



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه کارشناسی

سیستم تشخیص خواب‌آلودگی رانندگان با شبکه‌های عصبی پیچشی

نگارش

یاسمن حق‌بین

استاد راهنما

دکتر رحمتی

مرداد ۱۴۰۱

به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تاریخ: مرداد ۱۴۰۱

تعهد نامه اصالت اثر

اینجانب یاسمن حق‌بین آسیابر متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آن‌ها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک همسطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

یاسمن حق‌بین آسیابر

مرداد ۱۴۰۱

سپاس‌گزاری

سپاس پروردگار یکتا را که هستی‌مان بخشید و به طریق علم و دانش رهنمودمان شد و به همنشینی رهروان علم و دانش مفتخرمان نمود. از خانواده‌ی مهربانم که همواره در طول زندگی برای شادکامی من زحماتی به جان خریدند، تشکر می‌کنم.

بی تردید تهیه‌ی این گزارش بدون راهنمایی‌های ارزشمند استاد بزرگوار جناب آقای دکتر محمد رحمتی میسر نمی‌شد. پس بر خود واجب می‌دانم از راهنمایی‌های ایشان صمیمانه سپاس‌گزاری کنم.

از استاد گرانقدر، جناب آقای دکتر جوانمردی که زحمت داوری این پایان‌نامه را به عهده داشتند و به مطالعه‌ی آن پرداختند، سپاس‌گزاری می‌کنم.

یاسمن تقی‌زین

مرداد ۱۴۰۱

چکیده

خواب‌آلودگی رانندگان یکی از علل شایع تصادفات جاده‌ای است که منجر به جراحات، مرگ و خسارات اقتصادی قابل توجهی به رانندگان، خانواده‌ها و جامعه می‌شود. در میان الگوریتم‌های یادگیری عمیق، نوع خاصی از شبکه‌های عصبی عمیق به نام شبکه‌های عصبی پیچشی وجود دارد که عملکرد بسیار خوبی در بینایی کامپیوتری دارند، زیرا قادر به یافتن الگوها و تشخیص ویژگی‌ها در بین تصاویر هستند. لذا، این پروژه سیستمی را به کمک شبکه عصبی پیچشی برای تشخیص خواب‌آلودگی راننده پیشنهاد می‌کند. برای این کار ابتدا سه شبکه عصبی پیچشی آموزش داده می‌شوند تا وضعیت چشم راننده را بررسی کنند. برای تشخیص چهره و استخراج ناحیه چشم از تصاویر صورت، از الگوریتم تشخیص چهره ویولا و جونز استفاده شده است. پس از آموزش هر سه شبکه عصبی، مناسب‌ترین شبکه از نظر دقت و سرعت برای استفاده در سیستم تشخیص خواب‌آلودگی به کار گرفته می‌شود.

اولین شبکه عصبی پیچشی، یک شبکه عصبی کاملاً طراحی شده است. از تابع فعالیت سیگموئید^۱ نیز برای طبقه‌بندی راننده به عنوان خواب‌آلود یا غیرخواب‌آلود استفاده می‌شود. در دو شبکه عصبی دیگر، از مزیت یادگیری انتقالی^۲ برای آموزش شبکه‌های پیشنهادی در مجموعه داده آموزشی خود استفاده می‌کنیم. برای یادگیری انتقالی از مدل‌های VGG-16 و MobileNet-V2 استفاده کردیم که از نظر حافظه و پیچیدگی کارآمدتر هستند و می‌توانند ویژگی‌های مربوط به خواب‌آلودگی را به‌طور خودکار یاد بگیرند.

در نهایت پس از استخراج چشم از ویدیو، به کمک شبکه عصبی منتخب، خواب‌آلودگی آن را بررسی می‌کنیم. در صورت تشخیص خواب‌آلودگی، زنگ هشدار به صدا در می‌آید. رابط کاربری گرافیکی ساده‌ای نیز طراحی شده است تا کاربران به راحتی از پروژه استفاده کنند. همچنین کاربر می‌تواند زنگ هشدار موردنظر خود و آستانه زنگ هشدار را تعیین کند. علاوه بر این، به کمک داکر^۳ برنامه را پیکربندی کرده تا کاربران حرفه‌ای و برنامه‌نویسان بدون نصب پیش‌نیازها برنامه را روی سیستم‌های شخصی خود اجرا کنند.

واژه‌های کلیدی:

الگوریتم ویولا و جونز، شبکه عصبی پیچشی، یادگیری انتقالی، خواب‌آلودگی رانندگان

^۱ Sigmoid

^۲ Transfer learning

^۳ Docker

عنوان	صفحه
فصل اول مقدمه.....	۱
۱-۱- تعریف مسئله.....	۲
۱-۲- روش‌های موجود.....	۲
۱-۲-۱- ذهنی.....	۳
۲-۲-۱- فیزیولوژیکی.....	۳
۱-۲-۲-۱- سیگنال‌های الکتریکی قلب.....	۴
۲-۲-۲-۱- سیگنال‌های الکتریکی مغز.....	۴
۳-۲-۲-۱- سیگنال‌های پتانسیل الکتریکی چشم.....	۵
۳-۲-۱- مبتنی بر وسیله نقلیه.....	۵
۴-۲-۱- رفتاری.....	۶
۱-۴-۲-۱- وضعیت سر.....	۶
۲-۴-۲-۱- خمیازه.....	۶
۳-۴-۲-۱- حالت چشم.....	۷
۱-۳- مروری بر سامانه‌ها و پروژه‌های مشابه.....	۷
۱-۴- ساختار پایان‌نامه.....	۹
فصل دوم روش‌ها و الگوریتم‌ها.....	۱۰
۱-۲- چهارچوب سیستم پیشنهادی.....	۱۱
۲-۲- استخراج فریم.....	۱۲
۳-۲- تشخیص چهره و استخراج ناحیه چشم.....	۱۲
۱-۳-۲- ویژگی‌های هار.....	۱۲
۲-۳-۲- تصویر انتگرالی.....	۱۳
۳-۳-۲- آدابوست.....	۱۴
۴-۳-۲- مرحله آبخاری.....	۱۵
۴-۲- استخراج و طبقه‌بندی ویژگی‌ها.....	۱۵
۱-۴-۲- شبکه عصبی پیچشی.....	۱۶
۱-۴-۲-۱- لایه پیچشی.....	۱۶

۱۷	۲-۱-۴-۲- تابع فعال ساز غیرخطی
۱۸	۲-۱-۴-۳- لایه ادغام
۱۹	۲-۱-۴-۴- لایه کاملاً متصل
۱۹	۲-۴-۲- یادگیری انتقالی
۲۰	۲-۴-۱- MobileNetV2
۲۲	۲-۲-۴-۲- VGG-16
۲۳	۵-۲- فعال سازی زنگ هشدار
۲۳	۶-۲- جمع بندی
۲۴	فصل سوم ابزارهای پیاده سازی و مجموعه داده ها
۲۵	۳-۱-۱- ابزارها و کتابخانه های مورد استفاده
۲۵	۳-۱-۱-۱- تنسورفلو
۲۶	۳-۱-۱-۲- python-OpenCV
۲۷	۳-۱-۱-۳- Scikit-learn
۲۷	۳-۱-۱-۴- Numpy
۲۷	۳-۱-۱-۵- Pillow(PIL)
۲۷	۳-۲- مجموعه داده گان
۳۰	۳-۳- جمع بندی
۳۱	فصل چهارم پیاده سازی مدل شبکه عصبی
۳۲	۴-۱- پیش پردازش مجموعه داده ی آموزشی
۳۳	۴-۲- استخراج ویژگی و آموزش
۳۴	۴-۲-۱- شبکه عصبی تمام طراحی شده
۳۵	۴-۲-۲- شبکه انتقال یادگیری MobileNetV2
۳۸	۴-۲-۳- شبکه انتقال یادگیری VGG-16
۴۰	۴-۳- ابر پارامترها
۴۰	۴-۳-۱- ایپاک
۴۱	۴-۳-۲- اندازه بچ
۴۱	۴-۳-۳- تابع زیان
۴۱	۴-۳-۴- بهینه ساز
۴۲	۴-۳-۵- نرخ یادگیری
۴۳	۴-۴- ارزیابی مرحله آموزش

۴۵	۴-۵- آزمایش مدل های شبکه عصبی.....
۴۶	۴-۵-۱- ماتریس سردرگمی.....
۴۸	۴-۵-۲- دقت.....
۴۸	۴-۵-۳- صحت، پوشش و امتیاز F1.....
۴۹	۴-۵-۴- منحنی ROC.....
۵۱	۴-۵-۵- زمان آزمایش.....
۵۲	۴-۶- مدل منتخب.....
۵۳	۴-۷- جمع بندی.....
۵۴	فصل پنجم پیاده سازی سیستم تشخیص خواب آلودگی.....
۵۵	۵-۱- منطق سیستم تشخیص خواب آلودگی.....
۵۸	۵-۲- رابط کاربری گرافیکی.....
۵۸	۵-۲-۱- معرفی ابزار پیاده سازی رابط کاربری.....
۵۸	۵-۲-۲- پیاده سازی رابط کاربری.....
۶۱	۵-۳- مثال از عملکرد کلی سیستم.....
۶۲	۵-۴- برنامه قابل اجرا.....
۶۳	۵-۵- استفاده از داکر.....
۶۵	۵-۶- جمع بندی.....
۶۶	فصل ششم نتیجه گیری و پیشنهادات.....
۶۷	۶-۱- جمع بندی و نتیجه گیری.....
۶۸	۶-۲- پیشنهادات.....
۷۰	مراجع و منابع.....

فهرست اشکال

صفحه	عنوان
۴	شکل ۱-۱- نحوه نصب الکترودها برای استفاده از سیگنال‌های الکتریکی مغزی.....
۱۱	شکل ۱-۲- چهارچوب سیستم تشخیص خواب‌آلودگی.....
۱۳	شکل ۲-۲- نمونه‌ای از مستطیل‌های استفاده شده در الگوریتم ویولا و جونز.....
۱۳	شکل ۳-۲- قرارگیری ویژگی‌های هار روی چهره برای استخراج ویژگی.....
۱۴	شکل ۴-۲- نحوه محاسبه تصویر انتگرالی.....
۱۵	شکل ۵-۲- نحوه عملکرد مرحله آبخاری.....
۱۶	شکل ۶-۲- ضرب ماتریس عناصر و جمع نتایج بر روی فیچرکمپ در لایه پیچشی.....
۱۷	شکل ۷-۲- توابع غیرخطی در شبکه عصبی پیچشی.....
۱۸	شکل ۸-۲- تابع فعال‌ساز واحد یکسوساز خطی و نحوه اعمال به یک ورودی نمونه.....
۱۸	شکل ۹-۲- انواع ادغام.....
۱۹	شکل ۱۰-۲- لایه کاملاً متصل.....
۲۰	شکل ۱۱-۲- نمودار مقایسه‌ای فرآیند یادگیری بین یادگیری ماشین معمولی و یادگیری انتقالی.....
۲۱	شکل ۱۲-۲- ساختار موبایل‌نت نسخه ۲.....
۲۲	شکل ۱۳-۲- معماری مدل VGG-16.....
۲۸	شکل ۱-۳- تصاویر نمونه از مجموعه داده MRL.....
۲۹	شکل ۲-۳- ساختار نوشتاری توضیحات هر عکس مجموعه داده.....
۲۹	شکل ۳-۳- تشخیص چشم در مجموعه داده با هیستوگرام گرادیان‌های جهت دار و SVM.....
۳۲	شکل ۱-۴- قطعه کد مربوط به خواندن تصاویر مجموعه داده به همراه برچسب آن‌ها.....
۳۳	شکل ۲-۴- قطعه کد پیش‌پردازش تصاویر آموزشی.....
۳۴	شکل ۳-۴- معماری کلی شبکه عصبی تمام طراحی شده.....
۳۵	شکل ۴-۴- قطعه کد مربوط به ساخت مدل شبکه عصبی تمام طراحی شده.....
۳۶	شکل ۵-۴- گزارش ساخت شبکه عصبی پیچشی تمام طراحی شده.....
۳۷	شکل ۶-۴- معماری کلی شبکه MobileNetV2.....

شکل ۴-۷- قطعه کد پیاده‌سازی مدل MobileNetV2	۳۷
شکل ۴-۸- گزارش ساخت مدل MobileNetV2	۳۸
شکل ۴-۹- معماری کلی شبکه VGG-16	۳۹
شکل ۴-۱۰- قطعه کد پیاده‌سازی مدل VGG-16	۳۹
شکل ۴-۱۱- گزارش ساخت مدل شبکه عصبی پیچشی VGG-16	۴۰
شکل ۴-۱۲- قطعه کد مربوط به بهینه‌ساز و تابع زیان	۴۲
شکل ۴-۱۳- نمودار تغییر نرخ یادگیری به تعداد اپاک	۴۳
شکل ۴-۱۴- قطعه کد زمان‌بند نزول مبتنی بر زمان	۴۳
شکل ۴-۱۵- نمودار دقت و زیان مدل تمام طراحی شده	۴۴
شکل ۴-۱۶- نمودار دقت و زیان مدل MobileNetV2	۴۴
شکل ۴-۱۷- نمودار دقت و زیان مدل VGG-16	۴۵
شکل ۴-۱۸- ماتریس سردرگمی مدل تمام طراحی شده	۴۷
شکل ۴-۱۹- ماتریس سردرگمی مدل MobileNetV2	۴۷
شکل ۴-۲۰- ماتریس سردرگمی مدل VGG-16	۴۷
شکل ۴-۲۱- منحنی ROC مدل تمام طراحی شده	۵۰
شکل ۴-۲۲- منحنی ROC مدل MobileNetV2	۵۰
شکل ۴-۲۳- منحنی ROC مدل VGG-16	۵۱
شکل ۴-۲۴- زمان آزمایش هر مدل به ثانیه	۵۲
شکل ۵-۱- نحوه ساخت طبقه‌بندی‌کننده برای تشخیص چهره و چشم	۵۵
شکل ۵-۲- استفاده از متد detectMultiScale برای تشخیص چهره	۵۵
شکل ۵-۳- نحوه پیاده‌سازی تابع preprocessing	۵۶
شکل ۵-۴- قطعه کد تابع مربوط به طبقه‌بندی چشم	۵۷
شکل ۵-۵- قطعه کد مربوط به استفاده از کتابخانه playsound برای پخش زنگ هشدار	۵۷
شکل ۵-۶- نمای کلی رابط کاربری گرافیکی	۵۹
شکل ۵-۷- قسمت انتخاب حد آستانه	۵۹

- شکل ۵-۸- جزئیاتی از قسمت انتخاب زنگ هشدار..... ۶۰
- شکل ۵-۹- قسمت کنترلی رابط کاربری..... ۶۰
- شکل ۵-۱۰- مثالی از رانندگی غیر خواب‌آلود..... ۶۱
- شکل ۵-۱۱- مثالی از رانندگی خواب‌آلود..... ۶۱
- شکل ۵-۱۲- محتویات پوشه dist..... ۶۳
- شکل ۵-۱۳- فایل نیازمندی‌ها..... ۶۴
- شکل ۵-۱۴- داکر فایل..... ۶۴

فهرست جداول

صفحه	عنوان
۳۰.....	جدول ۱-۳- تقسیم‌بندی مجموعه داده برای آموزش و آزمایش
۴۶.....	جدول ۱-۴- ماتریس سردرگمی
۴۸.....	جدول ۲-۴- دقت مدل‌های شبکه عصبی پیچشی
۴۹.....	جدول ۳-۴- صحت، پوشش و امتیاز F1 مدل‌های شبکه عصبی پیچشی

فصل اول

مقدمه

۱-۱- تعریف مسئله

در فناوری ایمنی خودرو، تشخیص خواب‌آلودگی راننده برای جلوگیری از تصادفات جاده‌ای بسیار ضروری است [۱]. امروزه، بسیاری از مردم از خودرو برای رفت و آمد روزانه، استانداردهای زندگی بالاتر، راحتی و محدودیت‌های زمان برای رسیدن به مقصد استفاده می‌کنند. این روند منجر به حجم بالای ترافیک در مناطق شهری و بزرگراه‌ها می‌شود و به نوبه خود، تعداد تصادفات جاده‌ای را با عوامل متعددی افزایش می‌دهد. با توجه به گزارش‌های منتشر شده از سازمان بهداشت جهانی، حوادث جاده‌ای یکی از ۱۰ مورد برتر است که منجر به مرگ در جهان می‌شود [۲]. خواب‌آلودگی راننده می‌تواند یکی از دلایل تصادفات جاده‌ای باشد. یکی از راه‌های کاهش تصادفات، تشخیص زودهنگام خواب‌آلودگی راننده و هشدار به او است [۱]. آمارها و گزارش‌های اخیر نشان می‌دهد که ۲۰ تا ۵۰ میلیون نفر در تصادفات رانندگی در سراسر جهان کشته یا مجروح می‌شوند. ارزیابی‌های انجام‌شده توسط NHTSA (اداره ملی ایمنی ترافیک بزرگراه‌های ایالات متحده) نشان می‌دهد سالانه ۱۰۰ هزار تصادف رانندگی رخ می‌دهد که خواب‌آلودگی رانندگان یکی از عوامل اصلی آن است. این حوادث بیش از ۱۲.۵ میلیارد دلار هزینه دارد و منجر به ۱۵۵۰ کشته و ۷۱۰۰۰ مجروح می‌شود. بنیاد ملی خواب آمریکا اعلام کرده‌است که ۵۴ درصد از رانندگان در هنگام خواب‌آلودگی رانندگی کرده‌اند و ۲۸ درصد از این رانندگان کاملاً به خواب رفته‌اند. شورای ایمنی راه آلمان بیان می‌کند که ۲۵ درصد تصادفات رانندگی مرگبار در ترافیک بزرگراه‌ها به دلیل خواب‌آلودگی لحظه‌ای است. همچنین بررسی‌های انجام شده توسط پزشکی قانونی ایران نشان داده‌است که تصادفات رانندگی تعداد زیادی از تلفات کشور را تشکیل می‌دهد. بر اساس گزارش‌های این سازمان، این‌گونه حوادث سالانه بیش از ۴ میلیارد دلار هزینه دارد. همچنین بر اساس گزارش‌های ارائه شده توسط پلیس، تقریباً ۲۳ درصد از تصادفات رانندگی در ایران به دلیل خواب‌آلودگی و خستگی راننده بوده است [۳]. بنابراین با توجه به آمار ارائه شده، ضروری است تا سیستمی طراحی شود که راننده را تحت نظر گرفته و در صورت تشخیص خواب‌آلودگی به او هشدار دهد.

۲-۱- روش‌های موجود

خواب‌آلودگی می‌تواند به عنوان «نیاز به خواب رفتن» تعریف شود و نتیجه ریتم طبیعی و بیولوژیکی انسان است. این واژه مفهوم‌های مجزایی به معنی حالت قبل از به خواب رفتن، حالت فیزیولوژیکی ناشی از مدت طولانی نخوابیدن و یا یک بیماری مزمن را در بر می‌گیرد [۴].

راه‌های مختلفی برای تشخیص و اندازه‌گیری خواب‌آلودگی راننده وجود دارد که معمولاً به چهار دسته ذهنی^۴، فیزیولوژیکی^۵، مبتنی بر وسیله نقلیه^۶ و رفتاری^۷ تقسیم می‌شوند که در این بخش هر یک را به اختصار توضیح می‌دهیم [۴].

۱-۲-۱- ذهنی

خواب‌آلودگی را می‌توان به عنوان یک نیاز فیزیولوژیکی برای مبارزه با خستگی توضیح داد. هر چه بدن انسان خسته‌تر باشد، نیاز به خواب بیشتر احساس می‌شود که این موضوع نشان می‌دهد خواب‌آلودگی می‌تواند سطوح مختلفی داشته باشد. سازمان‌های علمی مانند Laboratory for Sleep و Division of Sleep Disorders مقیاس‌های توصیفی مختلفی از سطوح خواب‌آلودگی ایجاد کرده‌اند. این سازمان‌ها از پرسشنامه‌ها، آزمون‌ها و معیارهای الکتروفیزیولوژیکی برای ارزیابی خواب‌آلودگی استفاده می‌کنند [۴].

نتایج اندازه‌گیری ذهنی جمع‌آوری شده از تمام این آزمون‌ها به میزان کیفیت سوالات پرسیده شده و همچنین تفسیر و درک صحیح آن سوالات بستگی دارد. علاوه بر این، دیدگاه طراحان آزمون و پرسشنامه نقش بزرگی بر کیفیت داده‌های به دست آمده ایفا می‌کند. در نهایت، شایان ذکر است که دریافت بازخورد خواب‌آلودگی ذهنی از یک راننده در شرایط رانندگی در دنیای واقعی بسیار دشوار است.

۱-۲-۲- فیزیولوژیکی

روش‌های فیزیولوژیکی روشی عینی و دقیق برای اندازه‌گیری خواب‌آلودگی ارائه می‌دهند که مبتنی بر این واقعیت است که سیگنال‌های فیزیولوژیکی در مراحل اولیه خواب‌آلودگی شروع به تغییر می‌کنند [۴]. این موضوع می‌تواند به سیستم تشخیص خواب‌آلودگی راننده کمک کند تا هشدار به موقع به راننده خواب‌آلود دهد و از تصادف جلوگیری کند. ایده تشخیص خواب‌آلودگی در مراحل اولیه با موارد مثبت کاذب بسیار کم، بسیاری از محققان را برانگیخته است تا سیگنال‌های فیزیولوژیکی مختلف بدن انسان مانند سیگنال‌های الکتریکی قلب (ECG)،

^۴ subjective

^۵ Physiological

^۶ vehicle-based

^۷ behavioral

سیگنال‌های الکتریکی مغز (EEG) و سیگنال‌های پتانسیل الکتریکی چشم (EOG) را آزمایش کنند. در ادامه هر کدام توضیح داده شده است [۴].

۱-۲-۲-۱- سیگنال‌های الکتریکی قلب

به کمک سیگنال‌های الکتریکی قلب، فعالیت الکتریکی قلب انسان ثبت می‌شود. این سیستم می‌تواند با تشخیص تغییرات جزئی در رفتار قلب، مانند افزایش یا کاهش ضربان قلب، به طور بسیار دقیق تشخیص دهد که بدن انسان در چه وضعیتی است. تغییرات ضربان قلب را می‌توان به کمک فرکانس‌های پایین و بالای ضربان قلب توصیف کرد. زمانی که سوژه بیدار است، ضربان قلب به فرکانس‌های بالا نزدیک است. بنابراین، وقتی یک سوژه شروع به خواب‌آلودگی می‌کند، ضربان قلب شروع به کند شدن کرده و به سمت فرکانس‌های پایین حرکت می‌کند.

