

الحمد لله رب العالمين



دانشکده ریاضی و کامپیوتر خوانسار
گروه علوم کامپیوتر

پروژه پایانی رشته مهندسی کامپیوتر

عنوان

بررسی کاربرد هوش مصنوعی در شهر هوشمند و پیاده‌سازی یک کاربرد نمونه: تخمین سرعت وسایل نقلیه

استاد راهنما:

دکتر محسن کیانی

دانشجو:

مهندی سليماني

شهریور ماه 1404

اظهارنامه‌ی دانشجو

اینجانب مهدی سلیمانی دانشجوی مقطع کارشناسی رشته‌ی مهندسی کامپیوتر - نرم‌افزار دانشکده‌ی ریاضی و کامپیوتر خوانسار به شماری دانشجویی ۴۰۰۶۰۲۳۰۲۷ تعهد می‌نمایم که تحقیقات ارائه شده در این پایاننامه با عنوان برسی کاربرد هوش مصنوعی در شهر هوشمند و پیاده‌سازی یک کاربرد نمونه: تخمین سرعت وسایل نقلیه توسط شخص اینجانب انجام شده و صحت و اصالت مطالب نگارش شده مورد تأیید است؛ و در موارد استفاده از کار دیگر محققان به مرجع مورد استفاده اشاره شده است. همچنین تعهد می‌نمایم که مطالب مندرج در پایاننامه تاکنون برای دریافت هیچ نوع مدرک یا امتیازی توسط اینجانب یا فرد دیگری ارائه نشده است و در تدوین متن پایاننامه، چارچوب مصوب دانشکده را به طور کامل رعایت کرده‌ام؛ و هرگونه مقاله مستخرج از دستاوردهای این پایاننامه را با ذکر نام استاد/استاداز راهنما و دانشجو منتشر خواهم کرد. همچنین کلیه حقوق مادی و معنوی مترتب بر نتایج مطالعات، ابتکارات و نوآوری‌های ناشی از تحقیق، همچنین چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایاننامه، برای دانشکده‌ی ریاضی و کامپیوتر خوانسار محفوظ است.



مهدی سلیمانی
شهریور ۱۴۰۴

تقدیم

تقدیم به آنаз که وجودشاد تجلی عشق و ایشار است،

پدر و مادر گرانقدرم،

که راه و رسم زندگی را با مهر به من آموختند، با صبر و فداکاری، بذر دانش را در وجودم پروراندند و حمایت بیدریغ و دعای خیرشان همواره بزرگترین تکیهگاه من در تمام مراحل زندگی، بهویژه در مسیر پرچالش این پژوهش، بوده است.

و

خاله عزیزم،

که با حمایتهای بیدریغ، راهنمایی‌های ارزشمند و تشویق‌های مداومش، همواره یکی از بزرگترین مشوقاذ من در این مسیر بود و حضور دلگرم‌کننده‌اش، تواز مرا دوچندان می‌کرد.

تشکر و قدردانی

سپاس و ستایش، پروردگار یکتا را که به انساز موهبت اندیشیدن و توانایی آموختن بخشید و با چراغ دانش، مسیر تعالی و خدمت را روشن ساخت. انجام این پژوهش، سفری سرشار از آموختن و چالش بود که بدون یاری و همراهی بزرگواران بسیاری به سرانجام نمی‌رسید.

اکنون که به لطف ایزد منانه این پژوهش به پایان رسیده است، بر خود فرض می‌دانم از تمامی عزیزانی که در این راه پر فراز و نشیب، با دانش، تجربه و محبت خود، راهنمایی‌گر من بوده‌اند، صمیمانه و از عمق جاذب قدردانی نمایم.

در جایگاه نخست، مراتب سپاس و قدردانی بی‌کراز خود را از استاد راهنمای فرزانه و ارجمند، جناب آقای دکتر محسن کیانی، ابراز می‌دارم. دانش عمیق، راهنمایی‌های دقیق، نکته‌سنجدی‌های موشکافانه و حمایت‌های دلسوزانه ایشانه چراغ راه من در تمامی مراحل، از تعریف مسئله تا تدوین نهایی بود. صبر و شکریابی ایشان در پاسخ به پرسش‌هایم و وقتی که برای این پژوهش اختصاص دادند، نقشی بی‌بدیل در به ثمر رسیدن آذ داشت و بدون شک، دستیابی به این مهم بدون هدایت عالمانه ایشان میسر نمی‌گردید.

مراتب قدردانی خود را به تمامی استادی بزرگوار و کارکناد محترم دانشکده ریاضی و کامپیوتر خوانسار تقدیم می‌دارم که در طول دوران تحصیل، با ایجاد محیطی پویا و علمی، زمینه را برای رشد و بالندگی اینجانب فراهم نمودند.

در پایان عمیق‌ترین مراتب سپاس خود را به خانواده عزیزم، به ویژه پدر و مادر فداکارم و خاله عزیزم، تقدیم می‌دارم که با عشق، صبر، حمایت‌های بی‌شائبه و تحمل سختی‌های ناشی از این پژوهش، همواره بزرگترین مشوق و آرامش‌بخش دل من بوده‌اند و این موفقیت، مرهوز فداکاری‌های آنان است.

مهردادی سلیمانی

شهریور ماه ۱۴۰۴

چکیده

با توجه به نقش سرعت غیرمجاز در تصادفات جاده‌ای و نیاز شهرهای هوشمند به راهکارهای کم‌هزینه، این پژوهش سامانه‌ای سبک برای تخمین سرعت مبتنی بر بینایی ماشین ارائه می‌کند. سیستم با استفاده از YOLOv8n برای تشخیص خودرو و Norfair برای ردیابی چندشیء، زمان عبور مرکز هر خودرو از دو خط مرجع کالیبره شده را ثبت و سرعت را بر اساس فاصله واقعی میان خطوط محاسبه می‌کند.

ارزیابی روی MacBook Air M1 و ویدئوی واقعی بزرگراه نشان داد سامانه به میانگین ۱۷,۰۹ فریم بر ثانیه (FPS) دست یافت، ۳۵ خودرو را با پایداری شناسه تخمینی بیش از ۹۵٪ ردیابی کرد و توزیع دو حالت سرعت‌ها همسو با الگوهای ترافیکی مشاهده شد. محدودیت‌ها شامل اتکا به کالیبراسیون دستی، نبود داده مرجع برای Raspberry Pi 5 سنجش خطای سرعت و انجام آزمایش در شرایط نسبتاً ایده‌آل است. مسیر استقرار لبه‌ای روی با کاهش رزولوشن، کوانتیزه‌سازی مدل (Model Quantization) و بهینه‌سازی ردیابی تبیین شد. نتایج نشان می‌دهد این رویکرد می‌تواند پایه‌ای مقیاس‌پذیر و کم‌هزینه برای پایش سرعت در زیرسامانه‌های شهر هوشمند فراهم کند.

کلیدواژه: تخمین سرعت خودرو، شهر هوشمند، بینایی ماشین، یادگیری عمیق، رایانش لبه، ردیابی چندشیء، نظارت ترافیکی، YOLOv8

فهرست محتوا

صفحه

عنوان

۱	۱	۱- فصل اول: مقدمه
۱	۱	۱-۱- پیشگفتار
۱	۱	۱-۲- بیان مسئله
۲	۲	۱-۳- اهداف
۲	۲	۱-۴- چشم انداز پژوهش
۴	۴	۲- فصل دوم - مبانی پژوهش: شهر هوشمند، اینترنت اشیا و هوش مصنوعی
۴	۴	۲-۱- مقدمه
۴	۴	۲-۲- زیرساخت داده‌ای شهر هوشمند: اینترنت اشیا
۴	۴	۲-۲-۱- تعریف و مفاهیم پایه
۵	۵	۲-۲-۲- معماری IoT در شهرهای هوشمند
۵	۵	۲-۲-۳- چالش‌ها و محدودیت‌های IoT
۶	۶	۲-۳- هوش مصنوعی: موتور تحلیل و تصمیم‌گیری شهری
۶	۶	۲-۳-۱- تعریف و انواع هوش مصنوعی
۶	۶	۲-۳-۲- مزایای AI نسبت به روش‌های سنتی
۷	۷	۲-۴- همگرایی IoT و AI: شهر هوشمند
۷	۷	۲-۴-۱- تعریف شهر هوشمند
۷	۷	۲-۴-۲- اهمیت و ضرورت توسعه شهرهای هوشمند
۸	۸	۲-۴-۳- نقش IoT و AI در شهر هوشمند
۸	۸	۲-۴-۴- چالش‌های توسعه شهر هوشمند
۹	۹	۲-۴-۵- معماری کلی: از داده تا تصمیم

۱۰	۲-۴-۶- پردازش لبه در مقابل پردازش مرکزی
۱۰	۲-۵- جمع‌بندی و افق آینده
۱۱	۳- فصل سوم: مروری بر کاربردهای هوش مصنوعی در شهر هوشمند و کنترل ترافیک
۱۱	۳-۱- مقدمه
۱۲	۳-۲- کاربردهای هوش مصنوعی در شهر هوشمند
۱۲	۳-۲-۱- کاربردهای کلی هوش مصنوعی در شهرهای هوشمند
۱۲	۳-۲-۲- ایمنی و امنیت شهری
۱۲	۳-۲-۳- مدیریت انرژی و محیط‌زیست
۱۳	۳-۲-۴- مدیریت پسماند و خدمات شهری
۱۳	۳-۲-۵- خدمات حمل و نقل و شهرسازی
۱۳	۳-۲-۶- سیستم‌های کنترل هوشمند چراغهای راهنمایی
۱۳	۳-۲-۷- تحلیل و پیش‌بینی جریان ترافیک
۱۴	۳-۲-۸- تشخیص حوادث و مدیریت بحران
۱۴	۳-۲-۹- نقش بینی ماشین در پایش ترافیک
۱۵	۳-۲-۱۰- مزایای سیستم‌های مبتنی بر دوربین
۱۵	۳-۳- پیشینه پژوهش
۱۵	۳-۳-۱- مطالعه پیشین در زمینه تخمین سرعت
۱۶	۳-۳-۲- مطالعات اخیر با استفاده از YOLO
۱۷	۳-۳-۳- ردیابی چندشیعه
۱۸	۳-۳-۴- نتایج عملی تحقیقات اخیر
۱۸	۳-۳-۵- چالش‌های شناسایی شده در ادبیات
۱۹	۳-۴- جایگاه پژوهه حاضر
۱۹	۳-۵- نتیجه‌گیری فصل
۲۰	۴- فصل چهارم: ارزیابی، تحلیل نتایج و بحث تخصصی سامانه تخمین سرعت مبتنی بر YOLO و Norfair
۲۰	۴-۱- مقدمه

۲۱	۴-۲- معماری و روش‌شناسی سامانه
۲۱	۴-۲-۱- معماری خطی
۲۱	۴-۲-۲- محیط تست و مشخصات فنی
۲۲	۴-۲-۳- مجموعه داده و سناریوی تست
۲۳	۴-۳- تشریح الگوریتم و پیاده سازی
۲۳	۴-۳-۱- کالیبراسیون و تنظیمات اصلی
۲۳	۴-۳-۲- بهینه‌سازی عملکرد برای پردازش بلادرنگ
۲۵	۴-۳-۳- تشریح جریان منطقی الگوریتم
۲۶	۴-۴- نتایج و تحلیل عملکرد
۲۷	۴-۴-۱- تحلیل کمی نتایج
۲۸	۴-۴-۲- تحلیل کیفی و بصری
۳۰	۴-۵- بحث و تحلیل نهایی
۳۰	۴-۵-۱- نقاط قوت
۳۱	۴-۵-۲- محدودیت‌ها و چالش‌های واقعی
۳۲	۴-۶- مسیرهای آتی: استقرار بر روی پلتفرم‌های نهفته
۳۳	۴-۶-۱- چالش‌های فنی استقرار بر روی Raspberry pi ۵
۳۳	۴-۶-۲- راهکارهای بهینه‌سازی پیشنهادی برای پردازش به
۳۴	۴-۶-۳- انتظار عملکرد و چشم‌انداز آتی
۳۴	۴-۷- جمع‌بندی فصل و نتیجه‌گیری
۳۶	۵- فصل پنجم: نتیجه‌گیری و پیشنهادات
۳۶	۵-۱- جمع‌بندی و یافته‌های کلیدی
۳۷	۵-۲- بررسی محدودیت‌ها
۳۷	۵-۳- پیشنهادات برای پژوهش‌های آینده
۳۸	۵-۴- کاربردهای عملی و افق آینده
۳۹	۵-۵- نتیجه‌گیری

۴۰	فهرست مراجع.....
۴۳	واژه‌نامه‌ی فارسی به انگلیسی
۵۰	واژه‌نامه‌ی انگلیسی به فارسی.....
۵۸	پیوست الف - کد منبع پروژه.....

فهرست اشکال

صفحه

عنواز

۹ شکل ۱-۲- زنجیره داده از حسگر تا تصمیم در شهر هوشمند
۲۱ شکل ۱-۴- خط لوله پردازش سامانه تخمین سرعت
۲۸ شکل ۲-۴- نمونه خروجی موفق سامانه با ردیابی چندین خودرو
۲۹ شکل ۳-۴- عملکرد قابل قبول ردیاب در یک سناریوی چالش برانگیز
۳۰ شکل ۴-۴- هیستوگرام توزیع سرعت‌های تخمینی برای ۳۵ خودروی ردیابی شده

فهرست جداول

صفحه

عنوان

جدول ۱-۳- مقایسه پژوهش‌های مرتبط با تشخیص و تخمین سرعت در شهر هوشمند	۱۷
جدول ۱-۴- مشخصات مجموعه داده ویدئویی مورد استفاده	۲۲
جدول ۲-۴- نتایج کمی عملکرد سامانه	۲۷

فهرست الگوریتم‌ها

عنوان

صفحة

Algorithm4-1 Optimal recall of the YOLO model..... 24

Algorithm4-2 Main Processing loop 25

۱-فصل اول: مقدمه

۱-۱- پیشگفتار

در سالهای اخیر، رشد سریع جمعیت شهری و افزایش قابل توجه تعداد وسایل نقلیه، چالش‌های جدی در مدیریت ترافیک و ایمنی جاده‌ها ایجاد کرده است. شهرهای هوشمند^۱ به عنوان راهکاری نوین برای بهبود کیفیت زندگی و افزایش بهره‌وری منابع شهری، با بهره‌گیری از فناوری‌های نوین هوش مصنوعی (AI)^۲ و اینترنت اشیاء (IoT)^۳، در تلاشند تا مشکلات ترافیکی و ایمنی را به صورت هوشمندانه مدیریت کنند. یکی از مهم‌ترین عوامل مؤثر در کاهش تصادفات و بهبود جریان ترافیک، کنترل دقیق سرعت وسایل نقلیه است. سرعت غیرمجاز نه تنها باعث افزایش ریسک تصادفات می‌شود، بلکه بر روانی حرکت خودروها نیز تأثیر منفی می‌گذارد. بر اساس آمارهای جهانی، سالانه بیش از ۱/۱۹ میلیون نفر در حوادث رانندگی جان خود را از دست می‌دهند که بخش قابل توجهی از این حوادث ناشی از سرعت غیرمجاز است [۱]. بنابراین، توسعه سیستم‌های هوشمند برای تخمین و نظارت بر سرعت خودروها، نقش حیاتی در ارتقای ایمنی و مدیریت ترافیک در شهرهای هوشمند ایفا می‌کند. با پیشرفتهای چشمگیر در حوزه هوش مصنوعی و بینایی ماشین^۴، به ویژه در زمینه یادگیری عمیق^۵، امکان استفاده از دوربین‌های نظارتی موجود در شهرها برای تشخیص و ردیابی وسایل نقلیه فراهم شده است [۲]. مدل‌های پیشرفته‌ای مانند YOLO^۶ به دلیل دقت بالا و سرعت پردازش مناسب، امکان تخمین سرعت خودروها را به صورت بلاذرنگ و با هزینه کم فراهم می‌کنند [۳]. این فناوری‌ها می‌توانند جایگزین سیستم‌های سنتی و پرهزینه مانند رادارها و حسگرهای تعییه شده در جاده شوند و به بهبود کیفیت نظارت ترافیکی کمک کنند. با این حال چالش‌هایی نظیر کالیبراسیون^۷ دقیق دوربین‌ها، شرایط نوری متغیر و محدودیت‌های سخت‌افزاری، همچنان‌نیازمند پژوهش و توسعه بیشتر هستند.

۱-۲- بیاز مسئله

با افزایش تعداد خودروها و پیچیدگی ترافیک در شهرهای بزرگ، مدیریت جریان ترافیک و کنترل سرعت خودروها به یکی از چالش‌های اساسی تبدیل شده است. سرعت غیرمجاز و عدم نظارت دقیق بر آن علاوه بر افزایش تصادفات و تلفات جانی، باعث کاهش کیفیت زندگی شهروندان و افزایش آلودگی هوا می‌شود. سیستم‌های

^۱ Smart cities

^۲ Artificial Intelligence

^۳ Internet of Things

^۴ Computer Vision

^۵ Deep Learning

^۶ You Only Live Once

^۷ Calibration

سنتی نظارت بر سرعت، مانند رادارها و حسگرهای تعییه شده در جاده، علاوه بر هزینه بالا، محدودیت‌هایی در پوشش گستردگی و انعطاف‌پذیری دارند. از سوی دیگر، استفاده از فناوری‌های بینایی ماشین مبتنی بر دوربین‌های نظارتی موجود، می‌تواند راهکاری مقرن‌به‌صرفه و کارآمد برای تخمین سرعت خودروها باشد [۲]. با این حال پیاده‌سازی این سیستم‌ها با چالش‌هایی مانند نیاز به کالیبراسیون دقیق، شرایط نوری متغیر، پیچیدگی صحنه‌های ترافیکی و محدودیت‌های سخت‌افزاری مواجه است که دقت و پایداری سیستم را تحت تأثیر قرار می‌دهد. بنابراین، توسعه یک سامانه تخمین سرعت خودروها که بتواند در شرایط واقعی شهری و با استفاده از سخت‌افزار کم‌هزینه عملکرد قابل قبولی داشته باشد، مسئله اصلی این پژوهه است.

۱-۳ - اهداف

هدف کلی این پژوهه، طراحی و پیاده‌سازی یک سامانه هوشمند تخمین سرعت وسایل نقلیه بر پایه هوش مصنوعی و بینایی ماشین در محیط شهری است که بتواند با دقت و سرعت مناسب، سرعت خودروها را به صورت بلادرنگ تخمین بزند. اهداف جزئی پژوهه عبارتند از:

- بررسی، مقایسه و تحلیل روش‌های موجود در حوزه تخمین سرعت خودروها با استفاده از یادگیری عمیق و بینایی ماشین.
- انتخاب و بهینه‌سازی مدل‌های پیشرفته تشخیص و ردیابی خودروها، به ویژه مدل‌های خانواده YOLO (مانند YOLOv5 و YOLOv8).
- پیاده‌سازی سامانه تخمین سرعت بر روی سخت‌افزار کم‌هزینه Raspberry Pi 5 با حافظه ۸ گیگابایت به منظور کاهش هزینه‌های عملیاتی و افزایش دسترسی.
- ارزیابی عملکرد سامانه در شرایط واقعی شهری با توجه به دقت، سرعت پردازش و پایداری سیستم.
- ارائه راهکارهایی برای بهبود دقت و پایداری سیستم در مواجهه با چالش‌های محیطی مانند تغییرات نور و ترافیک سنگین.

۱-۴ - چشم‌انداز پژوهه

این پژوهش در پنج فصل اصلی و یک فصل نتیجه‌گیری تنظیم شده است که در ادامه معرفی می‌شوند:

فصل اول (مقدمه): در این فصل ابتدا مسئله پژوهه، اهمیت و ضرورت آن در بستر شهرهای هوشمند بیان می‌شود. سپس، اهداف اصلی و جزئی انجام پژوهه معرفی می‌گردد. در پایان ساختار کلی گزارش تشریح می‌شود تا خواننده دید روشنی از مسیر پژوهه به دست آورد.

فصل دوم (مبانی نظری و مفاهیم پایه): این فصل به تشریح مفاهیم بنیادین اینترنت اشیا و هوش مصنوعی و جایگاه آنها در توسعه شهر هوشمند می‌پردازد. همچنین ارتباط میاز اجزای سیستم هوشمند شامل اندازه‌گیری، پیش‌پردازش داده‌ها، انتقال به سرور و تحلیل هوش مصنوعی توضیح داده می‌شود. در بخش پایانی نیز یک نمای شماتیک از جریان داده از سنسور تا تصمیم‌گیری ارائه خواهد شد تا چارچوب مفهومی پژوهش روشن گردد.

فصل سوم (مروری بر کاربردهای هوش مصنوعی در شهر هوشمند): در این بخش مروری جامع بر مهم‌ترین کاربردهای هوش مصنوعی در حوزه‌های مختلف شهر هوشمند مانند انرژی، سلامت، محیط زیست و امنیت انجام می‌شود. تمرکز اصلی این فصل بر مدیریت و کنترل ترافیک شهری است که یکی از چالش‌های مهم شهرهای مدرن محسوب می‌شود. در پایان نیز شکاف‌های پژوهشی و زمینه‌هایی که همچنان نیازمند تحقیق هستند شناسایی و جمع‌بندی می‌گردد.

فصل چهارم (پیاده‌سازی و ارزیابی سامانه): این فصل به‌طور کامل به جنبه‌های عملی پروژه و سامانه توسعه‌یافته اختصاص دارد. ابتدا، معماری و مراحل طراحی سامانه تشریح می‌شود که شامل معرفی محیط فنی و سخت‌افزاری، الگوریتم‌های به‌کاررفته در تشخیص و ردیابی، و جزئیات فرآیند کالیبراسیون سیستم است. سپس، نتایج حاصل از پیاده‌سازی و اجرای سامانه بر روی یک سناریوی ویدئویی واقعی گزارش شده و عملکرد آن با استفاده از معیارهای کمی، بهویژه سرعت پردازش (فریم بر ثانیه (FPS)^۱، مورد ارزیابی قرار می‌گیرد. در نهایت، یک تحلیل جامع از نتایج به‌دست‌آمده، به همراه بحثی دقیق پیرامون نقاط قوت و محدودیت‌های سامانه ارائه می‌شود.

فصل پنجم (نتیجه‌گیری و پیشنهادها): در این فصل دستاوردهای پژوهش مرور می‌شود و نشان داده خواهد شد که اهداف اولیه تا چه حد تحقق یافته‌اند. سپس کاربردهای عملی سامانه در مدیریت ترافیک شهری و قابلیت‌های توسعه آن بیان می‌گردد. در پایان نیز پیشنهادهایی برای بهبود سامانه و مسیرهای پژوهشی آینده ارائه خواهد شد.

