کنترل
- هموارسازی تصمیم‌گیری در نواحی مرزی بین Low، Medium و High

در کاربرد صنعتی، تصمیم‌گیری ناگهانی می‌تواند منجر به رفتار ناپایدار سیستم شود؛ لذا استفاده از تابع گاوسی باعث انتقال نرم بین حالات مختلف می‌شود.



۴.۱.۲ قوانین فازی طراحی شده

قوانین به صورت ترکیب منطقی AND بین ورودی‌ها تعریف شده‌اند. نمونه‌ای از قوانین:

- اگر ریسک کم و اطمینان زیاد باشد \leftarrow افزایش سرعت
- اگر ریسک متوسط و اطمینان زیاد باشد \leftarrow کاهش سرعت
- اگر ریسک بالا باشد \leftarrow کاهش سرعت (صرف نظر از اطمینان)

به طور کلی:

- ریسک پایین \leftarrow تمایل به افزایش سرعت
- ریسک متوسط \leftarrow رفتار محافظه‌کارانه
- ریسک بالا \leftarrow کاهش سرعت قطعی

وجود متغیر اطمینان باعث می‌شود تصمیم تنها بر اساس اندازه عیب نباشد، بلکه میزان اعتماد مدل نیز لحاظ گردد. برای مثال:

- ریسک کم ولی اطمینان پایین \leftarrow حفظ سرعت (عدم تصمیم تهاجمی)
- ریسک بالا و اطمینان بالا \leftarrow ترمز قوی

۵.۱.۲ فرآیند استنتاج

مراحل پردازش به صورت زیر است:

۱. Fuzzification: تبدیل مقادیر عددی ریسک و اطمینان به درجات عضویت

۲. Evaluation Rule: اعمال قوانین با استفاده از عملگر AND (مینیمم)

۳. Aggregation: ترکیب خروجی قوانین

۴. Defuzzification: تبدیل خروجی فازی به مقدار عددی (مرکز سطح)

خروجی نهایی عددی در بازه $[1, -1]$ است:

- مقدار نزدیک به ۱ \leftarrow کاهش شدید سرعت
- مقدار نزدیک به ۰ \leftarrow حفظ سرعت
- مقدار نزدیک به -۱ \leftarrow افزایش سرعت

کد زیر مربوط به بخش فازی است. این بخش برای تمامی سه مدل YOLOV به یک شکل پیاده شده است:



```
 1 SAVE_DIR = "/content/drive/MyDrive/FINAL_Project/Fuzzy_System"
 2 os.makedirs(SAVE_DIR, exist_ok=True)
 3
 4 MODEL_PATH = "/content/drive/MyDrive/FINAL_Project/Training_Results2/YOLO11m_GSE_1280px2/
weights/best.pt"
 5 VAL_IMG_DIR = "dataset_yolo/images/val"
 6 VAL_LBL_DIR = "dataset_yolo/labels/val"
 7
 8 CLASSES = ["crazing", "inclusion", "patches", "pitted_surface", "rolled-in-scale", "scratches
"]
 9 COLORS = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255), (0, 255, 255)]
10 import os
11 import random
12 import cv2
13 import numpy as np
14 import matplotlib.pyplot as plt
15 import skfuzzy as fuzz
16 from skfuzzy import control as ctrl
17 from ultralytics import YOLO
18
19
20 RISK_FACTORS = {
21     0: 1.0, # crazing (High Risk)
22     1: 0.9, # inclusion
23     2: 0.6, # patches
24     3: 0.8, # pitted_surface
25     4: 0.5, # rolled-in-scale
26     5: 0.2 # scratches (Low Risk)
27 }
28
29 class AdvancedFuzzyController:
30     def __init__(self):
31
32         self.risk_score = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'risk_score')
33         self.confidence = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'confidence')
34         self.speed_change = ctrl.Consequent(np.arange(-1, 1.01, 0.01), 'speed_change')
35
36         self.risk_score['Low']      = fuzz.gaussmf(self.risk_score.universe, 0, 0.15)
37         self.risk_score['Medium']  = fuzz.gaussmf(self.risk_score.universe, 0.5, 0.15)
38         self.risk_score['High']    = fuzz.gaussmf(self.risk_score.universe, 1, 0.15)
39
40         self.confidence['Low']    = fuzz.gaussmf(self.confidence.universe, 0, 0.15)
41         self.confidence['Medium'] = fuzz.gaussmf(self.confidence.universe, 0.5, 0.15)
42         self.confidence['High']   = fuzz.gaussmf(self.confidence.universe, 1, 0.15)
43
```



```
٤٤         self.speed_change['Brake']      = fuzz.gaussmf(self.speed_change.universe, -1, 0.25)
٤٥         self.speed_change['Maintain']   = fuzz.gaussmf(self.speed_change.universe, 0, 0.15)
٤٦         self.speed_change['Accelerate'] = fuzz.gaussmf(self.speed_change.universe, 1, 0.25)
٤٧
٤٨         self.rules = [
٤٩
٥٠             ctrl.Rule(self.risk_score['Low'] & self.confidence['High'], self.speed_change['
Accelerate']),
٥١             ctrl.Rule(self.risk_score['Low'] & self.confidence['Medium'], self.speed_change['
Accelerate']),
٥٢             ctrl.Rule(self.risk_score['Low'] & self.confidence['Low'], self.speed_change['
Maintain']),
٥٣
٥٤             ctrl.Rule(self.risk_score['Medium'] & self.confidence['High'], self.speed_change
['Brake']),
٥٥             ctrl.Rule(self.risk_score['Medium'] & self.confidence['Medium'], self.
speed_change['Maintain']),
٥٦             ctrl.Rule(self.risk_score['Medium'] & self.confidence['Low'], self.speed_change['
Maintain']),
٥٧
٥٨             ctrl.Rule(self.risk_score['High'] & self.confidence['High'], self.speed_change['
Brake']),
٥٩             ctrl.Rule(self.risk_score['High'] & self.confidence['Medium'], self.speed_change
['Brake']),
٦٠             ctrl.Rule(self.risk_score['High'] & self.confidence['Low'], self.speed_change['
Brake']),
٦١         ]
٦٢
٦٣         self.control_system = ctrl.ControlSystem(self.rules)
٦٤         self.simulation = ctrl.ControlSystemSimulation(self.control_system)
٦٥
٦٦     def compute(self, risk_val, conf_val):
٦٧         self.simulation.input['risk_score'] = np.clip(risk_val, 0, 1)
٦٨         self.simulation.input['confidence'] = np.clip(conf_val, 0, 1)
٦٩         self.simulation.compute()
٧٠         return self.simulation.output['speed_change']
٧١
٧٢     def plot_result_manual(self, filename_suffix):
٧٣
٧٤         fig, ax = plt.subplots(figsize=(8, 4))
٧٥
٧٦         x = self.speed_change.universe
٧٧         mfs = self.speed_change.terms
٧٨         for label, term in mfs.items():
٧٩             ax.plot(x, term.mf, label=label)
```



```

1+         ax.fill_between(x, 0, term.mf, alpha=0.1)

2+
3+     try:
4+
5+         res = self.simulation.output['speed_change']
6+
7+         ax.vlines(res, 0, 1, colors='r', linestyles='dashed', linewidth=2, label='Result')
8+
9+     except:
10+        pass
11+
12+    ax.set_title(f"Fuzzy Result: {filename_suffix}")
13+    ax.legend()
14+
15+    save_path = os.path.join(SAVE_DIR, f"{filename_suffix}.png")
16+    plt.savefig(save_path, bbox_inches='tight')
17+    plt.close(fig)
18+    print(f"Plots saved: {save_path}")
19+
20+
21+
22+ fuzzy_brain = AdvancedFuzzyController()
23+
24+
25+ scenarios = [
26+     {"area": 0.85, "conf": 0.90, "name": "Scenario1_HighRisk"},
27+     {"area": 0.10, "conf": 0.85, "name": "Scenario2_LowRisk"},
28+     {"area": 0.50, "conf": 0.30, "name": "Scenario3_Ambiguous"}
29+ ]
30+
31+
32+ for sc in scenarios:
33+     res = fuzzy_brain.compute(sc["area"], sc["conf"])
34+
35+     print(f"Testing {sc['name']}: Risk={sc['area']}, Conf={sc['conf']} --> Speed Change={res:.2f}")
36+
37+     fname = f"risk_{sc['area']}_{sc['conf']}_{sc['name']}_result"
38+     fuzzy_brain.plot_result_manual(fname)
39+
40+
41+ def calculate_weighted_risk(results, img_shape):
42+     h, w = img_shape[:2]
43+
44+     total_pixels = h * w
45+
46+     total_weighted_risk = 0
47+
48+     confidences = []
49+
50+
51+     if len(results.boxes) == 0:
52+         return 0.0, 1.0
53+
54+
55+     for box in results.boxes:
56+         x1, y1, x2, y2 = map(int, box.xyxy[0])
57+
58+         box_area = (x2 - x1) * (y2 - y1)
59+
60+         confidence = fuzzy_brain.confidence((x1, y1), (x2, y2))
61+
62+         weighted_risk = confidence * box_area
63+
64+         total_weighted_risk += weighted_risk
65+
66+         confidences.append(confidence)
67+
68+     weighted_risk_per_pixel = total_weighted_risk / total_pixels
69+
70+     return weighted_risk_per_pixel, sum(confidences) / len(confidences)