۱-۲-۲-۲- سیگنال‌های الکتریکی مغز

سیگنال الکتریکی مغز، فعالیت الکتریکی مغز انسان را ثبت می‌کند و قابل اطمینان‌ترین و رایج‌ترین سیگنالی است که می‌تواند دقیقاً سطح هوشیاری انسان را توصیف کند. فرکانس‌های اندازه‌گیری شده با استفاده از این روش بسیار مستعد خطا هستند و برای اندازه‌گیری صحیح نیاز به شرایط بسیار خاصی دارند. علاوه بر این، برای اندازه‌گیری این سیگنال‌ها، دستگاه‌های حسگر باید با سوژه تماس فیزیکی داشته باشند. واضح است که در یک سناریوی رانندگی واقعی، داشتن الکترودهای متصل به سر راننده، منجر به ناراحتی راننده شده و توانایی‌های او برای رانندگی را سلب می‌کند و همین موضوع به طور بالقوه احتمال وقوع تصادف را افزایش می‌دهد. شکل ۱-۱ نحوه نصب الکترودها برای استفاده از سیگنال‌های الکتریکی مغزی را نشان می‌دهد.



شکل ۱-۱- نحوه نصب الکترودها برای استفاده از سیگنال‌های الکتریکی مغزی [۴]

۱-۲-۳- سیگنال‌های پتانسیل الکتریکی چشم

سیگنال‌های پتانسیل الکتریکی چشم، تفاوت پتانسیل الکتریکی بین قرنیه و شبکیه چشم انسان را ثبت می‌کنند که منجر به تعیین رفتار چشم می‌شود و می‌تواند برای نظارت بر سطح هوشیاری رانندگان استفاده شود. این روش نیاز به تماس مستقیم با سوژه دارد. اگر حرکت چشم آهسته‌تر از حرکت منظم چشم یک سوژه در مرحله بیداری تشخیص داده شود، سوژه در حال خواب‌آلود شدن است. اگرچه این نوع اندازه‌گیری بسیار دقیق است و منجر به خطاهای تشخیص بسیار کم می‌شود، اما به دلیل نیاز تماس مستقیم با سوژه و پیچیدگی دستگاه مورد نیاز برای اندازه‌گیری، برای پیاده‌سازی در دنیای واقعی و بلادرنگ روشی عملی نیست.

قابلیت اطمینان و دقت تشخیص خواب‌آلودگی راننده با استفاده از سیگنال‌های فیزیولوژیکی در مقایسه با سایر روش‌ها بسیار بالا است. با این حال، ماهیت اندازه‌گیری سیگنال‌های فیزیولوژیکی همچنان موضوعی است که از استفاده آن‌ها در سناریوهای دنیای واقعی جلوگیری می‌کند.

۱-۲-۳- مبتنی بر وسیله نقلیه

گزارش‌های جمع‌آوری شده از تصادفات نشان می‌دهد که رفتار وسیله نقلیه پیش از تصادف شامل یکی از موارد زیر بوده است [۴]:

- سرعت بالا بدون گرفتن ترمز
- خارج شدن از مسیر
- تنها بودن راننده در ماشین
- عدم اقدام راننده برای جلوگیری از تصادف

دو معیار متداول مبتنی بر وسیله نقلیه برای تشخیص خواب‌آلودگی راننده، حرکت فرمان و انحراف استاندارد موقعیت خط می‌باشد [۴].

- حرکت فرمان: این روش از اندازه‌گیری زاویه فرمان با استفاده از یک سنسور زاویه نصب شده بر روی ستون فرمان استفاده می‌کند که امکان تشخیص کوچک‌ترین تغییرات موقعیت فرمان را فراهم می‌کند. یک مشکل بالقوه در این رویکرد، تعداد بالای موارد مثبت کاذب است [۴].
- انحراف استاندارد موقعیت خط: ایده اصلی در این روش این است که موقعیت نسبی خودرو با دوربینی که در خارج نصب شده است نظارت می‌شود و از نرم‌افزار تخصصی برای تجزیه و تحلیل داده‌های به دست آمده توسط دوربین استفاده می‌شود [۴].

محدودیت‌های سیستم‌های مبتنی بر این روش، به وابستگی آن‌ها به عوامل خارجی مانند علامت‌گذاری جاده، آب و هوا و شرایط روشنایی مرتبط است.

۱-۲-۴- رفتار

روش‌هایی که تاکنون ذکر شد، برای کاربردهای دنیای واقعی غیرقابل اعتماد یا غیر عملی هستند. این مسئله منجر می‌شود که از روش‌های رفتاری مبتنی بر مشاهده وضعیت بیرونی راننده استفاده شود. این روش‌ها مبتنی بر شناسایی الگوهای رفتاری خاصی هستند که توسط راننده در حالت خواب‌آلود به نمایش گذاشته می‌شوند. ویژگی‌هایی مانند بسته بودن چشم، پلک زدن مداوم، تکان دادن سر، تاب خوردن سر و خمیازه مکرر الگوهای مناسبی برای تشخیص خواب‌آلودگی می‌باشند. به طور معمول، سیستم‌های مبتنی بر این روش از یک دوربین فیلمبرداری برای گرفتن تصویر استفاده می‌کنند و بر ترکیبی از تکنیک‌های بینایی رایانه و یادگیری ماشین برای تشخیص رویدادهای مورد علاقه، اندازه‌گیری آن‌ها و تصمیم‌گیری در مورد اینکه آیا راننده ممکن است خواب‌آلود باشد یا خیر، تکیه می‌کنند. اگر توالی تصاویر گرفته شده و پارامترهای اندازه‌گیری شده نشان دهد که راننده خواب‌آلود است، ممکن است اقدامی مانند به صدا درآوردن زنگ، صورت گیرد [۴]. در ادامه برخی از این الگوها و رفتارها را توضیح می‌دهیم.

۱-۲-۴-۱- وضعیت سر

وقتی راننده خواب‌آلود است، برخی از ماهیچه‌های بدن شروع به شل شدن می‌کنند که منجر به تکان دادن سر می‌شود. رفتار تکان دادن سر همان چیزی است که محققان در تلاش برای تشخیص آن هستند. تحقیقات برای استفاده از این ویژگی به تازگی آغاز شده است. تشخیص وضعیت سر یک مشکل پیچیده بینایی کامپیوتری است که ممکن است به دید استریوسکوپی^۸ یا دوربین‌های دید سه بعدی^۹ نیاز داشته باشد [۴].

۱-۲-۴-۲- خمیازه

خمیازه مکرر یک ویژگی رفتاری است که نشان می‌دهد بدن خسته است یا در حالت آرامش بیشتری قرار گرفته است و منجر به خواب‌آلودگی می‌شود. تشخیص خمیازه می‌تواند به عنوان یک اقدام پیشگیرانه برای هشدار

^۸ stereoscopic vision

^۹ 3D vision cameras

دادن به راننده عمل کند. البته باید توجه داشت که خمیازه کشیدن همیشه قبل از اینکه راننده به حالت خواب‌آلودگی برود رخ نمی‌دهد. بنابراین، نمی‌توان از آن به عنوان یک ویژگی مستقل استفاده کرد [۴].

۱-۲-۴-۳- حالت چشم

تشخیص وضعیت چشم‌ها تمرکز اصلی تحقیقات برای تعیین خواب‌آلودگی یا عدم خواب‌آلودگی راننده در سال‌های اخیر بوده‌است. چشم را می‌توان به یکی از سه حالت کاملاً باز، نیمه باز یا بسته دسته‌بندی کرد. دو مورد آخر را می‌توان به عنوان شاخصی استفاده کرد که راننده هنگام خواب‌آلودگی تجربه می‌کند. اگر چشم‌ها برای مدت طولانی در این دو حالت باقی بمانند، می‌توان نتیجه گرفت که راننده رفتار غیرعادی را تجربه می‌کند. یک سیستم تشخیص حالت چشم باید بتواند این حالات مختلف چشم را تشخیص داده و متمایز کند. الگوریتم‌های مختلفی با رویکردهای مختلف برای استخراج و فیلتر کردن ویژگی‌های مهم چشم‌ها در طول سالیان متمادی مورد استفاده قرار گرفته‌اند. به طور معمول، فرآیند استخراج ویژگی با آموزش و استفاده از الگوریتم‌های یادگیری ماشین دنبال می‌شود [۴].

در این پروژه از حالت چشم که یکی از معیارهای رفتاری می‌باشد برای تشخیص خواب‌آلودگی راننده استفاده می‌شود. همچنین برای تشخیص خواب‌آلودگی چشم از شبکه‌های عصبی پیچشی استفاده خواهیم کرد. در روش پیشنهادی، از یک مدل شبکه عصبی تمام طراحی شده^{۱۰} و دو مدل انتقال یادگیری استفاده خواهیم کرد و هر سه مدل را ارزیابی کرده و مناسب‌ترین مدل را برای سیستم نهایی انتخاب می‌کنیم.

۱-۳- مروری بر سامانه‌ها و پروژه‌های مشابه

به منظور دستیابی به دقت بهتر در سیستم تشخیص خواب‌آلودگی راننده، رویکردهای متنوعی پیاده‌سازی شده‌است. تشخیص وضعیت چشم‌ها، تمرکز اصلی بیش‌تر تحقیقات برای تعیین خواب‌آلودگی یا عدم خواب‌آلودگی رانندگان است. الگوریتم‌های مختلف با رویکردهای متفاوت برای استخراج و فیلتر کردن ویژگی‌های مهم چشم در طول سال‌ها مورد استفاده قرار گرفته‌اند و در این میان، آموزش و استفاده از الگوریتم‌های یادگیری ماشین بیش‌ترین استفاده را داشته‌اند. در ادامه برخی از این رویکردها را مرور می‌کنیم.

^{۱۰} Fully Designed Neural Network

پارک و همکاران [۵] یک شبکه مبتنی بر یادگیری عمیق را برای یافتن خواب‌آلودگی راننده پیشنهاد کردند. در این پروژه از سه ویژگی استفاده شده است که از سه شبکه عمیق مانند AlexNet، VGG-FaceNet و FlowImageNet برای یادگیری ویژگی‌ها استفاده می‌شود. ویژگی‌های مورد استفاده شامل ویژگی‌های رفتاری، محیطی و ویژگی‌های چهره می‌باشند. خروجی این مدل به چهار دسته غیر خواب‌آلود، خواب‌آلودگی همراه با خمیازه، تکان دادن سر و چشمک زدن دسته‌بندی می‌شود. خروجی این مدل از طریق دو معماری به طبقه‌بندی‌کننده و مجموعه SoftMax داده می‌شود که آن‌ها را معماری میانگین مستقل^{۱۱} و معماری ترکیبی ویژگی^{۱۲} نامیده‌اند. این معماری‌ها سه مدل را مقایسه و الحاق می‌کنند و بر اساس این ترکیب، خروجی طبقه‌بندی می‌شود. آزمایش‌ها بر روی مجموعه داده‌های ویدیویی خواب‌آلودگی راننده NTHU انجام شده و دقت حدود ۷۳ درصد به دست آمده است.

چوی و همکاران [۶] در مقاله خود از شبکه‌های عصبی پیچشی برای توسعه یک الگوریتم تشخیص موقعیت نگاه، استفاده کردند. برای شبکه عصبی از ساختار AlexNet استفاده شده است اما به دلیل سنگین بودن این ساختار تغییراتی در آن اعمال شده است. ورودی شبکه عصبی عکس‌هایی با سایز 277×277 می‌باشند. تعداد لایه‌های شبکه عصبی استفاده شده ۳ بوده و برای لایه ادغام^{۱۳}، از لایه ادغام موجود در ساختار Alex Net استفاده کرده‌اند. در لایه آخر نیز از تابع فعالیت واحد یکسوساز خطی^{۱۴} استفاده شده است.

جبارا و همکاران [۷] مدلی برای تشخیص خواب‌آلودگی راننده بر اساس یادگیری عمیق برای برنامه‌های اندرویدی معرفی کردند. آن‌ها مدلی را طراحی کردند که بر اساس تشخیص نقطه عطف چهره است. ابتدا تصاویر از فریم‌های ویدیویی استخراج می‌شوند و سپس از کتابخانه دی‌لیب^{۱۵} برای استخراج نقاط لندمارک استفاده می‌شود. این نقاط به عنوان ورودی به طبقه‌بندی‌کننده پرسپترون چندلایه داده می‌شوند. طبقه‌بندی‌کننده بر اساس این نقاط، خواب‌آلود یا غیر خواب‌آلود بودن راننده را تشخیص می‌دهد.

چیرا و همکاران [۸] یک چهارچوب جدید با استفاده از یادگیری عمیق برای تشخیص خواب‌آلودگی راننده بر اساس وضعیت چشم هنگام رانندگی وسیله نقلیه پیشنهاد کردند. برای تشخیص چهره و استخراج ناحیه چشم از تصاویر صورت، از الگوریتم تشخیص چهره ویولا و جونز استفاده شده است. پس از تشخیص چشم، این ناحیه به

^{۱۱} independently averaged architecture

^{۱۲} feature fused architecture

^{۱۳} pooling

^{۱۴} Rectified Linear Unit (ReLU)

^{۱۵} dlib

عنوان ورودی شبکه عصبی وارد می‌شود. در این پروژه، از ۴ لایه پیمشی و یک لایه کاملاً متصل^{۱۶} استفاده شده‌است. در لایه اول از فیلتر با سائز 3×3 برای کانالو با عکس ورودی استفاده می‌شود و در سه لایه دیگر از فیلتر با سائز 5×5 استفاده شده‌است. تابع فعالیت استفاده شده در این شبکه عصبی واحد یکسو ساز خطی می‌باشد و از Dropout با مقدار ۲۵٪ استفاده شده‌است. در این معماری، از لایه ادغام 2×2 نیز برای لایه‌های پیمشی استفاده می‌شود. خروجی طبقه‌بندی‌کننده دو حالتی است، بنابراین لایه خروجی تنها دو خروجی دارد. برای بهینه‌سازی نیز از آدام^{۱۷} استفاده می‌شود. همچنین از طبقه‌بندی‌کننده softmax برای طبقه‌بندی خروجی استفاده کرده‌اند.

۴-۱- ساختار پایان‌نامه

ادامه پایان‌نامه به شرح زیر ارائه می‌شود:

در فصل دوم به بررسی روش‌ها و الگوریتم‌های استفاده شده در پیاده‌سازی سیستم تشخیص خواب‌آلودگی از جمله شبکه‌های عصبی پیمشی، یادگیری انتقالی و الگوریتم ویولا و جونز می‌پردازیم. در فصل سه، ابزارها و مجموعه داده مورد استفاده برای پیاده‌سازی را معرفی خواهیم کرد و در فصل چهارم، جزئیات پیاده‌سازی روش‌ها برای آموزش و ارزیابی مدل‌های شبکه عصبی مورد بررسی قرار می‌گیرند. در فصل پنجم پیاده‌سازی سیستم تشخیص خواب‌آلودگی نهایی به همراه ویژگی‌های رابط کاربری و معماری پیاده‌سازی آن که جهت ارائه خدمات به کاربر می‌باشد، به همراه چندین مثال از این خدمات بررسی می‌شوند. در انتها، در فصل ششم نیز نتیجه‌گیری از پروژه‌ی انجام شده و پیشنهاداتی برای ادامه‌ی کار ارائه می‌شود.

^{۱۶} fully connected

^{۱۷} Adam

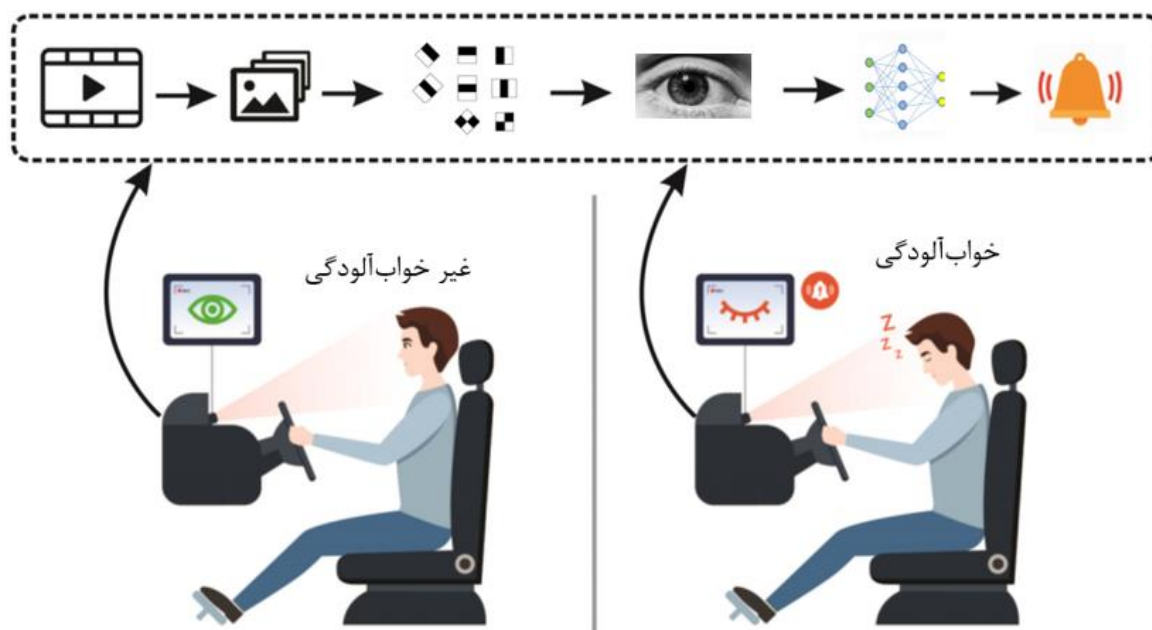
فصل دوم

روش‌ها و الگوریتم‌ها

۱-۲- چهارچوب سیستم پیشنهادی

چهارچوب سیستم پیشنهادی شامل مراحل زیر می‌باشد [۱] که نمای کلی آن در شکل ۱-۲ نشان داده شده‌است.

- ۱- فریم‌ها از ویدیو استخراج شده و به عنوان ورودی به بخش پیش‌پردازش فرستاده می‌شوند.
- ۲- الگوریتم تشخیص چهره ویولا و جونز برای تشخیص چهره در تصاویر به کار گرفته می‌شود. هنگامی که چهره تشخیص داده شد، الگوریتم تشخیص چشم ویولا و جونز برای استخراج ناحیه چشم از تصاویر صورت استفاده می‌شود و به عنوان ورودی به شبکه عصبی پیچشی داده می‌شود.
- ۳- شبکه عصبی پیچشی برای استخراج ویژگی‌ها به کار گرفته می‌شود. سپس این ویژگی‌ها به لایه کاملاً متصل منتقل می‌شوند و تصاویر به دو دسته خواب‌آلود و غیر خواب‌آلود تقسیم می‌شوند.
- ۴- در صورتی که راننده خواب‌آلود باشد، زنگ هشدار به صدا در می‌آید.



شکل ۱-۲- چهارچوب سیستم تشخیص خواب‌آلودگی [۸]

۲-۲- استخراج فریم

ویدئوها مجموعه‌ای از فریم‌های مختلف هستند که به ترتیب خاصی چیده شده‌اند. ویدیوها را نمی‌توان به طور مستقیم به مدل‌های شبکه عصبی داد. به همین دلیل، فریم‌ها از فیلم‌ها استخراج می‌شوند. مرحله استخراج فریم در چهارچوب پیشنهادی شامل گرفتن فیلم، شکستن آن به فریم و ذخیره فریم است [۹].

۲-۳- تشخیص چهره و استخراج ناحیه چشم

تشخیص چهره و اجزای صورت همواره یکی از موضوعات مورد مطالعه در علوم کامپیوتر بوده‌است. تا به امروز صدها الگوریتم برای تشخیص چهره و اجزای صورت به کار گرفته شده‌است. اما یکی از دقیق‌ترین و سریع‌ترین الگوریتم‌ها، الگوریتم ویولا و جونز می‌باشد. بنابراین، در مرحله اول با استفاده از الگوریتم تشخیص چهره ویولا و جونز، چهره از روی تصاویر شناسایی می‌شود. پس از تشخیص چهره، از الگوریتم تشخیص چشم ویولا و جونز برای استخراج ناحیه چشم از تصاویر صورت استفاده می‌شود.

این الگوریتم در سال‌های ۲۰۰۱ و ۲۰۰۴ به کمک دو شخص به نام‌های پاول ویولا و مایکل جونز منتشر شد که از چهار عنصر ویژگی‌های هار^{۱۸}، آدا بوست^{۱۹}، تصاویر انتگرالی^{۲۰} و مراحل آبشاری^{۲۱} تشکیل شده‌است [۱۰].

۲-۳-۱- ویژگی‌های هار

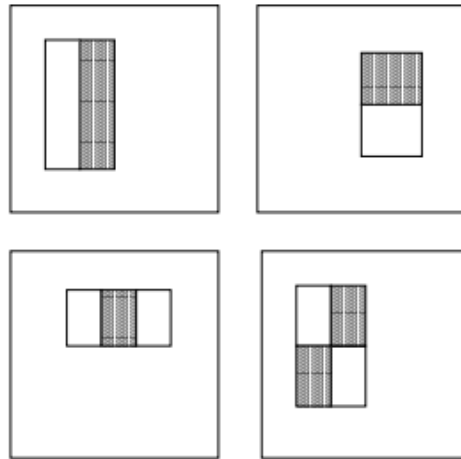
ویژگی‌های هار در واقع مستطیل‌هایی هستند که نشان‌دهنده بخش‌های مختلف صورت می‌باشند و بر روی عکس‌ها با روند تکراری قرار می‌گیرند. نمونه‌ای از این مستطیل‌ها را در شکل ۲-۲ مشاهده می‌کنید. دلایل زیادی برای استفاده از ویژگی‌ها به جای پیکسل وجود دارد که یکی از این دلایل این است که سیستم‌های مبتنی بر ویژگی، سریع‌تر از سیستم‌های مبتنی بر پیکسل عمل می‌کنند [۱۰].