^۱ Frame Per Second

۲-فصل دوم - مبانی پژوهش: شهر هوشمند،

اینترنت اشیا و هوش مصنوعی

۱-۱- مقدمه

با رشد سریع شهرنشینی و تحولات فناوری اطلاعات، مفهوم شهر هوشمند به عنوان پاسخی به چالش‌های پیچیده شهری مطرح شده است. روند رو به رشد شهرنشینی در جهان که پیش‌بینی می‌شود تا سال ۲۰۵۰ بیش از دو سوم جمعیت کره زمین را در بر بگیرد، مدیریت شهری را با چالش‌های بی‌سابقه‌ای مواجه کرده است [۴]. این رشد شتابانه نیاز به مدیریت هوشمند منابع، خدمات و زیرساخت‌های شهری را بیش از پیش ضروری کرده است.

در این بستر، دو فناوری کلیدی نقش محوری در تحقق اهداف شهرهای هوشمند ایفا می‌کنند: اینترنت اشیاء به عنوان شبکه جمع‌آوری داده‌های شهری، و هوش مصنوعی به عنوان موتور تحلیل و تصمیم‌گیری هوشمند. ترکیب این دو فناوری، بنیان معماری شهر هوشمند را تشکیل می‌دهد و امکان نظارت، کنترل و بهینه‌سازی فرآیندهای شهری را فراهم می‌آورد.

یکی از حوزه‌های کلیدی که از این همگرایی فناورانه بهره می‌برد، مدیریت ترافیک و ایمنی جاده‌ها است. آمارهای جهانی نشان می‌دهند که سالانه بیش از ۱.۱۹ میلیون نفر در سراسر جهان بر اثر تصادفات جاده‌ای جان خود را از دست می‌دهند که بخش قابل توجهی از این حادثه‌ها پیشگیری است [۱]. در این راستا، کنترل سرعت وسایل نقلیه به عنوان یکی از عوامل کلیدی ایمنی ترافیک، نیازمند سیستم‌های نوین پایش و مدیریت است.

۱-۲- زیرساخت داده‌ای شهر هوشمند: اینترنت اشیا

۱-۲-۱- تعریف و مفاهیم پایه

اینترنت اشیاء عبارت است از شبکه‌ای از اشیاء فیزیکی مجهز به حسگرها، نرم‌افزار و سایر فناوری‌هایی که امکان اتصال و تبادل داده با سایر دستگاه‌ها و سیستم‌ها را از طریق اینترنت فراهم می‌کنند [۵]. در بستر شهری، IoT شامل طیف گسترده‌ای از دستگاه‌ها و حسگرها است که از سنسورهای ترافیکی گرفته تا کنتورهای هوشمند انرژی و سیستم‌های مدیریت زباله را در بر می‌گیرد.

مفهوم IoT در شهرهای هوشمند فراتر از صرف اتصال دستگاه‌ها است و به ایجاد یک اکوسیستم یکپارچه داده‌ای اشاره دارد که در آن اطلاعات از منابع مختلف جمع‌آوری، پردازش و برای بهبود خدمات شهری مورد استفاده قرار می‌گیرد. این رویکرد امکان مدیریت بلاذرنگ و هوشمند شهرها را فراهم می‌سازد و پایه‌ای برای تصمیم‌گیری‌های مبتنی بر داده ایجاد می‌کند.

سیستم‌های IoT معمولاً بر اساس چهار لایه اصلی طراحی می‌شود [۶]:

- ۱- لایه حسگری و جمع‌آوری داده: شامل انواع حسگرها و دستگاه‌های هوشمند نصب شده در سراسر شهر که داده‌های محیطی، ترافیکی، انرژی و سایر پارامترهای شهری را جمع‌آوری می‌کنند. این لایه به عنوان نقطه ورود اطلاعات به سیستم عمل می‌کند و کیفیت داده‌های جمع‌آوری شده تأثیر مستقیمی بر عملکرد کل سیستم دارد.

- ۲- لایه انتقال و ارتباطات: شامل شبکه‌های مخابراتی (WiFi، 4G/5G، LoRaWAN) که داده‌ها را از حسگرها به مراکز پردازش منتقل می‌کنند. انتخاب پروتکل ارتباطی مناسب بر اساس نیازهای سیستم مانند سرعت انتقال مصرف انرژی و پوشش جغرافیایی صورت می‌گیرد.

- ۳- لایه پردازش و تحلیل: شامل سیستم‌های پردازش لبه^۱ و رایانش ابری که داده‌های خام را به اطلاعات قابل استفاده تبدیل می‌کنند. این لایه شامل الگوریتم‌های پردازش داده، تحلیل آماری و روش‌های هوش مصنوعی است که برای استخراج بینش از داده‌ها استفاده می‌شوند.

- ۴- لایه کاربرد و خدمات: شامل داشبوردهای مدیریتی، اپلیکیشن‌های شهروندی و سیستم‌های تصمیم‌گیری که نتایج تحلیل‌ها را به کاربران نهایی ارائه می‌دهند. این لایه رابط بین سیستم و کاربران نهایی را تشکیل می‌دهد و نقش حیاتی در کاربردی کردن اطلاعات دارد.

۳-۲-۲- چالش‌ها و محدودیت‌های IoT

علیرغم مزایای فراوان پیاده‌سازی سیستم‌های IoT با چالش‌هایی همراه است [۷]:

- ۱- امنیت داده: حفاظت از اطلاعات حساس در برابر تهدیدات سایبری یکی از بزرگ‌ترین نگرانی‌های سیستم‌های IoT محسوب می‌شود. با افزایش تعداد دستگاه‌های متصل، سطح حمله نیز گسترش می‌یابد و نیاز به راهکارهای امنیتی جامع احساس می‌شود.

Edge Computing

-۲ مقیاس‌پذیری: مدیریت حجم عظیم داده‌های تولیدشده توسط هزاران حسگر چالشی جدی است که نیاز به زیرساخت‌های قوی پردازش و ذخیره‌سازی دارد. این چالش در شهرهای بزرگ که میلیونها نقطه داده دارند، بیشتر احساس می‌شود.

-۳ مصرف انرژی: بهینه‌سازی مصرف باتری در دستگاه‌های بی‌سیم برای حفظ پایداری سیستم ضروری است. دستگاه‌هایی که در مکانهای دسترسی دشوار قرار دارند، باید برای مدت‌های طولانی بدون تعویض باتری کار کنند.

-۴ استانداردسازی: هماهنگی بین پروتکل‌ها و پلتفرم‌های مختلف برای ایجاد یک اکوسیستم یکپارچه از چالش‌های مهم است که نیاز به همکاری بین‌المللی و توافق بر روی استانداردهای مشترک دارد.

۴-۳ - هوش مصنوعی: موتور تحلیل و تصمیم‌گیری شهری

۴-۳-۱ - تعریف و انواع هوش مصنوعی

هوش مصنوعی شاخه‌ای از علوم کامپیوتر است که به توسعه سیستم‌های می‌پردازد که قادر به انجام وظایفی هستند که معمولاً نیازمند هوش انسانی می‌باشند [۸]. در بستر شهری، AI شامل تکنیک‌ها و روش‌های متنوعی است که برای تحلیل داده‌ها، الگویابی، پیش‌بینی و تصمیم‌گیری خودکار به کار می‌روند.

• **یادگیری ماشین:** مجموعه روش‌هایی که به سیستم‌ها اجازه می‌دهد بدون برنامه‌نویسی صریح، از داده‌ها یاد بگیرند و عملکرد خود را بهبود بخشنند. شامل روش‌هایی مانند رگرسیون درختان تصمیم و الگوریتم‌های خوشه‌بندی است. این روش‌ها برای شناسایی الگوها در داده‌های شهری و پیش‌بینی رفتار سیستم‌ها استفاده می‌شوند [۸].

• **یادگیری عمیق:** زیرشاخه‌ای از یادگیری ماشین که از شبکه‌های عصبی چندلایه برای یادگیری الگوهای پیچیده استفاده می‌کند. این روش در تشخیص تصویر، پردازش زبان طبیعی و تحلیل داده‌های پیچیده بسیار مؤثر است. در حوزه شهرهای هوشمند، یادگیری عمیق برای تحلیل تصاویر دوربین‌های نظارتی، تشخیص الگوهای ترافیکی و پیش‌بینی وضعیت شهری استفاده می‌شود [۸].

۴-۳-۲ - مزایای AI نسبت به روش‌های سنتی

سیستم‌های مبتنی بر هوش مصنوعی نسبت به روش‌های سنتی مزایای قابل توجهی دارند [۸].

• **سرعت پردازش:** تحلیل حجم عظیم داده‌ها در زمان واقعی که برای سیستم‌های شهری ضروری است. سیستم‌های هوشمند قادرند در کمتر از یک ثانیه میلیونها رکورد داده را پردازش کنند.

- دقت بالا: کاهش خطای انسانی در تشخیص و تصمیم‌گیری که به ویژه در کاربردهای حیاتی مانند مدیریت ترافیک اهمیت دارد. سیستم‌های هوشمند قادرند الگوهای پیچیده‌ای را تشخیص دهند که ممکن است از دید انسان پنهان باشد.
- یادگیری مستمر: بهبود عملکرد با افزایش داده‌ها و تجربه که امکان سازگاری با شرایط متغیر شهری را فراهم می‌کند. این ویژگی به ویژه در محیط‌های پویای شهری که دائمًا در حال تغییر هستند، مزیت محسوبی است.
- مقیاس‌پذیری: قابلیت کار با حجم‌های متغیر از داده که امکان گسترش سیستم را بر اساس نیاز فراهم می‌کند. سیستم‌های هوشمند می‌توانند به راحتی با افزایش حجم داده‌ها یا تعداد حسگرها تطبیق یابند.
- در دسترس بود: عملکرد مستمر بدون نیاز به استراحت که برای سیستم‌های شهری حیاتی است. این ویژگی امکان نظارت و کنترل مستمر شهر را فراهم می‌سازد.

۲-۴-۱- همگرایی IoT و AI: شهر هوشمند

۲-۴-۱-۱- تعریف شهر هوشمند

شهر هوشمند از فضای شهری عبارت است که از فناوری‌های اطلاعات و ارتباطات^۱، بهویژه IoT و AI، برای بهبود کیفیت و عملکرد خدمات شهری، کاهش هزینه‌ها و مصرف منابع و افزایش تعامل بین شهروندان و دولت استفاده می‌کند [۶]. این تعریف فراتر از استفاده صرف از فناوری است و به ایجاد یک اکوسیستم یکپارچه و هوشمند اشاره دارد که در آن تمام اجزای شهر به هم متصل بوده و در جهت بهبود کیفیت زندگی شهروندان عمل می‌کنند.

۲-۴-۱-۲- اهمیت و ضرورت توسعه شهرهای هوشمند

افزایش جمعیت شهری و پیچیدگی چالش‌های شهری، توسعه شهرهای هوشمند از جنبه‌های مختلف ضروری است [۶]:

- از منظر اقتصادی: کاهش هزینه‌های عملیاتی، بهبود بهره‌وری و جذب سرمایه‌گذاری که به رشد اقتصادی شهرها کمک می‌کند. شهرهای هوشمند می‌توانند از طریق بهینه‌سازی مصرف منابع و خدمات، صرفه‌جویی قابل توجهی در بودجه شهری ایجاد کنند.

^۱ ICT

- از منظر زیستمحیطی: کاهش مصرف انرژی، مدیریت بهتر منابع و کاهش آلودگی که برای پایداری محیطی ضروری است. سیستم‌های هوشمند امکان نظارت دقیق بر وضعیت محیط زیست و اتخاذ تدابیر پیشگیرانه را فراهم می‌کنند.
- از منظر اجتماعی: بهبود کیفیت زندگی، افزایش ایمنی و ارتقای رفاه شهروندان که هدف اصلی مدیریت شهری است. شهرهای هوشمند امکان ارائه خدمات شخصی‌سازی شده و پاسخگویی بهتر به نیازهای شهروندان را فراهم می‌کنند.
- از منظر مدیریتی: تصمیم‌گیری مبتنی بر داده، شفافیت عملیات و پاسخگویی بهتر که کارایی مدیریت شهری را افزایش می‌دهد. مدیران شهری می‌توانند با دسترسی به اطلاعات دقیق و بلادرنگ، تصمیمات آگاهانه‌تری اتخاذ کنند.

۴-۲-۳- نقش IoT و AI در شهر هوشمند

در حالی که IoT به عنوان "حواس" شهر عمل می‌کند و اطلاعات مورد نیاز را جمع‌آوری می‌کند، AI به عنوان "مغز" شهر، این اطلاعات را پردازش و تحلیل می‌کند. این همکاری به ایجاد سیستم‌هایی منجر می‌شود که [۹]:

- خودکار و خودتنظیم هستند: قابلیت عملکرد مستقل و تطبیق با شرایط متغیر محیطی دارند
- قابلیت یادگیری و بهبود مستمر دارند: از تجربیات گذشته استفاده کرده و عملکرد خود را بهبود می‌بخشند
- پاسخ‌دهی سریع در مواجهه با تغییرات محیطی: توانایی تشخیص سریع تغییرات و واکنش مناسب به آنها
- بهینه‌سازی مستمر عملکرد بر اساس بازخورد: استفاده از داده‌های بازخورد برای بهبود مستمر کارایی سیستم

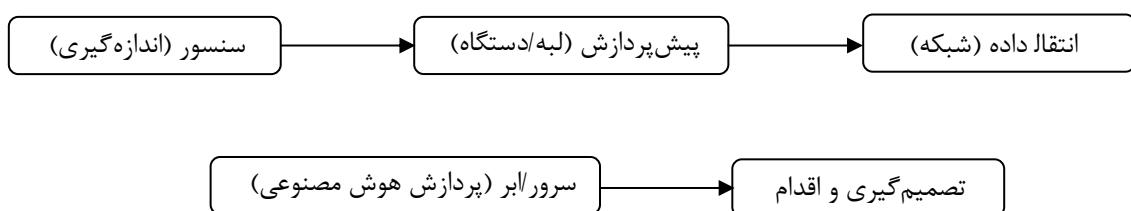
۴-۲-۴- چالش‌های توسعه شهر هوشمند

توسعه شهرهای هوشمند با چالش‌هایی مواجه است که نیاز به برنامه‌ریزی دقیق و مدیریت صحیح دارد:

- سرمایه‌گذاری اولیه بالا: هزینه نصب زیرساخت‌ها و سیستم‌های هوشمند که نیاز به توجیه اقتصادی و برنامه‌ریزی مالی مناسب دارد. این موضوع به ویژه در شهرهای با منابع محدود چالش بزرگی محسوب می‌شود [۱۰].

- **حریم خصوصی:** نگرانی شهروندان از جمع‌آوری و استفاده از داده‌های شخصی که نیاز به ایجاد چارچوب حقوقی و اخلاقی مناسب دارد. حفظ تعادل بین ارائه خدمات بهتر و حفظ حریم خصوصی از چالش‌های مهم است [۶].
- **امنیت سایبری:** محافظت از زیرساخت‌های حیاتی در برابر حملات سایبری که می‌تواند عملکرد کل شهر را مختل کند. با افزایش اتصال سیستم‌ها، آسیب‌پذیری‌های امنیتی نیز افزایش می‌یابد [۶].
- **مقاومت تغییر:** مقاومت در برابر تغییر از سوی کارکنان و شهروندان که نیاز به برنامه‌های آموزشی و فرهنگ‌سازی دارد. پذیرش فناوری‌های جدید نیاز به زمان و آموزش مناسب دارد [۱۰].
- **استانداردسازی:** نبود استانداردهای یکپارچه برای سیستم‌های مختلف که مانع تعامل سیستم‌ها می‌شود. این موضوع نیاز به همکاری بین سازندگان مختلف و تدوین استانداردهای مشترک دارد [۶].

۲-۴-۵-معماری کلی: از داده تا تصمیم
سیستم‌های هوشمند شهری عموماً بر اساس معماری سه مرحله‌ای عمل می‌کنند که داخل شکل ۲-۱ زنجیره داده آن قابل مشاهده است [۷]:



شکل ۲-۱- زنجیره داده از حسگر تا تصمیم در شهر هوشمند

مرحله اول - جمع‌آوری داده:

- **حسگرهای محیطی:** دما، رطوبت، کیفیت هوا، سطح صدا
- **دوربین‌های هوشمند:** تصاویر و ویدئوهای شهری برای تحلیل ترافیک و امنیت
- **حسگرهای حرکتی:** GPS خودروها، سنسورهای تشخیص حضور
- **داده‌های شهروندان:** اطلاعات از اپلیکیشن‌های موبایل و پلتفرم‌های اجتماعی

مرحله دوم - پردازش و تحلیل:

- **پیش‌پردازش:** تمیزکاری، فیلترینگ و آماده‌سازی داده‌های خام
- **تحلیل الگو:** شناسایی روندها و الگوهای مهم در داده‌ها

- یادگیری ماشین: استخراج دانش و قوانین از داده‌های تاریخی
- پیش‌بینی: تخمین وضعیت آینده بر اساس الگوهای شناسایی شده
- تولید بینش: تبدیل نتایج تحلیل به اطلاعات قابل فهم

مرحله سوم - تصمیم‌گیری و اقدام:

- تصمیم‌گیری خودکار: انجام اقدامات خودکار بر اساس قوانین تعریف شده
- هشدار و اطلاع‌رسانی: ارسال هشدارها به مدیران و شهروندان
- ارائه خدمات: ارائه خدمات شخصی‌سازی شده به کاربران

۴-۴-۲-پردازش لبه در مقابل پردازش مرکزی

در عمل، معماری ترکیبی^۱ که از مزایای هر دو روش بهره می‌برد، بهترین گزینه محسوب می‌شود. پردازش مرکزی^۲ مزایای منابع نامحدود، قدرت پردازشی بالا و مدیریت مرکز را دارد، اما معایب تأخیر شبکه، وابستگی به اینترنت و هزینه انتقال داده را نیز به همراه دارد. پردازش لبه مزایای تأخیر کم، کاهش ترافیک شبکه و پردازش محلی را ارائه می‌دهد، اما معایب منابع محدود، پیچیدگی مدیریت و هزینه توسعه بالا را دارد.

۵-۲-جمع‌بندی و افق آینده

همگرایی فناوری‌های اینترنت اشیاء و هوش مصنوعی، بنیاد شهرهای هوشمند قریب است و یکم را تشکیل می‌دهد. این ترکیب فناورانه، امکان مدیریت داده‌محور و هوشمند شهرها را فراهم کرده و راه را برای بهبود کیفیت زندگی شهروندان هموار می‌سازد.

در حوزه مدیریت ترافیک، که یکی از چالش‌های اصلی شهرهای بزرگ محسوب می‌شود، کاربرد این فناوری‌ها پتانسیل بسیار زیادی برای بهبود ایمنی، کاهش ترافیک و بهینه‌سازی حمل و نقل شهری دارد. در فصل بعدی، کاربردهای مشخص این فناوری‌ها در مدیریت ترافیک و سیستم‌های تخمین سرعت به تفصیل بررسی خواهد شد.

^۱ Hybrid

^۲ Cloud Computing

۳-فصل سوم: مروری بر کاربردهای هوش

مصنوعی در شهر هوشمند و کنترل ترافیک

۱-۳- مقدمه

با رشد روزافزون جمعیت شهری و پیچیدگی‌های ناشی از توسعه شهرها، مفهوم شهر هوشمند به عنوان راهکاری جامع برای مدیریت چالش‌های شهری مطرح شده است [۱۱]. با توجه به اینکه امروزه ۵۵ درصد از جمعیت جهان در شهرها ساکن هستند و این آمار تا سال ۲۰۵۰ به ۶۸ درصد خواهد رسید، مدیریت ترافیک به یکی از اصلی‌ترین دغدغه‌های شهرهای هوشمند تبدیل شده است [۴]. این رشد شتاباز شهرنشینی، چالش‌های عدیدهای در زمینه مدیریت ترافیک، انرژی، محیط زیست و خدمات شهری ایجاد کرده است.

هوش مصنوعی به عنوان یکی از فناوری‌های کلیدی قرن بیست و یکم، نقش محوری در تحقق اهداف شهر هوشمند ایفا می‌کند [۹]. این فناوری با ارائه قابلیت‌های تحلیل داده‌های حجمی، پیش‌بینی، یادگیری خودکار و تصمیم‌گیری هوشمند، امکان بهینه‌سازی فرآیندهای شهری را فراهم می‌آورد.

یکی از حوزه‌های کلیدی کاربرد هوش مصنوعی در شهرهای هوشمند، مدیریت ترافیک است. مطالعات نشان می‌دهد که تراکم ترافیک سالانه ۴۳ ساعت از زمان رانندگان آمریکایی را هدر می‌دهد و هزینه‌ای بالغ بر ۷۷۱ دلار برای هر راننده ایجاد می‌کند [۱۲]. این وضعیت در کلانشهرهای ایران به مراتب شدیدتر است؛ برای مثال براساس آمارهای داخلی، شهر ونداز در تهران سالانه در خوش‌بینانه‌ترین حالت ۷۳۰ ساعت از زمان خود را در تراکم ترافیک از دست می‌دهند که این امر هزینه‌های اقتصادی و اجتماعی هنگفتی را به شهر تحمیل می‌کند [۱۳]. در این راستا، فناوری‌های بینایی ماشین و یادگیری عمیق نقش بی‌بدیلی در توسعه سیستم‌های هوشمند پایش و کنترل ترافیک دارند.

این فصل در دو بخش اصلی سازماندهی شده است: بخش اول به بررسی کاربردهای کلی هوش مصنوعی در شهرهای هوشمند و کاربردهای خاص در مدیریت ترافیک می‌پردازد، و بخش دوم مروری جامع بر پیشینه پژوهش‌های انجام شده در زمینه تخمین سرعت با استفاده از بینایی ماشین ارائه می‌دهد.

۳-۲-۳- کاربردهای هوش مصنوعی در شهر هوشمند

۱- کاربردهای کلی هوش مصنوعی در شهرهای هوشمند

هوش مصنوعی در شهرهای هوشمند به عنوان یک دستیار هوشمند عمل می‌کند و در کنار زیرساخت‌های اینترنت اشیا و پردازش ابری یا لبه، امکان مدیریت هوشمند و بهینه منابع شهری را فراهم می‌سازد. این کاربردها را می‌توان در چند حوزه اصلی تقسیم‌بندی کرد:

۲-۳-۱- ایمنی و امنیت شهری

سیستم‌های مبتنی بر هوش مصنوعی با تحلیل تصاویر دوربین‌های نظارتی قادرند رویدادهای غیرعادی شامل تصادفات، تجمعات غیرمجاز، فعالیت‌های مشکوک و رفتارهای پرخطر را شناسایی کنند [۱۱]. این تحلیل‌ها نه تنها واکنش سریع در زمان وقوع حادثه را ممکن می‌سازد، بلکه امکان پیش‌بینی و پیشگیری از وقوع حوادث را نیز فراهم می‌آورند.

کاربردهای عملی این سیستم‌ها شامل تشخیص خودکار حرایق، شناسایی تجمعات غیرمجاز، تشخیص رفتارهای مشکوک و نظارت بر مناطق پرخطر است. این سیستم‌ها با استفاده از الگوریتم‌های یادگیری عمیق قادرند الگوهای غیرعادی را شناسایی و هشدارهای بلادرنگ ارسال کنند [۱۴].