```



```
122     normalized_area = box_area / total_pixels
123
124
125     cls_id = int(box.cls)
126     risk_factor = RISK_FACTORS.get(cls_id, 0.5)
127
128     total_weighted_risk += (normalized_area * risk_factor) * 10
129     confidences.append(float(box.conf))
130
131     mean_conf = sum(confidences) / len(confidences)
132
133     return min(total_weighted_risk, 1.0), mean_conf
134
135 def draw_gt(img, label_path):
136     h, w, _ = img.shape
137     if os.path.exists(label_path):
138         with open(label_path, 'r') as f:
139             for line in f.readlines():
140                 cls, x, y, nw, nh = map(float, line.split())
141                 x1, y1 = int((x - nw/2) * w), int((y - nh/2) * h)
142                 x2, y2 = int((x + nw/2) * w), int((y + nh/2) * h)
143                 cv2.rectangle(img, (x1, y1), (x2, y2), (255, 255, 255), 2)
144                 cv2.putText(img, f"GT: {CLASSES[int(cls)]}", (x1, y1-10),
145                             cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
146
147     return img
148
149
150 def draw_pred(img, results):
151     for box in results.bounding_boxes:
152         c = int(box.cls)
153         conf = float(box.conf)
154         xyxy = box.xyxy[0].cpu().numpy().astype(int)
155         color = COLORS[c]
156         cv2.rectangle(img, (xyxy[0], xyxy[1]), (xyxy[2], xyxy[3]), color, 2)
157         cv2.putText(img, f"{CLASSES[c]} {conf:.2f}", (xyxy[0], xyxy[1]-10),
158                     cv2.FONT_HERSHEY_SIMPLEX, 0.6, color, 2)
159
160     return img
161
162
163 model = YOLO(MODEL_PATH)
164 all_images = [f for f in os.listdir(VAL_IMG_DIR) if f.endswith('.jpg', '.png')]
165
166 sample_images = random.sample(all_images, min(3, len(all_images)))
167
168 for img_name in sample_images:
169     img_path = os.path.join(VAL_IMG_DIR, img_name)
170     lbl_path = os.path.join(VAL_LBL_DIR, img_name.replace('.jpg', '.txt'))
```



```
۱۶۸
۱۶۹     original_img = cv2.imread(img_path)
۱۷۰     if original_img is None: continue
۱۷۱     original_img = cv2.cvtColor(original_img, cv2.COLOR_BGR2RGB)
۱۷۲
۱۷۳     results = model(img_path, conf=0.25, verbose=False)[0]
۱۷۴
۱۷۵     calc_risk, calc_conf = calculate_weighted_risk(results, original_img.shape)
۱۷۶
۱۷۷     fuzzy_output = fuzzy_brain.compute(calc_risk, calc_conf)
۱۷۸
۱۷۹     gt_img = draw_gt(original_img.copy(), lbl_path)
۱۸۰     pred_img = draw_pred(original_img.copy(), results)
۱۸۱
۱۸۲     fig = plt.figure(figsize=(18, 10))
۱۸۳     gs = fig.add_gridspec(2, 2)
۱۸۴
۱۸۵     ax1 = fig.add_subplot(gs[0, 0])
۱۸۶     ax1.imshow(gt_img)
۱۸۷     ax1.set_title(f"Ground Truth: {img_name}", fontsize=12, fontweight='bold')
۱۸۸     ax1.axis("off")
۱۸۹
۱۹۰     ax2 = fig.add_subplot(gs[0, 1])
۱۹۱     ax2.imshow(pred_img)
۱۹۲     ax2.set_title(f"YOLOv8 Detection", fontsize=12, fontweight='bold')
۱۹۳     ax2.axis("off")
۱۹۴
۱۹۵     ax3 = fig.add_subplot(gs[1, 0])
۱۹۶     ax3.axis("off")
۱۹۷     decision_text = 'ACCELERATE'
۱۹۸     if fuzzy_output < -0.1: decision_text = 'BRAKE (Slow Down)'
۱۹۹     elif fuzzy_output < 0.1: decision_text = 'MAINTAIN'
۲۰۰
۲۰۱     text_info = (
۲۰۲         f"--- Fuzzy Input Analysis ---\n\n"
۲۰۳         f"1. Weighted Risk Score: {calc_risk:.3f} (0 to 1)\n"
۲۰۴         f"2. Mean Confidence:      {calc_conf*100:.1f}%\n\n"
۲۰۵         f"--- Fuzzy Output Decision ---\n\n"
۲۰۶         f"SPEED CHANGE FACTOR:    {fuzzy_output:.3f}\n"
۲۰۷         f"Decision: {decision_text}"
۲۰۸     )
۲۰۹     ax3.text(0.1, 0.5, text_info, fontsize=14, va='center', ha='left',
۲۱۰             bbox=dict(boxstyle="round", pad=0.5, fc="#f0f0f0", ec="black", alpha=0.9))
۲۱۱
۲۱۲     ax4 = fig.add_subplot(gs[1, 1])
```



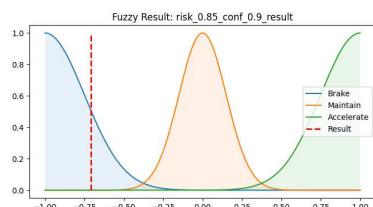
```

۲۱۳     x = fuzzy_brain.speed_change.universe
۲۱۴     mfs = fuzzy_brain.speed_change.terms
۲۱۵     for label, term in mfs.items():
۲۱۶         ax4.plot(x, term.mf, label=label, linewidth=1.5)
۲۱۷         ax4.fill_between(x, 0, term.mf, alpha=0.1)
۲۱۸
۲۱۹     ax4.vlines(fuzzy_output, 0, 1, colors='r', linestyles='dashed', linewidth=3, label='
Result')
۲۲۰     ax4.set_title("Fuzzy Speed Control Logic", fontsize=12, fontweight='bold')
۲۲۱     ax4.legend(loc='upper right')
۲۲۲     ax4.grid(True, alpha=0.3)
۲۲۳
۲۲۴     save_path = os.path.join(SAVE_DIR, img_name)
۲۲۵     plt.savefig(save_path, bbox_inches='tight', dpi=100)
۲۲۶     plt.close(fig)
۲۲۷     print(f"All saved : {save_path}")
۲۲۸
۲۲۹ print(f"Output plots path: \n{SAVE_DIR}")

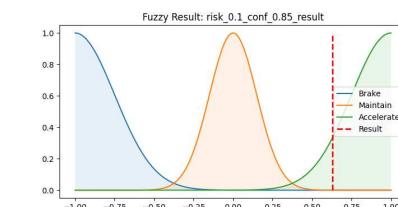
```

۲: کدهای مربوط به بخش پیاده‌سازی فازی

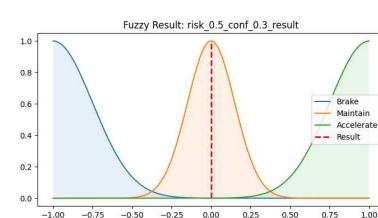
قبل از آنکه بر روی دیتای اصلی برویم، ابتدا این مدل زده شده را روی یکسری ورودی تست می‌نماییم تا از عملکرد مدل فازیمان مطمئن شویم. در واقع دو ورودی میزان اطمینان و مقدار ریسک را به مدل فازی می‌دهیم و از آن توقع داریم که سرعت مطلوب را بدهد. در سه تصویر زیر مشاهده می‌شود که مدل فازی عملکرد درستی نسبت به ورودی‌های تست داشته است: حال برای هر مدل Yolov، خروجی‌های