^{۱۸} Haar Features

^{۱۹} Ada boost

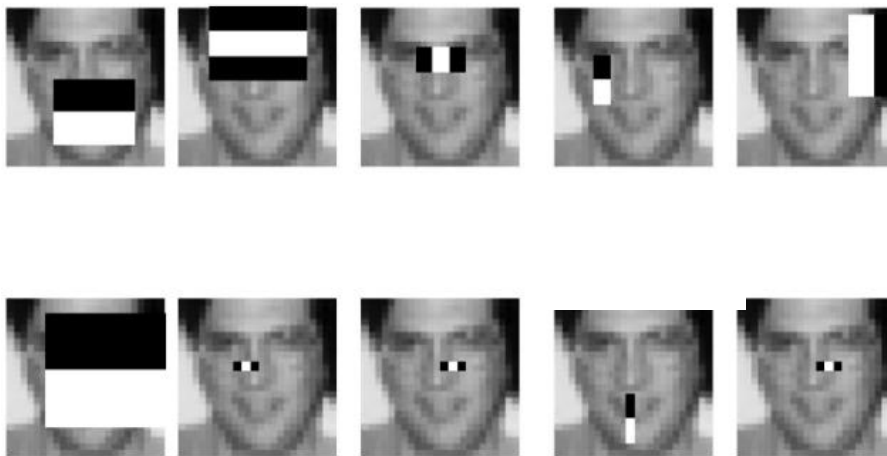
^{۲۰} Integral Images

^{۲۱} Cascading



شکل ۲-۲- نمونه‌ای از مستطیل‌های استفاده شده در الگوریتم ویولا و جونز [۱۱]

همانطور که در شکل زیر مشاهده می‌کنید، هر مستطیل با قرار گرفتن در بخش‌های مختلف صورت در طی مراحل متعدد با سایزهای مختلف شروع به محاسبه می‌کند. نتیجه نهایی از کم کردن پیکسل‌های زیر بخش‌های سیاه از جمع پیکسل‌های زیر بخش‌های سفید به دست می‌آید [۱۰].

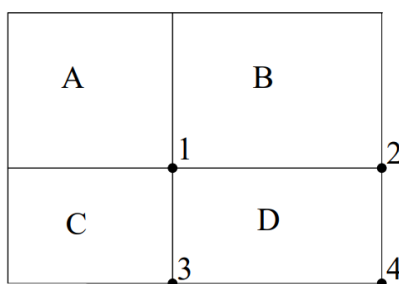


شکل ۳-۲- قرار گیری ویژگی‌های هار روی چهره برای استخراج ویژگی [۱۰]

۲-۳-۲- تصویر انتگرالی

همانطور که گفته شد، با قرار گرفتن هر ویژگی هار روی تصویر، نتیجه نهایی از کم کردن پیکسل‌های زیر بخش‌های سیاه از جمع پیکسل‌های زیر بخش‌های سفید به دست می‌آید. برای سریع‌تر انجام دادن این فرآیند از تصویر انتگرالی استفاده می‌شود. در یک تصویر انتگرالی مقدار پیکسل در مکان x, y برابر است با جمع مقادیر

پیکسل‌های بالا و چپ x, y [۱۰]. همانطور که در شکل ۲-۴ می‌بینید، در مستطیل زیر مجموع پیکسل‌ها در داخل مستطیل D می‌تواند با چهار ارجاع ارزیابی شود. مقدار تصویر انتگرالی در محل ۱ برابر است با مجموع پیکسل‌ها در مستطیل A . این مقدار در مکان ۲ برابر است با $A+B$ ، در مکان ۳ $A+C$ و در مکان ۴ $A+B+C+D$ می‌باشد.



شکل ۲-۴- نحوه محاسبه تصویر انتگرالی [۱۰]

الگوریتم ویولا و جونز از یک پنجره ۲۴ در ۲۴ برای طی این مراحل و قرار دادن مستطیل‌ها بر روی صورت استفاده می‌کند که با در نظر گرفتن تعداد و سائزهای مختلف این ویژگی‌ها (مستطیل‌ها)، برای محاسبه نتیجه نهایی به انجام ۱۸۰ هزار محاسبه نیاز داریم که تعداد بالایی است و هزینه و زمان زیادی را برای هر عکس خواهد گرفت [۱۰].

۲-۳-۳- آدابوست

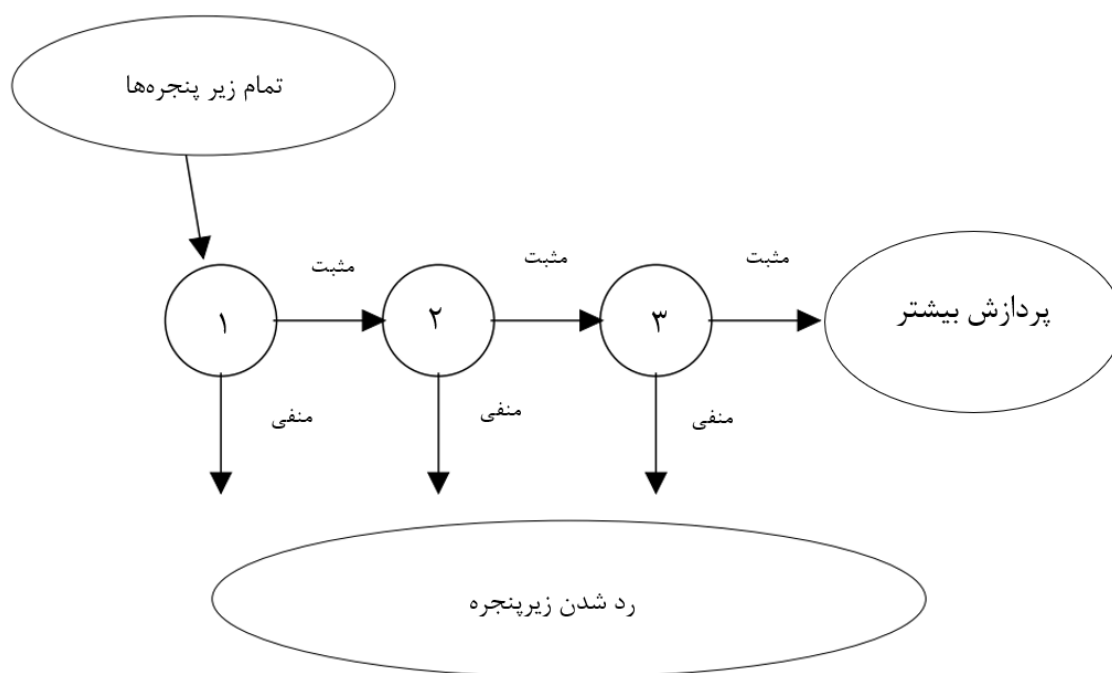
همانطور که در بخش قبل توضیح داده شد، به دلیل استفاده از ویژگی‌ها با سائز، ابعاد و مکان‌های مختلف مجبور به انجام محاسبات بسیار زیادی هستیم که به صرفه نمی‌باشد. اما تمام این ویژگی‌ها برای عکس مناسب نخواهد بود، به این معنی که بعضی از این ویژگی‌ها باید نادیده گرفته شوند و تنها مستطیل‌های مرتبط از بین صدها نوع ویژگی مختلف انتخاب شوند تا در زمان و محاسبات صرفه جویی شود. این کار در الگوریتم ویولا و جونز به صورت خودکار به کمک عنصر آدابوست انجام می‌شود [۱۰].

آدابوست یک الگوریتم یادگیری ماشین می‌باشد که وظیفه آن پیدا کردن ویژگی‌های کاربردی از میان تعداد زیادی ویژگی می‌باشد. برای تصمیم‌گیری در مورد نوع و اندازه یک ویژگی که در طبقه‌بندی کننده نهایی قرار می‌گیرد، آدابوست عملکرد همه طبقه‌بندی کننده‌ها را بررسی می‌کند. بنابراین، هنگامی که آدابوست را برای شناسایی ویژگی‌های مهم آموزش می‌دهیم، اطلاعات را در قالب داده‌های آموزشی به آن داده و آن را آموزش می‌دهیم تا به کمک این اطلاعات پیش‌بینی درستی انجام دهد [۱۰].

۴-۳-۲- مرحله آبشاری

در هر پنجره ۲۴ در ۲۴ پیکسلی نیازمند پردازش تعدادی ویژگی هستیم که از اجرای الگوریتم آدا بوست به دست آمده‌اند. تصور کنید که یک تصویر با ابعاد ۶۴۰ در ۴۸۰ دارید، نیاز دارید تا این ابعاد را به مربع‌های ۲۴ در ۲۴ پیکسلی تقسیم کرده و با پردازش هر بخش تشخیص دهید که آیا چهره‌ای در تصویر وجود دارد یا خیر. مرحله آبشاری به ما این امکان را می‌دهد که این پروسه را سریع‌تر و مفیدتر انجام دهیم.

همانطور که در شکل ۲-۵ مشاهده می‌کنید، یک سری طبقه‌بندی‌کننده برای هر زیر پنجره اعمال می‌شود. هر طبقه‌بند وظیفه دارد تعیین کند که آیا یک زیر پنجره معین صورت است یا خیر. اگر در هر مرحله، زیر پنجره تحت بازرسی رد شود، پردازش بیشتری روی آن اجرا نمی‌شود و جست‌وجو روی زیر پنجره بعدی انجام خواهد شد [۱۰].



شکل ۲-۵- نحوه عملکرد مرحله آبشاری [۱۰]

۴-۲- استخراج و طبقه‌بندی ویژگی‌ها

استخراج ویژگی یکی از انواع کاهش ابعاد است که در آن، بخش‌های مفید تصویر به‌عنوان بردار ویژگی نمایش داده می‌شود [۱]. در این مقاله، ویژگی تصاویر ناحیه چشم با استفاده از یک شبکه عصبی پیچشی استخراج شده‌اند.

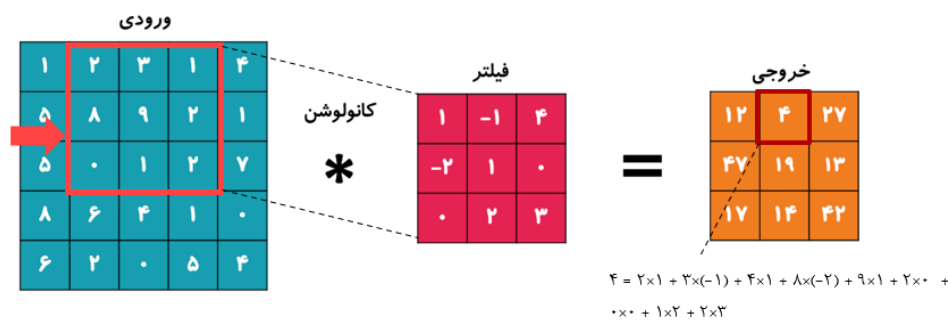
۲-۴-۱- شبکه عصبی پیچشی

شبکه عصبی پیچشی در دهه گذشته در زمینه‌های مختلف مرتبط با تشخیص الگو، نتایج مهمی داشته‌است. سودمندترین جنبه شبکه عصبی پیچشی کاهش تعداد پارامترها در شبکه عصبی مصنوعی است. شبکه عصبی پیچشی همانند سایر شبکه‌های عصبی از لایه‌های نورونی با وزن و بایاس با قابلیت یادگیری تشکیل شده‌است. این شبکه‌ها به گونه‌ای طراحی شده‌اند که برای ورودی‌های با ساختار ماتریسی (دوبعدی و سه‌بعدی) به خوبی کار می‌کنند. شبکه عصبی پیچشی برخلاف شبکه پرسپترون چند لایه، ساختار داده‌های ورودی را عوض نمی‌کند و به ارتباط بین پیکسل‌های همسایه اهمیت می‌دهد [۱۲]. در ادامه قسمت‌های اساسی شبکه عصبی پیچشی را توضیح خواهیم داد.

۲-۴-۱-۱- لایه پیچشی

در لایه پیچشی، ماتریسی به نام فیلتر از روی ماتریس ورودی عبور داده می‌شود تا یک فیچرماپ^{۲۲} برای لایه بعدی ایجاد شود. به عبارتی دیگر، فیلتر روی تصویر حرکت کرده و تصویر ورودی را اسکن می‌کند. هر فیلتر شامل مجموعه‌ای عدد است. با قرار گرفتن فیلتر روی هربخش از تصویر، اعداد فیلتر درایه به درایه در پیکسل‌های متناظر تصویر ضرب می‌شوند و در نهایت همه اعداد با هم جمع می‌شوند. این کار به کمک عملیات ریاضی کانولوشن انجام می‌شود. اگر یک ماتریس دو بعدی از عکس ورودی به نام I و یک فیلتر به نام k داشته باشیم، کانولوشن طبق فرمول ۱-۲ انجام می‌شود [۱۲]. نمونه‌ای از اجرای عملگر کانولوشن در لایه پیچشی را در شکل ۲-۶ مشاهده می‌کنید.

$$S(i, j) = \sum_m \sum_n I(m, n) k(i - m, j - n) \quad \text{فرمول (۱-۲)}$$



شکل ۲-۶- ضرب ماتریس عناصر و جمع نتایج بر روی فیچرماپ در لایه پیچشی [۱۳]

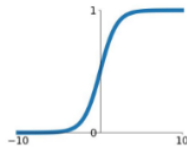
^{۲۲} feature map

۲-۴-۱-۲- تابع فعال‌ساز غیرخطی

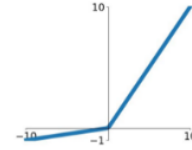
تابع فعال‌ساز بخشی از شبکه عصبی است که بعد از لایه پیچشی می‌آید و تبدیلی غیرخطی می‌باشد که روی سیگنال ورودی انجام می‌شود. برای سال‌های متمادی، سیگموئید و \tanh محبوب‌ترین توابع غیرخطی بودند [۱۲]. شکل ۲-۷، انواع رایج توابع غیرخطی را نشان می‌دهد.

Sigmoid

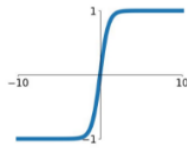
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

**Leaky ReLU**

$$\max(0.1x, x)$$

**tanh**

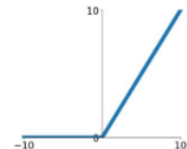
$$\tanh(x)$$

**Maxout**

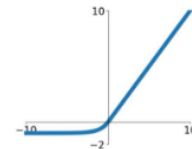
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$

**ELU**

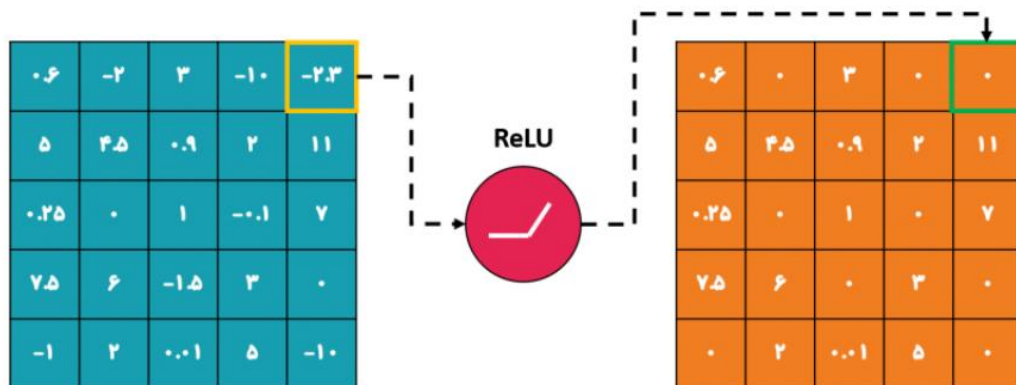
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



شکل ۲-۷- توابع غیرخطی در شبکه عصبی پیچشی [۱۳]

در بین تمام توابع غیرخطی، تابع واحد یکسوساز خطی بیشترین محبوبیت را دارد. این تابع مقادیر کوچکتر از صفر را صفر و مقادیر بزرگتر از صفر را بدون هیچ‌گونه تغییری به خروجی می‌برد [۱۲].

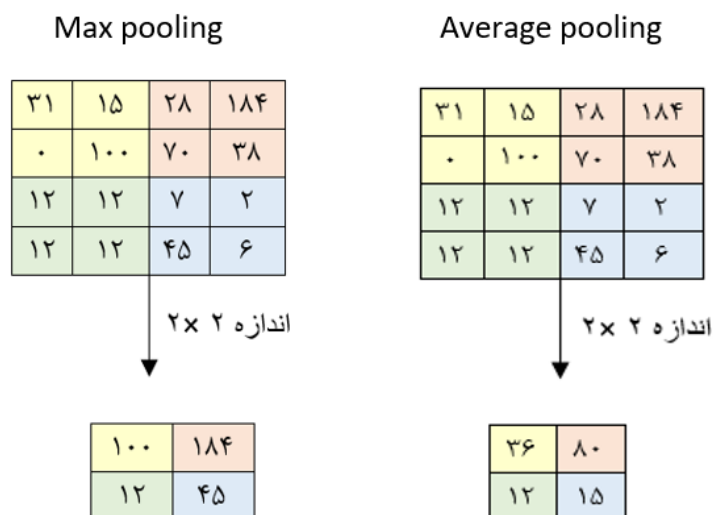
محاسبات موجود در تابع واحد یکسوساز خطی، ساده و صرفاً یک مقایسه است. به همین خاطر محاسبات در بخش تابع غیرخطی با استفاده از واحد یکسوساز خطی نسبت به سایر توابع غیرخطی مانند \tanh و سیگموئید با سرعت بیشتری انجام می‌شود. از طرفی فرآیند آموزش با این تابع نسبت به سایر توابع غیرخطی سریع‌تر است. توابع غیرخطی \tanh و سیگموئید در مقادیر خیلی بزرگ یا کوچک به اشباع می‌رسند و این باعث می‌شود که گرادیان این توابع به سمت صفر میل کند. در نتیجه کل فرآیند آموزش با سرعت پایین‌تری نسبت به واحد یکسوساز خطی انجام می‌شود [۱۲]. در شکل ۲-۸، خروجی تابع واحد یکسوساز خطی بر روی یک ماتریس ورودی را مشاهده می‌کنید.



شکل ۲-۸- تابع فعال‌ساز واحد یکسوساز خطی و نحوه اعمال به یک ورودی نمونه [۱۳]

۲-۴-۱-۳- لایه ادغام

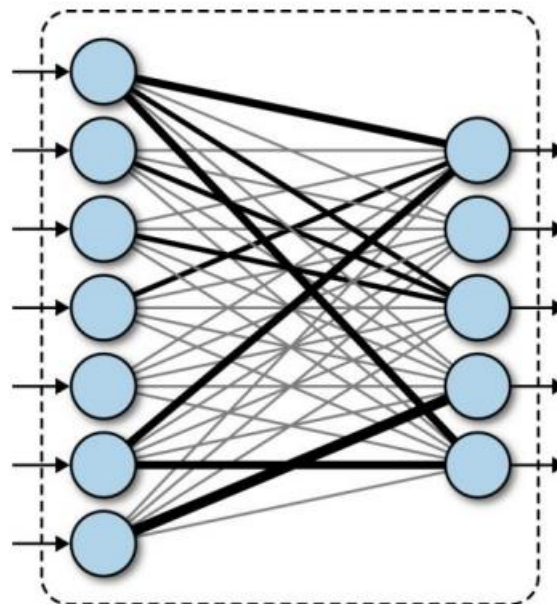
لایه ادغام یکی دیگر از لایه‌های مهم در شبکه عصبی پیچشی است. هدف لایه ادغام کاهش اندازه فیچرکمپ به دست آمده با استفاده از لایه پیچشی است. این لایه پارامتر قابل آموزش ندارد و صرفاً یک نمونه‌برداری ساده و موثر انجام می‌دهد. ادغام عملکردی شبیه کانولوشن دارد و یک پنجره را روی تصویر حرکت می‌دهد. رایج‌ترین نمونه ادغام max-pooling است که تصویر را به مستطیل‌های زیر منطقه تقسیم می‌کند و فقط حداکثر مقدار داخل آن زیر منطقه را برمی‌گرداند. یکی از رایج‌ترین اندازه‌هایی که در max-pooling استفاده می‌شود 2×2 است [۱۲]. در شکل ۲-۹ می‌توانید دو نمونه رایج ادغام را مشاهده کنید.



شکل ۲-۹- انواع ادغام [۱۲]

۲-۴-۱-۴-۲- لایه کاملاً متصل

در انتهای یک شبکه عصبی پیچشی، خروجی آخرین لایه ادغام به عنوان ورودی به لایه کاملاً متصل عمل می‌کند [۱۴]. لایه کاملاً متصل، مشابه روشی است که نورون‌ها در یک شبکه عصبی سنتی قرار می‌گیرند. بنابراین، همانطور که در شکل ۲-۱۰ نشان داده شده است، هر گره در یک لایه کاملاً متصل به طور مستقیم به هر گره در لایه قبلی و بعدی متصل است. این بخش بیشترین تعداد پارامترها در شبکه‌های عصبی پیچشی را دارد و زمان زیادی نیز برای آموزش نیاز دارد. ایراد اصلی یک لایه کاملاً متصل این است که شامل پارامترهای زیادی است که نیاز به محاسبات پیچیده دارد [۱۲].



شکل ۲-۱۰- لایه کاملاً متصل - هر گره در لایه اول به هر گره در لایه دوم متصل است [۱۴]

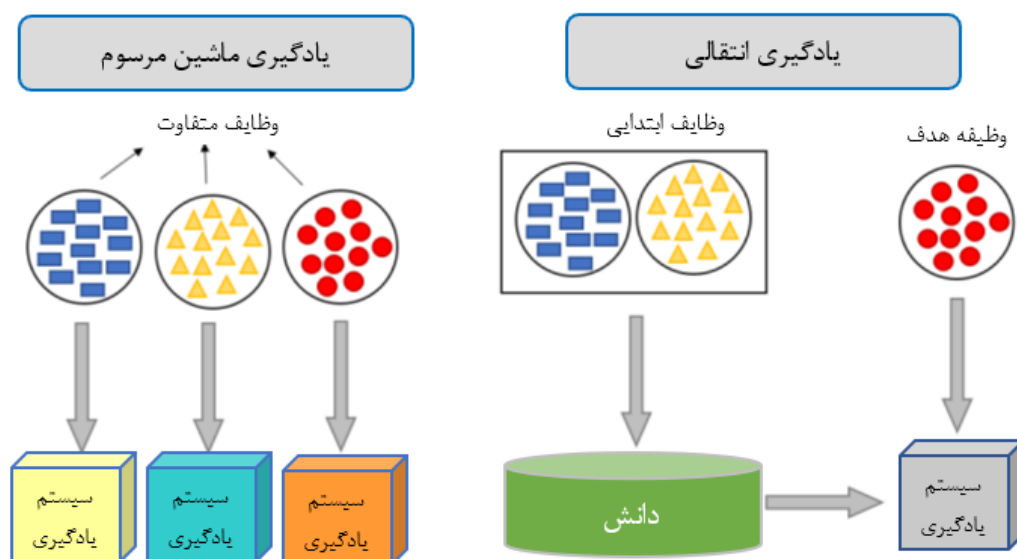
۲-۴-۲- یادگیری انتقالی

انسان‌ها دارای مهارت ذاتی برای انتقال دانش در فعالیت‌های مختلف هستند. دانشی که در حین انجام یک فعالیت خاص به دست می‌آوریم، برای حل فعالیت‌های مرتبط نیز از همان دانش و شیوه استفاده می‌کنیم. انتقال یک عمل شناختی است که به موجب آن تسلط یادگیرنده بر دانش یا مهارت‌ها در یک زمینه، آن‌ها را قادر می‌سازد تا آن دانش یا مهارت را در زمینه‌ای متفاوت به کار گیرند [۱۴].