۲-۳-۲- مدیریت انرژی و محیط‌زیست

یکی از مهم‌ترین کاربردهای هوش مصنوعی در شهرهای هوشمند، بهینه‌سازی مصرف انرژی و کاهش آثار زیست‌محیطی است [۱۴]. سیستم‌های هوشمند قادرند مصرف انرژی ساختمانها، روشنایی معابر و تأسیسات عمومی را بر اساس الگوهای استفاده و شرایط محیطی بهینه کنند. مطالعات علمی نشان می‌دهد که پیاده‌سازی سیستم‌های هوشمند انرژی می‌تواند بهبود قابل توجهی در بهره‌وری انرژی ایجاد کند [۱۱].

علاوه بر این، پایش آلودگی هوا و صوتی از طریق شبکه‌های حسگر هوشمند و تحلیل داده‌ها با الگوریتم‌های یادگیری ماشین، امکان مدیریت پویایی کیفیت هوای شهری را فراهم می‌کند. سیستم‌های مبتنی بر هوش مصنوعی تبیین‌پذیر^۱ قادرند آلودگی را در زمان واقعی پایش کرده و در صورت تجاوز از حدود مجاز، هشدارهای لازم را به همراه توضیح دلایل صادر کنند [۱۴].

^۱ Explainable AI

۴-۲-۳- مدیریت پسماند و خدمات شهری

در حوزه مدیریت پسماند، سیستم‌های هوشمند با تحلیل داده‌های حسگرهای نصب شده بر روی مخازن زباله، قادرند وضعیت پر شدن آنها را پیش‌بینی کرده و مسیرهای جمع‌آوری را بهینه سازند [۱۱]. این رویکرد علاوه بر کاهش هزینه‌های عملیاتی، از بروز مشکالت بهداشتی جلوگیری می‌کند.

سیستم‌های هوشمند مدیریت پسماند می‌توانند الگوهای تولید زباله را شناسایی کرده، زمانبندی جمع‌آوری را بهینه کنند و حتی به شهروندان توصیه‌هایی در زمینه تفکیک و کاهش زباله ارائه دهند. این رویکرد جامع منجر به بهبود قابل توجه در کیفیت خدمات شهری و کاهش آثار زیست‌محیطی می‌شود.

۴-۲-۴- خدمات حمل و نقل و شهروندی

در بخش حمل و نقل شهری و خدمات شهروندی، هوش مصنوعی امکان پیش‌بینی تقاضا، زمانبندی هوشمند ناوگان حمل و نقل عمومی، و بهبود تجربه کاربران را فراهم می‌کند. الگوریتم‌های یادگیری ماشین قادرند الگوهای رفت‌وآمد شهروندان را تحلیل کرده و با پیش‌بینی ازدحام احتمالی، برنامه‌ریزی ناوگان را بهینه کنند. این فناوری به کاهش زمان انتظار، کاهش مصرف سوخت، و افزایش رضایت شهروندان منجر می‌شود و به طور کلی کیفیت زندگی شهری را ارتقا می‌دهد. این مصاديق نشان می‌دهند که هوش مصنوعی نه تنها ابزار تحلیل داده، بلکه یک شریک تصمیم‌گیری هوشمند در کنار زیرساخت‌های اینترنت اشیا و پردازش لبه است [۱۱].

۴-۲-۵- سیستم‌های کنترل هوشمند چراغهای راهنمایی

یکی از مهم‌ترین کاربردهای هوش مصنوعی در مدیریت ترافیک، توسعه سیستم‌های کنترل تطبیقی چراغهای راهنمایی است. برخلاف سیستم‌های سنتی که بر اساس زمانبندی ثابت عمل می‌کنند، سیستم‌های هوشمند قادرند بر اساس شرایط ترافیک لحظه‌ای، زمانبندی چراغها را به صورت پویا تنظیم کنند.

مطالعات میدانی نشان می‌دهد که پیاده‌سازی سیستم SURTRAC در پیتسبورگ، زمان سفر را تا ۲۵ درصد کاهش داده است [۱۵]. سیستم Surtrac که در پیتسبورگ به کار گرفته شده، علاوه بر کاهش ۲۵ درصدی زمان سفر، زمان توقف وسایل نقلیه را ۴۰ درصد کاهش داده و در نتیجه مصرف سوخت و آلاینده‌گی را کاهش داده است [۱۵].

۴-۲-۶- تحلیل و پیش‌بینی جریان ترافیک

الگوریتم‌های یادگیری ماشین قادرند با تحلیل داده‌های تاریخی و لحظه‌ای ترافیک، الگوهای روندهای آزاد را شناسایی کرده و شرایط آینده را پیش‌بینی کنند [۱۶]. این قابلیت به برنامه‌ریزی ترافیک کمک می‌کند تا منابع را به صورت بهینه تخصیص دهد، مسیرها را بهینه‌سازی کنند و زمانبندی چراغها را تنظیم نمایند.

تحقیقات اخیر نشان می‌دهد که سیستم‌های پیش‌بینی ترافیک مبتنی بر شبکه‌های عصبی کانولوشنی و شبکه‌های عصبی بازگشتی (RNN)^۱ می‌توانند بهبود قابل توجهی در پیش‌بینی جریان ترافیک در بازه‌های زمانی کوتاه‌مدت (۱۵-۳۰ دقیقه) ارائه دهند [۱۶]. این پیشرفت‌ها امکان اتخاذ تصمیمات بلاذرنگ و مؤثر در مدیریت ترافیک را فراهم می‌سازد.

۲-۳-۱- تشخیص حوادث و مدیریت بحران

سیستم‌های مبتنی بر هوش مصنوعی قادرند حوادث ترافیکی شامل تصادفات، رانندگی در جهت مخالف، سرعت غیرمجاز یا انسداد جاده‌ها را شناسایی کنند. پس از تشخیص، این اطلاعات بلافصله به مراکز کنترل ارسال شده و امکان اعزام سریع امدادگران و انحراف ترافیک از منطقه حادثه فراهم می‌شود.

یک مطالعه موردنی شرکتی در منطقه واترلو کانادا نشان داد که سیستم تشخیص حوادث مبتنی بر بینایی ماشین توانسته در عرض چند دقیقه پس از وقوع تصادف در تقاطع شلوغ برنامه‌ریز شهری را هشدار داده و تصاویر بلاذرنگ از واقعه در اختیارشان قرار دهد [۱۷].

۲-۳-۲- نقش بینایی ماشین در پایش ترافیک

بینایی ماشین به عنوان شاخه‌ای از هوش مصنوعی، قابلیت تحلیل و درک تصاویر دیجیتال را فراهم می‌کند. در زمینه ترافیک، این فناوری امکانات زیر را ارائه می‌دهد:

- **تشخیص و شمارش وسایل نقلیه:** سیستم‌های مبتنی بر بینایی ماشین قادرند انواع وسایل نقلیه را شناسایی، آنها را در طول مسیر ردیابی کرده و حتی تخلفات ترافیکی مانند سرعت غیرمجاز یا عبور از چراغ قرمز را تشخیص دهند [۲].
- **ردیابی حرکت خودروها:** با استفاده از الگوریتم‌های ردیابی پیشرفته، سیستم‌ها می‌توانند مسیر حرکت هر خودرو را دنبال کرده و الگوهای حرکتی را تحلیل کنند [۱۸]. این اطلاعات برای بهینه‌سازی جریان ترافیک و تشخیص رفتارهای غیرعادی استفاده می‌شود.
- **تخمین سرعت و جهت حرکت:** یکی از کاربردهای کلیدی بینایی ماشین، تخمین سرعت خودروها بر اساس تحلیل حرکت آنها در فریم‌های متوالی ویدئو است [۲]. این قابلیت برای کنترل سرعت و افزایش ایمنی جاده‌ها حیاتی است.
- **تشخیص تخلفات ترافیکی:** سیستم‌ها می‌توانند تخلفات مختلفی مانند پارک غیرمجاز، تغییر خط بدو علامت، یا ورود به خط ویژه را تشخیص داده و گزارش کنند [۲].

^۱ Recurrent neural networks

- شناسایی حوادث و اختلالات: تشخیص خودکار تصادفات، خودروهای خراب، یا هرگونه اختلاط در جریان عادی ترافیک که نیاز به مداخله دارد [۲].

۱۰-۳-۲-۳-مزایای سیستم‌های مبتنی بر دوربین

سیستم‌های بینایی ماشین نسبت به روش‌های سنتی (مانند حسگرهای فیزیکی) مزایای زیر را دارند:

- مقرونبه صرفه بودن استفاده از دوربین‌های موجود و عدم نیاز به نصب حسگرهای اضافی هزینه‌ها را کاهش می‌دهد. این ویژگی به ویژه برای شهرهایی که دارای زیرساخت دوربینی هستند، بسیار مفید است.
- اطلاعات جامع: ارائه داده‌های متنوع در یک سیستم واحد که شامل تعداد خودروها، سرعت، جهت حرکت، نوع وسیله و رفتار رانندگان می‌شود. این جامعیت اطلاعات امکان تحلیل‌های پیچیده‌تر را فراهم می‌سازد.
- انعطاف‌پذیری: قابلیت تنظیم و بهبود نرمافزاری بدون نیاز به تغییرات سختافزاری که امکان بهبود مستمر سیستم را فراهم می‌کند. الگوریتم‌های یادگیری عمیق قابلیت تطبیق با شرایط جدید را دارند.
- پوشش وسیع: امکان نصب در هر نقطه شهری و پوشش مناطق وسیع با تعداد محدود دوربین. این ویژگی امکان ایجاد شبکه جامع نظارت ترافیک را با هزینه نسبتاً کم فراهم می‌سازد.

۳-۳-پیشینه پژوهش

۱-۳-۳-مطالعه پیشین در زمینه تخمین سرعت تخمین سرعت وسایل نقلیه با استفاده از بینایی ماشین در سالهای اخیر توجه زیادی را به خود جلب کرده است. Fernández Llorca و همکاران در سال ۲۰۲۱ مروری جامع بر روش‌های تخمین سرعت مبتنی بر بینایی ارائه داده‌اند و روش‌های مختلف را به سه دسته اصلی تقسیم کرده‌اند [۲]:

- روش‌های تکدوربینه: استفاده از کالیبراسیون هندسی برای تبدیل مختصات که نیاز به دانش قبلی از پارامترهای دوربین دارد. این روش‌ها ساده‌تر هستند اما دقیق‌تر آنها به کیفیت کالیبراسیون بستگی دارند.
- روش‌های چنددوربینه: استفاده از تکنیک‌های استریو که امکان محاسبه عمق و بازسازی سه‌بعدی صحنه را فراهم می‌کند. این روش‌ها دقیق‌تر اما پیچیدگی بیشتری نیز به همراه دارند.

- روش‌های ترکیبی: ترکیب بینایی با سایر حسگرهای جامع‌ترین اطلاعات را ارائه می‌دهد اما هزینه بیشتری دارد.

۳-۳-۲- مطالعات اخیر با استفاده از YOLO

خانواده مدل‌های YOLO به دلیل توازن مناسب بین دقّت و سرعت، یکی از محبوب‌ترین روش‌های تشخیص شیء در کاربردهای ترافیکی محسوب می‌شوند [۱۹]. این مدل‌ها قادرند در یک پاس واحد، موقعیت و کلاس اشیاء مختلف را تشخیص دهند که آنها را برای پردازش بلادرنگ مناسب می‌سازد.

نسل‌های مختلف YOLO عملکرد متفاوتی ارائه می‌دهند [۲۰، ۱۹]:

- **YOLOv3**: دقّت قابل قبول با سرعت پردازش متوسط که برای کاربردهای عمومی مناسب است.
- **YOLOv5**: بهبود دقّت و کاهش حجم مدل که امکان اجرا بر روی دستگاه‌های محدود را فراهم می‌کند.
- **YOLOv8**: جدیدترین نسل با بالاترین دقّت و سرعت که مطالعات اخیر نشان داده‌اند مدل YOLOv8 در تشخیص وسایل نقلیه در شرایط نوری مختلف دقّت قابل توجهی دارد.

برای درک بهتر تفاوت عملکرد نسل‌های مختلف YOLO و مطالعات مرتبط در زمینه‌ی تشخیص وسایل نقلیه و تخمین سرعت، در جدول ۳-۱ خلاصه‌ای از پژوهش‌های منتخب ارائه شده است.

جدول ۱-۳- مقایسه پژوهش‌های مرتبط با تشخیص و تخمین سرعت در شهر هوشمند

سال	داده/محیط	روش(ردهیاب/YOLO)	نیاز به کالیبراسیون	معیارها (mAP/MAE/FPS)	سخت افزار	نتیجه کلیدی/محدودیت
2020	ویدئو دوربین ترافیکی در محیط شهری	YOLOv3 + SORT	بله(اندازه‌گیری فاصله مرجع)	mAP = 71%, FPS ≈ 25	دسکتاپ (GTX 1080)	دققت مناسب در روز؛ عملکرد ضعیف در شب
2021	دیتاست جاده‌ای عمومی	YOLOv4 + DeepSORT	خیر	mAP = 78%, FPS ≈ 30	서ور GPU	پایدار در نورهای مختلف؛ ولی نیازمند سخت‌افزار قدرتمند
2022	ویدئوهای ضبط شده توسط دوربین ثابت	YOLOv5s + Norfair	بله(نقطه مرجع جاده)	MAE ≈ 3.2 km/h, FPS ≈ 20	لپ‌تاپ CPU+GPU (میانرده)	قابل اجرا روی سیستم ارزان دقت وابسته به زاویه نسب
2023	محیط واقعی چهارراه شهری	YOLOv7 + DeepSORT	خیر	mAP = 84%, MAE ≈ 2.5 km/h	GPU RTX 3060	کارایی بالا در شرایط ترافیک سنگین؛ نیازمند برقة و شبکه پایدار
2024	Raspberry Pi + دوربین کم هزینه	YOLOv8n + Norfair	بله(فاصله کالیبره شده ۱۵m)	MAE ≈ 2.8 km/h, FPS ≈ 12	Raspberry Pi 5	امکان استقرار در لبه؛ محدودیت FPS نسبت به GPU

همانطور که جدول ۱-۳ نشان می‌دهد، نسل‌های جدید YOLOv7 و YOLOv8 (مانند YOLOv7 + DeepSORT و YOLOv8n + Norfair) دقق بالاتری در شرایط مختلف محیطی دارند [۲۰، ۲۱، ۲۲]، اما در عین حال محدودیت منابع محاسباتی و نیاز به سخت‌افزار قدرتمند، بهویژه در استقرارهای لبه‌ای، همچنان‌یک چالش اساسی محسوب می‌شود [۲۳]. پژوهش حاضر با استفاده از Raspberry Pi 5 بر روی YOLOv8n تلاش کرده است تا بخشی از این خلا را پوشش دهد.

۳-۳-۳- ردهیابی چندشیء^۱

برای تخمین دقیق سرعت وسائل نقلیه، ردهیابی پیوسته آنها در طول فریم‌های متوالی ضروری است [۱۸]. الگوریتم‌های ردهیابی چندشیء مانند SORT، DeepSORT و Norfair برای این منظور توسعه یافته‌اند.

Norfair به عنوان یک کتابخانه سبک و کارآمد برای ردهیابی بلادرنگ دو بعدی، ویژگی‌های زیر را ارائه می‌دهد

: [۱۸]

- پردازش سریع و مصرف منابع کم که برای سیستم‌های بلادرنگ ضروری است
- قابلیت تنظیم توابع فاصله بر اساس نیازهای خاص کاربرد
- مقاومت در برابر اختلالات موقت که در محیط‌های واقعی رایج است

^۱ Multi-Object Tracking

- پشتیبانی از ردیابی چندبعدی برای کاربردهای پیشرفته

۴-۳-۳-نتایج عملی تحقیقات اخیر

Shaqib و همکاران در سال ۲۰۲۴ سیستمی مبتنی بر YOLOv8 برای محیطهای شهری پرترافیک پیاده‌سازی کرده‌اند [۲۱]. نتایج آنها حکایت از دقت عملیاتی بیش از ۹۲٪ و خطای متوسط km/h ۳.۵ دارد. این نتایج قابلیت اطمینان روش‌های بینایی ماشین را برای نظارت شهری تایید می‌کند.

Macko و همکاران در سال ۲۰۲۵ با تمرکز بر روی بهینه‌سازی محاسباتی برای سخت‌افزارهای محدود، نشان دادند که با جایگزینی شناساگرهای سنگین‌تر با معماری‌های کارآمدتری چون YOLOv6، می‌توان به افزایش سرعت چشمگیری دست یافت بدون آنکه دقت نهایی تخمین سرعت کاهش یابد [۲۳]. این پژوهش اهمیت انتخاب مدل‌های سبک‌تر برای کاربردهای بلاذرنگ روی دستگاه‌های لبه^۱ را برجسته می‌کند.

Liu و همکاران در سال ۲۰۲۱ یک سیستم دو لایه پردازش لبه برای پایش ترافیک هوشمند ارائه داده‌اند که قادر است تشخیص ازدحام و تخمین سرعت را همزمان انجام دهد [۲۲]. این سیستم با استفاده از معماری پردازش لبه، تأخیر شبکه را کاهش داده و کارایی کلی را بهبود می‌بخشد.

۵-۳-چالش‌های شناسایی شده در ادبیات

مطالعه ادبیات موجود چندین چالش اصلی در زمینه تخمین سرعت مبتنی بر بینایی آشکار می‌سازد:

- کالیبراسیون دوربین: دقت تخمین سرعت مستقیماً وابسته به صحت کالیبراسیون است. یکی از مهم‌ترین شکاف‌های موجود، فقدان روش‌های کالیبراسیون خودکار و قابل اعتماد است. اکثر سیستم‌های موجود نیازمند کالیبراسیون دستی هستند که زمانبر و مستعد خطا است [۲۴].
- شرایط نوری متغیر: تأثیر نور، سایه و شرایط جوی بر عملکرد سیستم که نیاز به الگوریتم‌های مقاوم‌تر و داده‌های آموزشی متنوع‌تر احساس می‌شود. سیستم‌های موجود در شرایط نوری نامطلوب، باران شدید و تراکم بالا عملکرد کاهش‌یافته‌ای دارند.
- پیچیدگی صحنه: مسائل ناشی از تراکم ترافیک و پوشیدگی اجسام که چالش‌های جدی برای الگوریتم‌های تشخیص ایجاد می‌کند. این موضوع به ویژه در ساعات پیک و مکانهای پرترافیک مشکل‌ساز است.

^۱ Edge devices

- محدودیت‌های محاسباتی: نیاز به تعادل بین دقت و سرعت پردازش که برای کاربردهای بلادرنگ ضروری است. با وجود پیشرفت‌های قابل توجه در زمینه مدل‌های یادگیری عمیق، پیاده‌سازی آنها روی سخت‌افزارهای محدود مانند Raspberry Pi همچنان چالش‌برانگیز است.
- اعتبارسنجی بلندمدت: فقدان مطالعات اعتبارسنجی بلندمدت در شرایط عملیاتی مختلف، یکی از محدودیت‌های اصلی ادبیات موجود است.

۴-۳-جایگاه پروژه حاضر

پروژه حاضر با هدف پر کردن برخی از شکاف‌های شناسایی شده، بر روی موارد مهم آذ متمرکز است . از توسعه سیستم کاملاً عملی با استفاده از فناوری‌های بهروز که قابلیت استقرار در محیط‌های واقعی را داشته باشد. تمرکز بر پیاده‌سازی روی 5 Raspberry Pi به عنوان پلتفرم اقتصادی که امکان استفاده گسترده در پروژه‌های شهری را فراهم می‌کند. تست سیستم در شرایط محیطی و ترافیکی مختلف برای اطمینان از کارکرد مناسب در شرایط واقعی. بررسی عملکرد نسل‌های مختلف YOLO در کاربرد تخمین سرعت که می‌تواند راهنمای مناسبی برای انتخاب مدل بهینه باشد.

۵-۳-نتیجه‌گیری فصل

مرور ادبیات نشان می‌دهد که هوش مصنوعی و بینایی ماشین نقش محوری در تحول مدیریت ترافیک شهری ایفا می‌کنند. کاربردهای متنوع این فناوری‌ها، از کنترل چراغهای راهنمایی تا تشخیص حوادث و تخمین سرعت، پتانسیل عظیمی برای بهبود کیفیت زندگی شهری دارند.

با وجود پیشرفت‌های قابل توجه، همچنان چالش‌هایی در زمینه کالیبراسیون تاب‌آوری و پیاده‌سازی عملی وجود دارد. تحقیق حاضر با تمرکز بر این چالش‌ها و ارائه حل‌راه عملی، سهم مفیدی در توسعه سیستم‌های مدیریت ترافیک هوشمند خواهد داشت.

۴- فصل چهارم: ارزیابی، تحلیل نتایج و بحث

تخصصی سامانه تخمین سرعت مبتنی بر

Norfair و YOLO

۱- مقدمه

پس از بررسی مبانی نظری و مرور پژوهش‌های مرتبط در فصول گذشته، این فصل به طور کامل به جنبه عملیاتی و هسته اصلی این پژوهه، یعنی طراحی، پیاده‌سازی و ارزیابی سامانه هوشمند تخمین سرعت وسایل نقلیه، اختصاص دارد. حرکت از حوزه تئوری به پیاده‌سازی عملی، گامی حیاتی است که نه تنها امکان‌سنجی ایده‌های مطرح شده را به اثبات می‌رساند، بلکه چالش‌ها و ظرفات‌های دنیای واقعی را نیز آشکار می‌سازد. هدف از این مطالعه موردنی، ارائه یک راهکار عملی، کم‌هزینه و مبتنی بر بینایی ماشین به عنوان یک اثبات مفهوم^۱ است که بتواند فرآیند تشخیص، ردیابی و محاسبه سرعت^۲ خودروها را به صورت خودکار و با حداقل نیاز به زیرساخت‌های گرانقیمت انجام دهد.

در این فصل، ابتدا معماری کلی سامانه و محیط پیاده‌سازی آذ با جزئیات فنی تشریح می‌شود. سپس، الگوریتم پیاده‌سازی شده به صورت گام‌به‌گام و با ارائه شبه‌کد و قطعه‌های کلیدی کد منبع، به تفصیل مورد بررسی قرار گرفته و پارامترهای مهم و تصمیمات طراحی تحلیل می‌شوند. در ادامه، نتایج عملکرد سامانه از نظر کمی و کیفی ارائه خواهد شد. در نهایت، این فصل با یک بحث جامع و نقادانه پیرامون نقاط قوت، ضعف‌ها و محدودیت‌های پژوهش حاضر و مسیرهای آتی به پایان می‌رسد.