(آ) ریسک یک دهم و میزان اطمینان
اطمینان نه دهم



(ب) ریسک یک دهم و میزان اطمینان
اطمینان نه دهم

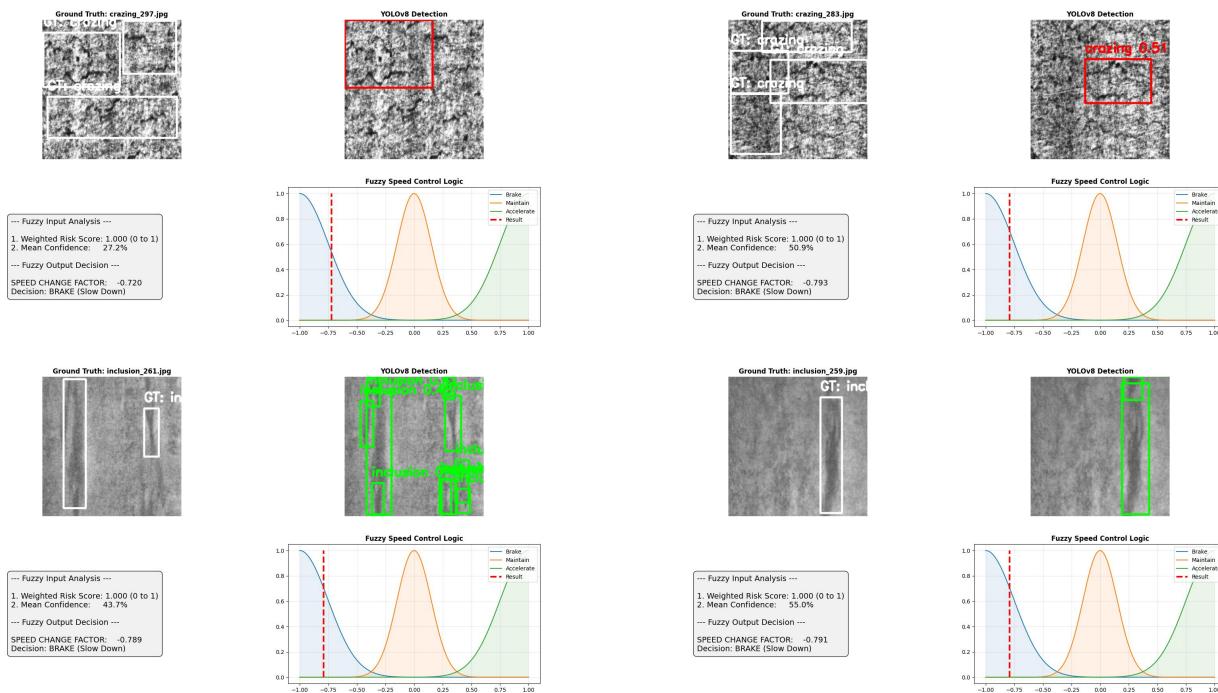


(ج) ریسک نیم و میزان اطمینان سه دهم

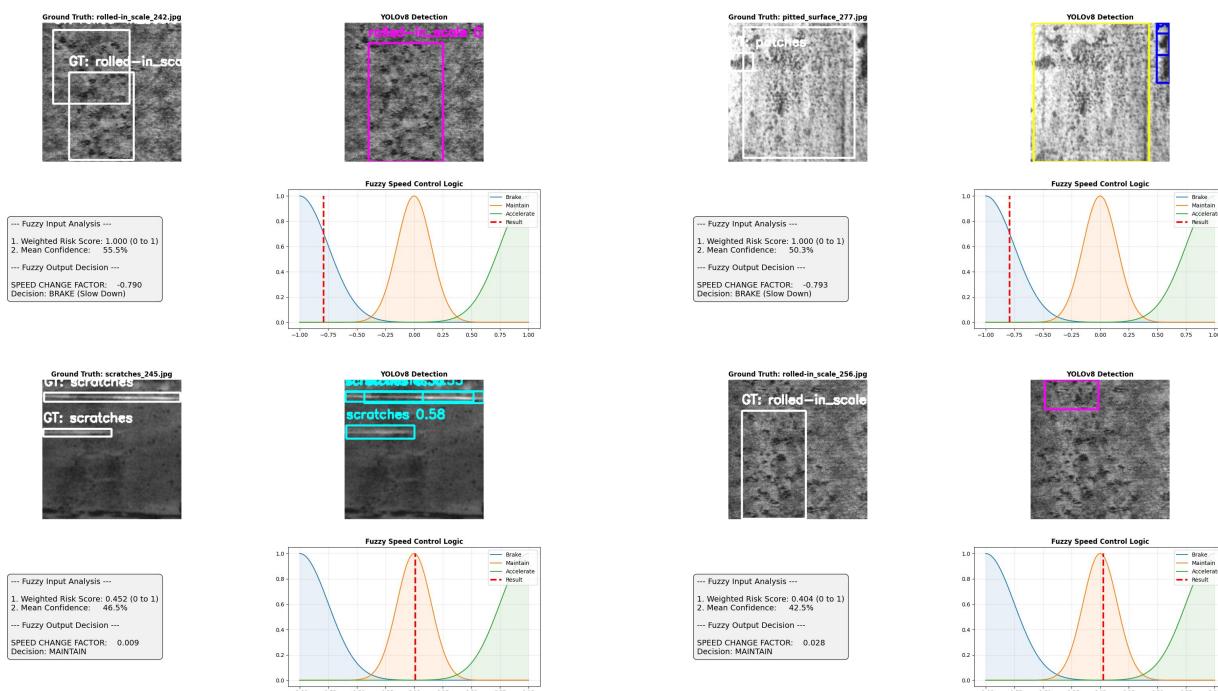
شکل ۸: نمایش نمونه داده‌های داده شده به مدل فازی و نمایش خروجی آن

مربوط به بخش فازی آن را نمایش می‌دهیم. بدیهی است که هرچه مدل Yolov اطلاعات دقیق‌تری به مدل فازی بدهد، نتایج این مدل نیز بهتر خواهد بود. بنابراین انتظار داریم که نتایج مدل Yolov11m، از دو مدل دیگر بهتر شود همچنین دو مدل دیگر نیز نتایجی تا حدی مشابه و در برخی موارد مدل Yolov8-Improvement بeter عمل نمایند.

۲.۲ خروجی فازی مربوط به مدل YOLO8n-GSE

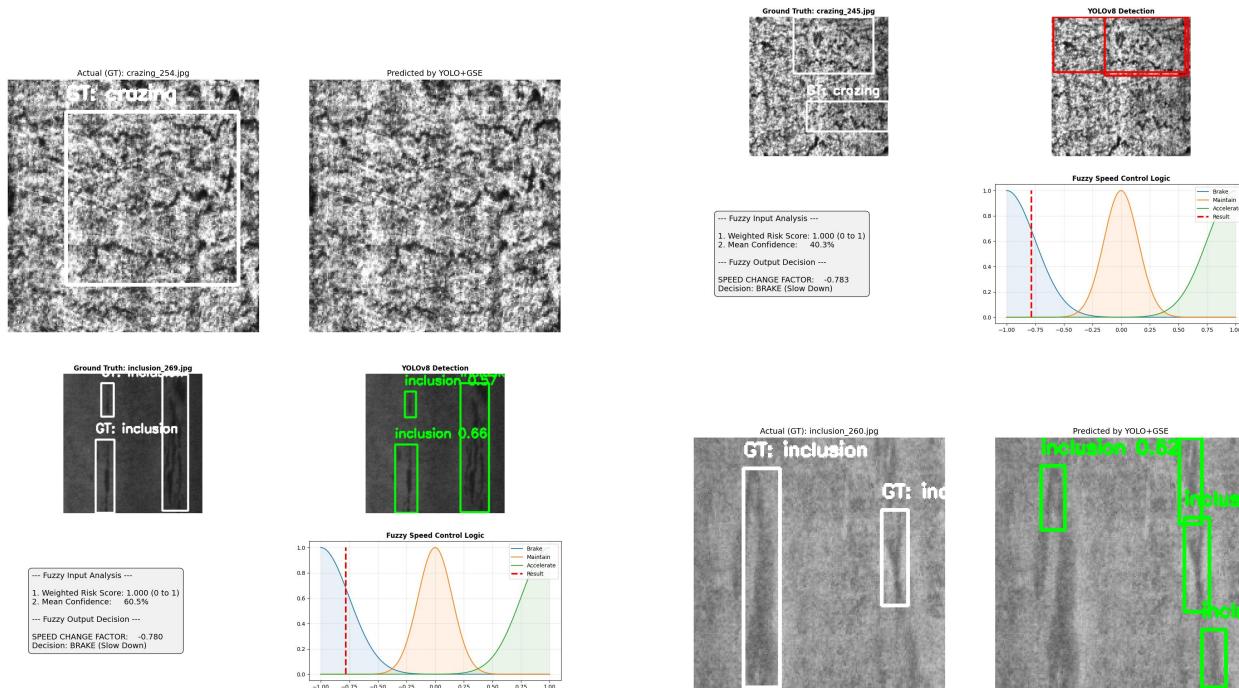


شکل ۹: خروجی‌های فازی مدل YOLOv8n-GSE (نمونه اول)