یادگیری انتقالی به عنوان یک روش یادگیری ماشین شناخته شده در نظر گرفته می‌شود که به موجب آن مدل‌های جدید می‌توانند دانش و تجربه را از یک کار از پیش آموخته‌شده برای بهبود عملکرد در یک کار جدید

به دست آورند. هدف از انتقال یادگیری در یادگیری عمیق، صرفه‌جویی در زمان و منابع با اجتناب از نیاز به آموزش چندین معماری شبکه عصبی از ابتدا برای انجام وظایف مشابه است. به این ترتیب، دانش کسب‌شده توسط مدل از پیش‌آموزش‌دیده به عنوان نقطه شروعی برای مدل جدید عمل می‌کند. ساخت و آموزش یک شبکه عصبی پیچشی جدید از ابتدا، زمان و تلاش زیادی را صرف می‌کند. بنابراین، مدل‌های از پیش‌آموزش‌دیده به طور گسترده مورد استفاده قرار می‌گیرند، زیرا با بهره‌گیری از دانش قبلی، نقطه شروع بهتری را ارائه می‌دهند و دقت بالاتری نیز دارند [۱۱].

شکل ۲-۱۱ تفاوت بین فرآیندهای یادگیری ماشین مرسوم و یادگیری انتقالی را نشان می‌دهد. همانطور که در یک یادگیری ماشین معمولی می‌بینیم، سعی می‌شود هر کار متفاوت به طور جداگانه با سیستم‌های یادگیری مختلف آموزش ببینند، در حالی که یادگیری انتقالی تلاش می‌کند دانش را از وظایف منبع قبلی به وظایف هدف استخراج کند و داده‌های برچسب‌گذاری شده کم‌تری نیز برای یادگیری نظارت شده نیاز دارد [۱۴].



شکل ۲-۱۱- نمودار مقایسه‌ای فرآیند یادگیری بین یادگیری ماشین معمولی و یادگیری انتقالی [۱۴]

در ادامه دو مدل MobileNetV2 و VGG16 که در پروژه استفاده شده‌است را شرح خواهیم داد.

۲-۴-۱- MobileNetV2

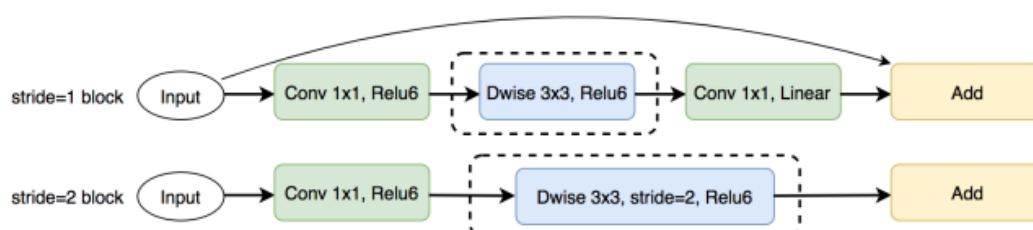
از زمان پیدایش شبکه عصبی الکسنت^{۲۳} در بسیاری از زمینه‌های بنیایی کامپیوتر تحول ایجاد شده‌است. سپس،

^{۲۳} AlexNet

شبکه‌های عمیق‌تر، پهن‌تر و البته پیچیده‌تر گوگل‌نت^{۲۴}، رزنت^{۲۵} و رزنکست^{۲۶} برای رسیدن به دقت بالاتر مطرح شدند. با این حال، همواره جای خالی شبکه‌هایی با سایز کوچک و سرعت بالا با قابلیت استفاده در رباتیک، بردهای مینی کامپیوتری و البته موبایل‌ها احساس می‌شد. بر همین اساس، دسته جدیدی از شبکه‌های پیچشی سبک با پارامترهای کم، سرعت اجرای بالا و البته دقت قابل قبول شکل گرفت. یکی از شاخص‌ترین شبکه‌های سبک، شبکه عصبی موبایل‌نت^{۲۷} نام دارد. این شبکه توسط محققان گوگل با هدف طراحی شبکه‌های کارآمد، سبک، سریع و با دقت قابل قبول مطرح شده‌است [۱۵].

در شبکه عصبی موبایل‌نت یک نوع کانولوشن جدید به نام depth-wise separable convolution معرفی شد. در کانولوشن dws ابتدا کانولوشن عمقی و سپس کانولوشن نقطه‌ای اعمال می‌شود. کانولوشن عمقی و نقطه‌ای به ترتیب نقش مراحل فیلتر و ادغام را در کانولوشن استاندارد دارند. با این تفاوت که در کانولوشن استاندارد M کرنل $k \times k$ داریم اما در کانولوشن عمقی تنها یک کرنل $k \times k$ وجود دارد. کانولوشن نقطه‌ای نیز شامل یک کانولوشن 1×1 است که M تا از این نوع کرنل در هر مرحله تعریف شده‌است [۱۵].

شکل ۲-۱۲ معماری شبکه MobileNetV2 را نشان می‌دهد. این نسخه از موبایل‌نت نسبت به نسخه اول آن ۳۰٪ تعداد پارامترهای کم‌تر، ۲ برابر تعداد عملیات‌های کمتر و ۳۰ تا ۴۰ درصد سریع‌تر است. MobileNetV2 دارای مزایایی همچون سریع، سبک وزن و دقت بالا است که آن را برای آموزش با مجموعه داده‌های محدود مناسب می‌کند [۸]. به این ترتیب، تصمیم گرفتیم MobileNetV2 را برای پیش‌بینی خواب‌آلودگی انتخاب کنیم.



شکل ۲-۱۲ - ساختار موبایل‌نت نسخه ۲ [۸]

^{۲۴} GoogleNet

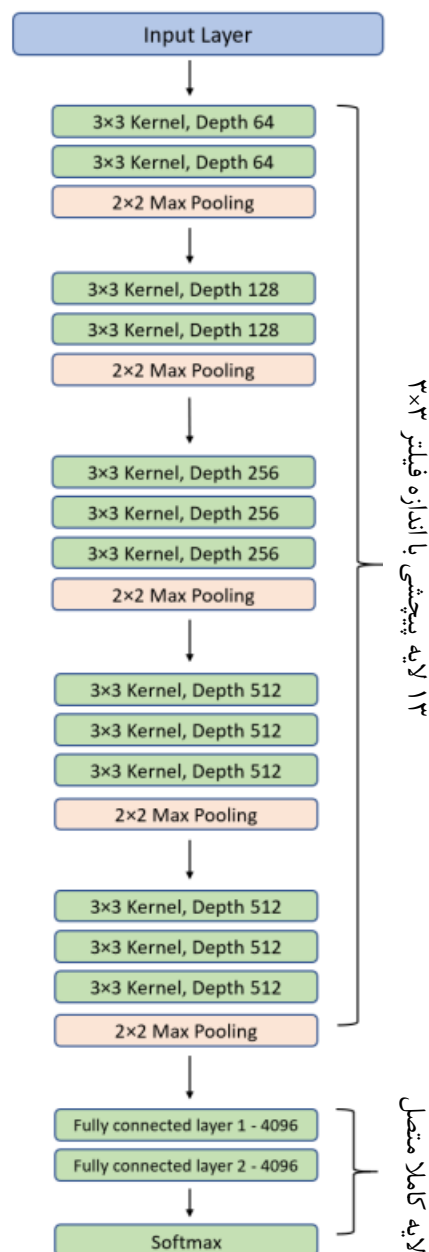
^{۲۵} ResNet

^{۲۶} ResNext

^{۲۷} MobileNet

VGG-16 - ۲-۲-۴-۲

کارن سیمونیان و اندرو زیسرمن معماری VGG-16 را در سال ۲۰۱۴ در مقاله‌ای به نام شبکه پیچشی بسیار عمیق برای تشخیص تصویر در مقیاس بزرگ معرفی کردند. در این معماری یک شبکه ۱۶ لایه‌ای متشکل از لایه‌های پیچشی و کاملاً متصل ایجاد شده است. همچنین برای سادگی، فقط از لایه‌های 3×3 پیچشی استفاده کرده‌اند [۱۴]. ساختار دقیق شبکه VGG-16 در شکل ۲-۱۳ نشان داده شده است.



شکل ۲-۱۳ - معماری مدل VGG-16 [۱۴]

ساختار این مدل به شرح زیر است [۱۴]:

- لایه‌های پیچشی اول و دوم از ۶۴ فیلتر تشکیل شده‌اند و اندازه هر فیلتر 3×3 است. با عبور تصویر ورودی (تصویر RGB با عمق ۳) به لایه پیچشی اول و دوم، ابعاد به $64 \times 64 \times 224$ تغییر می‌کنند. سپس خروجی به دست آمده با گام ۲ به لایه max pooling منتقل می‌شود.
- لایه‌های پیچشی سوم و چهارم از ۱۲۴ فیلتر و اندازه فیلتر 3×3 تشکیل شده‌اند. پس از این دو لایه، یک لایه max-pooling با گام ۲ قرار گرفته‌است و خروجی حاصل به $56 \times 56 \times 128$ کاهش می‌یابد.
- لایه‌های پنجم، ششم و هفتم لایه‌های پیچشی با اندازه فیلتر 3×3 هستند. هر سه از ۲۵۶ فیلتر استفاده می‌کنند. سپس یک لایه max-pooling با گام ۲ قرار گرفته‌است.
- لایه‌های هشتم تا سیزدهم با اندازه فیلتر 3×3 هستند. همه‌ی این لایه‌های پیچشی دارای ۵۱۲ فیلتر می‌باشند. پس از این لایه‌ها یک لایه max-pooling با گام ۱ قرار می‌گیرد.
- لایه‌های چهارده و پانزده لایه‌های کاملاً متصل متشکل از ۴۰۹۶ نورون هستند و پس از آن دو یک لایه خروجی softmax (لایه شانزدهم) ۱۰۰۰ تایی قرار گرفته‌است.

۲-۵- فعال‌سازی زنگ هشدار

در نهایت، نتیجه پیش‌بینی وضعیت چشم به کمک شبکه عصبی، به بخش فعال‌سازی هشدار فرستاده می‌شود تا تعیین شود که به راننده هشدار داده شود یا خیر. در صورتی که برای مدت زمان مشخصی، چشم فرد بسته تشخیص داده شود، زنگ هشدار برای هشیار کردن راننده به صدا در می‌آید.

۲-۶- جمع‌بندی

در این فصل به منطق، چهارچوب، اجزا و هدف الگوریتم‌ها و روش‌های مورد استفاده در سیستم تشخیص خواب‌آلودگی اشاره کردیم. در بخش ۱-۲ چهارچوب کلی سیستم تشخیص خواب‌آلودگی معرفی شد و در بخش‌های بعدی مراحل این چهارچوب را تشریح کردیم. در بخش ۲-۳ الگوریتم ویولا و جونز که برای تشخیص چهره و استخراج ناحیه چشم به کار می‌رود، تشریح شد و در بخش ۲-۴ شبکه‌های عصبی پیچشی و انتقال یادگیری معرفی شدند که از آن‌ها برای ساخت مدل شبکه عصبی و استخراج ویژگی‌های ناحیه چشم استفاده می‌شود. حال در فصل بعدی قصد داریم ابزارهای لازم برای پیاده‌سازی و مجموعه داده مورد نیاز برای آموزش شبکه عصبی پیچشی را معرفی کنیم.

فصل سوم

ابزارهای پیاده‌سازی و مجموعه داده‌ها

برای پیاده‌سازی این پروژه از زبان برنامه‌نویسی پایتون و محیط توسعه یکپارچه^{۲۸} جویپتر نوت‌بوک^{۲۹} استفاده شده‌است.

۳-۱- ابزارها و کتابخانه‌های مورد استفاده

۳-۱-۱- تنسورفلو

اصلی‌ترین ابزار استفاده شده در پیاده‌سازی این پروژه، تنسورفلو^{۳۰} می‌باشد. تنسورفلو یک کتابخانه‌ی نرم‌افزاری رایگان و منبع‌باز برای یادگیری ماشین و هوش مصنوعی است که می‌توان از آن در طیف وسیعی از وظایف استفاده کرد. اما تمرکز ویژه‌ای بر آموزش و استنتاج شبکه‌های عصبی دارد. این کتابخانه توسط تیم Brain Google برای استفاده داخلی گوگل در تحقیق و تولید توسعه داده شده‌است. نسخه اولیه‌ی آن تحت مجوز^{۳۱} آپاچی در سال ۲۰۱۵ منتشر شد و نسخه به روز شده‌ی آن را با نام TensorFlow 2.0 در سپتامبر ۲۰۱۹ منتشر کرد. تنسورفلو را می‌توان در طیف گسترده‌ای از زبان‌های برنامه‌نویسی، به ویژه پایتون استفاده کرد. تنسورفلو می‌تواند شبکه‌های عصبی عمیق را به منظور دسته‌بندی ارقام دست‌نویس، تشخیص تصاویر، تعبیه کلمات، شبکه‌های عصبی بازگشتی، مدل‌های دنباله به دنباله برای ترجمه‌ی ماشین، پردازش زبان طبیعی و شبیه‌سازی‌های مبتنی بر معادلات دیفرانسیل جزئی، تعلیم داده و اجرا کند. بزرگ‌ترین مزیتی که تنسورفلو برای توسعه‌ی یادگیری ماشین به ارمغان آورده است، انتزاع^{۳۲} است. توسعه‌دهنده می‌تواند به جای پرداختن به جزئیات زیربنایی و ریزه‌کاری‌های پیاده‌سازی الگوریتم‌ها یا کشف راه‌های مناسب برای وصل کردن خروجی یک تابع به ورودی تابعی دیگر، بر منطق کلی برنامه تمرکز کند [۱۶].

دو روش عمده در نحوه‌ی استفاده از تنسورفلو وجود دارد:

^{۲۸} Integrated Development Environment (IDE)

^{۲۹} Jupyter Notebook

^{۳۰} TensorFlow

^{۳۱} License

^{۳۲} abstraction

- استفاده از گوگل کولب^{۳۳}

گوگل کولب محصولی است که اجازه می‌دهد تا کد پایتون دلخواه را از طریق مرورگر بنویسید و اجرا کنید. اما به طور خاص برای یادگیری ماشین، تجزیه، تحلیل و آموزش داده‌ها مناسب است. از نظر فنی‌تر، کولب یک سرور بر پایه‌ی جویپتر است که برای استفاده نیازی به تنظیم و نصب کتابخانه‌هایی مانند تنسورفلو ندارد و دسترسی رایگان به منابع محاسباتی از جمله GPUها را فراهم می‌کند [۱۷].

- نصب تنسورفلو بر روی سیستم محلی

پروژه‌ی نصب و راه اندازی تنسورفلو بر روی سیستم محلی با سیستم عامل ویندوز، بسیار زمانبر و نیازمند پیش‌نیازهای زیادی می‌باشد. برای انجام این پروژه، تنسورفلو بر روی سیستم محلی طبق مراحل مرجع [۱۸] نصب شده و توصیه می‌گردد در صورت داشتن سیستم عامل ویندوز حتماً از همین مرجع برای نصب کمک گرفته شود. حال به بخشی از این پیش‌نیازها اشاره می‌کنیم [۱۸]:

۱- Microsoft Visual Studio

۲- NVIDIA CUDA toolkit

۳- cuDNN

توجه شود که سازگاری نسخه‌های موارد ذکرشده برای کارکرد نهایی تنسورفلو از اهمیت بالایی برخوردار است.

۳-۱-۲- python-OpenCV

کتابخانه OpenCV که مخفف Open Source Computer Vision Library یا کتابخانه منبع باز بینایی کامپیوتر است، یکی از پرستفاده‌ترین کتابخانه‌های برنامه‌نویسی برای کاربردهای بینایی کامپیوتر^{۳۴} مانند تشخیص چهره، ویرایش عکس، بینایی رباتیک پیشرفته محسوب می‌شود. کتابخانه Python-OpenCV، واسطه برنامه‌نویسی کاربردی برای کتابخانه OpenCV در زبان پایتون محسوب می‌شود [۱۹].

این کتابخانه نه تنها از سرعت بسیار بالایی برخوردار است، بلکه کد نویسی برنامه‌های کاربردی مرتبط با پردازش تصویر با پایتون و به کاراندازی آن‌ها را نیز تسهیل می‌بخشد. چنین ویژگی‌هایی، کتابخانه python-OpenCV را

^{۳۳} Google Colab

^{۳۴} Computer Vision

به بهترین انتخاب جهت پردازش تصویر با پایتون و پیاده‌سازی برنامه‌های بینایی کامپیوتر در این زبان بدل کرده است [۱۹].

Scikit-learn - ۳-۱-۳

Scikit Learn از کتابخانه‌های متن‌باز، مفید، پرکاربرد و قدرتمند در زبان برنامه‌نویسی پایتون است که برای اهداف یادگیری ماشین به کار می‌رود. این کتابخانه ابزارهای کاربردی زیادی به منظور یادگیری ماشین و مدل‌سازی آماری داده‌ها همچون طبقه‌بندی، رگرسیون، خوشه‌بندی و کاهش ابعاد فراهم می‌کند [۲۰].

Numpy - ۴-۱-۳

کتابخانه Numpy یکی از کتابخانه‌های برنامه‌نویسی کلیدی در زبان برنامه‌نویسی پایتون محسوب می‌شود و استفاده از آرایه را در پایتون فراهم می‌کند. این کتابخانه، یکی از مهم‌ترین کتابخانه‌های توسعه‌دهنده برای کاربردهای پردازش تصویر با پایتون نیز محسوب می‌شود که به طور رایگان در اختیار کاربران و برنامه‌نویسان قرار داده شده‌است. به طور کلی، یک تصویر یک آرایه استاندارد قابل تعریف توسط کتابخانه Numpy محسوب می‌شود که شامل پیکسل‌های متناظر با نقاط داده‌ای خواهد بود. بنابراین، با استفاده از عملیات‌های پایه‌ای تعریف شده در Numpy کاربر قادر خواهد بود تا مقادیر پیکسل‌های یک تصویر را تغییر دهد [۲۱].

Pillow(PIL) - ۵-۱-۳

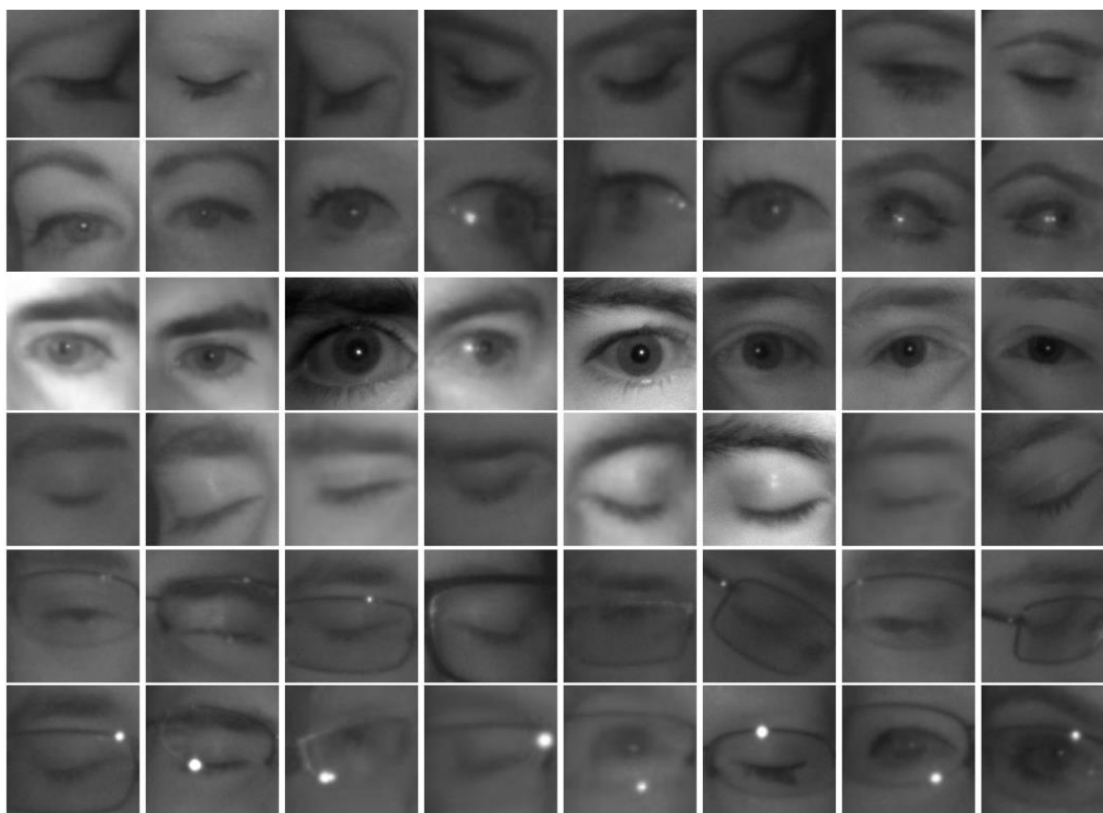
کتابخانه PIL که مخفف عبارت Python Imaging Library یا کتابخانه تصویر پایتون است، یکی از کتابخانه‌های پردازش تصویر با پایتون محسوب می‌شود. این کتابخانه، پشتیبانی از عملیات مرتبط با پردازش تصویر نظیر باز کردن، دستکاری و ذخیره‌سازی تصاویر در فرمت‌های مختلف را به زبان پایتون اضافه می‌کند [۲۲].