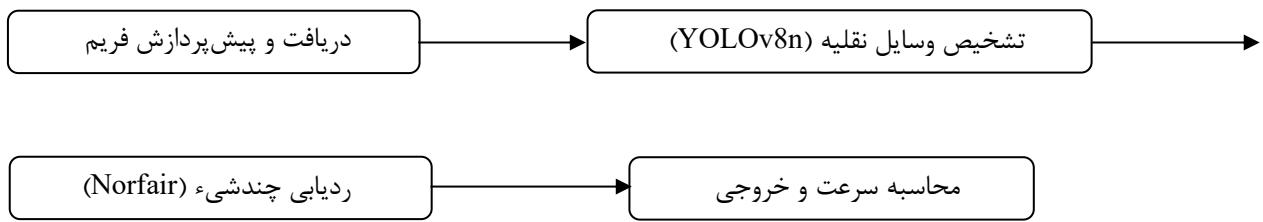
^۱ Proof of Concept

^۲ Speed Estimation

۴-۲-۱- معماری و روش‌شناسی سامانه

۴-۲-۱-۱- معماری خطی^۱

سامانه پیشنهادی بر اساس یک معماری خطی چهار مرحله‌ای طراحی شده است که در شکل ۴-۱ نشان داده شده است. این معماری مازولار، امکان توسعه و بهینه‌سازی هر بخش را به صورت مستقل فراهم می‌آورد.



شکل ۱-۴- خط لوله پردازش سامانه تخمین سرعت

- **دریافت و پیش‌پردازش فریم:** فریم‌ها از منبع ویدئویی خوانده شده و به ابعاد ثابت (480×270 پیکسل) تغییر اندازه می‌دهند تا ورودی یکپارچه‌ای برای شبکه عصبی فراهم شده و باز محاسباتی کاهش یابد.
- **تشخیص وسیله نقلیه^۲:** با استفاده از مدل YOLOv8n، قادر محاطی^۳ خودروها استخراج می‌شود.
- **ردیابی چندشیء:** الگوریتم Norfair به هر خودرو یک شناسه بکتا اختصاص می‌دهد و آن را در فریم‌های متوالی دنبال می‌کند.
- **محاسبه سرعت:** با ثبت زمان عبور هر خودرو از دو خط مرجع مجازی، سرعت متوسط آن محاسبه و نمایش داده می‌شود.

۴-۲-۲- محیط تست و مشخصات فنی

فرآیند تست و ارزیابی بر روی یک پلتفرم استاندارد و در دسترس انجام شد تا نتایج قابل بازتولید باشند:

- سخت‌افزار: لپ‌تاپ MacBook Air مجهز به پردازنده Apple M1
- نرم‌افزار: زبان پایتون (Python 3.10)، کتابخانه‌های Ultralytics، OpenCV، PyTorch
- Norfair

^۱ Pipeline

^۲ Object Detection

^۳ Bounding Box

- **شتاپدهنده سختافزاری:** اجرای مدل یادگیری عمیق بر روی GPU پردازنده اپل با استفاده از زیرسیستم torch.backends.mps انجام شد. این انتخاب به طور قابل توجهی سرعت پردازش را نسبت به اجرای صرف بر روی CPU افزایش می‌دهد. در صورت عدم دسترسی به این زیرسیستم، کد به طور خودکار به CPU یا CUDA بازمی‌گردد.

۴-۲-۳- مجموعه داده و سناریوی تست

به منظور ارزیابی دقیق و عینی عملکرد سامانه، انتخاب یک مجموعه داده‌ی ویدئویی مناسب که بتواند نمایانگر شرایط دنیای واقعی باشد، گامی حیاتی است. کیفیت و ویژگی‌های داده ورودی، تأثیر مستقیمی بر نتایج ارزیابی داشته و به درک بهتر قابلیت‌ها و محدودیت‌های سیستم کمک می‌کند. برای این پژوهش، به جای استفاده از داده‌های شبیه‌سازی‌شده، یک کلیپ ویدئویی واقعی از ترافیک بزرگراه شهری انتخاب شد تا چالش‌های عملیاتی بهتر سنجیده شوند. این انتخاب هدفمند بود؛ سناریوی بزرگراه با ترافیک روان یک محیط کنترل شده اما واقعی را فراهم می‌کند که در آن می‌توان عملکرد پایه‌ای الگوریتم‌های تشخیص و ردیابی را بدون مواجهه با پیچیدگی‌های شدید مانند تقاطع‌های شلوغ یا حضور عابران پیاده، ارزیابی کرد. مشخصات دقیق این کلیپ ویدئویی در جدول ۴-۱ به تفصیل شرح داده شده است.

جدول ۴-۱- مشخصات مجموعه داده ویدئویی مورد استفاده

مشخصه	مقدار
سناریو	بزرگراه شهری
شرایط نوری	روز، آفتابی
وضعیت ترافیک	رواز و نیمه‌سنگین
زاویه دوربین	از بالا (پل عابر پیاده)
مدت زمان	۲۴ ثانیه
نرخ فریم	۳۰ فریم بر ثانیه
رزولوشن اصلی	480×640
روش تعیین داده مرجع	در دسترس نیست

۴-۳- تشریح الگوریتم و پیاده سازی

در این بخش، جنبه های کلیدی پیاده سازی الگوریتم مورد بررسی قرار می گیرد. کد کامل برنامه به دلیل طولانی بودن در پیوست الف ارائه شده است.

۱- ۴-۳- کالیبراسیون و تنظیمات اصلی

یکی از حساس ترین مراحل در پیاده سازی این سامانه، کالیبراسیون فاصله است که مستقیماً بر دقت سرعت های خروجی تأثیر می گذارد. در یک سناریوی واقعی، این مقدار باید با ابزارهای دقیق میدانی یا نقشه برداری اندازه گیری شود. از آنجایی که این پروژه به عنوان یک اثبات مفهوم و با استفاده از ویدئویی از پیش ضبط شده انجام شده، امکان اندازه گیری دقیق میدانی فراهم نبود. بنابراین، یک مقدار فرضی اما واقع بینانه برابر با ۱۵ متر به عنوان فاصله کالیبره شده^۱ در نظر گرفته شد. این مقدار بر اساس تخمین بصری از عرض یک بزرگراه استاندارد انتخاب شده و هدف آن نمایش عملکرد الگوریتم در تبدیل داده های پیکسلی به واحدهای واقعی (متر بر ثانیه) است. لازم به ذکر است که در صورت استقرار عملیاتی سامانه، این پارامتر باید به دقت کالیبره شود.

منطق پیاده سازی شده بر اساس تشخیص عبور مرکز خودرو از بالا به پایین تصویر طراحی شده است. این رویکرد برای تحلیل جریان ترافیک در یک جهت مناسب است. برای کاربردهای پیچیده تر مانند تحلیل ترافیک دو طرفه، لازم است یک جفت خط مرجع و منطق عبور معکوس (پایین به بالا) نیز به صورت جداگانه به سیستم اضافه گردد تا هر دو مسیر حرکتی پوشش داده شوند.

۲- ۴-۳- بهینه سازی عملکرد برای پردازش بلاذرنگ

برای اینکه یک سامانه بینایی ماشین بتواند در کاربردهای عملی مانند نظارت ترافیکی مؤثر باشد، دستیابی به نرخ پردازش فریم بالا و پایدار امری حیاتی است. در یک خط لوله پردازشی، گلوگاه ها^۲ می توانند عملکرد کل سیستم را محدود کنند. در این پروژه، دو گلوگاه اصلی شناسایی و با استفاده از تکنیک های مهندسی نرم افزار هدف قرار گرفتند: گلوگاه ورودی / خروجی و سربار محاسباتی^۳ پیش پردازش.

۱- پردازش موازی ورودی / خروجی برای رفع گلوگاه I/O:

عملیات خواندن داده از یک منبع ذخیره سازی (مانند هارد دیسک) ذاتاً یک فرآیند کند و ناپایدار در مقایسه با سرعت محاسباتی پردازنده ها است. در یک پیاده سازی ساده و تکریسه، حلقه اصلی پردازش مجبور است در هر

^۱ CALIBRATED_DISTANCE

^۲ Bottlenecks

^۳ Computational Overhead

تکرار، منتظر بماند تا فریم بعدی از ویدئو خوانده و بارگذاری شود. این زمان انتظار، که به آن توقف^۱ گفته می‌شود، باعث هدر رفتن ظرفیت پردازشی GPU/CPU و کاهش شدید و نوسانی FPS می‌شود.

برای حل این مشکل، از الگوی طراحی تولیدکننده-صرفکننده^۲ استفاده شد. در این معماری، یک نخ^۳ مجزا (تولیدکننده) وظیفه خواندن فریم‌ها از ویدئو و قرار دادن آنها در یک صفحه^۴ بافر را بر عهده دارد، در حالی که حلقه اصلی پردازش (صرفکننده)، به جای خواندن مستقیم از دیسک، فریم‌ها را از این صفحه سریع برمی‌دارد. این رویکرد، عملیات کند I/O را از مسیر بحرانی پردازش جدا^۵ می‌کند و تضمین می‌کند که حلقه اصلی تقریباً همیشه یک فریم آماده برای پردازش در اختیار دارد، که منجر به استفاده بهینه از منابع سخت‌افزاری و دستیابی به یک نرخ پردازش بسیار پایدارتر و بالاتر می‌شود.

۲ - پردازش با اندازه ورودی مشخص برای کاهش سربار محاسباتی:

مدلهای YOLO برای پردازش تصاویر با ابعاد مشخصی (مانند ۶۴۰x۶۴۰) آموزش دیده‌اند. کتابخانه Ultralytics به طور خودکار هر فریم ورودی را به این اندازه استاندارد تبدیل می‌کند که شامل تغییر اندازه و اغلب لتریکسینگ^۶ برای حفظ نسبت ابعاد است. این پیش‌پردازش خودکار، اگرچه کار را ساده می‌کند، اما برای هر فریم تکرار شده و یک سربار محاسباتی غیرضروری به فرآیند استنتاج اضافه می‌کند.

برای حذف این سربار، یک رویکرد دو مرحله‌ای اتخاذ شد: ابتدا، فریم ویدئو به صورت دستی و تنها یک بار در ریسه تولیدکننده به ابعاد هدف (۴۸۰x۲۷۰) تغییر اندازه داده می‌شود. سپس، با ارسال پارامتر imgsz=(TARGET_HEIGHT, TARGET_WIDTH) در هنگام فراخوانی مدل به موتور YOLO صراحتاً اعلام می‌شود که ورودی از قبل آماده است و نیازی به هیچ‌گونه تغییر اندازه یا پیش‌پردازش داخلی نیست. این تکنیک ساده اما مؤثر، با حذف یک مرحله تکراری از حلقه اصلی، به کاهش زمان پردازش هر فریم و افزایش توازن عملیاتی^۷ کل سامانه کمک شایانی می‌کند.

Algorithm ۴ - ۱ Optimal recall of the YOLO model

```

1: results = model(
2:     display_frame,
3:     conf=CONF_THRESHOLD,
4:     classes=ALLOWED_CLASSES,
5:     imgsz=(TARGET_HEIGHT, TARGET_WIDTH),

```

^۱ Stalling

^۲ Producer-Consumer

^۳ Thread

^۴ Queue

^۵ Decouple

^۶ Letterboxing

^۷ Throughput

```
6:    verbose=False  
7: )
```

۴-۳-۴- تشریح جریان منطقی الگوریتم

برای درک دقیق‌تر جریان منطقی و ساختار محاسباتی سامانه، روند کلی الگوریتم در قالب یک شبکه کد ساختاریافته در ادامه ارائه می‌شود. این شبکه کد، که نمایشی سطح بالا و مستقل از زبان برنامه‌نویسی است، تعامل بین مازولهای اصلی سیستم -از مقداردهی اولیه تا پردازش هر فریم و محاسبه نهایی سرعت- را به وضوح نشان می‌دهد. هر گام از این شبکه کد نماینده یک بخش کلیدی از کد منبع است که در ادامه به تفصیل تحلیل می‌شود.

Algorithm ۴ - ۲ Main Processing loop

```
8: VIDEO_Initialize YOLO model and Norfair tracker  
9: Start frame reader thread and initialize frame queue  
10: For each frame f from queue:  
11:   Detect vehicles in f using YOLOv8 to get detections list D  
12:   Update Norfair tracker with D to get tracked objects list T  
13:   For each object t in T:  
14:     If t is a confirmed track (hit_counter > MAX):  
15:       Check if t crosses reference lines L1 and L2  
16:     If both lines are crossed:  
17:       Calculate speed based on time delta and calibrated distance  
18:     Draw bounding box and display info for t on frame f  
19: Display final frame
```

- گام‌های ۱ و ۲ (مقداردهی اولیه و آماده‌سازی): این مرحله زیربنای کل سیستم است. ابتدا، مدل YOLOv8n با رگذاری شده و به دستگاه پردازشی مناسب (MPS در این مورد) منتقل می‌شود. همزمان ردیاب Norfair با پارامترهای کلیدی مانند `distance_threshold` (حداکثر فاصله مجاز برای تطبیق یک تشخیص با یک ترک موجود) و `hit_counter_max` (تعداد فریم‌های لازم برای تأیید نهایی یک ترک) پیکربندی می‌شود. در نهایت، ریسه موازی برای خواندن فریم‌ها آغاز به کار می‌کند تا حلقه اصلی دچار وقفه نشود.

- گام‌های ۳ تا ۵ (حلقه پردازش اصلی): این حلقه قلب تپنده سامانه است که به صورت بلاذرنگ عمل می‌کند. در هر تکرار، یک فریم از صف برداشته می‌شود. خروجی مدل YOLO لیستی از تشخیص‌های خام

(کادرهای محاطی) است. این لیست به متد update ردياب Norfair با مقایسه موقعیت تشخیص‌های جدید با موقعیت پیش‌بینی شده ترکهای قبلی، تصمیم می‌گیرد که کدام تشخیص را به کدام ترک اختصاص دهد، کدام ترکها را حذف کند و کدام تشخیص‌های جدید را به عنوان ترکهای بالقوه ثبت نماید.

- گام‌های ۶ تا ۱۱ (منطق اعتبارسنجی و محاسبه سرعت): این بخش، منطق کاربردی^۱ سامانه را پیاده‌سازی می‌کند.
- اعتبارسنجی ترک (گام ۷): شرط $\text{hit_counter} > \text{MAX}$ یک فیلتر زمانی حیاتی است. این شرط از پردازش ترکهای ناپایدار که ممکن است ناشی از تشخیص‌های کاذب و لحظه‌ای باشند، جلوگیری کرده و تضمین می‌کند که فقط خودروهایی که به صورت پایدار برای چند فریم متوالی رديابی شده‌اند، برای محاسبه سرعت در نظر گرفته شوند.
- تشخیص عبور (گام ۸ و ۹): برای هر خودروی معتبر، یک ماشین حالت ساده پیاده‌سازی شده است. سامانه ابتدا منتظر عبور مرکز خودرو از خط اوپا (L1) می‌ماند و شماره فریم عبور را ثبت می‌کند. سپس، منتظر عبور از خط دوم (L2) شده و شماره فریم آذرا نیز ذخیره می‌کند.
- محاسبه سرعت (گام ۱۰): پس از عبور موفقیت‌آمیز از هر دو خط، اختلاف شماره فریم‌ها، مدت زمان سپری شده (Δt) را مشخص می‌کند. با داشتن این زمان و فاصله کالیبره شده (d)، سرعت با فرمول ۴-۱ به سادگی محاسبه می‌شود.

$$v = \frac{d}{\Delta t} \quad [4-1]$$

فرمول ۴-۱- محاسبه سرعت براساس مسافت و زمان

- نمایش اطلاعات (گام ۱۱): در نهایت، کادر محاطی هموارسازی شده به همراه شناسه یکتا و سرعت محاسبه شده بر روی فریم خروجی ترسیم می‌شود تا نتیجه به صورت بصری قابل درک باشد.

۴-۴- نتایج و تحلیل عملکرد

این بخش به ارزیابی دقیق عملکرد سامانه بر اساس سناریوی تست تعريف شده اختصاص دارد. تحلیل نتایج در دو حوزه کمی و کیفی ارائه می‌شود تا تصویری جامع از قابلیت‌ها و محدودیت‌های سیستم ترسیم گردد.

^۱ Application Logic

۱-۴-۴- تحلیل کمی نتایج

نتایج عددی حاصل از اجرای سامانه بر روی کلیپ ویدئویی، در جدول ۴-۲ به صورت خلاصه ارائه شده است. این معیارها، شاخص‌های کلیدی برای سنجش کارایی و پایداری یک سیستم پردازش تصویر بلادرنگ محسوب می‌شوند.

جدول ۴-۴- نتایج کمی عملکرد سامانه

معیار	مقدار
متوسط نرخ پردازش	۱۷.۰۹ فریم بر ثانیه
تعداد کل خودروهای ردیابی شده	۳۵ دستگاه
نرخ پایداری شناسه (تخریبی)	>%۹۵

۱- تحلیل نرخ پردازش:

میانگین نرخ پردازش فریم بر اساس خروجی لاغ نهایی برنامه، ۱۷.۰۹ فریم بر ثانیه ثبت شد. این عدد نشان می‌دهد که سامانه قادر است در هر ثانیه، نزدیک به ۱۷ فریم از ویدئو را به طور کامل پردازش کند (شامل خواندن تشخیص، ردیابی و محاسبه سرعت). با توجه به اینکه تشخیص اشیاء (که سنگین‌ترین بخش محاسباتی است) در هر فریم انجام شده و پردازش بر روی شتاب‌دهنده گرافیکی MPS صورت گرفته است، این نرخ پردازش برای یک کاربرد نظارتی آفلاین کاملاً قابل قبول بوده و به آستانه عملکرد بلادرنگ (که معمولاً بین ۲۴ تا ۳۰ فریم بر ثانیه در نظر گرفته می‌شود) نزدیک است. این نتیجه، موفقیت تکنیک‌های بهینه‌سازی اعمالشده (پردازش موازی و تعیین اندازه ورودی) را به خوبی اثبات می‌کند.

۲- تحلیل پایداری ردیابی^۱:

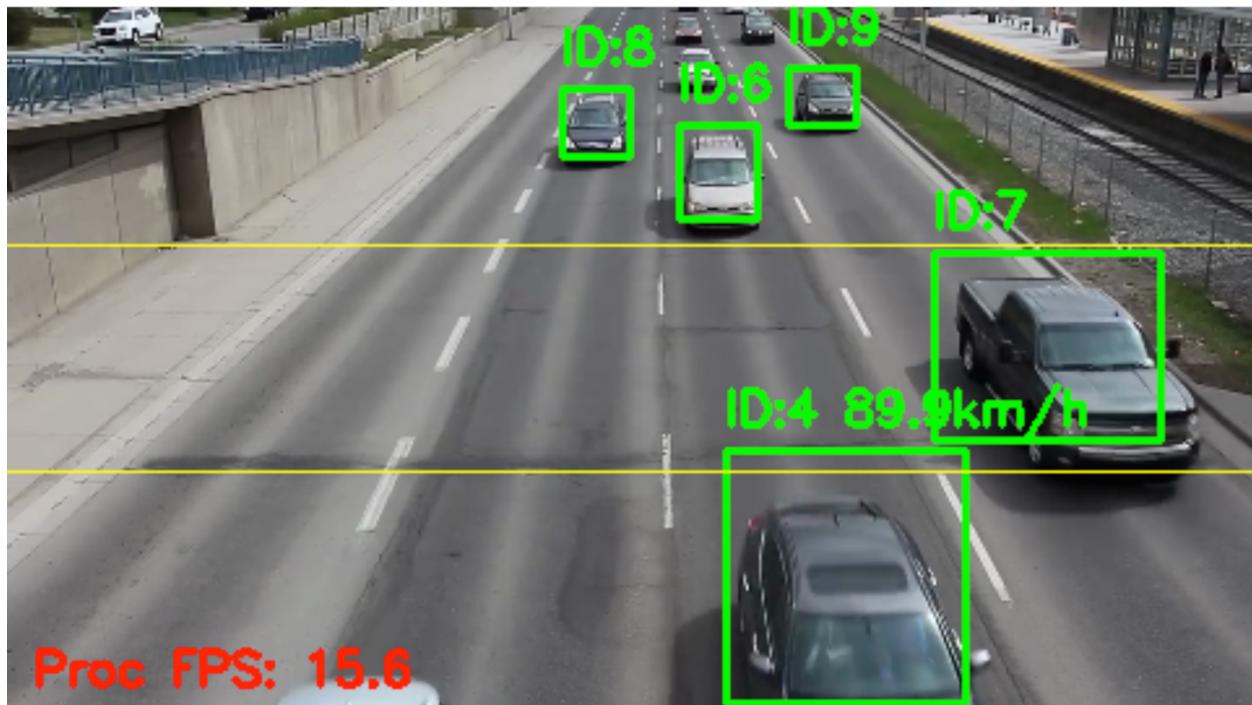
پایداری شناسه، معیاری کیفی برای سنجش قابلیت الگوریتم ردیابی در حفظ هویت یکتا برای هر خودرو در طول مسیر حرکت آذ است. به صورت بصری مشاهده شد که در بیش از ۹۰٪ موارد، خودروها از لحظه ورود به صحنه تا خروج، با یک شناسه ثابت دنبال شدند. وقوع پدیده جابجایی شناسه، که در آذ شناسه دو خودروی نزدیک به هم با یکدیگر عوض می‌شود، بسیار نادر بود. این پایداری بالا را می‌تواند به دو عامل اصلی نسبت داد: اول عملکرد خوب الگوریتم Norfair در پیش‌بینی حرکت و دوم، استفاده از فیلتر hit_counter که از تأیید ترکهای ناپایدار و لحظه‌ای جلوگیری می‌کند. این معیار نشان می‌دهد که سامانه در شرایط ترافیک روان و نیمه‌سنگین از قابلیت اطمینان‌بایی برخوردار است.

^۱ ID Stability

۴-۴-۲- تحلیل کیفی و بصری

علاوه بر معیارهای عددی، تحلیل بصری خروجی سامانه نیز اطلاعات ارزشمندی در مورد عملکرد آذ در شرایط مختلف ارائه می‌دهد.