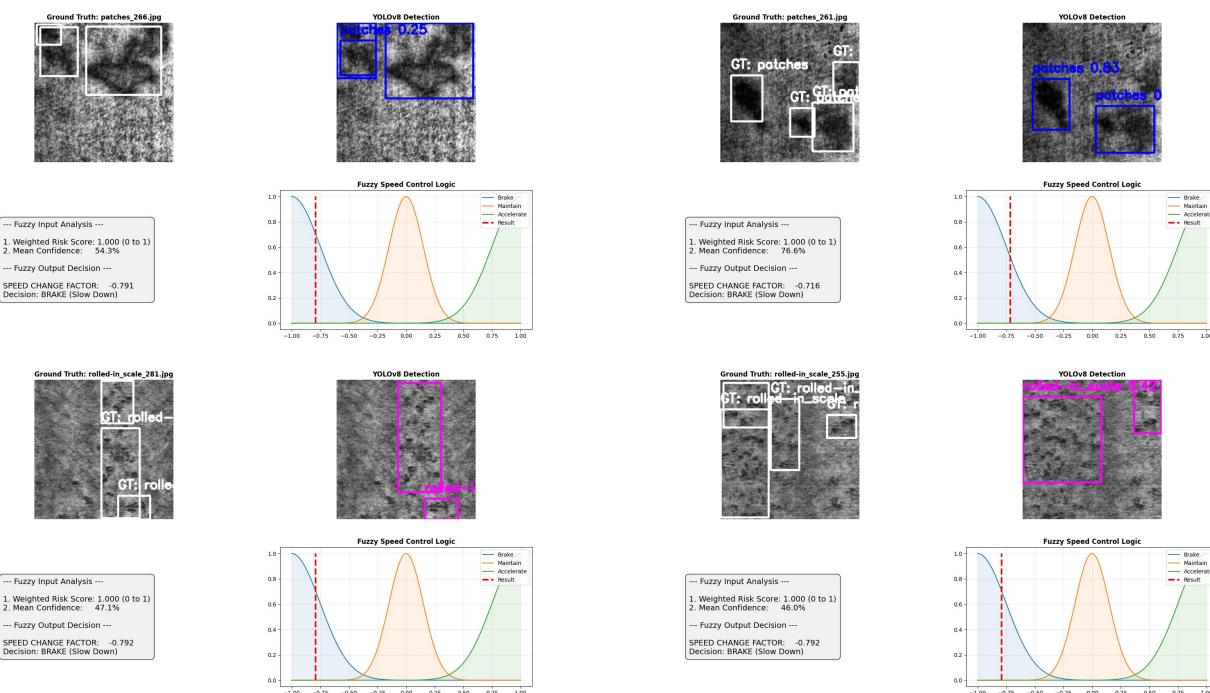


شکل ۱۰: خروجی‌های فازی مدل YOLOv8n-GSE (نمونه دوم)

۳.۲ خروجی فازی مربوط به مدل YOLOV11m

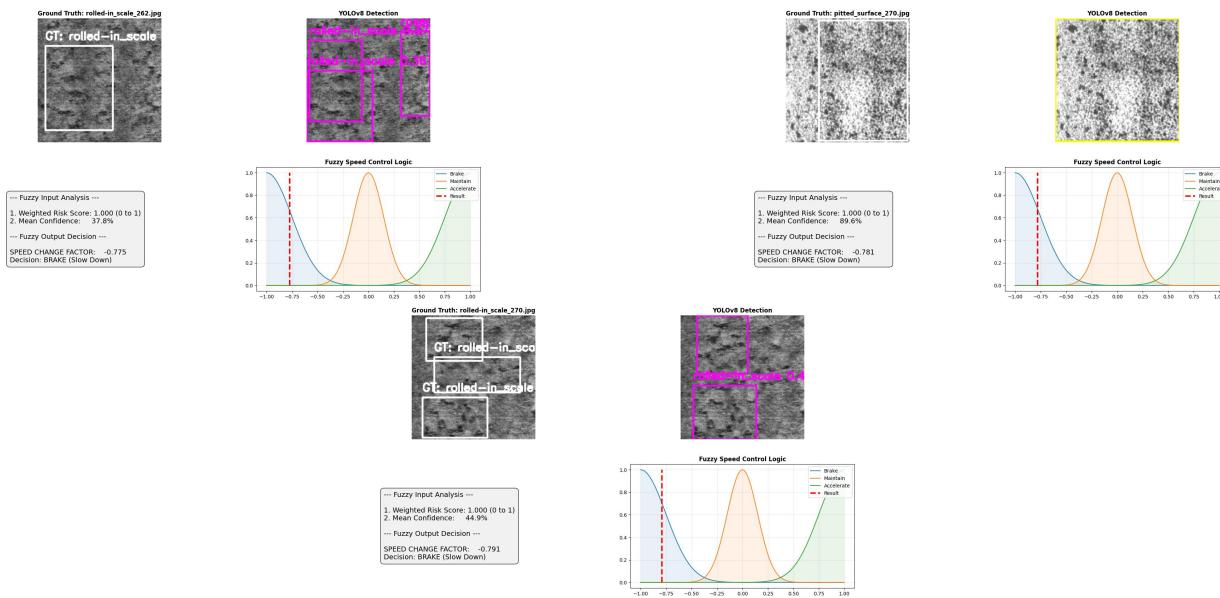


شکل ۱۱: خروجی‌های فازی مدل YOLOV11m (نمونه اول)

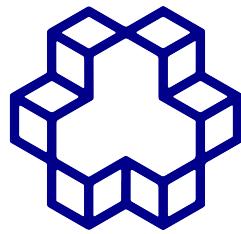


شکل ۱۲: خروجی‌های فازی مدل YOLOV11m (نمونه دوم)

۴.۲ خروجی فازی مربوط به مدل YOLOV8n-Improvement



شکل ۱۳: خروجی‌های مربوط به فازی برای مدل YOLOV8n-Improvement



دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق

درس مبانی سیستم‌های هوشمند

استاد: دکتر مهدی علیاری

پروژه پایانی بخش سوم

این پروژه با تایید جناب آقای دکتر علیاری انجام شده است

محمدامین محمدیون شبستری	نام و نام خانوادگی
۴۰۱۲۲۵۰۳	شماره دانشجویی
محمد سبحان سخایی	نام و نام خانوادگی
۴۰۱۱۹۲۵۳	شماره دانشجویی
۱۴۰۴ بهمن	تاریخ
لینک‌های مربوط به نتایج و کد :	
گوگل درایو لینک گیت‌هاب	



فهرست مطالب

۵	۱ شبیه‌سازی حلقه‌بسته کنترل هوشمند نوار نقاله	۱
۵	۱.۱ تولید ویدئوی شبیه‌سازی	۱.۱
۵	۲.۱ تعریف پارامترهای سیستم کنترل	۲.۱
۵	۳.۱ مدل‌سازی اثر سرعت بر کیفیت تصویر	۳.۱
۶	۴.۱ مراحل اجرای حلقه‌بسته	۴.۱
۶	۵.۱ محاسبه ریسک وزندار	۵.۱
۶	۶.۱ تصمیم‌گیری فازی	۶.۱
۷	۷.۱ نمایش اطلاعات و ذخیره نتایج	۷.۱
۷	۸.۱ تحلیل عملکرد سیستم	۸.۱
۷	۹.۱ ساختار کلی حلقه‌فیدبک	۹.۱
۹	 ۲ آموزش مدل RL	
۹	۱.۲ هدف طراحی عامل یادگیری تقویتی	۱.۲
۹	۲.۲ الگوریتم مورد استفاده: Q-Learning	۲.۲
۱۰	۳.۲ تعریف فضای حالت (State Space)	۳.۲
۱۱	۴.۲ فضای عمل (Action Space)	۴.۲
۱۱	۵.۲ استراتژی اکتشاف و بهره‌برداری (Exploration-Exploitation)	۵.۲
۱۱	۶.۲ طراحی تابع پاداش (Reward Function)	۶.۲
۱۲	۷.۲ فلسفه طراحی تابع پاداش	۷.۲
۱۲	 ۳ تحلیل رفتار و اثرباری عامل یادگیری تقویتی	
۱۲	۱.۳ ۱. پایداری یادگیری	۱.۳
۱۳	۲. تحلیل تعادل سرعت و ریسک	۲.۳
۱۳	۳. نقش هم‌راستایی با فازی	۳.۳
۱۳	۴. گذار از سیستم واکنشی به سیستم پیش‌بین	۴.۳



فهرست تصاویر

۸ YOLO8n-GSE	۱
۸	. YOLO8n-Improvement	۲
۹ YOLOv11m	۳
۲۰ Yolov8n-GSE	۴
۲۰	.. Yolov8n-Improvement	۵
۲۱ Yolov11m	۶



فهرست جداول



فهرست برنامه‌ها

۱۳	کدهای مربوط به بخش یادگیری تقویتی	۱
----	-----------------------------------	---



۱ شبیه‌سازی حلقه‌بسته کنترل هوشمند نوار نقاله

در این بخش، یک سیستم کنترل حلقه‌بسته مبتنی بر بینایی ماشین و منطق فازی طراحی و پیاده‌سازی شده است. هدف این سیستم، تنظیم دینامیکی سرعت نوار نقاله بر اساس میزان عیوب تشخیص‌داده شده توسط مدل YOLO می‌باشد. ساختار کلی سیستم ترکیبی از یادگیری عمیق برای تشخیص و منطق فازی برای تصمیم‌گیری کنترلی است.

۱.۱ تولید ویدئوی شبیه‌سازی

به منظور ایجاد یک سناریوی دینامیک مشابه شرایط واقعی خط تولید، تصاویر مجموعه اعتبارسنجی به یک فایل ویدئویی تبدیل شدند. در این فرآیند:

- تصاویر خوانده شده و بر اساس نام مرتب شدند.
- تمامی تصاویر به اندازه 640×640 تغییر اندازه داده شدند.
- با استفاده از VideoWriter و کدک mp4v، ویدئو با نرخ ۵ فریم بر ثانیه ساخته شد.
- هر تصویر چندین بار در ویدئو تکرار شد تا حرکت طبیعی تر شبیه‌سازی گردد.