۳-۲- مجموعه داده‌گان

در این پروژه از مجموعه داده چشم MRL استفاده شده‌است. این مجموعه داده شامل تصاویر مادون قرمز با وضوح متفاوت است که همگی در شرایط مختلف از نظر روشنایی و توسط دستگاه‌های مختلف گرفته شده‌اند. این مجموعه داده برای آزمایش چندین ویژگی یا طبقه‌بندی‌کننده قابل آموزش مناسب است. برای ساده‌تر شدن مقایسه الگوریتم‌ها، تصاویر به چند دسته تقسیم می‌شوند که آن‌ها را برای آموزش و آزمایش طبقه‌بندی‌کننده‌ها نیز مناسب

می‌کند. این مجموعه داده به صورت عمومی در اختیار همگان قرار دارد و می‌توانید آن را از [این لینک](#) دریافت کنید [۲۳].

تصاویر نمونه‌ای از این مجموعه داده در زیر نشان داده شده است.



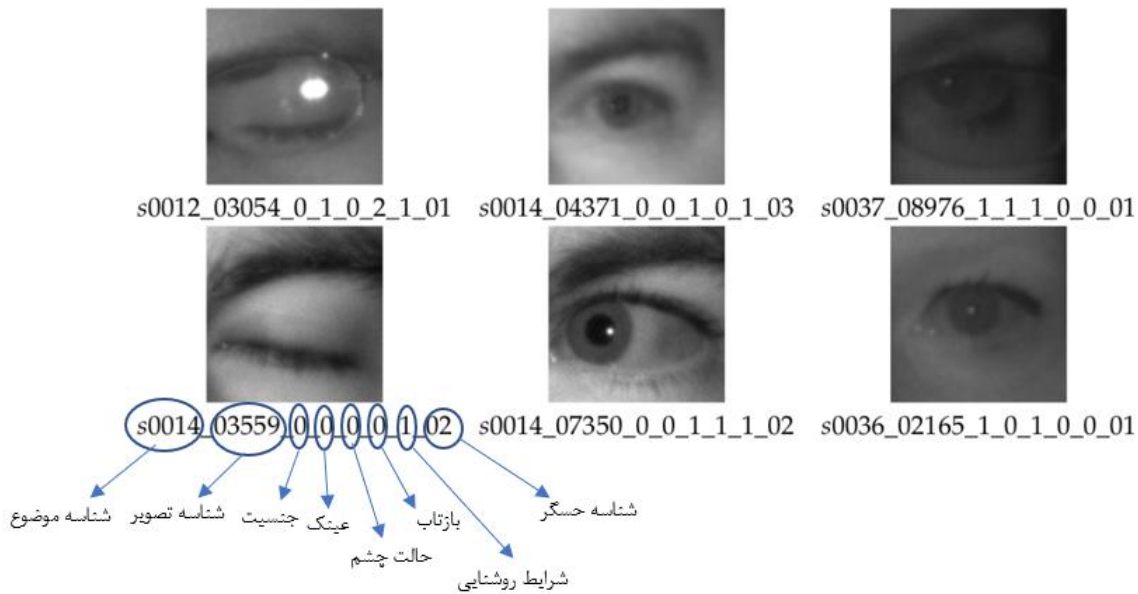
شکل ۳-۱- تصاویر نمونه از مجموعه داده MRL [۲۳]

در مجموعه داده MRL هر یک از ویژگی‌های زیر برای هر عکس تهیه گردیده است [۲۳].

- شناسه موضوع: در مجموعه داده، تصاویر ۳۷ فرد مختلف (۳۳ مرد و ۴ زن) جمع آوری شده است.
- شناسه تصویر: مجموعه داده شامل ۸۴۸۹۸ تصویر است.
- جنسیت: مجموعه داده حاوی اطلاعات مربوط به جنسیت برای هر تصویر است که ۰ نشانگر مرد و ۱ بیانگر زن بودن است.
- عینک: اگر تصویر چشم حاوی عینک باشد مقدار این بخش برابر ۱ و در غیر اینصورت برابر ۰ است.
- حالت چشم: این ویژگی حاوی اطلاعات مربوط به دو حالت چشم است که در صورت غیر خواب‌آلود بودن چشم مقدار آن ۱ و در صورت خواب‌آلود بودن مقدار آن ۰ است.
- بازتاب: سه حالت بازتاب بر اساس اندازه بازتاب‌ها (انعکاس‌های هیچ، کوچک و بزرگ) بیان شده است.

- شرایط روشنایی: هر تصویر بر اساس میزان نور در حین فیلمبرداری دارای دو حالت بد (مقدار صفر) و خوب (مقدار یک) است.
- شناسه حسگر: مجموعه داده شامل تصاویر گرفته شده توسط سه سنسور Intel RealSense RS300 با وضوح 640×480 (مقدار ۰۱)، حسگر IDS Imaging با وضوح 1280×1024 (مقدار ۰۲) و سنسور Aptina با وضوح 752×480 (مقدار ۰۳) است.

شکل ۲-۳ نمونه‌ای از ساختار نوشتاری توضیحات هر عکس را نشان می‌دهد.



شکل ۲-۳- ساختار نوشتاری توضیحات هر عکس مجموعه داده [۲۳]

برای به دست آوردن تصاویر چشم، از تشخیص‌دهنده چشم بر اساس هیستوگرام گرادیان‌های جهت دار^{۳۱} با طبقه‌بندی‌کننده SVM استفاده شده است [۲۳]. نمونه‌هایی از تشخیص چشم در تصاویر زیر نشان داده شده است.



شکل ۳-۳- تشخیص چشم در مجموعه داده با هیستوگرام گرادیان‌های جهت دار و SVM [۲۳]

برای پروژه تشخیص خواب‌آلودگی از ۱۴۵۰۰ عکس از مجموعه‌ی داده MRL استفاده شده‌است که از این تعداد ۱۲۰۰۰ عکس برای آموزش مدل شبکه عصبی و ۲۵۰۰ عکس برای ارزیابی به‌کار گرفته شده‌است. جدول ۱-۳ تقسیم‌بندی تصاویر برای آموزش مدل شبکه عصبی را نشان می‌دهد.

جدول ۱-۳- تقسیم‌بندی مجموعه داده برای آموزش و آزمایش

مجموعه داده	خواب‌آلود	غیر خواب‌آلود	کل
داده آموزش	۶۱۵۵	۵۸۴۵	۱۲۰۰۰
داده آزمایش	۱۳۰۰	۱۲۰۰	۲۵۰۰

۳-۳- جمع‌بندی

در این فصل ابتدا به ابزارهای موردنیاز برای پیاده‌سازی شبکه عصبی پیچشی اشاره کردیم و سپس مجموعه داده مورداستفاده برای آموزش مدل شبکه عصبی پیچشی را معرفی کردیم و جزئیات این مجموعه داده بیان شد. در فصل بعد به بررسی جزئیات پیاده‌سازی سه مدل شبکه عصبی پیچشی و ارزیابی هر یک از آن‌ها در مرحله آموزش و آزمایش می‌پردازیم.

فصل چهارم

پیاده‌سازی مدل شبکه عصبی

۴-۱- پیش‌پردازش مجموعه داده‌ی آموزشی

مجموعه داده مورد استفاده در این پروژه تصاویر چشم انسان در حالت‌ها و شرایط متفاوت است. این تصاویر توسط ماژول پیش‌پردازش دریافت می‌شوند که هدف آن تبدیل تصویر دریافتی به داده‌هایی است که می‌تواند توسط مدل تشخیص خواب‌آلودگی استفاده شود.

تصاویر به دو بخش خواب‌آلود و غیرخواب‌آلود تقسیم می‌شوند. تصاویر هر بخش در یک پوشه جداگانه با نام بسته و باز قرار گرفته‌اند که منظور همان چشم خواب‌آلود و غیرخواب‌آلود است. ابتدا وارد هر پوشه شده و به کمک یک حلقه تصاویر هر پوشه را با کتابخانه Python-OpenCV خوانده و هر تصویر را به یک تصویر جدید با سایز 128×128 تبدیل می‌کنیم. در نهایت نیز تصاویر به دست آمده که دارای سه کانال رنگی می‌باشند، به همراه برچسب مربوطه در یک لیست ذخیره می‌شوند. تصویر ۴-۱ قطعه کد نوشته شده به زبان پایتون را که به کمک آن این مراحل را طی می‌کنیم، نشان می‌دهد.

```
def create_training_data():
    training_data = []
    for category in Classes:
        path = os.path.join(train_dir, category)
        class_num = Classes.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
            backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB)
            backtorgb = cv2.resize(backtorgb, (img_size, img_size))
            training_data.append([backtorgb, class_num])
    return training_data
```

شکل ۴-۱- قطعه کد مربوط به خواندن تصاویر مجموعه داده به همراه برچسب آن‌ها

پس از آماده شدن لیست تصاویر، مجموعه داده آموزشی و داده اعتبارسنجی را تهیه می‌کنیم. برای این کار ابتدا لیست شامل تصاویر را مخلوط^{۳۵} می‌کنیم تا ترتیب اولیه عکس‌ها از بین برود. سپس هر تصویر را در لیستی به نام `X_train` و برچسب مربوط به آن را در `y_label` می‌ریزیم. هر یک از داده‌های تصویری شامل یک آرایه $3 \times 128 \times 128$ می‌باشد که هر خانه لیست نشانگر یک پیکسل از تصویر است که مقادیر بین ۰ تا ۲۵۵ را شامل می‌شود. هر عدد نشان‌دهنده یک کد رنگ است. هنگام استفاده از داده‌های تصویر با همین مقادیر و استفاده از آن در شبکه‌های عصبی، محاسبه مقادیر عددی بالا ممکن است منجر به پیچیدگی شود. بنابراین، مقادیر را بین

^{۳۵} Shuffle

تا ۱ نرمال می‌کنیم. برای این کار `X_train` به ۲۵۵ تقسیم می‌شود. در نهایت نیز برای اینکه بتوانیم از `X_train` و `y_label` در مدل شبکه عصبی استفاده کنیم، آن‌ها را به کمک کتابخانه `numpy` به فرمت آرایه در می‌آوریم. در گام آخر نیز، ۲۰ درصد از مجموعه داده آموزشی را به کمک تابع `train_test_split` در کتابخانه `sklearn` به داده اعتبارسنجی^{۳۶} اختصاص می‌دهیم. شکل ۲-۴ قطعه کد مربوط به تابع `loadData` را نشان می‌دهد. این تابع ابتدا لیست تصاویر به همراه برچسب آن‌ها را دریافت می‌کند سپس مراحل ذکر شده تا می‌کند و در نهایت مجموعه داده آموزش و اعتبارسنجی را برای آموزش مدل شبکه عصبی بر می‌گرداند.

```
def loadData():
    X_train = []
    y_label = []
    training_data = create_training_data()
    random.shuffle(training_data)

    for features, label in training_data:
        X_train.append(features)
        y_label.append(label)

    X_train = np.array(X_train)
    X_train = X_train/255.0
    y_label = np.array(y_label)

    return train_test_split(X_train, y_label, test_size=0.2, random_state=42)
```

شکل ۲-۴- قطعه کد پیش‌پردازش تصاویر آموزشی

اعتبارسنجی داده‌ها گامی حیاتی در دنیای تجزیه و تحلیل داده‌ها است. در طول آموزش مدل شبکه عصبی، داده‌های اعتبارسنجی، داده‌های جدیدی را به مدل وارد می‌کنند که قبلاً ارزیابی نشده‌اند. داده‌های اعتبارسنجی اولین آزمایش را در برابر داده‌های دیده نشده ارائه می‌کنند و به مهندسين داده اجازه می‌دهند تا میزان پیش‌بینی‌های مدل را بر اساس داده‌های جدید ارزیابی کنند [۲۴].

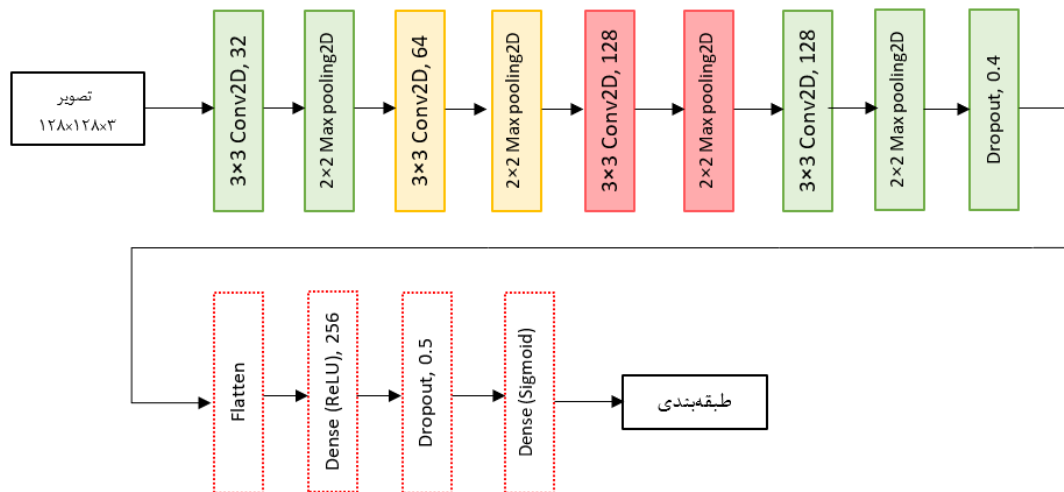
۲-۴- استخراج ویژگی و آموزش

به منظور استخراج ویژگی‌ها و آموزش، سه شبکه عصبی مختلف را طراحی کرده‌ایم که هر یک ویژگی‌های خود را دارند. این شبکه‌ها برای مجموعه داده نام برده استفاده می‌شوند و نتایج آن‌ها با هم مقایسه می‌شود.

^{۳۶} Validation data

۴-۲-۱- شبکه عصبی تمام طراحی شده

اولین شبکه عصبی پیچشی، یک شبکه عصبی کاملاً طراحی شده است. معماری کلی شبکه و لایه‌های مدل در شکل ۴-۳ نشان داده شده است. در این روش پیشنهادی از ۴ لایه پیچشی و دو لایه کاملاً متصل استفاده شده است.



شکل ۴-۳- معماری کلی شبکه عصبی تمام طراحی شده

تصاویر با اندازه $128 \times 128 \times 3$ به عنوان ورودی به لایه پیچشی-۱ (conv1) ارسال می‌شوند. در این لایه تصویر ورودی با ۳۲ فیلتر در اندازه 3×3 کانوالو شده است. سپس تبدیل غیرخطی واحد یکسوساز خطی و max-pooling با اندازه 2×2 در معماری گنجانده شده است. لایه پیچشی-۱ به ۸۹۶ پارامتر نیاز دارد. خروجی لایه پیچشی-۱ به لایه پیچشی-۲ (conv2) وارد می‌شود. در این لایه، ورودی با ۶۴ فیلتر با اندازه 3×3 کانوالو می‌شود. پس از آن، تبدیل غیرخطی واحد یکسوساز خطی و max-pooling با اندازه 2×2 انجام می‌شود. لایه پیچشی-۲ به ۱۸۴۹۶ پارامتر نیاز دارد. خروجی این لایه به لایه پیچشی-۳ (conv3) وارد می‌شود. در لایه پیچشی-۳، ورودی با ۱۲۸ فیلتر با اندازه 3×3 کانوالو شده و پس از آن تبدیل غیرخطی واحد یکسوساز خطی، max-pooling با اندازه 2×2 انجام می‌شود که به ۷۳۸۵۶ پارامتر نیاز دارد. خروجی لایه پیچشی-۳ به لایه پیچشی-۴ (conv4) وارد می‌شود. در لایه پیچشی-۴ همانند لایه پیچشی-۳، ورودی با ۱۲۸ فیلتر با اندازه 3×3 کانوالو می‌شود. پس از آن نیز، تبدیل غیرخطی واحد یکسوساز خطی، max-pooling با اندازه 2×2 انجام شده و به دنبال آن dropout با مقدار ۰.۴ اعمال می‌شود. dropout تکنیکی برای جلوگیری از بیش‌برازش^{۳۷} شبکه‌های عصبی است. dropout با غیرفعال

^{۳۷} Overfitting

کردن تصادفی نورون‌ها و اتصالات مربوط به آن‌ها کار می‌کند. این امر از وابستگی بیش از حد شبکه به تک نورون‌ها جلوگیری می‌کند و همه نورون‌ها را مجبور می‌کند تا تعمیم بهتری را یاد بگیرند. لایه پیچشی-۴ به ۱۴۷۵۸۴ پارامتر نیاز دارد. خروجی این لایه به کمک Flatten به یک آرایه تک بعدی تبدیل می‌شود و سپس یک لایه کاملاً متصل با ۲۵۶ نورون داریم که بعد از آن dropout با مقدار ۰/۵ اعمال شده‌است. مدل شبکه عصبی پیچشی پیشنهادی به ۱۴۲۰۹۹۳ پارامتر قابل آموزش نیاز دارد. برای طبقه‌بندی‌کننده نیز از یک لایه کاملاً متصل با یک نورون و تابع فعالیت سیگموئید استفاده می‌شود. از سیگموئید در طبقه‌بندی باینری استفاده می‌شود و تصویر آن در شکل ۷-۲ در صفحه ۱۷ آمده‌است. قطعه کد نوشته شده به زبان پایتون برای ساخت این مدل در تصویر ۴-۴ آمده‌است.

```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), name='conv1', activation='relu', input_shape=(img_size, img_size, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), name='conv2', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), name='conv3', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), name='conv4', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.4))

model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

شکل ۴-۴- قطعه کد مربوط به ساخت مدل شبکه عصبی تمام طراحی‌شده

گزارش ساخت شبکه عصبی پیچشی تمام طراحی شده را در تصویر ۴-۵ مشاهده می‌کنید.

۴-۲-۲- شبکه انتقال یادگیری MobileNetV2

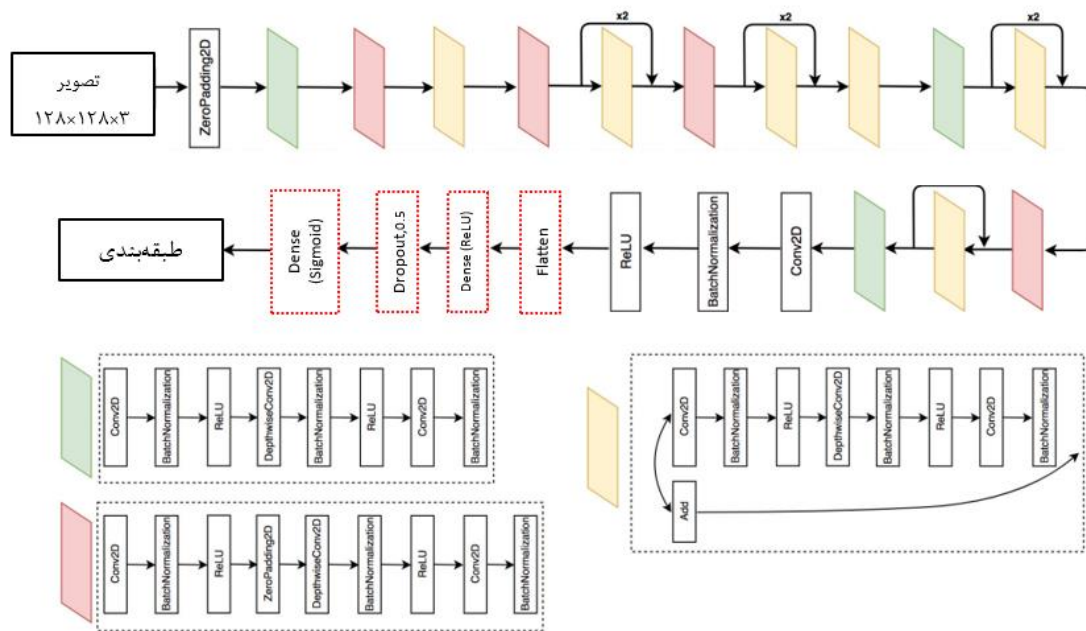
شبکه عصبی موبایل‌نت دسته جدیدی از شبکه‌های پیچشی سبک با پارامترهای کم، سرعت اجرای بالا و البته دقت قابل قبول با قابلیت استفاده در رباتیک، بردهای مینی کامپیوتری و موبایل‌ها می‌باشد [۱۵]. به دلیل حساسیت سیستم تشخیص خواب‌آلودگی نسبت به زمان و اهمیت کارکرد سیستم با سرعت بالا، نیاز است که از یک شبکه عصبی سبک که قابلیت اجرا با سرعت مناسب را دارد، استفاده کنیم. بر همین اساس، از شبکه عصبی MobileNetV2 که یکی از شاخص‌ترین شبکه‌های سبک می‌باشد، استفاده شده‌است. این نسخه از موبایل‌نت

نسبت به نسخه اول آن ۳۰٪ تعداد پارمترهای کمتر، ۲ برابر تعداد عملیات‌های کمتر و ۳۰ تا ۴۰ درصد سریع‌تر است [۸].

Layer (type)	Output Shape	Param #
=====		
conv1 (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2 (Conv2D)	(None, 61, 61, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv3 (Conv2D)	(None, 28, 28, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
conv4 (Conv2D)	(None, 12, 12, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout (Dropout)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257
=====		
Total params: 1,420,993		
Trainable params: 1,420,993		
Non-trainable params: 0		

شکل ۴-۵- گزارش ساخت شبکه عصبی پیچشی تمام طراحی شده

برای بهبود عملکرد مدل شبکه عصبی MobileNetV2، تغییراتی در ویژگی‌های سطح بالا اعمال شده‌است. این تغییرات شامل افزودن لایه‌هایی مانند Flatten، Dense و Dropout می‌باشد. معماری کلی شبکه و لایه‌های مدل در شکل ۴-۶ آمده‌است.



شکل ۴-۶- معماری کلی شبکه MobileNetV2 [۸]

تصویر ورودی به مدل شبکه عصبی، $3 \times 128 \times 128$ می‌باشد. این تصویر از لایه‌های پیچشی، max-pooling و سایر اجزای شبکه عصبی موبایل نت عبور می‌کند و در نهایت خروجی آن به کمک Flatten به یک آرایه یک بعدی تبدیل می‌شود. این آرایه یک بعدی به یک لایه کاملاً متصل با ۲۵۶ نورون متصل می‌شود که بعد از آن Dropout با مقدار ۰/۵ اعمال شده‌است. در آخرین لایه نیز یک لایه کاملاً متصل با یک نورون و تابع فعالیت سیگموئید داریم که کار طبقه‌بندی تصاویر را انجام می‌دهد. این مدل در مجموع ۷,۵۰۱,۳۷۷ پارامتر دارد که از این تعداد ۵,۲۴۳,۳۹۳ پارامتر نیاز به آموزش دارند. قطعه کد نوشته شده به زبان پایتون برای پیاده‌سازی این مدل در تصویر ۴-۷ نشان داده شده‌است.