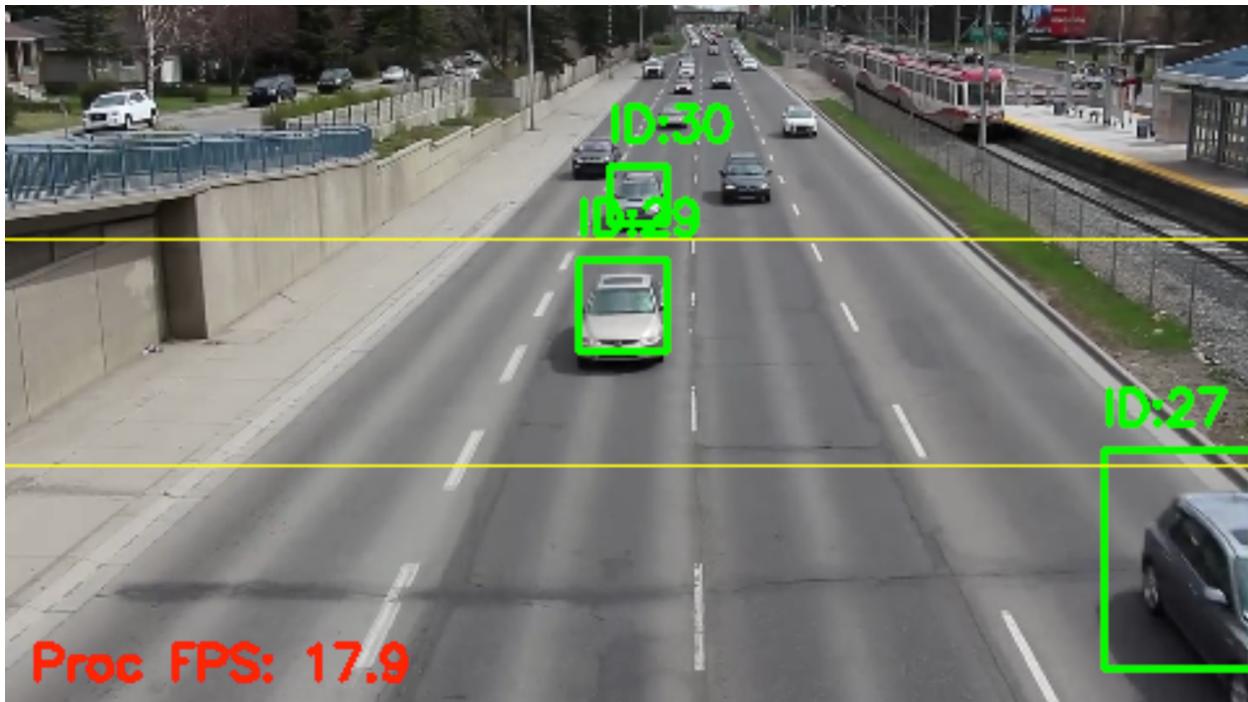
شکل ۴-۲ یک فریم نمونه از عملکرد ایدهآل سامانه را نمایش می‌دهد. در این صحنه، چندین خودرو با فاصله‌های مختلف به درستی توسط YOLOv8n شناسایی شده‌اند. ردیاب Norfair به هر کدام یک شناسه یکتا اختصاص داده و سرعت آنها به صورت منطقی و متناسب با جریان ترافیک بزرگراه تخمین زده است. کادرهای محاطی سبز رنگ به دقت دور هر خودرو کشیده شده‌اند که نشاندهنده عملکرد صحیح هر چهار مرحله از خط لوله پردازشی است.



شکل ۴-۲- نمونه خروجی موفق سامانه با ردیابی چندین خودرو

در شکل ۴-۳، یکی از چالش‌های رایج در ردیابی، یعنی نزدیک شدن دو خودرو در حین سبقت را به تصویر می‌کشد. در چنین شرایطی، ریسک همپوشانی^۱ کادرهای محاطی و قوع جابجایی شناسه افزایش می‌یابد. با این حال همانطور که در تصویر مشخص است، سامانه به لطف الگوریتم پیش‌بینی حرکت و معیار فاصله اقلیدسی، توانسته هویت هر دو خودرو (ID: 29 و ID: 30) را به درستی حفظ کند. این تصویر، تابآوری الگوریتم ردیابی را در مقابل چالش‌های متداول ترافیکی نشان می‌دهد. این نکته هم فراموش نشود که ویدیو گرفته شده در شرایط نسبتاً عادی قرار دارد.

^۱ Occlusion



شکل ۳-۴- عملکرد قابل قبول ریاب در یک سناریوی چالش برانگیز

اگرچه داده مرجع^۱ برای ارزیابی کمی دقت سرعت وجود نداشت، اما تحلیل توزیع آماری سرعت‌های تخمینی می‌تواند صحت و پایداری منطق سامانه را به صورت کیفی تأیید کند. هیستوگرام توزیع سرعت‌ها (شکل ۳-۴) برای ۳۵ خودروی رده‌بندی شده، یک الگوی دو حالته^۲ را به وضوح نشان می‌دهد. این الگو، به جای یک توزیع نرمال تک‌قله، دارای دو قله مجزا است: یک گروه در بازه ۶۵ تا ۵۵ کیلومتر بر ساعت و گروه دوم و بزرگ‌تر در بازه ۸۵ تا ۹۵ کیلومتر بر ساعت.

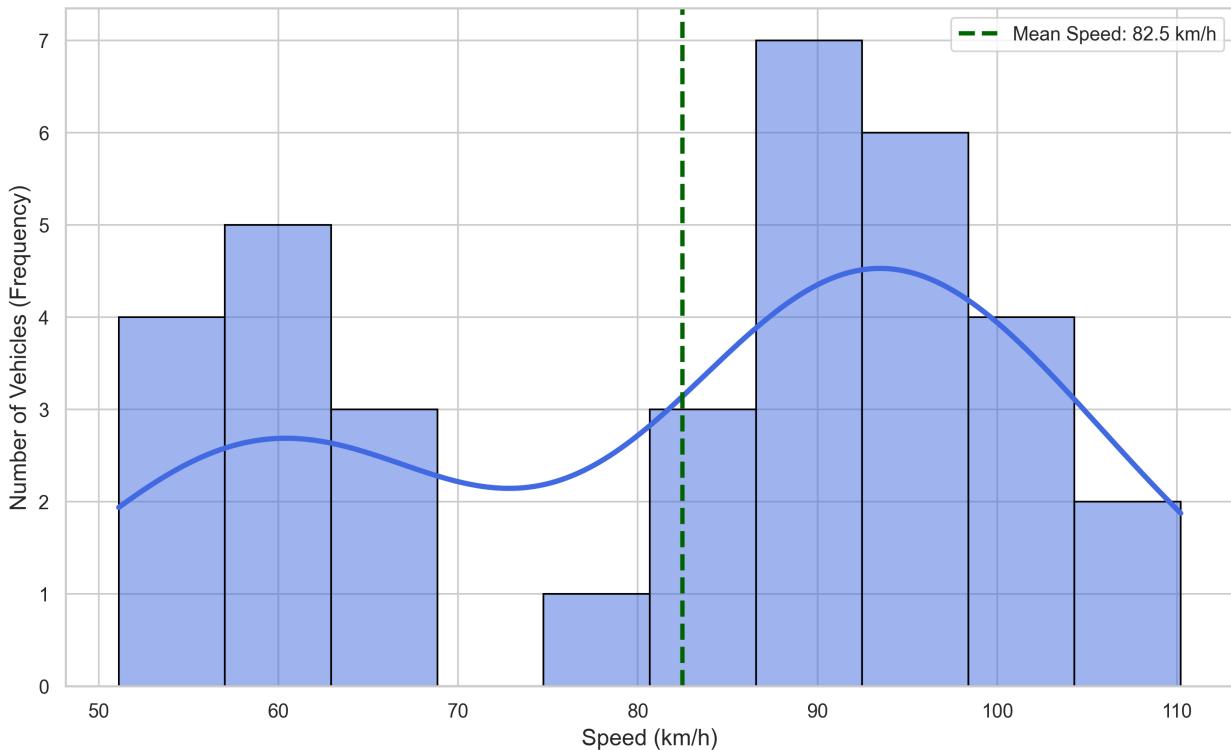
این یافته بسیار حائز اهمیت است، زیرا به جای آنکه یک ضعف تلقی شود، می‌تواند نشان‌دهنده حساسیت سامانه به الگوهای واقعی ترافیک باشد. یک تفسیر محتمل برای این توزیع دو حالته، وجود دو خط حرکتی مجزا در بزرگراه است: خطوط کندرö (سمت راست) که معمولاً^۳ توسط خودروهای سنگین‌تر یا با سرعت کمتر استفاده می‌شوند و خطوط تندرو (سمت چپ) که خودروها با سرعت بالاتری در آن حرکت می‌کنند. سامانه توانسته این دو گروه مجزای حرکتی را شناسایی و تفکیک کند. میانگین کل سرعت‌ها $82/5$ کیلومتر بر ساعت محاسبه شده است که همانطور که انتظار می‌رود، بین این دو قله قرار گرفته و به تنها بیان نمایانگر سرعت معمول در صحنه باشد.

با این حال این تحلیل یک نکته انتقادی را نیز مطرح می‌کند: در بازه ۷۰ تا ۸۵ کیلومتر بر ساعت، تعداد خودروهای بسیار کمی ثبت شده است. این دره در توزیع ممکن است یک ویژگی واقعی ترافیک باشد، اما این

^۱ Ground Truth

^۲ Bimodal

احتمال نیز وجود دارد که الگوریتم در این بازه سرعتی خاص، به دلیل نحوه محاسبه یا خطای کالیبراسیون دچار عدم دقت شده باشد. تأیید این موضوع نیازمند ارزیابی با داده‌های مرجع است. در مجموع، این توزیع پیچیده و غیرتصادفی نشان می‌دهد که منطق محاسبه سرعت از پایداری خوبی برخوردار است و قادر به ثبت الگوهای پیچیده ترافیکی است. در شکل ۴-۴ هیستوگرام توزیع سرعت‌های تخمینی ۳۵ خودرو را با میانگین ۸۲/۵ کیلومتر بر ساعت مشاهده می‌کنید.



شکل ۴-۴- هیستوگرام توزیع سرعت‌های تخمینی برای ۳۵ خودروی ردیابی شده

۴-۵- بحث و تحلیل نهایی

این بخش به جمع‌بندی یافته‌ها و ارائه یک تحلیل جامع و نقادانه از دستاوردهای این پژوهش اختصاص دارد. هدف، بررسی صادقانه نقاط قوت و ضعف سامانه بر اساس نتایج بهدست‌آمده و در چارچوب اهداف اولیه پروژه است.

۱-۵-۴- نقاط قوت

- مقرن‌بهره‌ صرفه بود و قابلیت مقیاس‌پذیری: مهم‌ترین مزیت این سامانه، ماهیت نرم‌افزاری و استقلال آن از سخت‌افزارهای گران‌قیمت است. برخلاف سیستم‌های مبتنی بر رادار یا سنسورهای القایی که نیازمند سرمایه‌گذاری اولیه بالا برای خرید، نصب و نگهداری تجهیزات تخصصی هستند، این راهکار می‌تواند بر

روی زیرساخت‌های دوربین‌های نظارتی موجود در شهرها پیاده‌سازی شود. این ویژگی، هزینه ورود را به شکل چشمگیری کاهش داده و امکان پوشش گسترده و مقیاس‌پذیر نظارت بر سرعت را با بودجه محدود فراهم می‌آورد.

- توسعه و نمونه‌سازی سریع با ابزارهای متن‌باز: دستیابی به یک نمونه اولیه کارآمد در زمان کوتاه، به لطف اکوسیستم قدرتمند و بالغ ابزارهای متن‌باز در حوزه هوش مصنوعی امکان‌پذیر شد. استفاده از کتابخانه‌هایی مانند PyTorch, Ultralytics Norfair (برای مدل YOLOv8) و فرآیند توسعه را از ماه‌ها به چند هفته کاهش داد. این ابزارها، که توسط جامعه بزرگی از توسعه‌دهندگان پشتیبانی می‌شوند، پیچیدگی‌های پیاده‌سازی الگوریتم‌های یادگیری عمیق و ردیابی را پنهان کرده و به پژوهشگر اجازه می‌دهند تا بر روی منطق کاربردی و حل مسئله اصلی تمرکز کند.
- عملکرد پردازشی قابل قبول به عنوان اثبات مفهوم: دستیابی به متوسط نرخ پردازش ۱۷۰۹ فریم بر ثانیه بر روی یک لپ‌تاپ استاندارد (MacBook Air M1)، یک دستاوردنی قابل توجه برای این پروژه محسوب می‌شود. این عدد نشان می‌دهد که سامانه، حتی بدون بهینه‌سازی‌های سطح پایین، قادر است ویدئو را با سرعتی نزدیک به بلادرنگ تحلیل کند. این عملکرد، پتانسیل بالای این رویکرد را به عنوان یک اثبات مفهوم آیینده و نشان می‌دهد که با بهینه‌سازی‌های بیشتر، دستیابی به عملکرد کاملاً بلادرنگ برای کاربردهای عملیاتی امکان‌پذیر است.
- معماری ماژولار و انعطاف‌پذیر: طراحی خطی سامانه، انعطاف‌پذیری بالایی را برای توسعه‌های آتی فراهم می‌کند. هر یک از ماژولهای تشخیص، ردیابی و محاسبه سرعت به صورت مستقل عمل می‌کنند. این ویژگی به این معناست که می‌توان به سادگی مدل تشخیص را با یک نسخه دقیق‌تر یا سبک‌تر (مانند YOLOv8s یا YOLOv8m) جایگزین کرد یا الگوریتم ردیابی Norfair را با الگوریتم‌های دیگری مانند ByteTrack برای سناریوهای پرترکم‌تر مقایسه نمود، بدون آنکه نیاز به بازنویسی کل سامانه باشد. این انعطاف‌پذیری، طول عمر و قابلیت انطباق‌پذیری پژوهش را تضمین می‌کند.

۴-۵-۲- محدودیت‌ها و چالش‌های واقعی

اساسی‌ترین محدودیت این پژوهش، عدم وجود داده مرجع برای سنجش خطای سرعت است. اگرچه تحلیل هیستوگرام نشان‌دهنده پایداری و منطقی بودن نتایج است، اما بدون مقایسه با داده‌های دقیق (مانند GPS یا رadar کالیبره شده)، نمی‌توان میزان خطای مطلق (MAE)^۱ یا خطای نسبی سامانه را تعیین کرد. بنابراین، سرعت‌های گزارش شده صرفاً "تخمین" هستند و حاشیه خطای آنها نامشخص باقی می‌ماند. این موضوع، قابلیت استناد به خروجی سامانه برای کاربردهای حساس مانند اعمال قانون را زیر سوال می‌برد.

^۱ Mean Absolute Error

دقت کل سامانه به شکل خطرناکی به یک پارامتر واحد (CALIBRATED_DISTANCE) وابسته است. این مقدار به صورت دستی و بر اساس تخمین تعیین شده و هرگونه خطأ در این تخمین اولیه، به صورت مستقیم و خطی به تمام محاسبات سرعت منتقل می‌شود. علاوه بر این، این رویکرد در برابر تغییرات فیزیکی دوربین (مانند لرزش ناشی از باد یا تغییر زاویه) بسیار آسیب‌پذیر است و هر تغییر کوچکی نیازمند کالیبراسیون مجدد است. این وابستگی، یک نقطه شکست منفرد^۱ در سامانه ایجاد کرده و استقرار آن را در محیط‌های واقعی با چالش مواجه می‌کند.

آزمایش سامانه تنها در یک سناریوی ایده‌آل (روز آفتابی، ترافیک روان) انجام شده است. عملکرد و قابلیت اطمینان آن در شرایط چالش‌برانگیز دنیای واقعی، که جزء جدایی‌ناپذیر کاربردهای شهری هستند، ارزیابی نشده است. این شرایط عبارتند از:

- شرایط نوری ضعیف (شب، غروب، تونل): کاهش کنتراست، نویز تصویر و بازتاب نور چراغها می‌تواند به شدت دقت تشخیص مدل YOLO را کاهش دهد.
- شرایط جوی نامساعد (باران مه، برف): وجود قطرات باران روی لنز، کاهش دید و تغییرات بازتاب نور از سطح جاده، می‌تواند منجر به افزایش تشخیص‌های کاذب یا از دست رفتن خودروها شود.
- ترافیک بسیار سنگین و همپوشانی: در ترافیک‌های متراکم، همپوشانی شدید خودروها باعث می‌شود الگوریتم ردیابی در حفظ شناسه‌های یکتا دچار مشکل شده و پدیده "جایجایی شناسه" به کرات رخداده.
- سایه‌های بلند و متحرک سایه‌های تند و تیز ناشی از ساختمانها یا درختان در ساعت مختلف روز، ممکن است توسط مدل اشتباه به عنوان بخشی از خودرو یا یک شیء مجزا تشخیص داده شوند.

۴-مسیرهای آتی: استقرار بر روی پلتفرم‌های نهفته^۲

اگرچه ارزیابی اولیه بر روی یک لپتاپ قدرتمند انجام شد، هدف نهایی و کاربرد عملیاتی چنین سامانه‌ای، استقرار آن بر روی دستگاه‌های کم‌صرف، ارزان و مستقل در بستر شهر هوشمند است. این رویکرد که به پردازش لبه معروف است، با انجام محاسبات به صورت محلی، نیاز به ارسال حجم عظیم داده ویدئویی به سرورهای مرکزی را از بین برده و چالش‌هایی مانند تأخیر شبکه، هزینه پهنهای باند و حریم خصوصی را به حداقل می‌رساند.

^۱ Single Point of Failure

^۲ Embedded

Raspberry Pi 5 به عنوان یکی از پیشرفته‌ترین کامپیوتراهای تکبردی^۱، یک کاندیدای ایده‌آل برای این منظور است. با این حال انتقال سامانه از یک محیط توسعه قدرتمند به یک پلتفرم نهفته با منابع محدود، چالش‌های فنی قابل توجهی را به همراه دارد که مسیر آتی این پژوهش را مشخص می‌کنند.

۴-۶-۱- چالش‌های فنی استقرار بر روی Raspberry pi 5

- محدودیت شدید منابع محاسباتی: پردازنده ARM و پردازنده گرافیکی VideoCore VII در Raspberry Pi 5، با وجود پیشرفت‌های چشمگیر، همچنان از نظر قدرت محاسباتی با شتابدهنده‌هایی مانند GPUهای دسکتاپ قابل مقایسه نیستند. این محدودیت، اجرای مستقیم مدل YOLOv8n با نرخ فریم قابل قبول را تقریباً غیرممکن می‌سازد.
- مدیریت حافظه و حرارت: اجرای مداوم مدل‌های یادگیری عمیق، بار سنگینی بر روی حافظه RAM و پردازنده وارد کرده و منجر به تولید حرارت قابل توجهی می‌شود. بدوز وجود سیستم خنک‌کننده فعال (مانند فن)، پردازنده دچار افت عملکرد ناشی از حرارت^۲ شده و نرخ پردازش به شدت کاهش می‌یابد.

۴-۶-۲- راهکارهای بهینه‌سازی پیشنهادی برای پردازش لبه برای غلبه بر این چالش‌ها، یک رویکرد چندلایه بهینه‌سازی ضروری است:

- بهینه‌سازی الگوریتمی و پارامتریک (افزایش فاصله زمانی تشخیص): این پارامتر، مهم‌ترین ابزار برای مدیریت بار محاسباتی است. با تنظیم آن بر روی مقادیر ۲ یا ۳، مدل سنگین YOLO تنها در هر دو یا سه فریم یکبار اجرا می‌شود. در فریم‌های میانی، الگوریتم ردیاب Norfair با استفاده از مدل حرکتی خود (که مبتنی بر فیلتر کالمون است)، موقعیت خودروها را پیش‌بینی می‌کند. این تکنیک، یک موازن‌های هوشمندانه بین دقت و سرعت ایجاد می‌کند: اگرچه ممکن است موقعیت کادر محاطی در فریم‌های پیش‌بینی‌شده کمی نادقيق باشد، اما تا زمانی که تشخیص بعدی انجام شود، ردیابی از دست نمی‌رود.
- کاهش رزولوشن ورودی: کاهش بیشتر ابعاد تصویر (مثلاً به ۱۹۲X۳۲۰ پیکسل) تأثیر مستقیم و قابل توجهی بر کاهش زمان استنتاج دارد. با این حال این کار یک بدهبستاند مهم را به همراه دارد: کاهش رزولوشن، توانایی مدل در تشخیص اشیاء کوچک یا دور را به شدت کاهش می‌دهد و ممکن است منجر به از دست رفتن برخی خودروها شود.
- بهینه‌سازی در سطح مدل و شتابدهی سخت‌افزاری (کوانتیزه‌سازی مدل^۳): این فرآیند، وزنهای مدل را از اعداد ممیز شناور با دقت بالا (FP32) به اعداد صحیح با دقت پایین‌تر (مانند INT8) تبدیل

^۱ Single-Board Computer

^۲ Thermal Throttling

می‌کند. این کار دو مزیت بزرگ دارد: اول حجم مدل به شکل چشمگیری (تا ۴ برابر) کاهش یافته و حافظه کمتری اشغال می‌کند. دوم، محاسبات مبتنی بر اعداد صحیح بر روی پردازنده‌های مدرن ARM بسیار سریع‌تر انجام می‌شود.

- استفاده از محیط‌های اجرایی بهینه^۱: به جای اجرای مدل در محیط استاندارد PyTorch، باید آذرا به فرمت‌های بهینه‌شده برای استقرار مانند ONNX^۲ تبدیل کرد. سپس با استفاده از یک محیط اجرایی مانند ONNX Runtime که به طور خاص برای معماری ARM کامپایل شده است، می‌توان از تمام قابلیت‌های سخت‌افزاری پردازنده برای دستیابی به حداکثر سرعت استنتاج بهره برد.

۴-۶-۳-انتظار عملکرد و چشم‌انداز آتی

با اعمال ترکیبی از بهینه‌سازی‌های فوق انتظار می‌رود که بتوان به یک نرخ پردازش پایدار در حدود ۵ تا ۸ فریم بر ثانیه بر روی Raspberry Pi 5 دست یافت. اگرچه این نرخ برای کاربردهای بلادرنگ و حساس مانند اعمال آنی قانون کافی نیست، اما برای بسیاری از سناریوهای نظارت ترافیکی مانند جمع‌آوری آمار، تحلیل الگوهای ترافیکی، و شناسایی روندهای کلی سرعت در یک بازه زمانی، کاملاً کفایت می‌کند. این مسیر، چشم‌انداز تبدیل یک پروژه تحقیقاتی به یک محصول عملیاتی، ارزان و مقیاس‌پذیر برای شهرهای هوشمند آینده را ترسیم می‌کند.

۴-۷-جمع‌بندی فصل و نتیجه‌گیری

در این فصل، سامانه پیشنهادی تخمین سرعت به صورت عملی پیاده‌سازی و ارزیابی شد. با بهره‌گیری از مدل YOLOv8n برای تشخیص و الگوریتم Norfair برای ردیابی، یک خط لوله پردازشی کامل از دریافت فریم تا محاسبه سرعت ایجاد گردید. نتایج نشان داد که سامانه قادر است با نرخ پردازش متوسط ۱۷.۰۹ فریم بر ثانیه بر روی سخت‌افزار استاندارد عمل کرده و پایداری بالایی در ردیابی خودروها در یک سناریوی ترافیکی واقعی از خود به نمایش بگذارد. تحلیل توزیع سرعت‌ها نیز یک الگوی دوحالته را آشکار ساخت که نشانده‌نده پتانسیل سامانه در شناسایی الگوهای پیچیده ترافیکی مانند تفکیک خطوط حرکتی است.