این ویدئو به عنوان ورودی سیستم کنترل مورد مورد استفاده قرار گرفت.

۲.۱ تعریف پارامترهای سیستم کنترل

پارامترهای اصلی کنترل به صورت زیر تعریف شدند:

- سرعت اولیه: 0.5
- حداقل سرعت مجاز: 0.1
- حداکثر سرعت مجاز: 1.0
- ضریب تغییر نرم سرعت: 0.1

سرعت به صورت نرمال شده در بازه $[0, 1]$ مدل‌سازی شده است.

۳.۱ مدل‌سازی اثر سرعت بر کیفیت تصویر

به منظور شبیه‌سازی شرایط واقعی، افزایش سرعت منجر به تاری حرکتی تصویر می‌شود. این اثر با استفاده از یک کرنل خطی افقی پیاده‌سازی شد. اندازه کرنل متناسب با مقدار سرعت تغییر می‌کند:

$$\text{KernelSize} \propto \text{Speed}$$

هر چه سرعت بیشتر باشد، تاری تصویر افزایش یافته و تشخیص دشوارتر می‌شود. این بخش نقش مهمی در ایجاد فیدبک واقعی در سیستم دارد.



۴.۱ مراحل اجرای حلقه بسته

برای هر فریم از ویدئو، مراحل زیر اجرا می‌شود:

۱. اعمال تاری حرکتی متناسب با سرعت فعلی
۲. اجرای مدل YOLO برای تشخیص عیوب
۳. محاسبه ریسک وزنده بر اساس مساحت عیوب و ضرایب ریسک کلاس‌ها
۴. محاسبه میانگین اطمینان تشخیص
۵. ارسال ریسک و اطمینان به سیستم فازی
۶. دریافت خروجی کنترلی و به روزرسانی سرعت

۵.۱ محاسبه ریسک وزنده

ریسک کلی تصویر به صورت زیر محاسبه می‌شود:

$$Risk = \sum_{i=1}^N \left(\frac{Area_i}{ImageArea} \times RiskFactor_i \right) \times 10$$

که در آن:

- مساحت جعبه مرزی عیوب $Area_i$
- مساحت کل تصویر $ImageArea$
- ضریب اهمیت هر کلاس $RiskFactor_i$

مقدار نهایی در بازه $[0, 1]$ محدود می‌شود. همچنین میانگین اطمینان مدل نیز محاسبه شده و به عنوان ورودی دوم سیستم فازی استفاده می‌شود.

۶.۱ تصمیم‌گیری فازی

سیستم فازی با دریافت دو ورودی:

- امتیاز ریسک
- میانگین اطمینان

یک خروجی در بازه $[1, -1]$ تولید می‌کند:

- مقدار منفی: کاهش سرعت



- مقدار نزدیک صفر: حفظ سرعت

- مقدار مثبت: افزایش سرعت

سرعت جدید به صورت زیر محاسبه می‌شود:

$$Speed_{new} = Speed_{current} + (\alpha \times Output_{fuzzy})$$

که در آن $\alpha = 0.1$ ضریب نرم‌سازی تغییر سرعت است.

در نهایت سرعت در بازه معجاز محدود می‌شود:

$$Speed \in [0.1, 1.0]$$

۷.۱ نمایش اطلاعات و ذخیره نتایج

در هر فریم اطلاعات زیر روی تصویر نمایش داده می‌شود:

- سرعت فعلی (با رنگ‌بندی سطح خطر)

- مقدار ریسک

- نوع تصمیم کنترلی (کاهش، حفظ یا افزایش سرعت)

ویدئوی خروجی با تمامی حاشیه‌نویسی‌ها ذخیره می‌شود.

۸.۱ تحلیل عملکرد سیستم

در پایان شبیه‌سازی، نمودار تغییرات سرعت و ریسک در طول زمان رسم می‌شود. این نمودار امکان بررسی موارد زیر را فراهم می‌کند:

- واکنش سرعت نسبت به افزایش ریسک

- میزان پایداری سیستم

- وجود یا عدم وجود نوسانات شدید

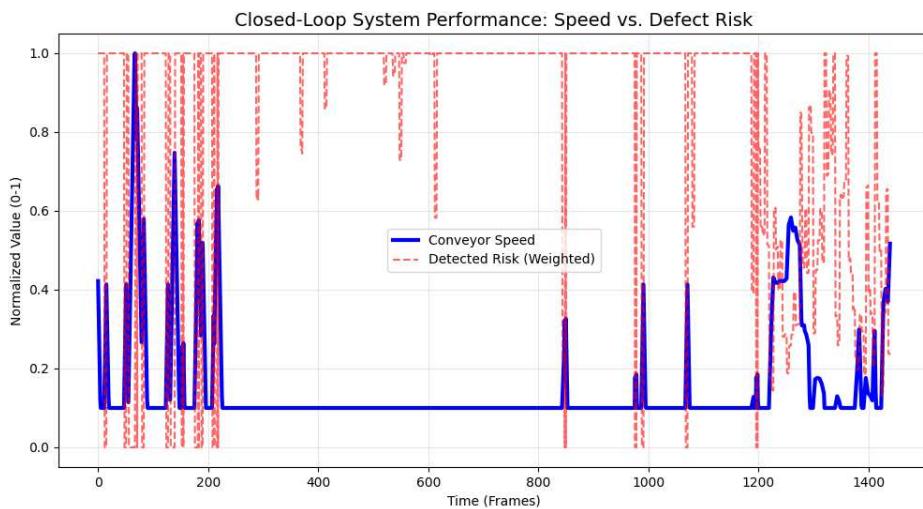
۹.۱ ساختار کلی حلقه فیدبک

ساختار سیستم به صورت زیر قابل نمایش است:

Image → YOLO → Risk Estimation → Fuzzy Controller → Speed Update → Motion Blur → Image

این ساختار یک حلقه کنترل بسته واقعی ایجاد می‌کند که در آن سرعت بر کیفیت تصویر اثر گذاشته و کیفیت تصویر نیز تصمیم‌گیری کنترلی را تحت تأثیر قرار می‌دهد.

حال نمودار تغییرات سرعت و ریسک را در طول زمان برای هر سه مدل ارائه می‌دهیم:



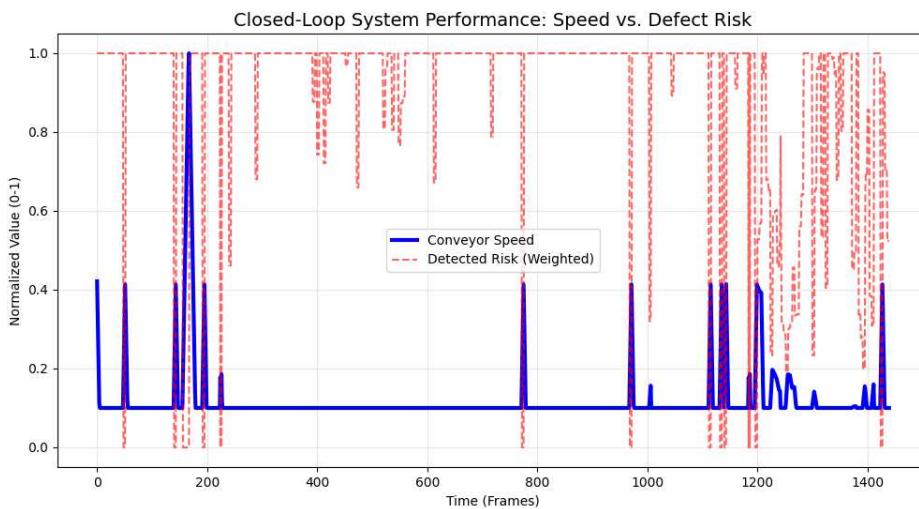
شکل ۱: نمودار تغییرات سرعت و ریسک بر حسب زمان (در طول فریم) برای خروجی مدل YOLO8n-GSE

همانطور که از نمودار مشخص است با کاهش ریسک، سرعت خط تولید بالا رفته است و با افزایش آن سرعت خط تولید پایین آمده است. در این نمودار نوسان‌های ناگهانی نیز در خط مشاهده می‌شود. این نوسان‌ها ناشی از تغییرات سرعت در طول زمان می‌باشد.



شکل ۲: نمودار تغییرات سرعت و ریسک بر حسب زمان (در طول فریم) برای خروجی مدل YOLO8n-Improvement

در این نمودار نیز همان موارد گفته شده در بالا دیده می‌شود. خیلی نمودار تغییری نکرده است اما مقداری نوسان‌های ناگهانی کمتر شده است و مقداری از تیزی نمودار کاسته شده است اما باز هم نمی‌توان گفت که تصمیمات در تغییر سرعت به صورت پیوسته هستند.