```
conv_base = MobileNetV2(weights='imagenet', input_shape=(img_size, img_size, 3), include_top=False)
conv_base.trainable = False
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

شکل ۴-۷- قطعه کد پیاده‌سازی مدل MobileNetV2

گزارش ساخت مدل موبایل نت در شکل ۴-۸ آمده‌است.

Model: "sequential"

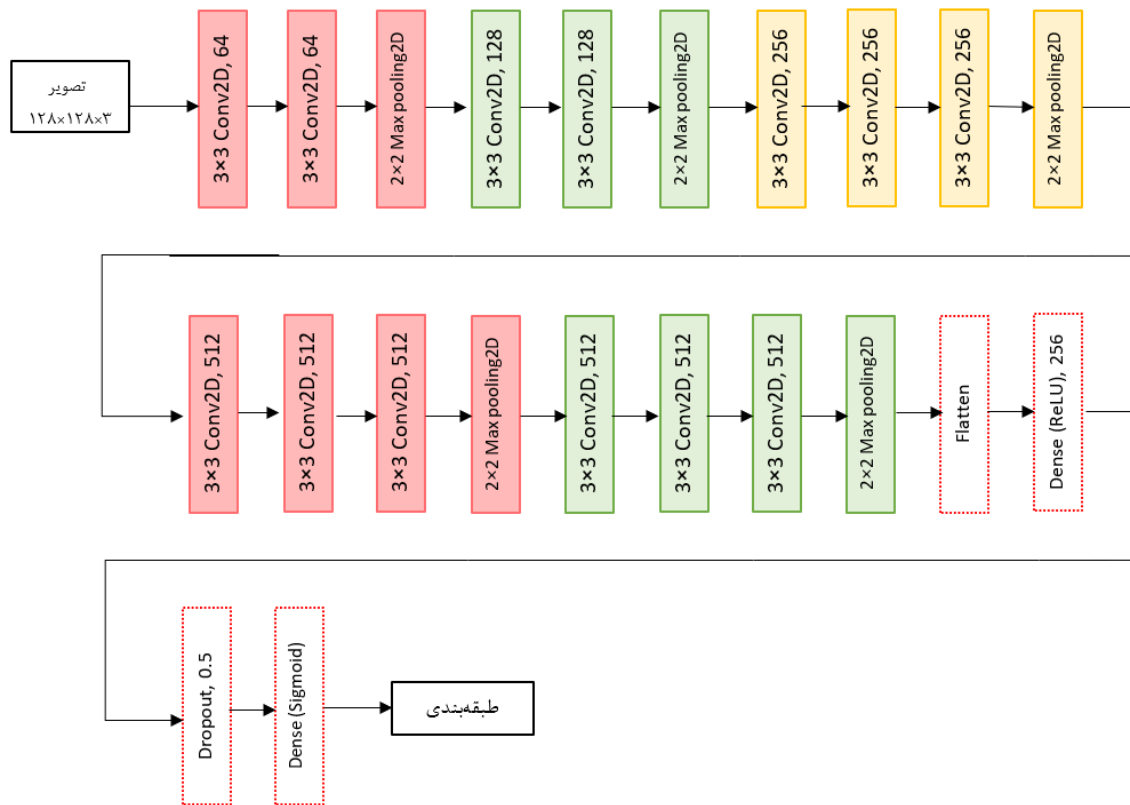
Layer (type)	Output Shape	Param #
mobilenetv2_1.00_128 (Functional)	(None, 4, 4, 1280)	2257984
flatten (Flatten)	(None, 20480)	0
dense (Dense)	(None, 256)	5243136
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257
Total params: 7,501,377		
Trainable params: 5,243,393		
Non-trainable params: 2,257,984		

شکل ۴-۸- گزارش ساخت مدل MobileNetV2

۴-۲-۳- شبکه انتقال یادگیری VGG-16

VGG-16 یک مدل شبکه عصبی پیچشی است که توسط سیموینان و زیرسمن ارائه شده‌است. این مدل در مجموعه داده ImageNet به دقت ۹۲/۷٪ دست می‌یابد [۲۵]. شبکه پیشنهادی به این صورت است که ویژگی‌های سطح پایین با وزن‌های از پیش مشخص‌شده‌ی مجموعه داده ImageNet مشخص می‌شوند و در ویژگی‌های سطح بالا بهبودهایی حاصل شده‌است. همانطور که در شکل ۴-۹ مشاهده می‌کنید، این تغییرات شامل افزودن لایه‌هایی مانند Flatten، Dense(ReLU)، Dropout و Dense(Sigmoid) است.

تصویر ورودی به مدل شبکه عصبی $3 \times 128 \times 128$ می‌باشد و تصویر پس از عبور از لایه‌های پیچشی 3×3 و ۵ لایه‌ی max-pooling با اندازه 2×2 ، به کمک Flatten به یک آرایه یک بعدی تبدیل می‌شود. خروجی این بخش به یک لایه کاملاً متصل با ۲۵۶ نورون متصل می‌شود که بعد از آن dropout با مقدار ۰/۵ اعمال شده‌است. در آخرین لایه نیز یک لایه کاملاً متصل با یک نورون و تابع فعالیت sigmoid داریم که کار طبقه‌بندی را برای ما انجام می‌دهد. برای این مدل در مجموع ۱۶,۸۱۲,۳۵۳ پارامتر داریم که از این تعداد ۲,۰۹۷,۶۶۵ نیاز به آموزش دارند و مابقی از وزن‌های آموزش‌دیده شده توسط مجموعه داده ImageNet استفاده می‌کنند.



شکل ۴-۹- معماری کلی شبکه VGG-16 [۱۴]

قطعه کد نوشته شده به زبان پایتون برای ساخت این مدل در شکل ۴-۱۰ نشان داده شده‌است.

```
conv_base = VGG16(weights='imagenet', include_top=False, input_shape=(img_size, img_size, 3))
conv_base.trainable = False
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

شکل ۴-۱۰- قطعه کد پیاده‌سازی مدل VGG-16

گزارش ساخت مدل VGG-16 در شکل ۴-۱۱ آمده‌است.

Model: "sequential"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14714688
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 256)	2097408
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 1)	257
Total params: 16,812,353		
Trainable params: 2,097,665		
Non-trainable params: 14,714,688		

شکل ۴-۱۱- گزارش ساخت مدل شبکه عصبی پیچشی VGG-16

۴-۳- ابرپارامترها

در این قسمت به ابرپارامترهای استفاده‌شده در مدل شبکه عصبی پیچشی و تنظیم آن‌ها می‌پردازیم [۲۶].

۴-۳-۱- ایپاک^{۳۸}

ایپاک به معنی آموزش شبکه عصبی با تمام داده‌های آموزشی برای یک چرخه است. در یک ایپاک، ما از تمام داده‌ها دقیقاً یک بار استفاده می‌کنیم. ارسال کل مجموعه داده به شبکه عصبی برای یک بار کافی نیست و باید مجموعه داده را چندین بار به یک شبکه عصبی ارسال کنیم. تعداد ایپاک استفاده شده در شبکه عصبی کاملاً طراحی شده و VGG-16 برابر ۵۰ و در موبایل‌نت ۲۵ است. تعداد کم ایپاک منجر به کم‌برازش^{۳۹} و تعداد زیاد ایپاک منجر به بیش‌برازش می‌شود.

^{۳۸} Epoch

^{۳۹} Underfitting

۴-۳-۲- اندازه بچ^{۴۰}

اندازه بچ، تعداد داده‌هایی است که در یک زمان به شبکه ارسال می‌شوند. یک بچ را می‌توان به عنوان یک حلقه برای تکرار در یک یا چند نمونه در نظر گرفت. در پایان هر بچ، پیش‌بینی‌های انجام شده با متغیرهای خروجی مورد انتظار مقایسه شده و یک خطا محاسبه می‌شود. الگوریتم به روزرسانی از این خطا برای بهبود مدل استفاده می‌کند. یک مجموعه داده آموزشی را می‌توان به یک یا چند بچ تقسیم کرد. هنگامی که اندازه بچ بیش از یک و کمتر از اندازه مجموعه داده آموزشی باشد، الگوریتم یادگیری را mini-batch gradient descent می‌نامند. در روش پیشنهادی، در هر سه مدل شبکه عصبی از بچ با اندازه ۶۴ استفاده شده‌است.

۴-۳-۳- تابع زیان^{۴۱}

تابع زیان یکی از اجزای مهم شبکه‌های عصبی است که خطای پیش‌بینی شبکه را مشخص می‌کند. به عبارت ساده، از مقدار زیان برای محاسبه گرادیان استفاده شده و از گرادیان نیز برای به روز رسانی وزن شبکه عصبی استفاده می‌شود و به این ترتیب شبکه عصبی آموزش داده می‌شود. یکی از توابع زیان که برای طبقه‌بندی مسائل باینری استفاده می‌شود، Binary Crossentropy است. هنگام استفاده از این تابع زیان، فقط به یک گره خروجی نیاز است تا داده‌ها به دو کلاس طبقه‌بندی شوند. مقدار خروجی باید از طریق یک تابع فعال‌ساز سیگموئید منتقل شود و محدوده خروجی آن نیز بین صفر و یک است. با توجه به باینری بودن شبکه عصبی استفاده شده در سیستم تشخیص خواب‌آلودگی، از Binary Crossentropy به عنوان تابع زیان استفاده کرده‌ایم.

۴-۳-۴- بهینه‌ساز^{۴۲}

بهینه‌سازها در یادگیری ماشین برای تنظیم پارامترهای یک شبکه عصبی به منظور به حداقل رساندن تابع هزینه استفاده می‌شوند. بنابراین، انتخاب بهینه‌ساز یک جنبه مهم است که می‌تواند منجر به یک آموزش خوب یا یک آموزش بد شود. الگوریتم بهینه‌سازی آدام، توسعه‌یافته الگوریتم نزول گرادیان تصادفی^{۴۳} است که در سال‌های

^{۴۰} Batch size^{۴۱} Loss function^{۴۲} Optimizer^{۴۳} stochastic gradient descent

اخیر برای کاربردهای یادگیری عمیق در بینایی رایانه و پردازش زبان طبیعی مورد استفاده گسترده‌تری قرار گرفته‌است. آدام هر پارامتری را با نرخ یادگیری منحصر به فردی به روز می‌کند. به عبارتی دیگر، هر پارامتر در شبکه دارای نرخ یادگیری خاصی است. بنابراین، در هر سه مدل شبکه عصبی آموزش دیده‌شده از بهینه‌ساز آدام با نرخ یادگیری^{۴۴} اولیه ۰/۰۰۱ استفاده شده‌است. شکل ۴-۱۲ قطعه کدی که در آن بهینه‌ساز و تابع زیان تعریف شده است را نشان می‌دهد.

```
initial_learning_rate = 0.0001
optimizer = keras.optimizers.Adam(learning_rate=initial_learning_rate)
model.compile(loss=keras.losses.BinaryCrossentropy(),
              optimizer=optimizer,
              metrics=['accuracy'])
```

شکل ۴-۱۲- قطعه کد مربوط به بهینه‌ساز و تابع زیان

۴-۳-۵- نرخ یادگیری

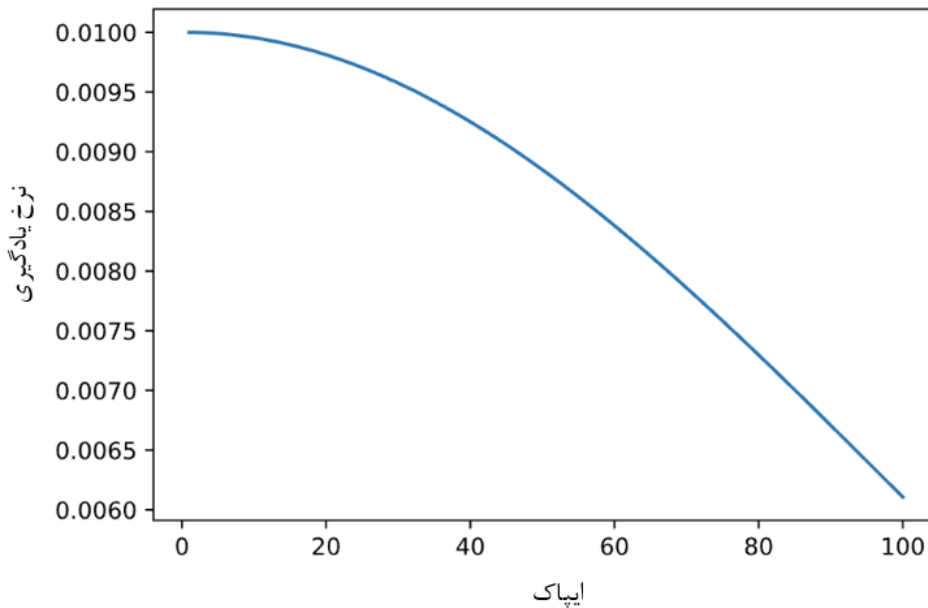
نرخ یادگیری یک ابرپارامتر قابل تنظیم است که در آموزش شبکه‌های عصبی استفاده می‌شود که مقدار مثبت کمی دارد و اغلب در محدوده بین ۰ و ۱ قرار می‌گیرد. نرخ یادگیری سرعت انطباق مدل با مسئله را کنترل می‌کند. اگر نرخ یادگیری کم باشد، آموزش شبکه عصبی بسیار آهسته پیشرفت خواهد کرد زیرا، وزن‌های شبکه عصبی با مقادیر بسیار کم تغییر می‌کنند. هم‌چنین اگر نرخ یادگیری مقدار زیادی در نظر گرفته‌شود، می‌تواند باعث ایجاد رفتار واگرایی نامطلوب در عملکرد شبکه عصبی شود.

هنگام آموزش شبکه عصبی، بهتر است که نرخ یادگیری با پیشرفت آموزش کاهش یابد. این کار را می‌توان با استفاده از زمان بندی‌های نرخ یادگیری^{۴۵} انجام داد. در هر سه مدل شبکه عصبی از زمان‌بند نزول مبتنی بر زمان^{۴۶} استفاده کرده‌ایم. نمودار تغییر نرخ یادگیری به تعداد ایپاک در شکل ۴-۱۳ نشان داده شده‌است.

^{۴۴} Learning Rate

^{۴۵} Learning rate schedules

^{۴۶} Time-based decay



شکل ۴-۱۳- نمودار تغییر نرخ یادگیری به تعداد ایپاک

قطعه کدی که برای زمان‌بند نزول مبتنی بر زمان استفاده شده‌است، در تصویر ۴-۱۴ آمده‌است.