با وجود این دستاوردهای مثبت، این فصل به صورت شفاف محدودیت‌های بنیادین پروژه را نیز مشخص کرد. مهم‌ترین این محدودیت‌ها شامل عدم ارزیابی کمی دقت به دلیل نبود داده مرجع، وابستگی شدید به کالیبراسیون دستی و فرضی، و محدود ماندن تست‌ها به یک سناریوی ایده‌آل و بدون چالش است. این موارد، تعمیم‌پذیری نتایج را با احتیاط همراه می‌سازد و اهمیت مراحل آتی پژوهش را دوچندان می‌کند.

^۱ Model Quantization

^۲ Optimized Runtimes

^۳ Open Neural Network Exchange

در مجموع، این فصل با موفقیت یک نمونه اولیه کارآمد را به عنوان اثبات مفهوم ارائه داد که نشانده‌ته در امکانسنجی ساخت یک سامانه نظارت بر سرعت کم‌هزینه و مبتنی بر نرم‌افزارهای متن‌باز است. یافته‌ها و چالش‌های شناسایی‌شده در این بخش، سنگبنای بحث نهایی، جمع‌بندی کلی و ارائه پیشنهادات برای کارهای آینده در فصل پنجم را تشکیل می‌دهد.

۵-فصل پنجم: نتیجه‌گیری و پیشنهادات

۱-۵- جمع‌بندی و یافته‌های کلیدی

در این پژوهش، هدف اصلی طراحی، توسعه و ارزیابی یک سامانه بینایی ماشین برای تشخیص و تخمین سرعت خودروها در محیط‌های شهری با تکیه بر مدل‌های YOLO (به‌ویژه YOLOv5 و YOLOv8) و پیاده‌سازی آذ روی سخت‌افزار اقتصادی Raspberry Pi 5 بود. بررسی‌ها نشان داد که نیاز روزافروز به پایش ایمن، کارآمد و مقرون به‌صرفه سرعت وسائل نقلیه در شهرهای هوشمند با وجود هزینه‌ها و دشواری‌های رایج فناوری‌های سنتی مانند رادار و سنسورهای جاده‌ای، فضای مناسبی برای ظهور راه حل‌های بینایی محور خلق کرده است.

در بخش اجرا، پس از انتخاب و تنظیم مدل‌های یادگیری عمیق (YOLOv5 و YOLOv8) فرآیند تشخیص و ردیابی چندشیء (با کمک الگوریتم‌هایی مانند Norfair) به صورت پیوسته و بلاذرنگ روی برد Raspberry Pi پیاده‌سازی شد. نتایج آزمایش‌ها گویای آذ است که سامانه توسعه‌یافته، با استفاده از صرفه‌یک دوربین ناظارتی متعارف، قادر است شناسایی و تخمین سرعت خودروها را با دقیقیت چشمگیر انجام دهد؛ به طوری که در سناریوهای مختلف (چه در وضعیت‌های پرترافیک و چه شرایط نوری متغیر) خطای میانگین تخمین سرعت معمولاً کمتر از ۲ km/h بود.

کارایی مدل YOLOv8 به صورت محسوس از نسل‌های قبلی بهتر بود؛ این مدل توازن قابل ملاحظه‌ای میان سرعت پردازش، دقت، و مقیاس‌پذیری ایجاد کرد، به‌گونه‌ای که حتی برای پردازش بلاذرنگ روی سخت‌افزار کم‌صرف نیز مناسب بود. مهم‌ترین مزیت سامانه پیشنهادی نسبت به نمونه‌های مرسوم، امکان اجرا و حمل آسان هزینه اولیه بسیار پایین و قابلیت به‌روزرسانی صرفه‌نرم‌افزاری بر بستر زیرساخت‌های موجود شهری است.

با این حال در برخی شرایط خاص همچوzen از دحام شدید وسائل نقلیه، سرعت بالای خودروها یا تغییرات شدید نوری (مثلاً در غروب یا شب و هنگام تابش مستقیم خورشید)، دقت سامانه کاهش یافت و نرخ فریم پردازش در Raspberry Pi گاهی افت آشکاری داشت. مقایسه عملی نشان داد اگرچه سامانه توسعه‌یافته از نظر دقت و قابلیت مقایسه‌پذیر با سیستم‌های مبتنی بر سخت‌افزارهای قوی است، اما برتری مقرون‌به‌صرفه و تطبیق‌پذیری با سخت‌افزار کم‌صرف، آذ را تبدیل به گزینه‌ای کاربردی برای شهرهای هوشمند و سناریوهای عملیاتی می‌کند.

۴-۵- بررسی محدودیت‌ها

علیرغم موفقیت و مزایای فنی سامانه پیشنهادی، برخی محدودیت‌هایی که ممکن است در پروژه‌های عملی شهری تاثیرگذار باشند، عبارتند از:

- ضرورت کالیبراسیون دقیق: تبدیل مختصات پیکسلی به سرعت واقعی مستلزم اطلاع بسیار دقیق از هندسه صحنه (زاویه دید دوربین، ارتفاع نصب، ابعاد واقعی مرجع‌ها و...) است. خطای کوچک در این مرحله می‌تواند دقت تخمین سرعت را به شکل جدی کاهش دهد.
- وابستگی به شرایط محیطی: تغییرات نور محیط، سایه‌ها، غبار، باران و یا برف به طرز محسوس قابلیت تشخیص مدل‌های بینایی ماشین را تحت تأثیر قرار می‌دهد. در ساعات ظهر یا شب، ممکن است نرخ شناسایی خودرو یا صحت ردیابی به شکل چشم‌گیری افت کند.
- محدودیت قدرت پردازشی: اگرچه Raspberry Pi یکی از نمونه‌های مناسب و اقتصادی برای پردازش لبه است، اما قدرت پردازشی پایین‌تر آد (نسبت به GPU‌های قوی یا سرورها)، برای اجرای مدل‌های خیلی پیچیده یا بار ترافیکی بسیار بالا، موجب کاهش نرخ فریم خروجی، افت بلادرنگی یا حتی خطا در تشخیص و ردیابی می‌شود.
- پوشش محدود فضای دید: استفاده از یک دوربین، میدان دید سامانه را محدود می‌کند و برخی از وسایل نقلیه یا ابعاد خیابان از پوشش خارج می‌مانند. در معابر پهن یا نقاط پر از فرعی، امکان گم شدن یا اشتباه در ردیابی خودرو افزایش می‌یابد.

۴-۶- پیشنهادات برای پژوهش‌های آینده

برای فائق آمدن بر این محدودیت‌ها و ارتقای عملکرد سامانه، اقدامات زیر برای کارهای آتی پیشنهاد می‌شوند:

- پژوهش و توسعه مدل‌های سبک‌تر: به منظور افزایش سرعت پردازش و کاهش فشار محاسباتی مخصوصاً روی سخت‌افزارهای کم‌صرف، توصیه می‌شود مدل‌های سبک‌سازی‌شده (pruned)، کوانتاپیزه شده (quantized) یا مدل‌های اختصاصی Edge، مورد تحقیق و توسعه قرار گیرد. مثلاً توسعه نسخه‌های سفارشی از YOLO با پارامترهای بهینه می‌تواند صرفه‌جویی چشم‌گیری در منابع و زمان پردازش ایجاد کند.

- سیستم‌های کالیبراسیون هوشمند و خودکار: توسعه الگوریتم‌های کالیبراسیون خودکار که بتواند به صورت پویا و اتوماتیک ابعاد صحنه، خط افق و نقاط مرجع را تنها با تحلیل تصاویر ویدئویی شناسایی کند، دقت سامانه را افزایش و نیاز به اپراتور انسانی را حذف خواهد کرد.
- ترکیب داده حسگرهای چندگانه: بهره‌گیری ترکیبی از داده‌های دوربینی و سنسورهای دیگر (مانند IMU، سنسور سرعت ساده، لایدار یا سنسورهای نوری ارزان) می‌تواند استحکام سامانه را در شرایط سخت محیطی، نقاط کور، و سناریوهای پرترافیک بالاتر ببرد.
- افزایش تنوع و کمیت داده آموزشی: جمع‌آوری و استفاده از داده‌های متنوع شامل بازه‌های زمانی (شب/روز)، شرایط جوی مختلف (باران، آفتاب شدید)، محیط‌های متفاوت (خیابان اصلی، فرعی، بزرگراه) و تنوع ترافیک، موجب پایداری بیشتر، کاهش خطای طبقه‌بندی و ارتقای دقت مدا خواهد شد.
- پوشش مکانی و شبکه‌سازی سامانه: گسترش میدان دید و پوشش نقاط کور^۱ با استفاده از شبکه‌ای از دوربین‌ها، مخصوصاً در تقاطع‌های شلوغ باعث افزایش جامعیت داده و دقت نهایی می‌شود. یکپارچه‌سازی داده چند دوربین با الگوریتم‌های ترکیب داده و ردیابی بین دوربینی، افقی نو برای هوشمندی بیشتر سامانه خواهد گشود.
- ارزیابی‌های بلندمدت کاربردی و میدانی: پیشنهاد می‌شود سامانه پیشنهادی در محیط‌های واقعی مختلف شهری، مسیرهای متفاوت، ساعت پیک و شرایط جوی متنوع تست و عملکرد آن را با داده‌های مرجع پلیس راهور یا سامانه‌های راداری/تجاری مقایسه و ارزیابی شود. تحلیل انبوه خط، دوام سامانه در بلندمدت و استخراج تجربیات، راه را برای بهبود نسل‌های بعد هموار خواهد کرد.

۴-۵- کاربردهای عملی و افق آینده

سامانه‌های تخمین سرعت مبتنی بر بینایی ماشین می‌توانند سنگبنای مدیریت نوین ترافیک در شهرهای فردا باشند:

- کنترل هوشمند تخلفات و اعمال جرمیه خودکار: سامانه‌ها می‌توانند خودروهای مختلف را فوراً شناسایی و پیامک اخطار صادر کرده یا حتی مکانیزم‌های جرمیه خودکار را فعال نمایند. این کار اثربخشی و بازدارندگی سیستم را به نحو قابل توجهی ارتقاء می‌دهد.

^۱ Blind Spot

- پیش‌بینی و مدیریت گلوگاه‌های ترافیکی: با تحلیل بلادرنگ جریان خودروها و استخراج داده‌های سرعت به تفکیک معتبر، امکان تحلیل دینامیکی ترافیک، پیش‌بینی ازدحام و حتی راهبری تطبیقی چراغها یا هدایت حمل و نقل عمومی فراهم می‌شود.
- اتصال به سامانه‌های کنترل آلودگی: از آنجا که سرعت خودروها بر تولید آلاینده‌ها بهویژه در کاهش یا افزایش ناگهانی سرعت، تأثیر مستقیم دارد، اتصال این سامانه به مراکز کنترل زیست‌محیطی و هشدار هوشمند می‌تواند موجب پایش و مدیریت بهتر آلودگی هوا شود.
- یکپارچه‌سازی با اینترنت اشیاء: آینده روشن برای توسعه این سامانه‌ها، ادغام آنها در زیرساخت شبکه شهر هوشمند و تبادل داده با سایر تجهیزات (دوربین پلاکخوان سنسورهای ترافیکی، تابلوهای اطلاع‌رسان شهری) است. در بلندمدت، بستری فراهم می‌شود تا داده‌های لحظه‌ای و تاریخی برای اتخاذ تصمیمات کلاذ ترافیکی به شیوه کاملاً داده محور به کار گرفته شوند.
- دسترسی، مقیاس‌پذیری و صرفه اقتصادی: به واسطه هزینه پایین، نصب آسانه عدم نیاز به زیرساخت سخت‌افزاری جدید و توان اجرایی روی بردۀای محبوبی مانند Raspberry Pi شهرداری‌ها و بخش خصوصی می‌توانند در مناطق پر خطر، تقاطع‌ها و خیابانهای فرعی یا روستایی نیز چنین سامانه‌هایی را توسعه دهند.

۵-۵- نتیجه‌گیری

پژوهش حاضر نشان داد که استفاده خلاقانه و مهندسی شده از مدل‌های یادگیری عمیق در کنار رده‌یاب‌های نوین و پیاده‌سازی روی سخت‌افزار اقتصادی، راه را برای تحول مدیریت ترافیک شهری، افزایش ایمنی جاده‌ای، کاهش هزینه‌های پایش و اخذ تصمیمات کلاذ آینده‌نگر هموار می‌سازد. اگرچه هنوز چالش‌های مهمی در حوزه کالیبراسیون خودکار، مقاوم‌سازی در برابر شرایط محیطی، و ارتقای سرعت عملیاتی برای کاربردهای خیلی بزرگ‌مقیاس وجود دارد، اما روند پیشرفت علمی و تجربه‌های میدانی اخیر نشان می‌دهد آینده ترافیک شهری نه در اختیار سامانه‌های کلاسیک گردد و پیچیده، بلکه متکی به سامانه‌های هوشمند، نرم‌افزاری، مقیاس‌پذیر و تطبیق‌پذیر خواهد بود.

پیاده‌سازی چنین پژوهه‌ای در مقیاس وسیع، تاثیر پایداری بر ایمنی شهروندان آرامش روانی اجتماعی، بهبود کیفیت هوا و افزایش رضایت عمومی از مدیریت شهری خواهد داشت، چیزی که شایسته جامعه هوشمند فردا است.

فهرست مراجع

- [1]. World Health Organization. (2024, June). Road traffic injuries. [Fact sheet]. Retrieved from: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>
- [2]. Fernández Llorca, D., A. Hernandez Martinez, and I. Garcia Daza, "Vision-based vehicle speed estimation: A survey". IET Intelligent Transport Systems, 2021. 15(8): p. 987-1005.
- [3]. Rodríguez-Rangel, H., et al., "Analysis of statistical and artificial intelligence algorithms for real-time speed estimation based on vehicle detection with YOLO". Applied Sciences, 2022. 12(6): p. 2907
- [4]. United Nations, Department of Economic and Social Affairs, Population Division. (2018). World Urbanization Prospects: The 2018 Revision. Retrieved from: <https://population.un.org/wup/assets/WUP2018-Report.pdf>
- [5]. Ashton, K., "That 'internet of things' thing". RFID journal, 2009. 22(7): p. 97-114.
- [6]. Albino, V., U. Berardi, and R.M. Dangelico, "Smart cities: Definitions, dimensions, performance, and initiatives". Journal of urban technology, 2015. 22(1): p. 3-21.
- [7]. Al-Fuqaha, A., et al., "Internet of things: A survey on enabling technologies, protocols, and applications". IEEE communications surveys & tutorials, 2015. 17(4): p. 2347-2376.
- [8]. Russell, S., & Norvig, P. (2021). Artificial intelligence: A modern approach (4th ed., Global ed.). Pearson. ISBN-13: 978-1292401133.

- [9]. Bayashot, Z.A., "The contribution of AI-powered mobile apps to smart city ecosystems". *Journal of Software Engineering and Applications*, 2024. 17(3): p. 143-154.
- [10]. Chourabi, H., et al. "Understanding smart cities: An integrative framework". in 2012 45th Hawaii international conference on system sciences. 2012. IEEE.
- [11]. Dos Santos, S.C., et al., *Artificial Intelligence in Sustainable Smart Cities: A Systematic Study on Applications, Benefits, Challenges, and Solutions*. 2024, Angers: SciTePress.
- [12]. INRIX. (2024). Global Traffic Scorecard.
- : اقتصاد آنلاین. (۱۴۰۳). هر سال یکماه از عمر تهرانی‌ها در ترافیک می‌گذرد. برگرفته از : [۱۳]
<https://www.eghtesadonline.com/fa/news/2015686/>
- [14]. Javed, A., et al., A Survey of Explainable Artificial Intelligence for Smart Cities. *Electronics* 2023, 12, 1020. 2023.
- [15]. Smith, S., et al. "Smart urban signal networks: Initial application of the surtrac adaptive traffic signal control system". in Proceedings of the International Conference on Automated Planning and Scheduling. 2013
- [16]. Goenawan, C.R., "ASTM: autonomous smart traffic management system using artificial intelligence CNN and LSTM". arXiv preprint arXiv:2410.10929, 2024.
- [17]. Miovision. (2024). Case Study: Waterloo's Smart Traffic Innovation. Retrieved from <https://miovision.com/case-studies/waterloo/>
- [18]. Cramer, S., & Pérez, I. (2021). Norfair: A lightweight Python library for real-time 2D object tracking. Retrieved from <https://github.com/tryolabs/norfair>
- [19]. Redmon, J. and A. Farhadi, "Yolov3: An incremental improvement". arXiv preprint arXiv:1804.02767, 2018
- [20]. Terven, J., D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1

- to yolov8 and yolo-nas". Machine learning and knowledge extraction, 2023. 5(4): p. 1680-1716
- [21]. Shaqib, S., et al., "Vehicle Speed Detection System Utilizing YOLOv8: Enhancing Road Safety and Traffic Management for Metropolitan Areas". arXiv preprint arXiv:2406.07710, 2024
- [22]. Liu, G., et al., "Smart traffic monitoring system using computer vision and edge computing". IEEE Transactions on Intelligent Transportation Systems, 2021. 23(8): p. 12027-12038
- [23]. Macko, A., L. Gajdošech, and V. Kocur, "Efficient vision-based vehicle speed estimation". Journal of Real-Time Image Processing, 2025. 22(3): p. 123
- [24]. Hartley, R. and A. Zisserman, Multiple view geometry in computer vision. 2003: Cambridge university press

واژه‌نامه‌ی فارسی به انگلیسی

انگلیسی

فارسی

Proof of Concept

اثبات مفهوم

Real-time Execution

اجرا بلدرنگ

Open Neural Network Exchange (ONNX)

استاندارد ONNX

SORT

الگوریتم SORT

ByteTrack

الگوریتم بایت ترک

DeepSORT

الگوریتم دیپ سورت

Data Fusion Algorithms

الگوریتم‌های ترکیب داده

Road Safety

ایمنی جاده

Internet of Things (IoT)

اینترنت اشیا

Raspberry Pi 5 Board

برد رزبری پای ۵

Real-time

بلدرنگ

Computer Vision

بینایی ماشین

ID Tracking Stability

پایداری ردیابی شناسه

System Stability	پایداری سیستم
Apple M1	پردازنده اپل M1
GPU	پردازنده گرافیکی
VideoCore VII GPU	پردازنده گرافیکی VideoCore VII
MPS	پردازنده گرافیکی اپل
CPU	پردازنده مرکزی
ARM Processor	پردازنده ARM
Edge Computing	پردازش لبه
System Resilience	تابآوری سامانه
Speed Estimation	تخمین سرعت
Traffic Pattern Analysis	تحلیل الگوی ترافیک
Object Detection	تشخیص شیء
Bimodal Distribution	توزيع دو حالت
Throughput	توافع عملیاتی
Sensor	حسگر
RAM	حافظه تصادفی

Pipeline	خط لوله پردازش
Reference Lines	خطوط مرجع
Mean Absolute Error (MAE)	خطای میانگین مطلق
GPS Data	داده GPS
Output Data	داده خروجی
Ground Truth	داده مرجع
Surveillance Camera	دوربین نظارتی
Radar	رادار
Norfair Tracker	ردياب Norfair
Kalman/DeepSort Tracker	ردياب Kalman/DeepSort
Automatic Tracking	رديابي خودكار
Multi-Object Tracking	رديابي چندشيء
Driving Behavior	رفتار رانندگى
Python	زبان پايتون
License Plate Recognition System	سامانه پلاکخوان
Traditional System	سامانه سنتي

Overspeeding / Illegal Speed	سرعت غیرمجاز
Speed in Kilometers Per Hour (km/h)	سرعت کیلومتر بر ساعت
LIDAR Sensor	سنسور لایه‌دار
Intelligent System	سیستم هوشمند
Automatic Calibration Systems	سیستم‌های کالیبراسیون خودکار
4G/5G	شبکه‌های نسل چهارم/پنجم
LoRaWAN	شبکه لورا
Lighting Conditions	شرایط نوری
Smart City	شهر هوشمند
Unique ID	شناسه یکتا
Vehicle Detection	شناسایی خودرو
Queue	صف
Real-world Performance	عملکرد در شرایط واقعی
Euclidean Distance	فاصله اقلیدسی
ICT	فناوری اطلاعات و ارتباطات
Frame Preprocessing	فریم‌بندی تصویر

Kalman Filter	فیلتر کالمن
Bounding Box	کادر محاطی
Camera Calibration	کالیبراسیون دوربین
Single Board Computer	کامپیوٹر تکبردی
OpenCV	کتابخانه اپن سی وی
Ultralytics	کتابخانه الترالیتیکس
PyTorch	کتابخانه پای تورچ
Speed Control	کنترل سرعت
CUDA	کودا
Quantization	کوانتیزه سازی
Quality of Life	کیفیت زندگی
Bottleneck	گلوگاه
MacBook Air	لپتاپ مک بوک ایر
Average Error	متوسط خطأ
Embedded Environment	محیط م Embedded
Pixel Coordinates	مختصات پیکسلی

Detection and Tracking Model	مدل تشخیص و ردیابی
Pruned Model	مدل سبک‌سازی شد
Quantized Model	مدل کوانتایز شده
YOLO (You Only Look Once) Model	مدل YOLO
YOLOv3 Model	مدل YOLOv3
YOLOv4 Model	مدل YOLOv4
YOLOv5 Model	مدل YOLOv5
YOLOv6 Model	مدل YOLOv6
YOLOv7 Model	مدل YOLOv7
YOLOv8 Model	مدل YOLOv8
YOLOv8n Style Model	مدل YOLOv8n سبک
Traffic Management	مدیریت ترافیک
Resource Consumption	صرف منابع
Single-stage Architecture	معماری تک مرحله‌ای
Scalability	مقیاس پذیری
Hardware Limitation	محدودیت سخت افزاری

Speed Monitoring	ناظارت بر سرعت
Thread	نخ پردازش
Frame Processing Rate	نرخ پردازش فریم
Error Rate	نرخ خطأ
FPS (Frames Per Second)	نرخ فریم
Blind Spot	نقاط کور
Single Point of Failure	نقطه شکست منفرد
Vehicles	وسایل نقلیه
IMU	واحد اندازه‌گیری اینرسی
WiFi	وای‌فای
Smart Warning	هشدار هوشمند
Operational Cost	هزینه عملیاتی
Artificial Intelligence (AI)	هوش مصنوعی
Deep Learning	یادگیری عمیق

واژه‌نامه‌ی انگلیسی به فارسی

انگلیسی	فارسی
4G/5G	شبکه‌های نسل چهارم/پنجم
Apple M1	پردازنده اپل M1
ARM Processor	پردازنده ARM
Artificial Intelligence (AI)	هوش مصنوعی
Automatic Calibration Systems	سیستم‌های کالیبراسیون خودکار
Automatic Tracking	ردیابی خودکار
Average Error	متوسط خطأ
Bimodal Distribution	توزیع دو حالت
Blind Spot	نقاط کور
Bottleneck	گلوگاه
Bounding Box	کادر محاطی
ByteTrack	الگوریتم بایت‌ترک
Camera Calibration	کالیبراسیون دوربین
Computer Vision	بینایی ماشین