شکل ۳: نمودار تغییرات سرعت و ریسک برحسب زمان(در طول فریم) برای خروجی مدل YOLOv11m

در این نمودار نیز باز رفتار مشابه دیده می‌شود و در برخی زمان‌ها به صورت ناگهانی تصمیم به تغییر سرعت داریم. این موضوع باعث می‌شود تا خط ناپایدار شود. این رفتار به رفتار مدل فازی برمی‌گردد. چرا که هرچند ما با یکتابع گاوی سعی کردیم تا رفتار مدل را پیوسته‌تر کنیم اما باز هم این پیوستگی خودش را در طول زمان نشان نمی‌دهد و رفتار مدل فازی به صورت صفر و یکی و نوسانی می‌شود برای اینکه رفتار آن را بهتر کنیم و سرعتی پیوسته‌تر در طول خط داشته باشیم و همچنین از تجربیات محیط هم برای سرعت خط تولید استفاده نماییم، سرانجام آخر این پژوهه می‌رویم و یک مدل یادگیری تقویتی آموخته می‌دهیم.

۲ آموزش مدل RL

۱.۲ هدف طراحی عامل یادگیری تقویتی

هدف از به کارگیری Reinforcement Learning در این پژوهه، ایجاد یک مکانیزم تصمیم‌گیری تطبیقی برای تنظیم سرعت خط تولید است؛ به گونه‌ای که بین دو هدف متضاد زیر تعادل برقرار شود:

- بیشینه‌سازی بهره‌وری تولید (افزایش سرعت)

- کمینه‌سازی عبور عیوب پر ریسک (کاهش سرعت در شرایط خطر)

در این چارچوب، عامل یادگیرنده به صورت تعاملی با محیط (ویدیو شبیه‌سازی شده خط تولید) تعامل کرده و با دریافت پاداش، سیاست بهینه را می‌آموزد.

۲.۲ الگوریتم مورد استفاده: Q-Learning

روش یادگیری مورد استفاده، الگوریتم Q-Learning مبتنی بر جدول مقدار (Q-Table) است.



رابطه به روزرسانی به صورت زیر تعریف می‌شود:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') \right)$$

که در آن:

s : حالت فعلی •

a : عمل انتخاب شده •

r : پاداش دریافتی •

s' : حالت بعدی •

(Learning Rate) α : نرخ یادگیری •

(Discount Factor) γ : ضریب تنزیل •

در پیاده‌سازی انجام شده:

$$\alpha = 0.1 \quad \gamma = 0.9$$

۳.۲ تعریف فضای حالت (State Space)

حالت سیستم ترکیبی از دو مؤلفه گستته است:

۱. وضعیت پیشنهاد سیستم فازی

۲. سطح سرعت فعلی نوار نقاله

هر یک به سه سطح گستته تقسیم شده‌اند:

$$\text{State Fuzzy} \in \{\text{Brake}, \text{Hold}, \text{Accelerate}\}$$

$$\text{State Speed} \in \{\text{Low}, \text{Medium}, \text{High}\}$$

در نتیجه فضای حالت برابر است با:

$$3 \times 3 = 9 \text{ حالت}$$

که به صورت یک جدول سه‌بعدی با ابعاد (3, 3, 3) مدل شده است.



۴.۲ فضای عمل (Action Space)

عامل می‌تواند یکی از سه عمل زیر را انتخاب کند:

- کاهش سرعت: $\Delta v = -0.1$
- حفظ سرعت: $\Delta v = 0$
- افزایش سرعت: $\Delta v = +0.1$

بنابراین:

$$|A| = 3$$

سرعت در بازه زیر محدود شده است:

$$0.1 \leq v \leq 1.0$$

۵.۲ استراتژی اکتشاف و بهره‌برداری (Exploration-Exploitation)

برای جلوگیری از گیر افتادن در بهینه محلی، از سیاست ϵ -greedy استفاده شده است:

$$a = \begin{cases} \text{action random} & \text{probability with } \epsilon \\ \arg \max Q(s, a) & \text{otherwise} \end{cases}$$

پارامتر ϵ به صورت کاهشی تنظیم شده است:

$$\epsilon_{new} = \max(0.05, \epsilon \times 0.93)$$

این مکانیزم باعث می‌شود سیستم در مراحل اولیه بیشتر اکتشاف انجام دهد و در مراحل پایانی به سمت بهره‌برداری حرکت کند.

۶.۲ طراحی تابع پاداش (Reward Function)

تابع پاداش به صورت تهاجمی طراحی شده است تا رفتار صنعتی واقع‌گرایانه ایجاد کند.



مولفه‌های پاداش

۱. پاداش بهره‌وری:

$$R_{speed} = 100 \times Speed$$

۲. جریمه ریسک بالا:

$$R_{risk} = -300 \times (Risk \times Speed) \quad \text{if } Risk > 0.4$$

۳. هم‌راستایی با پیشنهاد فازی:

$$R_{fuzzy} = \begin{cases} +50 & \text{aligned if} \\ -50 & \text{otherwise} \end{cases}$$

فرم کلی:

$$R = R_{speed} + R_{risk} + R_{fuzzy}$$

۷.۲ فلسفه طراحی تابع پاداش

این تابع سه رفتار را تشویق می‌کند:

- حرکت به سمت سرعت بالا در شرایط اینمن
- کاهش سرعت هنگام افزایش ریسک
- هم‌گرایی تدریجی رفتار RL با منطق انسانی (خروجی فازی)

در واقع، سیستم فازی نقش نوعی Prior Knowledge را ایفا می‌کند و عامل RL به مرور زمان می‌آموزد چه زمانی از آن تبعیت کند یا در صورت نیاز از آن فاصله بگیرد.

۳ تحلیل رفتار و اثرگذاری عامل یادگیری تقویتی

۱.۳. ۱. پایداری یادگیری

به دلیل کوچک بودن فضای حالت (۹ حالت) و فضای عمل (۳ عمل)، الگوریتم Q-Learning هم‌گرایی سریعی دارد. کاهش تدریجی باعث ثبات سیاست نهایی شده و از نوسانات شدید جلوگیری می‌کند.



۲.۳ تحلیل تعادل سرعت و ریسک

با توجه به ساختار پاداش:

- در ریسک پایین جمله R_{speed} غالب است سیستم تمایل به افزایش سرعت دارد.
- در ریسک بالا جمله R_{risk} غالب می‌شود کاهش سرعت بهینه می‌شود.

بنابراین سیاست نهایی یک سیاست وابسته به شرایط محیطی است و رفتار ثابت ندارد.

۳.۳ نقش هم‌راستایی با فازی

مولفه R_{fuzzy} باعث می‌شود:

- در مراحل اولیه سریع‌تر همگرا شود.
- از رفتارهای بسیار پر ریسک جلوگیری کند.

در عمل، سیستم یک ساختار Hybrid Intelligent Control ایجاد می‌کند که در آن:

- نقش راهنمای اولیه را دارد.
- سیاست بهینه بلندمدت را استخراج می‌کند.

۴.۳ گذار از سیستم واکنشی به سیستم پیش‌بین

در سیستم‌های سنتی، کاهش سرعت تنها پس از مشاهده خطای انجام می‌شود. اما در این معماری، عامل با یادگیری از توالی حالات، رفتار آینده‌نگرانه اتخاذ می‌کند و پیش از بحرانی شدن وضعیت، سرعت را تنظیم می‌کند.