```
initial_learning_rate = 0.0001
decay = initial_learning_rate / epochs
def lr_time_based_decay(epoch):
    return initial_learning_rate * 1 / (1 + decay * epoch)
```

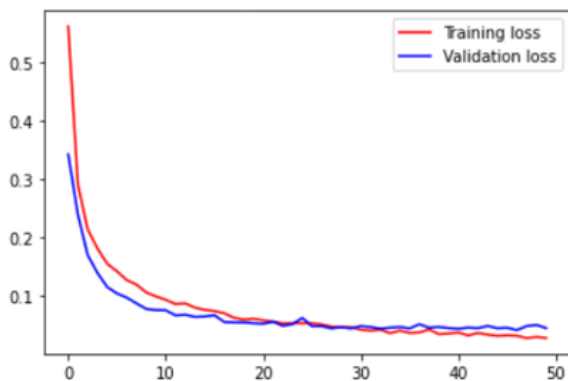
شکل ۴-۱۴- قطعه کد زمان‌بند نزول مبتنی بر زمان

۴-۴- ارزیابی مرحله آموزش

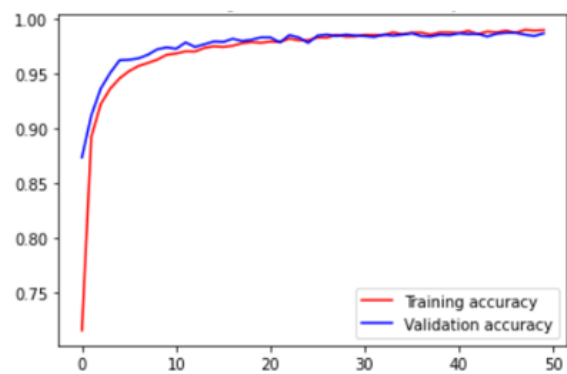
برای آموزش هر سه مدل شبکه عصبی از ۱۲۰۰۰ داده آموزشی استفاده شده‌است که از این تعداد ۶۱۵۵ تصویر مربوط به چشم خواب‌آلود و ۵۸۴۵ تصویر مربوط به چشم غیر خواب‌آلود است. از بین ۱۲۰۰۰ داده آموزشی، ۲۴۰۰ داده را به داده اعتبارسنجی اختصاص داده‌ایم. مجموعه اعتبارسنجی، مجموعه داده‌ای جدا از مجموعه‌ی آموزشی است که برای اعتبارسنجی مدل در طول آموزش استفاده می‌شود. این فرآیند، اطلاعاتی ارائه می‌کند که که در تنظیم ابرپارامترها در آموزش مدل شبکه عصبی استفاده می‌شود. داده‌های اعتبارسنجی برای ارزیابی مدل شبکه عصبی به کار می‌روند تا بررسی کنیم که شبکه عصبی آموزش داده شده، داده‌های آموزشی را حفظ نکرده باشد، بلکه آن‌ها را یاد گرفته‌باشد [۲۴].

نمودارهای ۴-۱۵، ۴-۱۶ و ۴-۱۷ به ترتیب نمودار دقت^{۴۷} و زیان مدل تمام طراحی شده، MobileNetV2 و VGG-16 را در حین آموزش و اعتبارسنجی برای یادگیری ویژگی‌های تشخیص خواب‌آلودگی راننده نشان می‌دهد. محور y دقت و مقدار زیان مدل و محور x تعداد اپیاک را نشان می‌دهد. همانطور که مشاهده می‌کنید، مقدار دقت با افزایش تعداد اپیاک به تدریج افزایش می‌یابد، در حالی که مقدار زیان کاهش می‌یابد. از آنجایی که مقدار زیان داده اعتبارسنجی در هر سه مدل هم‌گام با مقدار زیان داده آموزشی کاهش پیدا کرده‌است، بنابراین مدل‌ها دچار بیش‌برازش نشده‌اند.

مقدار زیان داده آموزشی و اعتبارسنجی مدل تمام طراحی شده

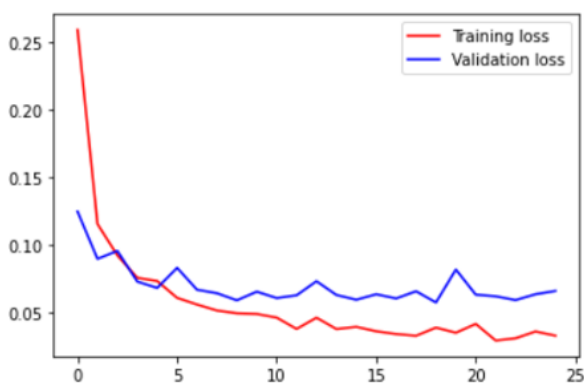


دقت داده آموزشی و اعتبارسنجی مدل تمام طراحی شده

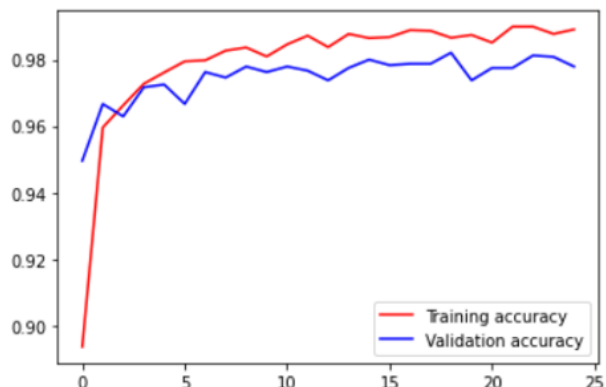


شکل ۴-۱۵- نمودار دقت و زیان مدل تمام طراحی شده

مقدار زیان داده آموزشی و اعتبارسنجی مدل MobileNetV2

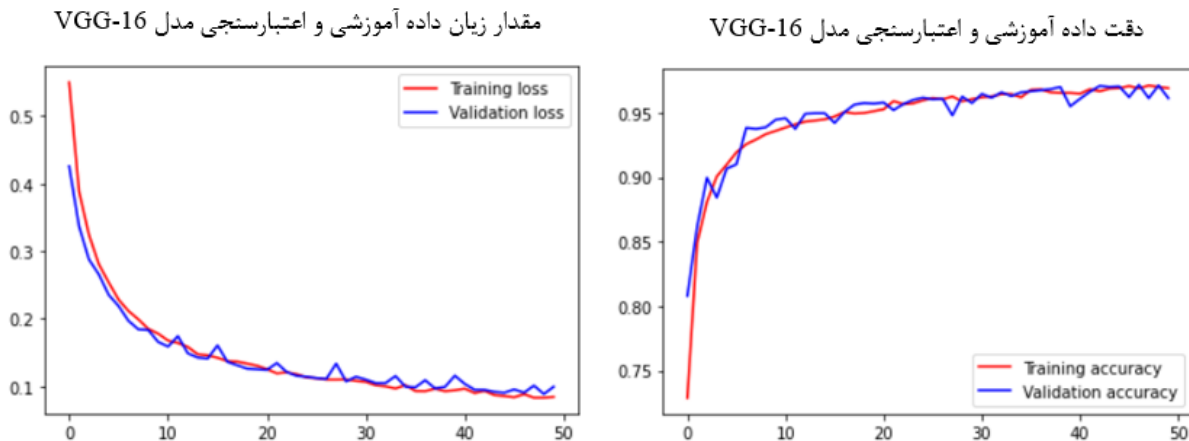


دقت داده آموزشی و اعتبارسنجی مدل MobileNetV2



شکل ۴-۱۶- نمودار دقت و زیان مدل MobileNetV2

^{۴۷} Accuracy



شکل ۴-۱۷- نمودار دقت و زیان مدل VGG-16

دقت مدل شبکه عصبی تمام طراحی شده بر روی داده اعتبارسنجی از ۸۷/۳٪ شروع شده و پس از ۵۰ اپیک به ۹۸/۷٪ می‌رسد. هم‌چنین مقدار زیان روی داده اعتبارسنجی برای این مدل از ۰.۳۴ شروع شده و تا ۰/۰۴ کاهش می‌یابد. مقدار دقت مدل MobileNetV2 بر روی داده اعتبارسنجی از ۹۴/۹٪ شروع شده و پس از ۲۵ اپیک تا ۹۷/۷٪ افزایش می‌یابد. مقدار زیان برای این مدل نیز از ۰/۱۲ تا ۰/۰۶ نزول پیدا کرده‌است. دقت مدل VGG-16 پس از ۵۰ اپیک ۲۰٪ افزایش داشته‌است و از ۸۰٪ به ۹۶٪ رسیده‌است. به طور مشابه مقدار زیان برای این مدل کاهش ۷۷ درصدی داشته و از ۰/۴ به ۰/۰۹ رسیده‌است.

۴-۵- آزمایش مدل‌های شبکه عصبی

برای ارزیابی عملکرد مدل‌های شبکه عصبی پیاده‌سازی شده، قبل از پیاده‌سازی سیستم تشخیص خواب‌آلودگی، ماتریس سردرگمی^{۴۸}، دقت، صحت^{۴۹}، پوشش^{۵۰}، امتیاز F1، منحنی ROC و زمان آزمایش را محاسبه می‌کنیم. پیش‌پردازش روی داده‌های آزمایش همانند پیش‌پردازش روی داده آموزش است. برای جزئیات این پیش‌پردازش می‌توانید به بخش ۴-۱ در صفحه ۳۲ مراجعه کنید. معیارهای مطرح شده برای ارزیابی مدل نیز به کمک کتابخانه sklearn محاسبه شده‌است.

^{۴۸} Confusion Matrix

^{۴۹} Precision

^{۵۰} Recall

تمام آزمایش‌ها در این مطالعه با استفاده از لپ تاپ Intel(R) Core(TM) i5-8265U و ۱۲ گیگابایت حافظه در جوپیتر نوت‌بوک انجام شده‌است. در مجموع از ۲۵۰۰ تصویر برای آزمایش استفاده شده‌است که از این تعداد ۱۳۰۰ تصویر خواب‌آلود و ۱۲۰۰ تصویر دیگر، تصاویر غیر خواب‌آلود هستند.

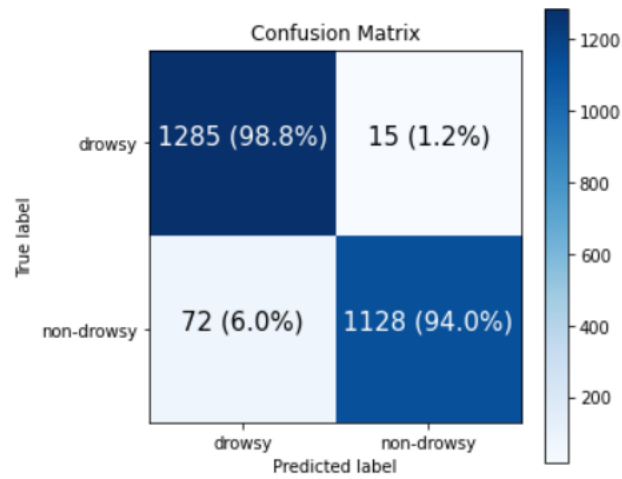
۴-۵-۱- ماتریس سردرگمی

ماتریس سردرگمی معمولاً به صورت جدول برای توصیف عملکرد مدل‌های طبقه‌بندی نشان داده می‌شود [۸]. نمونه‌ای از یک ماتریس سردرگمی برای طبقه‌بندی باینری در جدول ۴-۱ نشان داده شده است.

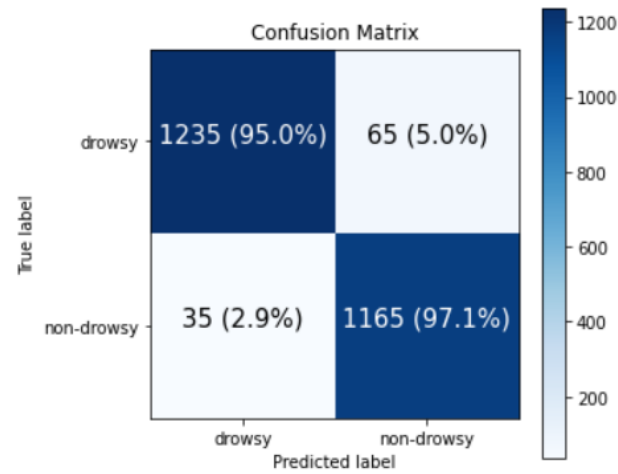
جدول ۴-۱- ماتریس سردرگمی [۸]

کلاس پیش‌بینی شده کلاس واقعی	منفی	مثبت
منفی	TN	FP
مثبت	FN	TP

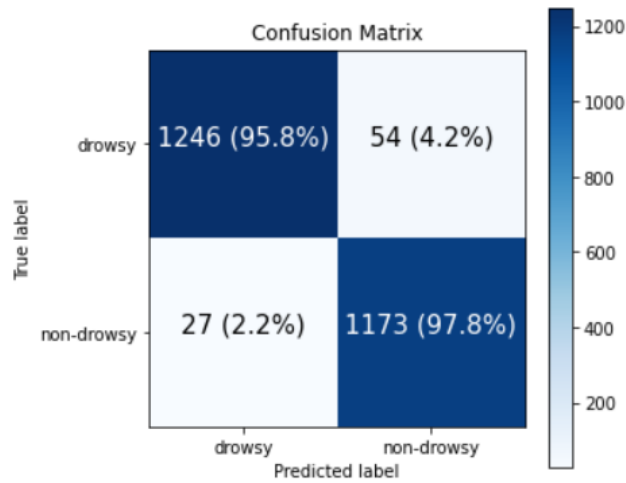
ماتریس سردرگمی نشان‌دهنده تعداد مقادیر پیش‌بینی شده و واقعی است. "TN" مخفف True Negative است و تعداد نمونه‌های منفی طبقه‌بندی شده را که به درستی تشخیص داده شده‌اند، نشان می‌دهد. به طور مشابه، "TP" مخفف True Positive است و تعداد نمونه‌های مثبت طبقه‌بندی شده را که به درستی تشخیص داده شده‌اند، نشان می‌دهد. "FP" مقدار مثبت کاذب را نشان می‌دهد و "FN" به معنای مقدار منفی کاذب است که تعداد نمونه‌های مثبت واقعی طبقه‌بندی شده به عنوان منفی را نشان می‌دهد [۸]. در کار پیشنهادی، TN نشان‌دهنده پیش‌بینی‌های صحیح است، درحالی که فرد خواب‌آلود می‌باشد و TP نشان‌دهنده پیش‌بینی‌های صحیح است در حالی که فرد خواب‌آلود نیست. به طور مشابه، مقادیر مثبت کاذب (FP) و منفی کاذب (FN) به ترتیب نشان‌دهنده پیش‌بینی‌های نادرست است درحالی که فرد خواب‌آلود است و فرد خواب‌آلود نیست. ماتریس سردرگمی مدل‌های شبکه عصبی پیچشی تمام طراحی شده را در تصویر ۴-۱۸، ۴-۱۹ و ۴-۲۰ مشاهده می‌کنید. مقدار مثبت کاذب در مدل تمام طراحی شده از سایر مدل‌ها کمتر است که نشان می‌دهد این مدل تعداد کمی از کلاس‌های خواب‌آلود را متعلق به کلاس غیر خواب‌آلود می‌داند. همچنین تعداد نمونه‌های منفی که به درستی پیش‌بینی شده، در این مدل بهتر از سایر مدل‌ها است. نرخ منفی کاذب در مدل VGG-16 از سایر مدل‌ها کمتر است، یعنی تعداد کلاس‌های غیر خواب‌آلود که خواب‌آلود پیش‌بینی می‌شوند، کمتر است.



شکل ۴-۱۸- ماتریس سردرگمی مدل تمام طراحی شده



شکل ۴-۱۹- ماتریس سردرگمی مدل MobileNetV2



شکل ۴-۲۰- ماتریس سردرگمی مدل VGG-16

۴-۵-۲- دقت

هنگام ساخت یک مدل طبقه‌بندی، باید دقت آن مدل را دانست. به طور کلی، دقت نسبتی از مشاهدات پیش‌بینی‌شده درست به کل مشاهدات است. دقت به ما کمک می‌کند تا عملکرد یک مدل را بر روی مجموعه داده ارزیابی کنیم [۸]. این پارامتر به صورت فرمول ۴-۱ محاسبه می‌شود.

$$\text{دقت} = \frac{TP + TN}{TP + FP + FN + TN} \quad \text{فرمول (۴-۱)}$$

جدول ۴-۲ دقت مدل‌های شبکه عصبی آموزش داده شده را نشان می‌دهد. مقدار دقت مدل VGG-16 از دو مدل دیگر بیشتر است و مقدار آن برابر ۹۶/۷۶٪ می‌باشد و پس از آن مدل تمام طراحی‌شده با دقت ۹۶/۵۲٪ بیشترین دقت را دارد.

جدول ۴-۲- دقت مدل‌های شبکه عصبی پیچشی

دقت	مدل شبکه عصبی پیچشی
۹۶/۵۲۰٪	مدل تمام طراحی‌شده
۹۶/۰۰۰٪	مدل MobileNetV2
۹۶/۷۶۰٪	مدل VGG-16

۴-۵-۳- صحت، پوشش و امتیاز F1

نسبت مشاهدات مثبت پیش‌بینی شده صحیح به کل مشاهدات مثبت پیش‌بینی شده را صحت می‌گویند. هر چه مقدار صحت بیشتر باشد، مدل در طبقه‌بندی مثبت بهتر است. دقت به کمک فرمول ۴-۲ محاسبه می‌شود [۸].

$$\text{صحت} = \frac{TP}{TP + FP} \quad \text{فرمول (۲-۴)}$$

پوشش، میزان پیش‌بینی صحیح موارد مثبت را در تمام نمونه‌های متعلق به گروه مثبت اندازه‌گیری می‌کند که به کمک فرمول ۴-۳ محاسبه می‌شود [۸].

$$\text{پوشش} = \frac{TP}{TP + FN} \quad \text{فرمول (۳-۴)}$$

امتیاز F1 میانگین وزنی صحت و پوشش است. بنابراین، در ارزیابی مدل از نظر صحت و پوشش مفیدتر است. امتیاز F1 به کمک فرمول ۴-۴ محاسبه می‌شود [۸].

$$\text{امتیاز F1} = \frac{\text{صحت} \times \text{پوشش} \times 2}{\text{صحت} + \text{پوشش}} \quad \text{فرمول (۴-۴)}$$

جدول ۳-۴ مقادیر صحت، پوشش و امتیاز F1 را برای هر سه مدل شبکه عصبی پیچشی، گزارش می‌کند. مقدار صحت در مدل تمام طراحی شده از سه مدل دیگر بیشتر می‌باشد، بدین معنا که این مدل بهتر از سایر مدل‌ها می‌تواند بین کلاس خواب‌آلود و غیر خواب‌آلود تمایز قائل شود. مقدار پوشش نیز برای مدل VGG-16 از سایر مدل‌ها بیشتر بوده که به این معنا است که این مدل تعداد تصاویر غیر خواب‌آلود بیشتری را به درستی پیش‌بینی کرده‌است. امتیاز F1 که میانگین وزنی صحت و پوشش را بیان می‌کند، در مدل VGG-16 بیشترین مقدار را دارد. در صورتی که بخواهیم مدل‌ها را بر اساس هر دو پارامتر پوشش و صحت مقایسه کنیم، مقایسه امتیاز F1 بهترین گزینه است.

جدول ۳-۴- صحت، دقت و امتیاز مدل‌های شبکه عصبی پیچشی

امتیاز F1	پوشش	صحت	مدل شبکه عصبی پیچشی
٪۹۶/۲۸۷	٪۹۴	٪۹۶/۶۸۸	مدل تمام طراحی شده
٪۹۵/۸۸۵	٪۹۷/۰۸۳	٪۹۴/۷۱۵	مدل MobileNetV2
٪۹۶/۶۶۳	٪۹۷/۷۵	٪۹۵/۵۹۹	مدل VGG-16

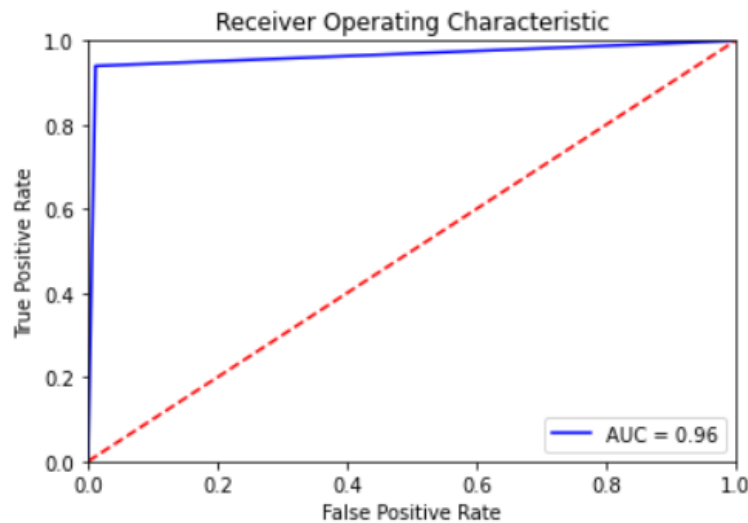
۴-۵-۴- منحنی ROC^{۵۱}

منحنی ROC یکی از مهم‌ترین معیارهای ارزیابی برای بررسی عملکرد مدل طبقه‌بندی است. این منحنی معمولاً در طبقه‌بندی باینری برای بررسی عملکرد یک طبقه‌بندی‌کننده استفاده می‌شود. مساحت زیر منحنی^{۵۲} توانایی یک طبقه‌بندی‌کننده برای تمایز بین کلاس‌ها را نشان می‌دهد و به عنوان چکیده منحنی ROC استفاده می‌شود. هر چه مساحت زیر منحنی بیشتر باشد، عملکرد مدل در تشخیص کلاس‌های مثبت و منفی بهتر است.

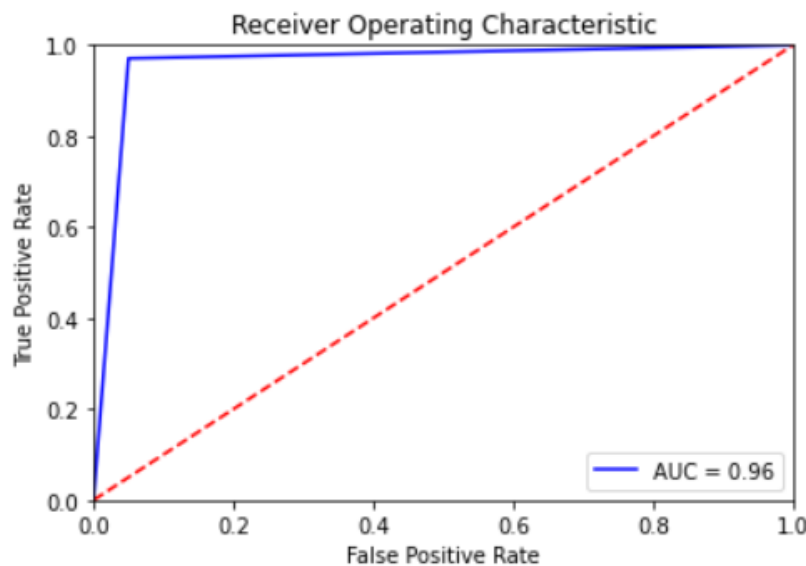
^{۵۱} Receiver operating characteristic

^{۵۲} Area Under the Curve (AUC)

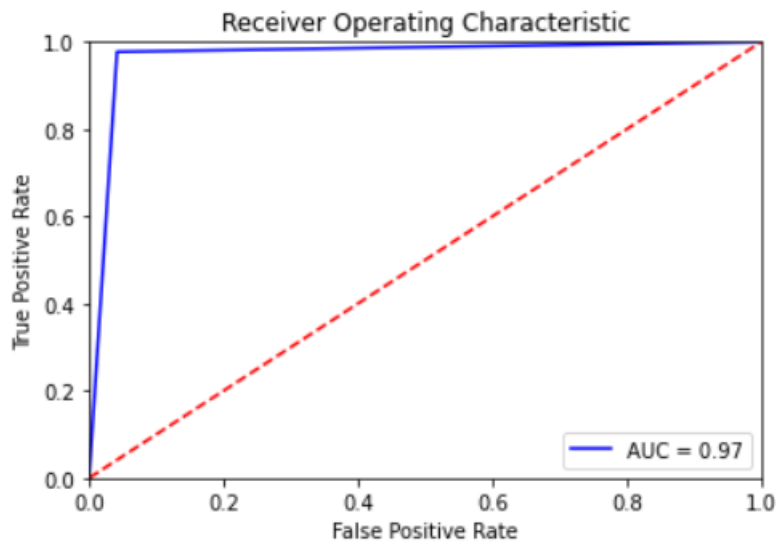
وقتی مساحت سطح زیر منحنی برابر یک است یعنی طبقه‌بندی‌کننده می‌تواند بین تمام کلاس‌های مثبت و منفی به درستی تمایز قائل شود. منحنی‌های ROC در محور Y نرخ مثبت واقعی را نشان می‌دهند که بیانگر تعداد کلاس‌های خواب‌آلود است که به درستی طبقه‌بندی شده‌اند. هم‌چنین در محور X نرخ مثبت کاذب قرار دارد که تعداد کلاس‌های خواب‌آلود است که به عنوان غیر خواب‌آلود طبقه‌بندی شده‌اند. هر چه منحنی ROC به سمت بالا و چپ متمایل شود، بیانگر عملکرد بهتر مدل می‌باشد. تصاویر ۴-۲۱، ۴-۲۲ و ۴-۲۳ منحنی ROC برای مدل‌های شبکه عصبی طراحی‌شده را نشان می‌دهند.



شکل ۴-۲۱- منحنی ROC مدل تمام طراحی‌شده



شکل ۴-۲۲- منحنی ROC مدل MobileNetV2



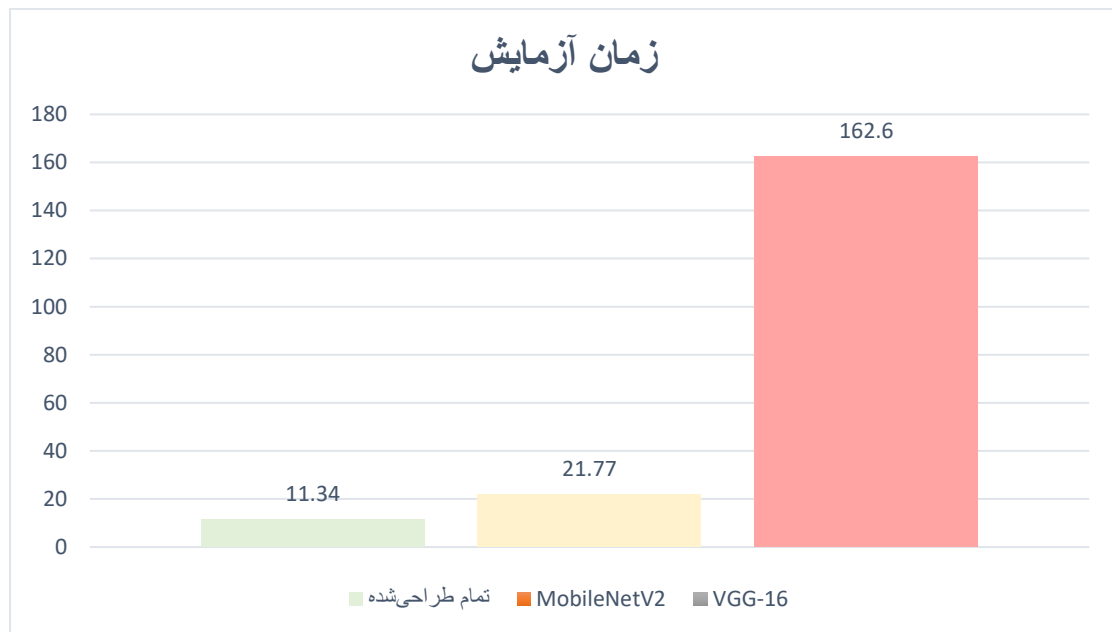
شکل ۴-۲۳- منحنی ROC مدل VGG-16

مساحت سطح زیر نمودار برای مدل تمام طراحی شده برابر ۰/۹۶ است. این مقدار توانایی مدل در تمایز بین کلاس خواب‌آلود و غیر خواب‌آلود را نشان می‌دهد. مساحت سطح زیر نمودار برای MobileNetV2 همانند مدل تمام طراحی شده برابر ۰/۹۶ و این مقدار برای VGG-16 برابر ۰/۹۷ است.

۴-۵-۵- زمان آزمایش

با توجه به اینکه سیستم تشخیص خواب‌آلودگی، تصاویر راننده را به صورت پایپ‌لاین^{۵۳} دریافت و ارزیابی می‌کند، لذا زمان یکی از موارد قابل توجه در ارزیابی مدل می‌باشد. تصویر ۴-۲۴ زمان استفاده شده برای آزمایش هر مدل را به ثانیه نشان می‌دهد. این مقدار، زمان موردنیاز هر مدل برای طبقه‌بندی داده‌های آزمایش است که تعداد آن‌ها برابر ۲۵۰۰ می‌باشد. مدل تمام طراحی شده، کم‌ترین زمان را برای طبقه‌بندی تصاویر نیاز دارد و VGG-16 بیشترین زمان را به خود اختصاص می‌دهد. زمان‌های به دست آمده نتیجه آزمایش روی لپ تاپ Intel(R) Core(TM) i5-8265U و ۱۲ گیگابایت حافظه در جوپیتر نوت‌بوک است.

^{۵۳} Pipeline



شکل ۴-۲- زمان آزمایش هر مدل به ثانیه

۴-۶- مدل منتخب

با توجه به ارزیابی هر سه مدل شبکه عصبی که در بخش ۴-۵ مطرح شد، مدل VGG-16 از نظر دقت بهتر است. هم‌چنین این مدل مقدار امتیاز F1 بیشتری نیز از سایر مدل‌ها دارد. اما یکی از مهم‌ترین معیارها در سیستم تشخیص خواب‌آلودگی زمان است. زیرا، تصاویر به صورت بی‌درنگ از دوربین دریافت می‌شوند و برای طبقه‌بندی به مدل شبکه عصبی ارسال می‌شوند. اگرچه VGG-16 از نظر دقت و امتیاز F1 بهتر از سایر مدل‌ها عمل کرده‌است، اما زمان صرف‌شده توسط این مدل در آزمایش داده‌ها بسیار زیاد است. بنابراین این مدل به عنوان مدل نهایی مدنظر قرار نمی‌گیرد. پس از VGG-16 مدل تمام طراحی شده بیشترین دقت را دارد و اختلاف دقت این مدل با VGG-16 حدود ۰/۲٪ است. با این حال، زمان طبقه‌بندی این مدل برای هر تصویر بسیار کم است و این مدل را به گزینه مناسبی برای استفاده در سیستم تشخیص خواب‌آلودگی بدل می‌کند. علاوه بر این، مدل تمام طراحی شده از نظر صحت، TN و FP از مدل MobileNetV2 بهتر عمل کرده‌است و در سایر موارد نیز تفاوت ناچیزی با دو مدل دیگر دارد. در ارتباط با سایز مدل‌ها، مدل تمام طراحی شده ۱۶/۷۱۶ مگابایت، MobileNetV2 ۷۰/۵۷۹ مگابایت و VGG-16 ۸۲/۱۳ مگابایت می‌باشد و از این نظر نیز مدل تمام طراحی شده نسبت به بقیه مدل‌ها برتری دارد، زیرا مقدار حافظه و منابع کم‌تری مصرف می‌کند. مدل تمام طراحی شده در زمان موردنیاز برای پیش‌بینی تصاویر از دو مدل دیگر بهتر است و چون زمان یکی از مهم‌ترین معیارها برای مدل منتخب است، مدل تمام طراحی شده را به عنوان مدل منتخب در سیستم تشخیص خواب‌آلودگی به کار برده‌ایم.

۴-۷- جمع‌بندی

در این بخش جزئیات طراحی و پیاده‌سازی مدل‌های شبکه عصبی را بیان کردیم و خروجی‌های به دست آمده را نشان دادیم. در ابتدا جزئیات پیش‌پردازش روی مجموعه داده آموزشی مطرح شد و سپس در بخش ۴-۲ جزئیات پیاده‌سازی سه شبکه عصبی پیچشی را بیان کردیم. در بخش بعد ابرپارامترهای استفاده شده برای آموزش شبکه‌ها را معرفی کردیم. سپس نتایج آزمایش هر یک از مدل‌ها روی مجموعه داده آزمایشی را مطرح کردیم. در نهایت نیز به کمک ارزیابی انجام شده روی هر مدل، مدل تمام طراحی شده به عنوان مدل منتخب برای سیستم تشخیص خواب‌آلودگی انتخاب شد. در فصل بعد به پیاده‌سازی سیستم تشخیص خواب‌آلودگی به همراه رابط کاربری گرافیکی جهت ارائه خدمات به کاربران پرداخته می‌شود.

فصل پنجم

پیاده‌سازی سیستم تشخیص خواب‌آلودگی

در این بخش به پیاده‌سازی سیستم تشخیص خواب‌آلودگی به کمک مدل انتخاب شده می‌پردازیم و یک واسط کاربری گرافیکی ساده نیز برای استفاده کاربران ارائه می‌دهیم. هم‌چنین به معرفی ابزار مورد استفاده برای پیاده‌سازی رابط کاربری خواهیم پرداخت.

۵-۱- منطق سیستم تشخیص خواب‌آلودگی

برای پیاده‌سازی منطق این سیستم، ابتدا به کمک کتابخانه python-OpenCV فریم تصاویر را از وب‌کم کاربر دریافت می‌کنیم. سپس یک تارای گاوسی^{۵۴} بر روی تصویر دریافت شده اعمال می‌کنیم. محو کردن تصاویر یک تکنیک رایج است که برای صاف کردن لبه‌ها و حذف نویز از تصویر استفاده می‌شود [۲۷]. سپس به سراغ تشخیص چهره و استخراج چشم‌ها می‌رویم. برای این کار، ابتدا تصویر را به تصویر سیاه و سفید تبدیل می‌کنیم، سپس برای تشخیص چهره به کمک الگوریتم ویولا و جونز از کتابخانه python-OpenCV کمک می‌گیریم. در این کتابخانه چندین مدل Haar Cascade آموزش‌دیده داریم که به عنوان فایل XML ذخیره شده‌اند. بنابراین، به جای ایجاد و آموزش مدل ویولا و جونز از ابتدا، از این فایل‌ها استفاده می‌کنیم. برای تشخیص چهره از تصویر به دست‌آمده، از فایل "haarcascade_frontalface_default.xml" استفاده می‌کنیم. هم‌چنین برای تشخیص چشم از فایل‌های "haarcascade_righeye_2splits.xml" و "haarcascade_lefteye_2splits.xml" استفاده می‌کنیم. در مرحله بعد، طبقه‌بندی‌کننده ویولا و جونز برای تشخیص چهره و چشم را بارگذاری می‌کنیم که در اینجا از سه طبقه‌بندی‌کننده استفاده شده‌است. شکل زیر نحوه ساخت این طبقه‌بندی‌کننده‌ها را نشان می‌دهد.