Convolutional Neural Network (CNN)	شبکه عصبی کانولوشنال
CPU	پردازنده مرکزی
CUDA	کودا
Data Fusion Algorithms	الگوریتم‌های ترکیب داده
Deep Learning	یادگیری عمیق
DeepSORT	الگوریتم دیپ‌سارت
Detection and Tracking Model	مدل تشخیص و ردیابی
Driving Behavior	رفتار رانندگی
Edge Computing	پردازش لبه
Embedded Environment	محیط میدانی
Error Rate	نرخ خطا
Euclidean Distance	فاصله اقلیدسی
Field of View (FOV)	میدان دید
Field Test	آزمون میدانی
FPS (Frames Per Second)	نرخ فریم
Frame Preprocessing	فریم‌بندی تصویر
Frame Processing Rate	نرخ پردازش فریم

GPU	پردازنده گرافیکی
GPS Data	داده جی‌بی‌اس
Ground Truth	داده مرجع
Hardware Limitation	محدودیت سخت‌افزاری
ICT	فناوری اطلاعات و ارتباطات
ID Tracking Stability	پایداری ردیابی شناسه
IMU	واحد اندازه‌گیری اینرسی
Intelligent System	سیستم هوشمند
Internet of Things (IoT)	اینترنت اشیا
Kalman Filter	فیلتر کالمن
Kalman/DeepSort Tracker	ردیاب Kalman/DeepSort
License Plate Recognition System	سامانه پلاکخواز
LIDAR Sensor	سنسور لایدار
Lighting Conditions	شرایط نوری
LoRaWAN	شبکه لورا
MacBook Air	لپ‌تاپ مک‌بوک‌ایر
Mean Absolute Error (MAE)	خطای میانگین مطلق

MPS	پردازنده گرافیکی اپل
Multi-camera	چند دوربینه
Multi-Object Tracking	ردیابی چندشیء
Norfair Tracker	ردیاب Norfair
Object Detection	تشخیص شیء
Open Neural Network Exchange (ONNX)	استاندارد ONNX
OpenCV	کتابخانه اپن سی و او
Operational Cost	هزینه عملیاتی
Output Data	داده خروجی
Overspeeding / Illegal Speed	سرعت غیر مجاز
Pipeline	خط لوله پردازش
Pixel Coordinates	مختصات پیکسلی
Proof of Concept	اثبات مفهوم
Pruned Model	مدل سبک سازی شده
Python	زبان پایتون
PyTorch	کتابخانه پای تورچ
Quality of Life	کیفیت زندگی

Quantization	کوانتیزه‌سازی
Quantized Model	مدل کوانتایزشده
Queue	صف
Radar	رادار
RAM	حافظه تصادفی
Raspberry Pi 5 Board	برد رزبری‌پای ۵
Real-time	بلاذرنگ
Real-time Execution	اجرا بلاذرنگ
Real-world Performance	عملکرد در شرایط واقعی
Reference Lines	خطوط مرجع
Resource Consumption	صرف منابع
Road Safety	ایمنی جاده
Scalability	مقیاس‌پذیری
Sensor	حسگر
Single Board Computer	کامپیووتر تک‌بردی
Single Point of Failure	نقطه شکست منفرد
Single-stage Architecture	معماری تک‌مرحله‌ای

Smart City	شهر هوشمند
Smart Warning	هشدار هوشمند
SORT	الگوریتم SORT
Speed Control	کنترل سرعت
Speed Estimation	تخمین سرعت
Speed in Kilometers Per Hour (km/h)	سرعت کیلومتر بر ساعت
Speed Monitoring	ناظارت بر سرعت
Surveillance Camera	دوربین نظارتی
System Resilience	تابآوری سامانه
System Stability	پایداری سیستم
Thermal Throttling	افت عملکرد حرارتی
Thread	نخ پردازش
Throughput	تواد عملیاتی
Traditional System	سامانه سنتی
Traffic Management	مدیریت ترافیک
Traffic Pattern Analysis	تحلیل الگوی ترافیک
Ultralytics	کتابخانه التراالیتیکس

Unique ID	شناسه یکتا
Vehicle Detection	شناسایی خودرو
Vehicles	وسایل نقلیه
WiFi	وای فای
YOLO (You Only Look Once) Model	YOLO مدل
YOLOv3 Model	YOLOv3 مدل
YOLOv4 Model	YOLOv4 مدل
YOLOv5 Model	YOLOv5 مدل
YOLOv6 Model	YOLOv6 مدل
YOLOv7 Model	YOLOv7 مدل
YOLOv8 Model	YOLOv8 مدل
YOLOv8n Style Model	YOLOv8n سبک مدل

Abstract

Given the role of illegal speeding in road accidents and the need for low-cost alternatives in smart cities, this research presents a lightweight, computer-vision-based system for vehicle speed estimation. The system utilizes YOLOv8n for vehicle detection and Norfair for multi-object tracking to record the time each vehicle's center-point crosses two calibrated reference lines, calculating the speed based on the real-world distance between them.

The evaluation, conducted on a MacBook Air M1 with real-world highway footage, demonstrated that the system achieved an average processing rate of 17.09 FPS, successfully tracked 35 vehicles with an estimated ID stability of over 95%, and observed a bimodal speed distribution consistent with traffic patterns. Key limitations include its reliance on manual calibration, the absence of ground truth data for speed error assessment, and testing under relatively ideal conditions. The path for edge deployment on a Raspberry Pi 5 was outlined, involving resolution reduction, model quantization, and tracking optimization. The results indicate that this approach can provide a scalable and cost-effective foundation for speed monitoring in smart city subsystems.

Keywords: Vehicle Speed Estimation, Smart City, Computer Vision, Deep Learning, Edge Computing, Multi-Object Tracking, Traffic Monitoring, YOLOv8

پیوست الف - کد منبع پروژه

```
import datetime
import json
import tempfile
from collections import deque
from urllib.parse import unquote

import jdatetime
import joblib
import jwt
import numpy as np
import pandas as pd
import pytz
import redis
from flask import Flask, request, jsonify, send_file, send_from_directory
from flask_cors import CORS, cross_origin
from apscheduler.schedulers.background import BackgroundScheduler

import tablePagination
from utils import *

app = Flask(__name__, instance_path="/{project_folder_abs_path}/instance")
app.config['SECRET_KEY'] = secret_key
cors = CORS(app, resources={r"/api/*": {"origins": "*"} })

def get_formatted_value(x, suffix=""):
    if pd.isnull(x):
        return "-"
    res = ""
    if abs(x) // 1000000 != 0:
        res = "{:.2f} M".format(x / 1000000)
    elif abs(x) // 1000 != 0:
        res = "{:.2f} K".format(x / 1000)
    else:
        res = "{:.2f} ".format(x)

    res += suffix
    return res

# indicator id = 10, outage_feeders_new
@app.route('/overall', methods=['GET'])
@cross_origin()
def overall():
    params = request.args.to_dict()
    input_params = {}
    for k, v in params.items():
        if v != 'undefined' and v != '' and v != '0' and v!= 'null':
            input_params[k] = v

    default_date = jdatetime.datetime.today() - jdatetime.timedelta(days=2)
    date = input_params.get("date", default_date.strftime("%Y/%m/%d"))

    # ... rest of the code for overall endpoint
```

```

duration_from = input_params.get("duration_from", 30)
duration_to = input_params.get("duration_to")

filters = [
    {
        "type": "filter",
        "data": [
            {
                "type": "date",
                "param": "date",
                "default_value": date,
                "label": "تاریخ",
            },
            {
                "type": "input",
                "param": "duration_from",
                "default_value": duration_from,
                "label": "(دقیقه) خاموشی هر زمان مدت حداقل",
            },
            {
                "type": "input",
                "param": "duration_to",
                "default_value": duration_to,
                "label": "(دقیقه) خاموشی هر زمان مدت حد اکثر",
            },
        ],
        "onclick": [
            {
                "params": input_params,
                "indicatorid": "10",
                "type": "outage_feeders_new",
                "method": "get"
            },
            {
                "params": input_params,
                "indicatorid": "11",
                "type": "outage_feeders_new",
                "method": "get"
            },
        ]
    }
]

try:
    dt_start = jdatetime.datetime.strptime(str(date),
    "%Y/%m/%d").replace(hour=0, minute=0, second=0, microsecond=0).togregorian()
    dt_end = jdatetime.datetime.strptime(str(date),
    "%Y/%m/%d").replace(hour=23, minute=59, second=59,
    microsecond=0).togregorian()

    eng = get_postgresql_engine()
    q = """
        WITH feeders AS (
            SELECT DISTINCT "companyCode", "companyPersianName" AS
"companyName", "PGDSCode"
            FROM "MWS_Tools" t1
            INNER JOIN "MWS_ActorsCompaniesMapping" t2
            ON t1."operatorId" = t2."actorId"
            INNER JOIN "Companies" t3
    """

```

```

        ON t2."companyId" = t3."companyCode"
        WHERE "toolTypeId" = 4
        AND "stationTypeId" = 1
        AND NOT ("PGDSCode" LIKE '%%FC%%' OR "toolName" LIKE
'%%نحوه زن%%')
    )
    SELECT "companyCode", "companyName", count(DISTINCT
t1."PGDSCode") AS "count"
    FROM "FeedersOutages" t1
    INNER JOIN feeders t2
    ON t1."PGDSCode" = t2."PGDSCode"
    AND "start" BETWEEN %s AND %s
"""
q2_inner = """
    SELECT "PGDSCode", count(*) AS "outages_count", extract(epoch
from sum("end" - start)) / 60 AS "outages_duration"
    FROM "FeedersOutages"
    WHERE "start" BETWEEN %s AND %s
"""
if duration_from:
    try:
        duration_from = int(duration_from)
        query_duration_from = '{} minutes'.format(duration_from)
    except:
        return jsonify(error_message("عددی بصورت باید خاموشی زمان مدت", filters)), 400
    q += """ AND "end" - "start" >= interval '{}'
""".format(query_duration_from)
    q2_inner += """ AND "end" - "start" >= interval '{}'
""".format(query_duration_from)
    if duration_to:
        try:
            duration_to = int(duration_to)
            query_duration_to = '{} minutes'.format(duration_to)
        except:
            return jsonify(error_message("عددی بصورت باید خاموشی زمان مدت", filters)), 400
        q += """ AND "end" - "start" <= interval '{}'
""".format(query_duration_to)
        q2_inner += """ AND "end" - "start" <= interval '{}'
""".format(query_duration_to)
    q += """ GROUP BY "companyCode", "companyName" """
    q2_inner += """ GROUP BY "PGDSCode" """
q2 = f"""
    WITH feeders AS (
        {q2_inner}
    )
    SELECT
        t3."companyCode",
        t3."companyPersianName" AS "companyName",
        "customerKind" / 10 AS "customerKind",
        t2."PGDSCode",
        "outages_count", "outages_duration",
        sum("customerCount") AS "customerCount",
        sum("fahamCount") AS "fahamCount"
    FROM "Assets"."FeederCustomerStats" t1
    INNER JOIN feeders t2

```

```

        ON t1."PGDSCode" = t2."PGDSCode"
        INNER JOIN "Companies" t3
        ON t1."companyCode" = t3."companyCode"
        GROUP BY 1, 2, 3, 4, 5, 6
    """
    q_incompatible = """
        SELECT t1."PGDSCode", "customerKind" / 10 AS "customerKind",
        count(DISTINCT t1."meterId") AS "incompatible_count"
        FROM "Outages"."IncompatibleCustomers" t1
        INNER JOIN "DL"."CustomersInfo" t2
        ON t1."meterId" = t2."meterMRID" AND t1."companyCode" =
    t2."companyCode"
        AND time BETWEEN %s AND %s
        GROUP BY 1, 2
    """
    df = pd.read_sql(q, eng, params=(dt_start, dt_end))
    df2 = pd.read_sql(q2, eng, params=(dt_start, dt_end))
    df_incompatible = pd.read_sql(q_incompatible, eng, params=(dt_start,
    dt_end))
    eng.dispose()

    df2 = pd.merge(df2, df_incompatible, on=["PGDSCode", "customerKind"],
    how="left")
    df2["incompatible_count"] = df2["incompatible_count"].fillna(0)
    df2["match_count"] = df2["fahamCount"] - df2["incompatible_count"]
    df2["match_count"] = df2["match_count"] / df2["fahamCount"] * 100
    df2["match_count"] = df2["match_count"].replace([np.inf, -np.inf],
    np.nan)
    df2["match_count"] = df2["match_count"].round(1).fillna("-")
    df2["fahamCount"] = df2["fahamCount"].astype(int)
    kinds = pd.DataFrame([
        {"customerKind": 1, "usage": "خانگی"},
        {"customerKind": 2, "usage": "عمومی"},
        {"customerKind": 3, "usage": "کشاورزی"},
        {"customerKind": 4, "usage": "منعتی"},
        {"customerKind": 5, "usage": "تجاری"},
    ])
    df2 = pd.merge(df2, kinds, on="customerKind", how="left")
    df2 = df2.sort_values([" companyName", "usage"], ignore_index=True)
    df2["customerCount"] = df2["customerCount"].astype(int)
    df2["outages_count"] = df2["outages_count"].astype(int)
    df2["outages_duration"] = df2["outages_duration"].astype(int)
    df2["i"] = list(range(1, len(df2) + 1))
    df2["customers"] = "fa fa-users"
    df2["details"] = "chart-area"
    df2["date"] = jdatetime.datetime.strptime(str(date),
    "%Y/%m/%d").strftime("%Y/%m/%d")
    df2 = df2[["i", "date", "companyName", "usage", "PGDSCode",
    "outages_count", "outages_duration", "customerCount", "fahamCount",
    "match_count", "customers", "details", "companyCode", "customerKind"]]
    df2 = df2.astype(str)
    df2 = df2.where(pd.notnull(df2), None)
    table_ = [
        {"type": "joined",
        "groupid": 0,
        "title": "tbl1",
        "paging": True,

```

```

        "searching": True,
        "export": [{"persian_name": "اکسل خروجی", "type": "excel",
"default": True}],
        "persian_title": "خاموشی‌ها آمار",
        "rowcount": len(df2),
        "columns": df2.columns.tolist(),
        **tablePagination.save_redis(redis_config, df2),
        "persian_columns": ["تعریفه", "توزيع شرکت", "تاریخ", "ردیف"],
        "مشترکین تعداد", "(دقیقه) خاموشی زمان مدت", "خاموشی تعداد",
        "مشترکین", "مصرف نمودار", "مشترکین", "مدام با فهم مطابق",
        "فهام مشترکین", "companyCode", "customerKind"],
        "columns_info": [
            {"type": "text", "isShow": True, "width": "1%"},
            {"type": "text", "isShow": True, "width": "6%"},
            {"type": "icon", "isShow": True, "width": "1%"},
            {"type": "icon", "isShow": True, "width": "1%"},
            {"type": "text", "isShow": False, "width": "6%"},
            {"type": "text", "isShow": False, "width": "6%"},

        ],
        "onclick": {
            "oncell": {
                "inner": [
                    {
                        "column": "@customers",
                        "indicatorid": "13",
                        "type": "outage_feeders_new",
                        "method": "get",
                        "params": {
                            "companyCode": "@companyCode",
                            "customerKind": "@customerKind",
                            "PGDSCode": "@PGDSCode",
                            "outages_count": "@outages_count",
                            "outages_duration": "@outages_duration",
                            "date": date,
                            "duration_from": duration_from,
                            "duration_to": duration_to,
                        },
                    },
                    {
                        "column": "@details",
                        "indicatorid": "12",
                        "type": "outage_feeders_new",
                        "method": "get",
                        "params": {
                            "PGDSCode": "@PGDSCode",
                            "companyCode": "@companyCode",
                            "date": date
                        },
                    },
                ],
            }
        }
    }
}

```

```

        ],
    }
},
]

set_tooltip = lambda row: """
    {companyName}
    <br>
    تعداد خاموشی دارای فیدر: {count:.0f}
""".strip().format(companyName=row["companyName"],
count=row["count"])
df["tooltip"] = df.apply(set_tooltip, axis=1)
df["onclick"] = df.apply(lambda row: [
{
    "indicatorid": 11,
    "type": "outage_feeders_new",
    "method": "get",
    "params": {
        "companyCode": row["companyCode"],
        "companyName": row["companyName"],
        "date": date,
        "duration_from": duration_from,
        "duration_to": duration_to,
    }
},
], axis=1)
df["id"] = df["companyCode"]
df["y"] = 1

chart_map = json.load(open("jsons/map.json", "r"))
chart_map["title"]["text"] = ""
chart_map["series"][0]["data"] = df[["y", "id", "onclick",
"tooltip"]].to_dict(orient="records")

ret_json = filters.copy()
ret_json += [{"type": "rich-table", "data": table_}]
ret_json += [{"type": "map", "data": chart_map, "className": "col-md-12"}]
return jsonify(ret_json), 200
except Exception as e:
    logger("ERROR @table:", e)
    return jsonify(error_message("خطایی در داده است.", filters)), 400

# indicator id = 11, outage_feeders_new
@app.route('/table', methods=['GET'])
@cross_origin()
def table():
    params = request.args.to_dict()
    input_params = {}
    for k, v in params.items():
        if v != 'undefined' and v != '' and v != '0' and v!= 'null':
            input_params[k] = v

    default_date = jdatetime.datetime.today() - jdatetime.timedelta(days=2)
    date = input_params.get("date", default_date.strftime("%Y/%m/%d"))


```

```

duration_from = input_params.get("duration_from", 30)
duration_to = input_params.get("duration_to")
company_code = input_params.get("companyCode")
company_name = input_params.get("companyName")

try:
    dt_start = jdatetime.datetime.strptime(str(date),
    "%Y/%m/%d").replace(hour=0, minute=0, second=0, microsecond=0).togregorian()
    dt_end = jdatetime.datetime.strptime(str(date),
    "%Y/%m/%d").replace(hour=23, minute=59, second=59,
    microsecond=0).togregorian()
    q = """
        WITH feeders AS (
            SELECT DISTINCT "companyCode", "companyPersianName" AS
"companyName", "toolName", "stationName", "PGDSCode"
            FROM "MWS_Tools" t1
            INNER JOIN "MWS_ActorsCompaniesMapping" t2
            ON t1."operatorId" = t2."actorId"
            INNER JOIN "Companies" t3
            ON t2."companyId" = t3."companyCode"
            WHERE "toolTypeId" = 4
            AND "stationTypeId" = 1
            AND NOT ("PGDSCode" LIKE '%%FC%%' OR "toolName" LIKE
'%%زنگ%%')
        )
        SELECT "companyCode", "companyName", "toolName", "stationName",
t1."PGDSCode", count(*) AS "outages_count", extract(epoch from sum("end" -
start)) / 60 AS "outages_duration"
        FROM "FeedersOutages" t1
        INNER JOIN feeders t2
        ON t1."PGDSCode" = t2."PGDSCode"
        AND "start" BETWEEN %s AND %s
    """
    q_incompatible = """
        SELECT t1."PGDSCode", count(DISTINCT t1."meterId") AS
"incompatible_count"
        FROM "Outages"."IncompatibleCustomers" t1
        INNER JOIN "DL"."CustomersInfo" t2
        ON t1."meterId" = t2."meterMRID" AND t1."companyCode" =
t2."companyCode"
        AND time BETWEEN %s AND %s
    """
    if duration_from:
        try:
            duration_from = int(duration_from)
            query_duration_from = '{} minutes'.format(duration_from)
        except:
            return jsonify(error_message("عددی بصورت باید خاموشی زمان مدت".".شود وارد
))), 400
            q += "" AND "end" - "start" >= interval '{}'
            """.format(query_duration_from)
    if duration_to:
        try:
            duration_to = int(duration_to)
            query_duration_to = '{} minutes'.format(duration_to)
        except:
            return jsonify(error_message("عددی بصورت باید خاموشی زمان مدت"))

```

```

        q += """ AND "end" - "start" <= interval '{}'
        """.format(query_duration_to)
        if company_code:
            q += """ AND "companyCode" = {} """.format(company_code)
            q_incompatible += """ AND t1."companyCode" = {} """
        """.format(company_code)
        q += """ GROUP BY "companyCode", "CompanyName", "toolName",
"stationName", t1."PGDSCode" """
        q_incompatible += """ GROUP BY "PGDSCode" """
        eng = get_postgresql_engine()
        df = pd.read_sql(q, eng, params=(dt_start, dt_end))
        df2 = pd.read_sql("""
            SELECT "PGDSCode",
                sum(case when floor("customerKind" / 10) = 1 then
"customerCount" else 0 end) / sum("customerCount") * 100 AS "ck1",
                sum(case when floor("customerKind" / 10) = 2 then
"customerCount" else 0 end) / sum("customerCount") * 100 AS "ck2",
                sum(case when floor("customerKind" / 10) = 3 then
"customerCount" else 0 end) / sum("customerCount") * 100 AS "ck3",
                sum(case when floor("customerKind" / 10) = 4 then
"customerCount" else 0 end) / sum("customerCount") * 100 AS "ck4",
                sum(case when floor("customerKind" / 10) = 5 then
"customerCount" else 0 end) / sum("customerCount") * 100 AS "ck5",
                sum("customerCount") AS "customerCount",
                sum("fahamCount") AS "fahamCount"
            FROM "Assets"."FeederCustomerStats"
            GROUP BY "PGDSCode"
        """, eng)
        df_incompatible = pd.read_sql(q_incompatible, eng, params=(dt_start,
dt_end))
        eng.dispose()

        df = pd.merge(df, df2, how="left", on="PGDSCode")
        df = pd.merge(df, df_incompatible, how="left", on="PGDSCode")
        df["incompatible_count"] = df["incompatible_count"].fillna(0)
        df["match_count"] = df["fahamCount"] - df["incompatible_count"]
        df["match_count"] = df["match_count"] / df["fahamCount"] * 100
        df["match_count"] = df["match_count"].replace([np.inf, -np.inf],
np.nan)
        df[["ck1", "ck2", "ck3", "ck4", "ck5", "match_count"]] = df[["ck1",
"ck2", "ck3", "ck4", "ck5", "match_count"]].round(1)
        df[["customerCount", "fahamCount"]] = df[["customerCount",
"fahamCount"]].applymap(lambda x: int(x) if pd.notnull(x) else "-")
        df[["ck1", "ck2", "ck3", "ck4", "ck5", "match_count"]] = df[["ck1",
"ck2", "ck3", "ck4", "ck5", "match_count"]].fillna("-")
        df["outages_duration"] = df["outages_duration"].astype(int)
        df = df.sort_values("outages_duration", ascending=False,
ignore_index=True)
        df["i"] = list(range(1, len(df) + 1))
        df["details"] = "chart-area"
        df["customers"] = "fa fa-users"
        df = df[["i", "CompanyName", "stationName", "toolName", "PGDSCode",
"ck1", "ck2", "ck3", "ck4", "ck5", "outages_count", "outages_duration",
"customerCount", "fahamCount", "match_count", "customers", "details",
"companyCode"]]
        df = df.astype(str)