کد زده شده مربوط به این بخش به صورت زیر است:

```
1 import numpy as np
2 import pickle
3 import os
4 import cv2
5 import matplotlib.pyplot as plt
6 from tqdm import tqdm
7 from ultralytics import YOLO
8 import skfuzzy as fuzz
9 from skfuzzy import control as ctrl
10
11 INPUT_VIDEO = "/content/drive/MyDrive/FINAL_Project/simulated_video/raw_simulation.mp4"
```



```
۱۲ MODEL_PATH = "/content/drive/MyDrive/FINAL_Project/Training_Results/YOLO_GSE_v14/weights/best
.pt"
۱۳
۱۴
۱۵ RL_MODEL_SAVE_PATH = "/content/drive/MyDrive/FINAL_Project/RL/hybrid_rl_model.pkl"
۱۶ EPISODES = 100
۱۷
۱۸ model = YOLO(MODEL_PATH)
۱۹
۲۰
۲۱
۲۲ class HybridRLAgent:
۲۳     def __init__(self, learning_rate=0.1, discount_factor=0.9, epsilon=1.0, epsilon_decay
=0.93):
۲۴         self.q_table = np.zeros((3, 3, 3))
۲۵         self.lr = learning_rate
۲۶         self.gamma = discount_factor
۲۷         self.epsilon = epsilon
۲۸         self.epsilon_decay = epsilon_decay
۲۹
۳۰         self.actions = {0: -0.1, 1: 0.0, 2: 0.1}
۳۱
۳۲     def get_state(self, fuzzy_output, current_speed):
۳۳         if fuzzy_output < -0.1: fuzzy_state = 0
۳۴         elif fuzzy_output > 0.1: fuzzy_state = 2
۳۵         else: fuzzy_state = 1
۳۶
۳۷         if current_speed < 0.3: speed_state = 0
۳۸         elif current_speed < 0.7: speed_state = 1
۳۹         else: speed_state = 2
۴۰
۴۱         return (fuzzy_state, speed_state)
۴۲
۴۳     def choose_action(self, state):
۴۴         if np.random.rand() < self.epsilon:
۴۵             return np.random.randint(0, 3)
۴۶         return np.argmax(self.q_table[state])
۴۷
۴۸     def learn(self, state, action, reward, next_state):
۴۹         old_value = self.q_table[state][action]
۵۰         next_max = np.max(self.q_table[next_state])
۵۱         new_value = (1 - self.lr) * old_value + self.lr * (reward + self.gamma * next_max)
۵۲         self.q_table[state][action] = new_value
۵۳
۵۴     def calculate_reward_aggressive(speed, risk, action, fuzzy_suggestion):
```



```
55     reward = 0
56     reward += speed * 100
57
58     if risk > 0.4:
59         penalty = (risk * speed) * 300
60         reward -= penalty
61
62     if fuzzy_suggestion < -0.1: sugg_idx = 0
63     elif fuzzy_suggestion > 0.1: sugg_idx = 2
64     else: sugg_idx = 1
65
66     if action == sugg_idx:
67         reward += 50
68     else:
69         reward -= 50
70
71     return reward
72
73 RISK_FACTORS = {0: 1.0, 1: 0.9, 2: 0.6, 3: 0.8, 4: 0.5, 5: 0.2}
74 def calculate_weighted_risk(results, img_shape):
75     h, w = img_shape[:2]
76     total_weighted_risk = 0
77     confidences = []
78     if len(results.boxes) == 0: return 0.0, 1.0
79     for box in results.boxes:
80         x1, y1, x2, y2 = map(int, box.xyxy[0])
81         normalized_area = ((x2 - x1) * (y2 - y1)) / (h * w)
82         cls_id = int(box.cls)
83         total_weighted_risk += (normalized_area * RISK_FACTORS.get(cls_id, 0.5)) * 10
84         confidences.append(float(box.conf))
85     return min(total_weighted_risk, 1.0), sum(confidences) / len(confidences)
86
87 def apply_motion_blur(image, speed):
88     factor = 20
89     kernel_size = int(speed * factor)
90     if kernel_size < 1: return image
91     kernel_motion_blur = np.zeros((kernel_size, kernel_size))
92     kernel_motion_blur[int((kernel_size-1)/2), :] = np.ones(kernel_size)
93     kernel_motion_blur /= kernel_size
94     return cv2.filter2D(image, -1, kernel_motion_blur)
95
96 if os.path.exists(RL_MODEL_SAVE_PATH):
97     try:
98         os.remove(RL_MODEL_SAVE_PATH)
99         print("The previous version of model has been deleted.")
```



```
100     except OSError:
101         print("couldn't remove the previous model. The new model will be overwritten")
102
103 fuzzy_brain = AdvancedFuzzyController()
104 rl_agent = HybridRLAgent()
105
106 print(f"Train hybrid RL model and save in: {RL_MODEL_SAVE_PATH}")
107
108 cap_check = cv2.VideoCapture(INPUT_VIDEO)
109 total_frames = int(cap_check.get(cv2.CAP_PROP_FRAME_COUNT))
110 cap_check.release()
111
112 for episode in range(EPISODES):
113     cap = cv2.VideoCapture(INPUT_VIDEO)
114     current_speed = 0.5
115     total_reward = 0
116
117     ret, frame = cap.read()
118     if not ret: break
119
120     results = model(frame, conf=0.25, verbose=False)[0]
121     risk, conf = calculate_weighted_risk(results, frame.shape)
122     fuzzy_suggestion = fuzzy_brain.compute(risk, conf)
123     state = rl_agent.get_state(fuzzy_suggestion, current_speed)
124
125     cached_risk, cached_conf = risk, conf
126
127     for _ in range(total_frames - 1):
128         ret, next_frame = cap.read()
129         if not ret: break
130
131         action_idx = rl_agent.choose_action(state)
132         speed_change = rl_agent.actions[action_idx]
133         new_speed = np.clip(current_speed + speed_change, 0.1, 1.0)
134
135         if _ % 3 == 0:
136             blurred_frame = apply_motion_blur(next_frame, new_speed)
137             results = model(blurred_frame, conf=0.25, verbose=False)[0]
138             cached_risk, cached_conf = calculate_weighted_risk(results, next_frame.shape)
139
140         next_risk, next_conf = cached_risk, cached_conf
141
142         reward = calculate_reward_aggressive(new_speed, next_risk, action_idx,
143                                             fuzzy_suggestion)
144         total_reward += reward
```



```
۱۴۴
۱۴۵     next_fuzzy_suggestion = fuzzy_brain.compute(next_risk, next_conf)
۱۴۶     next_state = rl_agent.get_state(next_fuzzy_suggestion, new_speed)
۱۴۷     rl_agent.learn(state, action_idx, reward, next_state)
۱۴۸
۱۴۹     state = next_state
۱۵۰     fuzzy_suggestion = next_fuzzy_suggestion
۱۵۱     current_speed = new_speed
۱۵۲
۱۵۳     cap.release()
۱۵۴     rl_agent.epsilon = max(0.05, rl_agent.epsilon * rl_agent.epsilon_decay)
۱۵۵
۱۵۶     if (episode+1) % 10 == 0:
۱۵۷         print(f"Ep {episode+1}/{EPISODES} | Reward: {total_reward:.0f} | Epsilon: {rl_agent.
۱۵۸     epsilon:.2f} | End Speed: {current_speed:.2f}")
۱۵۹
۱۶۰     import shutil
۱۶۱     if not os.path.exists("/content/drive/MyDrive/FINAL_Project/RL/"):
۱۶۲         os.makedirs("/content/drive/MyDrive/FINAL_Project/RL/")
۱۶۳         print(f"Created new directory: {"/content/drive/MyDrive/FINAL_Project/RL/"})
۱۶۴     else:
۱۶۵         print(f"Directory already exists: {"/content/drive/MyDrive/FINAL_Project/RL/"})
۱۶۶
۱۶۷     with open(RL_MODEL_SAVE_PATH, 'wb') as f:
۱۶۸         pickle.dump(rl_agent.q_table, f)
۱۶۹
۱۷۰     print(f"Model saved in: \n{RL_MODEL_SAVE_PATH}")
۱۷۱     import pickle
۱۷۲     import os
۱۷۳
۱۷۴
۱۷۵     RL_MODEL_PATH = "/content/drive/MyDrive/FINAL_Project/RL/hybrid_rl_model.pkl"
۱۷۶     OUTPUT_RL_VIDEO = "/content/drive/MyDrive/FINAL_Project/RL/hybrid_rl_smart_conveyor.mp4"
۱۷۷     SAVE_PLOT_PATH = "/content/drive/MyDrive/FINAL_Project/RL/hybrid_rl_performance.png"
۱۷۸
۱۷۹     rl_agent = HybridRLAgent()
۱۸۰
۱۸۱     if os.path.exists(RL_MODEL_PATH):
۱۸۲         with open(RL_MODEL_PATH, 'rb') as f:
۱۸۳             rl_agent.q_table = pickle.load(f)
۱۸۴         print(f"RL model successfully loaded from: {RL_MODEL_PATH}")
۱۸۵     else:
۱۸۶         print("Not found.")
۱۸۷         raise FileNotFoundError("Model file not found")
۱۸۸
۱۸۹
```



```
188 rl_agent.epsilon = 0
189
190 cap = cv2.VideoCapture(INPUT_VIDEO)
191 width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
192 height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
193 out_rl = cv2.VideoWriter(OUTPUT_RL_VIDEO, cv2.VideoWriter_fourcc(*'mp4v'), 5, (width, height)
194 )
195
196 current_speed = 0.5
197 risk_log, speed_log = [], []
198
199 print("Creating Final video...")
200
201 ret, frame = cap.read()
202 if ret:
203     results = model(frame, conf=0.25, verbose=False)[0]
204     risk, conf = calculate_weighted_risk(results, frame.shape)
205
206     fuzzy_suggestion = fuzzy_brain.compute(risk, conf)
207     state = rl_agent.get_state(fuzzy_suggestion, current_speed)
208
209 print("Max value in Q-Table:", np.max(rl_agent.q_table))
210 print("Non-zero elements:", np.count_nonzero(rl_agent.q_table))
211 while True:
212     action_idx = rl_agent.choose_action(state)
213     speed_change = rl_agent.actions[action_idx]
214     if np.random.rand() < 0.05:
215         print(f"Speed: {current_speed:.2f} | Fuzzy: {fuzzy_suggestion:.2f} | RL Action: {
216             action_idx} | Q-Values: {rl_agent.q_table[state]}")
217
218     if np.all(rl_agent.q_table[state] == 0):
219         print("All values are zero. weather the model couldn't learn any thing or nothing is
220 loaded from table")
221         break
222     current_speed = np.clip(current_speed + speed_change, 0.1, 1.0)
223
224     ret, frame = cap.read()
225     if not ret: break
226
227     blurred_frame = apply_motion_blur(frame, current_speed)
228     results = model(blurred_frame, conf=0.25, verbose=False)[0]
229     risk, conf = calculate_weighted_risk(results, frame.shape)
230
231     fuzzy_suggestion = fuzzy_brain.compute(risk, conf)
232     state = rl_agent.get_state(fuzzy_suggestion, current_speed)
```