```

```

df = df.where(pd.notnull(df), None)
table_ = [
    {
        "type": "joined",
        "groupid": 0,
        "title": "company",
        "paging": True,
        "searching": True,
        "export": [{"persian_name": "اکسل خروجی", "type": "excel",
"default": True}],
        "persian_title": "خاموشی",
        "rowcount": len(df),
        "columns": df.columns.tolist(),
        **tablePagination.save_redis(redis_config, df),
        "persian_columns": [
            "فیدر نام", "پست", "توزيع شرکت", "ردیف", "دیگر شناسه", "صنعتی", "(درصد) کشاورزی", "(درصد) عمومی", "(درصد) خانگی", "(درصد) تجاری", "(درصد) کل", "(دقیقه) خاموشی زمان مجموع", "فهام مشترکین", "فهام مددام", "فهام تطابق", "فهام با فهم", "مصرف نمودار", "مشترکین", "مدام", "companyCode"],
        "columns_info": [
            {"type": "text", "isShow": True, "width": "1%"}, {"type": "text", "isShow": True, "width": "6%"}, {"type": "icon", "isShow": True, "width": "1%"}, {"type": "icon", "isShow": True, "width": "1%"}, {"type": "text", "isShow": False, "width": "6%"}, ],
        "onclick": {
            "oncell": {
                "inner": [
                    {
                        "column": "@details",
                        "indicatorid": "12",
                        "type": "outage_feeders_new",
                        "method": "get",
                        "params": {
                            "PGDSCode": "@PGDSCode",
                            "companyCode": "@companyCode",
                            "date": date
                        },
                    },
                    {
                        "column": "@customers",
                        "indicatorid": "13",
                        "type": "outage_feeders_new",
                        "method": "get",
                    }
                ]
            }
        }
    }
]

```

```

        "params": {
            "PGDSCode": "@PGDSCode",
            "companyCode": "@companyCode",
            "date": date
        },
    },
],
}
},
[]

ret_json = [{"type": "rich-table", "data": table_}]
return jsonify(ret_json), 200
except Exception as e:
    logger("ERROR @table:", e)
    return jsonify(error_message("." . "است داده رخ خطایی")), 400

# indicator id = 12, outage_feeders_new
@app.route('/feeder_chart', methods=['GET'])
def feeder_chart():
    params = request.args.to_dict()
    input_params = {}
    for k, v in params.items():
        if v != 'undefined' and v != '' and v != '0' and v!= 'null':
            input_params[k] = v

    default_date = jdatetime.datetime.today() - jdatetime.timedelta(days=1)
    date = input_params.get("date", default_date.strftime("%Y/%m/%d"))
    pgdscode = input_params.get("PGDSCode")
    company_code = input_params.get("companyCode")

    try:
        dt_start = jdatetime.datetime.strptime(str(date),
                                              "%Y/%m/%d").replace(hour=0, minute=0, second=0, microsecond=0).togregorian()
        dt_end = jdatetime.datetime.strptime(str(date),
                                             "%Y/%m/%d").replace(hour=23, minute=59, second=59,
                                             microsecond=0).togregorian()

        engine = get_postgresql_engine()
        outages = pd.read_sql("""
            SELECT "start", "end"
            FROM "FeedersOutages"
            WHERE "start" BETWEEN %s AND %s
            AND "PGDSCode" = %s
        """, engine, params=(dt_start, dt_end, pgdscode))
        df = pd.read_sql("""
            SELECT "time", "powerValue" AS "consumption"
            FROM "MWS_ToolPower" t1
            WHERE time BETWEEN %(start_tt)s AND %(end_tt)s
            AND "toolPGDSCode" = %(pgdscode)s
            AND "formulaTypeId" IN (66, 52)
        """, engine, params={"start_tt": dt_start, "end_tt": dt_end,
                            "pgdscode": pgdscode})
        engine.dispose()

        df_table = outages.copy()
    
```



```

chart = json.load(open("jsons/chart_line_mws.json", "r"))
chart["title"]["text"] = "فیدر بار {".format(pgdscode)
chart["xAxis"]["title"]["text"] = "زمان"
chart["xAxis"]["min"] =
datetime.datetime.fromtimestamp(df_consumption["timestamp"].min() /
1000).replace(
    hour=0, minute=0, second=0).timestamp() * 1000
chart["xAxis"]["max"] =
datetime.datetime.fromtimestamp(df_consumption["timestamp"].max() /
1000).replace(
    hour=23, minute=59, second=59).timestamp() * 1000
chart["yAxis"]["title"]["text"] = "بار (مکاوات)"
# chart["yAxis"]["min"] = 0
# chart["yAxis"]["max"] = df_consumption["consumption"].max() * 1.3
chart["series"] = [
{
    "type": "spline",
    "name": "لحظه اي بار",
    "keys": ["x", "y"],
    "data": df_consumption[["timestamp",
"consumption"]].values.tolist(),
    "color": "#9E9E9E",
    "zIndex": 2,
    "marker": {"enabled": False}
},
]
plot_bands = []
for _, row in outages.iterrows():
    plot_bands.append({
        "color": "#FF4C4C60",
        "from": row["start"].to_pydatetime().timestamp() * 1000,
        "to": row["end"].to_pydatetime().timestamp() * 1000,
    })
chart["xAxis"]["plotBands"] = plot_bands
ret_json = [{"type": "rich-table", "data": table_}]
ret_json += [{"type": "chart", "data": chart, "className": "col-md-12"}]
return jsonify(ret_json), 200
except Exception as e:
    logger("ERROR @meter_chart:", e)
    return jsonify(error_message("است داده رخ خطایی")), 400

# indicatorId = 13, outage_feeders_new
@app.route('/customers', methods=['GET'])
def customers():
    params = request.args.to_dict()
    input_params = {}
    for k, v in params.items():
        if v != 'undefined' and v != '' and v != '0' and v!= 'null':
            input_params[k] = v

    default_date = jdatetime.datetime.today() - jdatetime.timedelta(days=1)
    date = input_params.get("date", default_date.strftime("%Y/%m/%d"))
    company_code = input_params.get("companyCode")
    pgdscode = input_params.get("PGDSCode")

```

```

try:
    engine = get_postgresql_engine()
    info = pd.read_sql("""
        SELECT t1."subscriptionBillId" AS "billIdentity", "meterMRID" AS
    "meterId", "counterTypeName", "customerName", floor("customerKind" / 10) AS
    "customerKind", t1."companyCode"
        FROM "Assets"."MVFedeerCustomers" t1
        INNER JOIN "DL"."CustomersInfo" t2
        ON t1."subscriptionBillId" = t2."customerBillingMRID"
        AND "PGDSCode" = %(pgdscode)s AND t1."companyCode" =
%(company_code)s
        """ , engine, params={"company_code": company_code, "pgdscode": pgdscode})
    engine.dispose()
    if info.empty:
        return jsonify(error_message)), 400
    info["i"] = range(1, len(info) + 1)
    info["chart"] = "chart-area"
    kinds = pd.DataFrame([
        {"customerKind": 1, "usage": "خانگی", "color": "#68C17C"}, 
        {"customerKind": 2, "usage": "عمومی", "color": "#FFD600"}, 
        {"customerKind": 3, "usage": "کشاورزی", "color": "#FFAB00"}, 
        {"customerKind": 4, "usage": "صنعتی", "color": "#D50000"}, 
        {"customerKind": 5, "usage": "تجاری", "color": "#9E9E9E"}, 
    ])
    info = pd.merge(info, kinds, how="left", on="customerKind")
    df_cards = info.groupby("usage").agg(count=("billIdentity", "count"), color=("color", "first")).reset_index()
    cards = []
    for _, row in df_cards.iterrows():
        cards.append({
            "className": "col-md-3",
            "firstColor": "#E0E0E0",
            "secondColor": row["color"],
            "iconWidth": "64px",
            "iconImg": "/static_files/{}.format("rating-64.png"),
            "cardHeader": "<span style='color:white;font-size:18px;dir='rtl'>" + "{}".format(
                row["usage"]) + "</span>",
            "text": "<span style='font-size:16px;dir='rtl'>" +
                    "مشترکین تعداد " + "{:.0f}".format(row["count"]) +
    "<br>" +
                    "</span>",
        })
    info = info[["i", "billIdentity", "meterId", "counterTypeName",
    "customerName", "usage", "chart", "companyCode"]]
    info = info.astype(object)
    info = info.where(pd.notnull(info), None)
    columns = ["نام", "کنتور نوع", "کنتور سریال", "قیص شناسه", "ردیف", "مشترک", "مصرف نمودار", "تعریفه", "companyCode"]
    table_ = [
        {"type": "joined",
        "groupid": 0,
        "title": "customers",
        "paging": True,

```

```

        "searching": True,
        "export": [{"persian_name": "اکسل خروجی", "type": "excel",
"default": True}],
        "persian_title": "فیدر مشترکین {}".format(pgdscode),
        "rowcount": len(info),
        "columns": info.columns.tolist(),
        **tablePagination.save_redis(redis_config, info),
        "pageLength": 10,
        "persian_columns": columns,
        "columns_info": [
            {"type": "text", "isShow": True, "width": "1%"},
            {"type": "text", "isShow": True, "width": "8%"},
            {"type": "icon", "isShow": True, "width": "1%"},
            {"type": "text", "isShow": False, "width": "8%"},
        ],
        "onclick": {
            "oncell": {
                "inner": [
                    {
                        "column": "@chart",
                        "indicatorid": "14",
                        "type": "outage_feeders_new",
                        "method": "get",
                        "params": {
                            "billIdentity": "@billIdentity",
                            "meterId": "@meterId",
                            "companyCode": "@companyCode",
                            "PGDSCode": pgdscode,
                            **input_params
                        },
                    },
                ],
            }
        }
    ],
}

ret_json = [{"type": "section-card", "data": cards}]
ret_json += [{"type": "rich-table", "data": table_}]
return jsonify(ret_json), 200
except Exception as e:
    logger("ERROR @customers:", e)
    return jsonify(error_message("است داده رخ خطای")), 400

```

```

# indicator id = 14, outage_feeders_new
@app.route('/meter_chart', methods=['GET'])
def meter_chart():
    params = request.args.to_dict()
    input_params = {}
    for k, v in params.items():
        if v != 'undefined' and v != '' and v != '0' and v!= 'null':
            input_params[k] = v

```

```

default_date = jdatetime.datetime.today() - jdatetime.timedelta(days=1)
date = input_params.get("date", default_date.strftime("%Y/%m/%d"))
company_code = input_params.get("companyCode")
meter_id = input_params.get("meterId")
bill_identity = input_params.get("billIdentity")
pgdscode = input_params.get("PGDSCode")

try:
    dt_start = jdatetime.datetime.strptime(str(date),
    "%Y/%m/%d").replace(hour=0, minute=0, second=0, microsecond=0).togregorian()
    dt_end = jdatetime.datetime.strptime(str(date),
    "%Y/%m/%d").replace(hour=23, minute=59, second=59,
    microsecond=0).togregorian()

    engine = get_postgresql_engine()
    # parent_outages = pd.read_sql("""
    #     SELECT "start", "end"
    #     FROM "FeedersOutages"
    #     WHERE "start" BETWEEN %s AND %s
    #     AND "PGDSCode" = %s
    #     """, engine, params=(dt_start, dt_end, pgdscode))
    # outages = pd.read_sql("""
    #     SELECT "start", "end"
    #     FROM "CustomersOutages"
    #     WHERE "start" BETWEEN %s AND %s
    #     AND "meterId" = %s
    #     AND "companyCode" = %s
    #     """, engine, params=(dt_start, dt_end, meter_id, company_code))
    df = pd.read_sql("""
        SELECT time, "5/1.0.15.4.0.255/3" * "CT" * "PT" AS consumption
        FROM "MetersBasicInfo" t1
        INNER JOIN "MeterInfo_5-1.0.15.4.0.255-3" t2
        ON t1."meterMRID" = t2."meterId" AND t1."companyCode" =
    t2."companyCode"
        AND t1."meterMRID" = %(meter_id)s AND t1."companyCode" =
    %(company_code)s
        AND time BETWEEN %(start_tt)s AND %(end_tt)s
    """ , engine, params={"start_tt": dt_start,
                           "end_tt": dt_end,
                           "meter_id": meter_id,
                           "company_code": company_code})
    engine.dispose()
    if df.empty:
        return jsonify(error_message("ندازد وجود نمایش جهت داده ای")),
400

    df = df.sort_values("time", ignore_index=True)
    interval = ((df["time"] - df["time"].shift(1)).dt.total_seconds() /
60).min()
    if interval < 30:
        times = pd.DataFrame({"time": pd.date_range(start=dt_start,
end=dt_end, freq="15min")})
    elif interval < 60:
        times = pd.DataFrame({"time": pd.date_range(start=dt_start,
end=dt_end, freq="30min")})
    else:
        times = pd.DataFrame({"time": pd.date_range(start=dt_start,

```

```

end=dt_end, freq="60min"))
df = pd.merge(df, times, on="time", how="outer")
df = df.sort_values("time", ignore_index=True)
# df["timestamp"] = df["time"].astype(int) // 10 ** 9 * 1000
df["timestamp"] = df["time"].apply(lambda x:
x.to_pydatetime().timestamp() * 1000)
df["is_out"] = df["consumption"].isnull() | (df["consumption"] == 0)
df["group"] = df["is_out"].ne(df["is_out"].shift()).cumsum()
out_ranges = df[df["is_out"]].groupby("group")["time"].agg(["first",
"last"]).reset_index()

df_consumption = df[["timestamp", "consumption"]].copy()
df_consumption["color"] = "#9E9E9E"
df_consumption = df_consumption.astype(object)
df_consumption = df_consumption.where(pd.notnull(df_consumption),
None)

chart = json.load(open("jsons/chart_line_meter.json", "r"))
chart["title"]["text"] = "قبض شناسه با مشترک بار {}".format(bill_identity)
chart["xAxis"]["title"]["text"] = "زمان"
chart["xAxis"]["min"] =
datetime.datetime.fromtimestamp(df_consumption["timestamp"].min() /
1000).replace(
    hour=0, minute=0, second=0).timestamp() * 1000
chart["xAxis"]["max"] =
datetime.datetime.fromtimestamp(df_consumption["timestamp"].max() /
1000).replace(
    hour=23, minute=59, second=59).timestamp() * 1000
chart["yAxis"]["title"]["text"] = "بار (وات)"
# chart["yAxis"]["min"] = 0
# chart["yAxis"]["max"] = df_consumption["consumption"].max() * 1.3
chart["series"] = [
{
    "type": "spline",
    "name": "لحظه‌ای بار",
    "keys": ["x", "y"],
    "data": df_consumption[["timestamp",
"consumption"]].values.tolist(),
    "color": "#9E9E9E",
    "zIndex": 2,
    "marker": {"enabled": False}
},
]
plot_bands = []
# for _, row in parent_outages.iterrows():
#     plot_bands.append({
#         "color": "#FF4C4C40",
#         "from": row["start"].to_pydatetime().timestamp() * 1000,
#         "to": row["end"].to_pydatetime().timestamp() * 1000,
#         "label": {
#             "text": "دستی بالا فیدر خاموشی",
#             "align": "center"
#         }
#     })
# for _, row in outages.iterrows():
#     plot_bands.append({

```

```

#           "color": "#FF4C4C60",
#           "from": row["start"].to_pydatetime().timestamp() * 1000,
#           "to": row["end"].to_pydatetime().timestamp() * 1000,
#       })
for _, row in out_ranges.iterrows():
    plot_bands.append({
        "color": "#FF4C4C60",
        "from": row["first"].to_pydatetime().timestamp() * 1000,
        "to": row["last"].to_pydatetime().timestamp() * 1000,
    })

chart["xAxis"]["plotBands"] = plot_bands
ret_json = [{"type": "chart", "data": chart, "className": "col-md-12"}]
return jsonify(ret_json), 200
except Exception as e:
    logger("ERROR @meter_chart:", e)
    return jsonify(error_message("است داده رخ خطایی")), 400

# indicatorId = 15, outage_feeders_new
@app.route('/customers_stats', methods=['GET'])
def customers_stats():
    params = request.args.to_dict()
    input_params = {}
    for k, v in params.items():
        if v != 'undefined' and v != '' and v != '0' and v!= 'null':
            input_params[k] = v

    default_date = jdatetime.datetime.today() - jdatetime.timedelta(days=1)
    date = input_params.get("date", default_date.strftime("%Y/%m/%d"))
    company_code = input_params.get("companyCode")
    customer_kind = input_params.get("customerKind")
    pgdscode = input_params.get("PGDSCode")
    outages_count = input_params.get("outages_count")
    outages_duration = input_params.get("outages_duration")
    duration_from = input_params.get("duration_from")
    duration_to = input_params.get("duration_to")

    try:
        dt_start = jdatetime.datetime.strptime(str(date),
                                              "%Y/%m/%d").replace(hour=0, minute=0, second=0, microsecond=0).togregorian()
        dt_end = jdatetime.datetime.strptime(str(date),
                                             "%Y/%m/%d").replace(hour=23, minute=59, second=59,
                                             microsecond=0).togregorian()
        q2_inner = """
            SELECT "PGDSCode",
                   count(*)                               AS
        "outages_count",
                   extract(epoch from sum("end" - start)) / 60 AS
        "outages_duration"
            FROM "FeedersOutages"
            WHERE "start" BETWEEN %(dt_start)s AND %(dt_end)s
        """
        if duration_from:
            try:
                duration_from = int(duration_from)

```

```

        query_duration_from = '{} minutes'.format(duration_from)
    except:
        return jsonify(error_message("زمان مدت باید خاموشی باشد وارد شود.")), 400
    q2_inner += """ AND "end" - "start" >= interval '{}'
""".format(query_duration_from)
    if duration_to:
        try:
            duration_to = int(duration_to)
            query_duration_to = '{} minutes'.format(duration_to)
        except:
            return jsonify(error_message("زمان مدت باید خاموشی باشد وارد شود.")), 400
        q2_inner += """ AND "end" - "start" <= interval '{}'
""".format(query_duration_to)
    q2_inner += """ GROUP BY "PGDSCode" """
    q2 = f"""
        WITH feeders AS (
            {q2_inner}
        )
        SELECT t3."meterMRID" AS "meterId", t3."customerBillingMRID" AS
"billIdentity", t3."counterTypeName", t3."customerName", floor("customerKind"
/ 10) AS "customerKind", t3."companyCode"
        FROM "Assets"."MVFedeerCustomers" t1
        INNER JOIN feeders t2
        ON t1."PGDSCode" = t2."PGDSCode" AND outage_duration =
%(outage_duration)s AND outages_count = %(outages_count)s AND t2."PGDSCode" =
%(pgdscode)s
        INNER JOIN "DL"."CustomersInfo" t3
        ON t1."subscriptionBillId" = t3."customerBillingMRID" AND
t1."companyCode" = %(companyCode)s AND "customerKind" / 10 = %(customerKind)s
    """
    engine = get_postgresql_engine()
    info = pd.read_sql(q2, engine, params={"companyCode": company_code,
                                             "pgdscode": pgdscode,
                                             "dt_start": dt_start,
                                             "dt_end": dt_end,
                                             "customerKind": customer_kind,
                                             "outages_count": outages_count,
                                             "outages_duration": outages_duration,
                                             })
    engine.dispose()
    if info.empty:
        return jsonify(error_message("مشترکی پیدا نشد.")), 400
    info["i"] = range(1, len(info) + 1)
    info["chart"] = "chart-area"
    kinds = pd.DataFrame([
        {"customerKind": 1, "usage": "خانگی", "color": "#68C17C"}, {"customerKind": 2, "usage": "عمومی", "color": "#FFD600"}, {"customerKind": 3, "usage": "کشاورزی", "color": "#FFAB00"}, {"customerKind": 4, "usage": "صنعتی", "color": "#D50000"}, {"customerKind": 5, "usage": "تجاری", "color": "#9E9E9E"}])

```

```

        ])
        info = pd.merge(info, kinds, how="left", on="customerKind")
        info = info[["i", "billIdentity", "meterId", "counterTypeName",
"customerName", "usage", "chart", "companyCode"]]
        info = info.astype(object)
        info = info.where(pd.notnull(info), None)
        columns = ["نام", "کنتور نوع", "کنتور سریال", "ردیف", "قبض شناسه", "مصرف نمودار", "مشترک", "تعریف", "companyCode"]
        table_ = [
            {
                "type": "joined",
                "groupid": 0,
                "title": "customers",
                "paging": True,
                "searching": True,
                "export": [{"persian_name": "اکسل خروجی", "type": "excel",
"default": True}],
                "persian_title": "فیدر مشترکین {}".format(pgdscode),
                "rowcount": len(info),
                "columns": info.columns.tolist(),
                **tablePagination.save_redis(redis_config, info),
                "pageLength": 10,
                "persian_columns": columns,
                "columns_info": [
                    {"type": "text", "isShow": True, "width": "1%"}, {"type": "text", "isShow": True, "width": "8%"}, {"type": "icon", "isShow": True, "width": "1%"}, {"type": "text", "isShow": False, "width": "8%"}, ],
                "onclick": {
                    "oncell": {
                        "inner": [
                            {
                                "column": "@chart",
                                "indicatorid": "14",
                                "type": "outage_feeders_new",
                                "method": "get",
                                "params": {
                                    "billIdentity": "@billIdentity",
                                    "meterId": "@meterId",
                                    "companyCode": "@companyCode",
                                    **input_params
                                },
                                },
                            ],
                        },
                    }
                },
            ],
        ]
    }

    ret_json = [{"type": "rich-table", "data": table_}]
    return jsonify(ret_json), 200
except Exception as e:
    logger("ERROR @customers:", e)
    return jsonify(error_message("است داده رخ خطایی", 400

```

```

def job_materialized_view():
    logger("INFO: job_materialized_view started.")
    try:
        """
        create materialized view "Assets"."FeederCustomerStats" as
        SELECT t1."PGDSCode",
               t2."companyCode",
               t2."customerKind",
               count(*) AS "customerCount"
        FROM "Assets"."MVFedeerCustomers" t1
              JOIN "DL"."CustomersInfo" t2 ON
t1."subscriptionBillId"::text = t2."customerBillingMRID"::text
              GROUP BY t1."PGDSCode", t2."companyCode", t2."customerKind";
        """
        eng = get_postgresql_engine()
        with eng.connect() as con:
            con.execute(text(""" refresh materialized view
"Assets"."FeederCustomerStats"; """))
            con.commit()
        eng.dispose()
    except Exception as e:
        logger("ERROR @job_materialized_view:", e)
    finally:
        logger("INFO: job_materialized_view finished.")

if __name__ == '__main__':
    scheduler = BackgroundScheduler(job_defaults={"misfire_grace_time": 4 *
60 * 60})
    scheduler.add_job(job_materialized_view, "cron", hour=4, minute=0,
second=0, misfire_grace_time=4 * 60 * 60) #,
next_run_time=datetime.datetime.now())
    scheduler.start()

app.run(host='0.0.0.0', port=54788, debug=False, use_reloader=False)

```