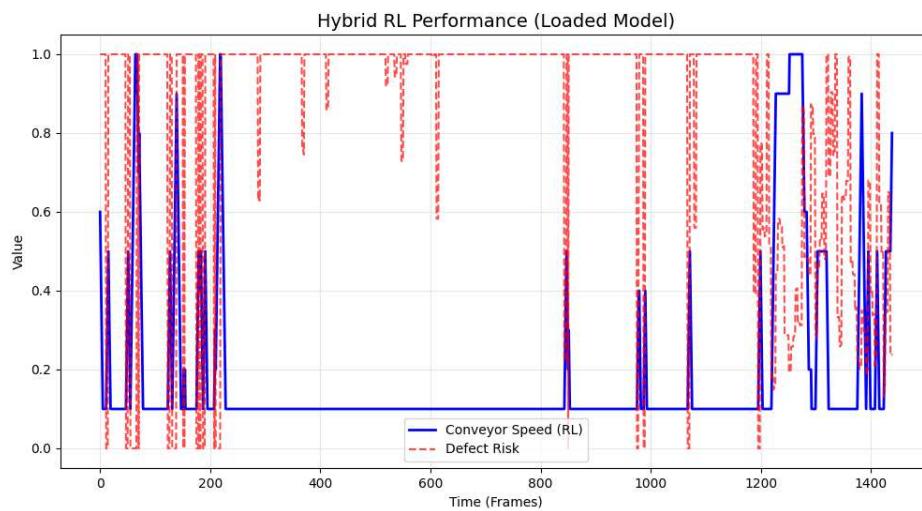


```
۲۳۰
۲۳۱     annotated = draw_pred(blurred_frame.copy(), results)
۲۳۲
۲۳۳     cv2.rectangle(annotated, (0, 0), (width, 100), (0, 0, 0), -1)
۲۳۴
۲۳۵     color_speed = (0, 255, 0) if current_speed > 0.7 else (0, 0, 255)
۲۳۶     cv2.putText(annotated, f"SPEED: {current_speed*100:.0f}%", (20, 40), cv2.
۲۳۷     FONT_HERSHEY_SIMPLEX, 0.8, color_speed, 2)
۲۳۸
۲۳۹     cv2.putText(annotated, f"RISK: {risk:.2f}", (250, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
۲۴۰     (255, 255, 255), 2)
۲۴۱
۲۴۲     fuz_text = "BRAKE" if fuzzy_suggestion < -0.1 else "ACCEL" if fuzzy_suggestion > 0.1 else
۲۴۳     "HOLD"
۲۴۴     cv2.putText(annotated, f"Fuzzy: {fuz_text}", (20, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
۲۴۵     (200, 200, 200), 1)
۲۴۶
۲۴۷     action_name = ["BRAKE", "HOLD", "ACCEL"][action_idx]
۲۴۸     color_act = (0, 255, 255) if action_name == fuz_text else (0, 0, 255)
۲۴۹     cv2.putText(annotated, f"RL Action: {action_name}", (250, 80), cv2.FONT_HERSHEY_SIMPLEX,
۲۵۰     0.8, color_act, 2)
۲۵۱
۲۵۲     out_rl.write(annotated)
۲۵۳
۲۵۴     risk_log.append(risk)
۲۵۵     speed_log.append(current_speed)
۲۵۶
۲۵۷     cap.release()
۲۵۸     out_rl.release()
۲۵۹     print(f"Final video saved in: {OUTPUT_RL_VIDEO}")
۲۶۰
۲۶۱     plt.figure(figsize=(12, 6))
۲۶۲     plt.plot(speed_log, label='Conveyor Speed (RL)', color='blue', linewidth=2)
۲۶۳     plt.plot(risk_log, label='Defect Risk', color='red', linestyle='--', alpha=0.7)
۲۶۴     plt.title("Hybrid RL Performance (Loaded Model)", fontsize=14)
۲۶۵     plt.xlabel("Time (Frames)")
۲۶۶     plt.ylabel("Value")
۲۶۷     plt.legend()
۲۶۸     plt.grid(True, alpha=0.3)
۲۶۹     plt.savefig(SAVE_PLOT_PATH)
۲۷۰     plt.show()
```

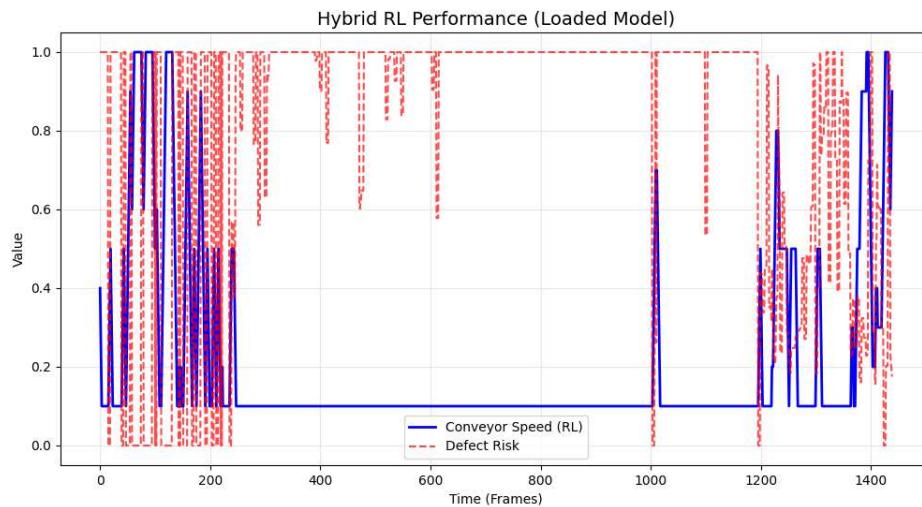
۱: کدهای مربوط به بخش یادگیری تقویتی Code

حال خروجی‌های نهایی نمودار سرعت و مقدار ریسک بر حسب زمان(فریم) ارائه می‌شوند. همانطور که در شکل‌های زیر مشخص می‌باشد، سرعت نوار نقاله پیوسته‌تر تغییر می‌کند و از آن حالت نوسانی دورتر شده است چرا که یادگیری تقویتی باعث می‌شود که تصمیمات

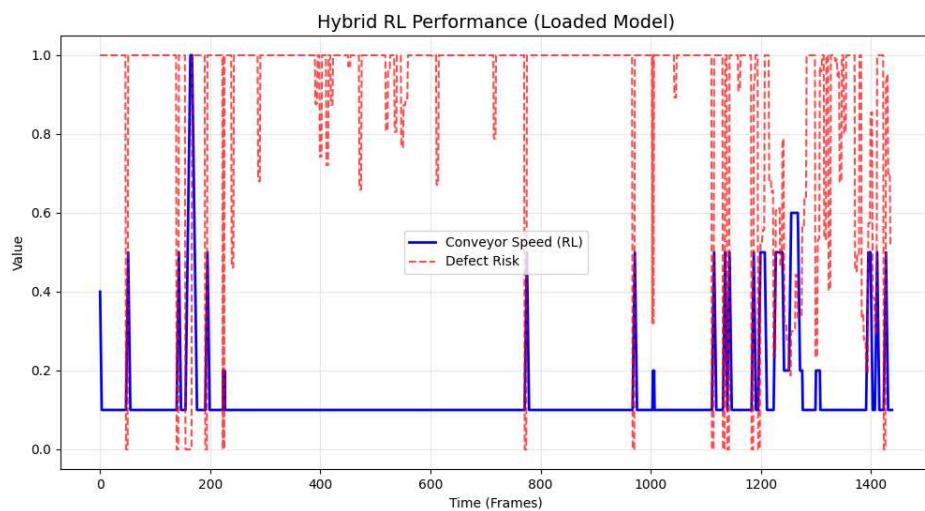
فازی در طول زمان پیوسته‌تر و نه فقط صفر و یکی شود. علاوه بر این پالس ضربه، یک تغییراتی را در بازه زمانی مشاهده نماییم.



شکل ۴: عملکرد مدل هیبریدی یادگیری تقویتی و فازی در طول زمان(فریم) برای خروجی Yolov8n-GSE



شکل ۵: عملکرد مدل هیبریدی یادگیری تقویتی و فازی در طول زمان(فریم) برای خروجی Yolov8n-Improvement



شکل ۶: عملکرد مدل هیبریدی یادگیری تقویتی و فازی در طول زمان(فریم) برای خروجی Yolov11m

همانطور که مشاهده می‌شود، در نمودارهای دوم و سوم تغییرات سرعت پیوسته‌تر است که دلیل آن هم به خاطر خروجی‌های مناسب‌تری است که مدل‌های Yolov11m و Yolov8n-Improvement فراهم می‌نمایند. در نهایت تمامی این سه ساختار به صورت سه ویدیو درآمده است که آن سه ویدیو به همراه فایل گزارش کار پژوهه ارسال گردیده است و عملکرد نهایی سیستم نیز روی این سه ویدیو قابل مقایسه می‌باشد.