```
self.face_cascade = cv2.CascadeClassifier(os.path.join(THIS_FOLDER, 'data/haarcascade_frontalface_default.xml'))
self.right_eye_cascade = cv2.CascadeClassifier(os.path.join(THIS_FOLDER, 'data/haarcascade_righeye_2splits.xml'))
self.left_eye_cascade = cv2.CascadeClassifier(os.path.join(THIS_FOLDER, 'data/haarcascade_lefteye_2splits.xml'))
```

شکل ۵-۱- نحوه ساخت طبقه‌بندی‌کننده برای تشخیص چهره و چشم

شی face_cascade دارای یک متد به نام detectMultiScale است که یک فریم (تصویر) را به عنوان آرگومان دریافت می‌کند و آبشار طبقه‌بندی‌کننده^{۵۵} را روی تصویر اجرا می‌کند. قطعه کد به کار گرفته‌شده برای استفاده از این متد در شکل ۵-۲ آورده شده‌است.

^{۵۴} Gaussian blur

^{۵۵} Classifier cascade

```
faces = self.faceCascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5)
```

شکل ۵-۲- استفاده از متد detectMultiScale برای تشخیص چهره

در ادامه آرگومان‌های این تابع را مرور می‌کنیم:

- **scaleFactor**: پارامتری که مشخص می‌کند اندازه تصویر در هر مقیاس تصویر چقدر کاهش می‌یابد. با تغییر مقیاس تصویر ورودی، می‌توانید اندازه یک چهره بزرگ‌تر را به یک چهره کوچک‌تر تغییر دهید و آن را توسط الگوریتم قابل تشخیص کنید.
- **minNeighbors**: پارامتری که مشخص می‌کند هر مستطیل نامزد، باید چند همسایه داشته باشد تا آن را حفظ کند. این پارامتر بر کیفیت چهره‌های شناسایی شده تأثیر می‌گذارد. مقدار بالاتر این پارامتر، باعث تشخیص کم‌تر اما با کیفیت بالاتر می‌شود.

تشخیص چهره به عنوان مختصات پیکسل ذخیره می‌شود. برای نشان دادن چهره شناسایی شده، یک مستطیل روی آن می‌کشیم که این کار توسط متد `rectangle` در کتابخانه `python-OpenCV` و به کمک مختصات‌هایی که نشان‌دهنده ردیف و ستون پیکسل چهره در تصویر است، انجام می‌شود. حال در چهره به دست آمده، چشم را تشخیص می‌دهیم. این کار در تابع `preprocessing` انجام می‌شود. صورت تصویری متقارن است، از این رو برای تشخیص خواب‌آلودگی، بررسی یک چشم کافی است. این مسئله منجر می‌شود که زمان محاسبات نیز کاهش پیدا کند. لذا ابتدا به کمک شی `right_eye_cascade` تلاش می‌کنیم تا چشم راست را شناسایی کنیم، در صورتی که موفق به تشخیص این چشم نشدیم، اقدام به تشخیص چشم چپ می‌کنیم. حال باید چشم به دست آمده را آماده ارسال به مدل طبقه‌بند کنیم. ابتدا تصویر چشم را به اندازه 128×128 تغییر اندازه می‌دهیم و سپس مقادیر پیکسل‌ها را نرمال می‌کنیم. شکل ۵-۳ نحوه پیاده‌سازی این تابع را نشان می‌دهد.

```
def preprocessing(self, frame, roi_face_gray, roi_face_rgb):
    eye_cascade = self.right_eye_cascade
    eyes = eye_cascade.detectMultiScale(roi_face_gray, scaleFactor=1.1, minNeighbors=5)
    if(len(eyes)==0): # change eye_cascade if couldn't detect eye with current eye_cascade
        eye_cascade = self.left_eye_cascade
        eyes = eye_cascade.detectMultiScale(roi_face_gray, scaleFactor=1.1, minNeighbors=5)

    for (ex,ey,ew,eh) in eyes:
        eye = roi_face_rgb[ey:ey+eh, ex:ex+ew]
        eye = cv2.resize(eye, (self.img_size, self.img_size)) #Resize image
        cv2.rectangle(roi_face_rgb,(ex, ey),(ex+ew, ey+eh),(0,255,0), 2) #Draw green rectangle
        eye = np.expand_dims(eye, axis=0)
        eye= eye / 255
        self.prediction(frame, eye)
    break
```

شکل ۵-۳- نحوه پیاده‌سازی تابع `preprocessing`

طبقه‌بندی چشم در تابع prediction انجام می‌شود. همانطور که در شکل ۵-۴ مشاهده می‌کنید، در این تابع از مدل شبکه عصبی که در ابتدا برنامه بارگذاری کردیم، برای پیش‌بینی استفاده می‌کنیم. از آنجایی که در لایه آخر مدل از تابع سیگموئید استفاده شده‌است، خروجی مدل بین ۰ و ۱ خواهد بود. در صورتی که این مقدار کم‌تر از ۰.۵ باشد، چشم به عنوان خواب‌آلود طبقه‌بندی می‌شود.

```
def prediction(self, frame, eye):
    prediction = self.model.predict(eye)
    if(prediction[0][0]> 0.5):
        cv2.putText(frame, 'Open', (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 255, 0), 2)
        self.score = False
    else:
        cv2.putText(frame, 'Close', (100, 100), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 2)
        self.score = True
```

شکل ۵-۴- قطعه کد تابع مربوط به طبقه‌بندی چشم

در صورتی که چشم برای مدت زمان مشخصی خواب‌آلود تشخیص داده شود، زنگ هشدار به صدا در می‌آید. تغییرات در فرکانس، دامنه و مدت پلک زدن چشم، در پاسخ به افزایش خواب‌آلودگی ناشی از کم‌خوابی و اثرات ریتم شبانه‌روزی رخ می‌دهد. در حالی که مدت پلک زدن در شرایط عادی کم‌تر از ۲۰۰ میلی ثانیه طول می‌کشد، محرومیت از خواب منجر به افزایش این مقدار به بیش از ۵۰۰ میلی ثانیه می‌شود [۲۸]. لذا برای مدت زمان خواب‌آلود بودن چشم از سه حد آستانه استفاده می‌کنیم که کاربر با توجه به وضعیت خود می‌تواند یکی را انتخاب کند. این مقدار به صورت پیش‌فرض ۶۰۰ میلی ثانیه می‌باشد و دو مقدار دیگر نیز ۵۵۰ و ۶۵۰ میلی ثانیه هستند. هم‌چنین کاربران می‌توانند زنگ هشدار مدنظر خود را انتخاب کنند. در صورتی که زنگ هشدار توسط کاربر انتخاب نشود، از زنگ هشدار پیش‌فرض استفاده خواهد شد. برای پخش زنگ هشدار، از کتابخانه playsound استفاده شده‌است. نحوه استفاده از این کتابخانه را در تصویر ۵-۵ مشاهده می‌کنید.

```
from playsound import playsound

if alarmFilePath:
    playsound(alarmFilePath)
else:
    playsound(os.path.join(THIS_FOLDER, 'data/alarm.wav'))
```

شکل ۵-۵- قطعه کد مربوط به استفاده از کتابخانه playsound برای پخش زنگ هشدار

منظور از متغیر alarmFilePath مسیر زنگ هشدار انتخاب شده توسط کاربر است. تمام مراحل پیاده‌سازی

منطق سیستم در کلاس VideoCapture انجام شده‌است.

۵-۲- رابط کاربری گرافیکی

ابتدا ابزار مورد استفاده برای پیاده‌سازی رابط کاربری گرافیکی را معرفی می‌کنیم و سپس نحوه به کار گیری آن در سیستم تشخیص خواب‌آلودگی را توضیح خواهیم داد.

۵-۲-۱- معرفی ابزار پیاده‌سازی رابط کاربری

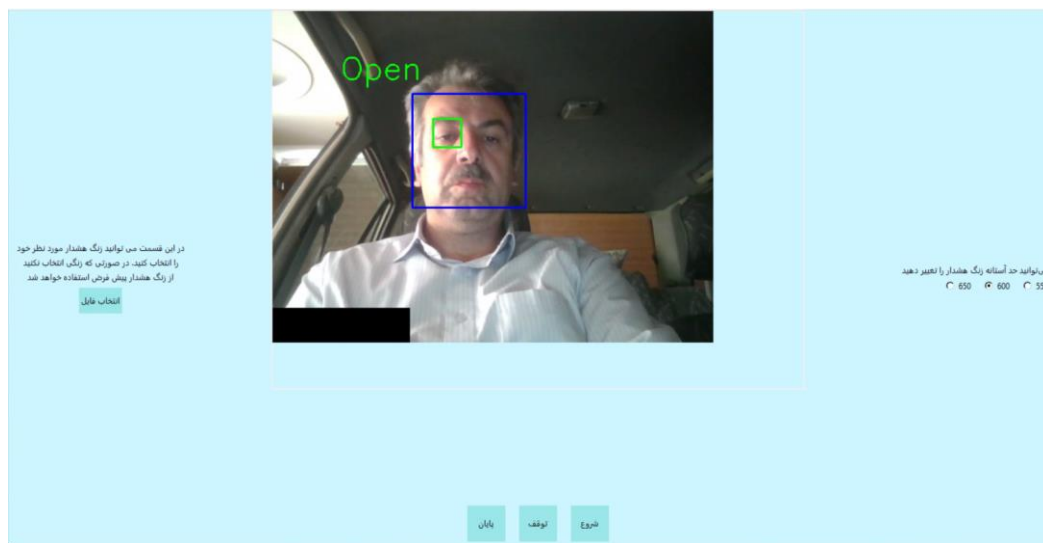
چندین ابزار معروف برای پیاده‌سازی رابط کاربری در زبان پایتون وجود دارند. از جمله‌ی آن‌ها می‌توان به موارد زیر اشاره کرد:

- Kivy
- PyQT
- TKinter
- WxPython
- PyGui
- PySide

ما برای این پروژه، کتابخانه‌ی Tkinter را انتخاب کرده‌ایم که یک رابط استاندارد پایتون است و در همه توزیع‌های استاندارد پایتون نیز گنجانده شده‌است. در مقایسه با سایر گزینه‌های موجود، Tkinter سبک‌تر و استفاده از آن آسان‌تر است. بنابراین، این گزینه برای ساخت سریع برنامه‌هایی است که می‌توانند در پلتفرم‌های مختلف کار کنند و نیازی به ظاهر مدرن ندارند [۲۸].

۵-۲-۲- پیاده‌سازی رابط کاربری

پیاده‌سازی رابط کاربری گرافیکی در کلاسی به نام App انجام شده‌است. همانطور که در شکل ۵-۶ مشاهده می‌کنید، رابط کاربری از ۴ بخش تشکیل شده‌است. قسمت سمت راست مربوط به انتخاب آستانه زنگ هشدار، قسمت سمت چپ برای انتخاب زنگ هشدار، قسمت مرکزی برای دریافت تصاویر راننده و قسمت پایینی، دکمه‌های کنترلی می‌باشند. در ادامه هر قسمت را به تفکیک شرح می‌دهیم.



شکل ۵-۶- نمای کلی رابط کاربری گرافیکی

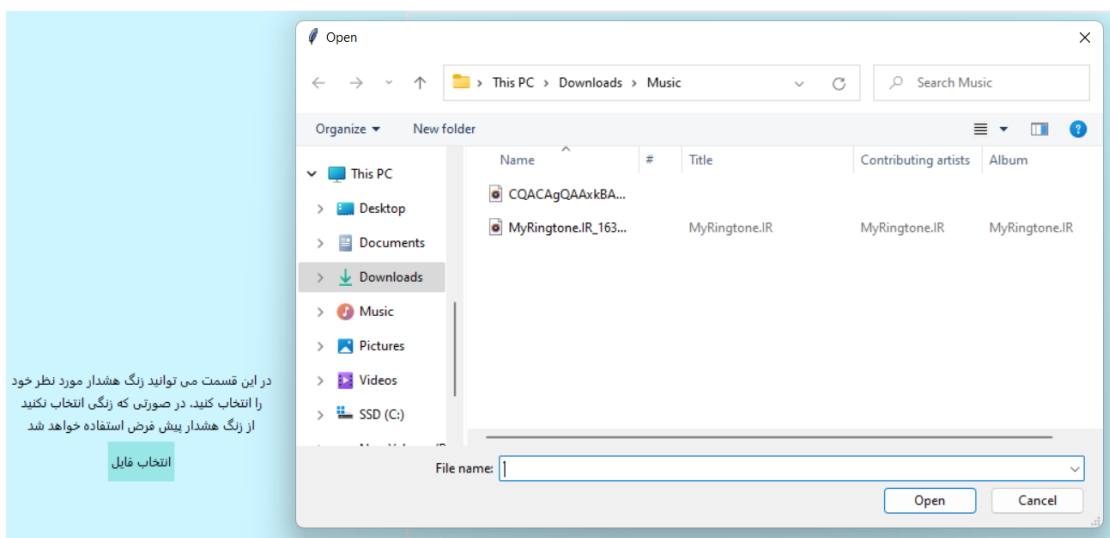
در قسمت سمت راست که مربوط به انتخاب حد آستانه برای زنگ هشدار می‌باشد، از رادیوباتن^{۵۶} برای ساخت گزینه‌ها استفاده کرده‌ایم. گزینه پیش‌فرض روی ۶۰۰ میلی ثانیه قرار دارد و کاربر می‌تواند آن را تغییر دهد. جزئیات این قسمت را در شکل ۵-۷ مشاهده می‌کنید.



شکل ۵-۷- قسمت انتخاب حد آستانه

همانطور که در شکل ۵-۸ مشاهده می‌کنید، زمانی که کاربر گزینه «انتخاب فایل» را انتخاب کند، پنجره‌ای باز شده و می‌تواند فایل موردنظر خود را برای زنگ هشدار انتخاب کند. پس از انتخاب زنگ هشدار، مسیر این فایل در متغیر alarmFilePath ذخیره می‌شود تا به کلاس VideoCapture فرستاده شود. در صورتی که زنگ هشدار توسط کاربر انتخاب نشود، از زنگ هشدار پیش‌فرض سیستم برای هشدار به راننده استفاده خواهد شد.

^{۵۶} Radiobutton



۵-۸- جزئیاتی از قسمت انتخاب رنگ هشدار

قسمت کنترلی که مربوط کنترل ضبط ویدیو و پخش آن است در شکل ۵-۹ نشان داده شده‌است. این قسمت از سه دکمه «شروع»، «توقف» و «پایان» تشکیل شده‌است. هنگامی که کاربر دکمه شروع را انتخاب کند، به کمک نمونه‌ای که از کلاس VideoCapture ساخته‌ایم، فریم‌ها را از متد `get_frame` این کلاس دریافت می‌کنیم. تصویر دریافت شده، پس از بررسی خواب‌آلودگی چشم ارسال می‌شود، لذا به کمک برجستگی که روی فریم تحت «Open» یا «Close» قرار دارد، کاربر نیز می‌تواند وضعیت خواب‌آلودگی خود را مشاهده کند. همچنین در صورتی که کاربر رنگ هشدار انتخاب کرده باشد، مسیر آن در این قسمت به عنوان آرگومان به متد `get_frame` فرستاده می‌شود.

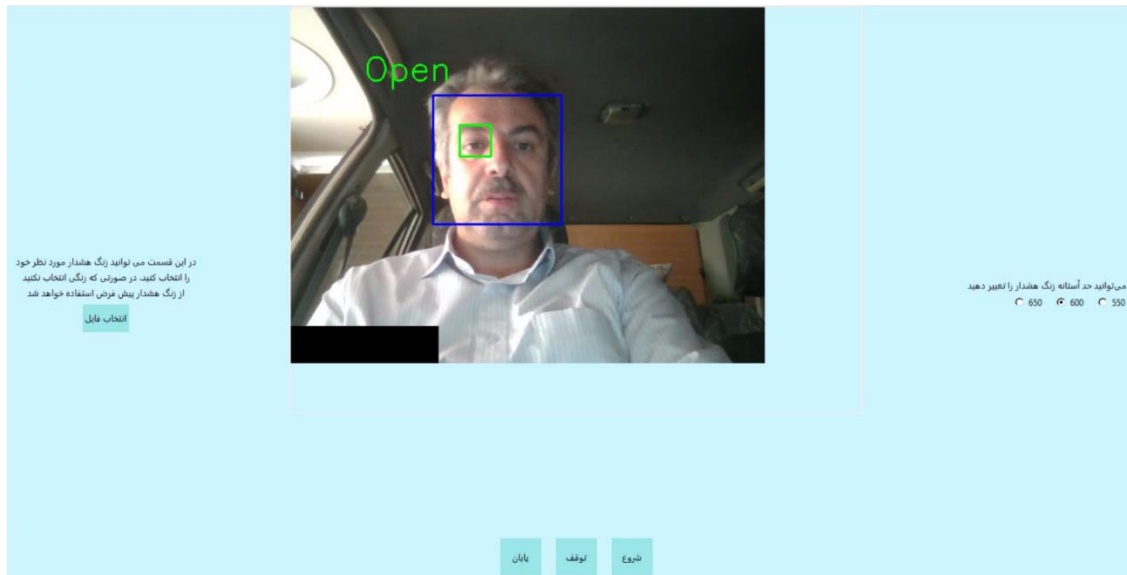


۵-۹- قسمت کنترلی رابط کاربری

با انتخاب دکمه توقف، گرفتن ویدیو و طبقه‌بندی وضعیت چشم متوقف شده و کاربر می‌تواند با انتخاب گزینه شروع، مجدد آن را آغاز کند. برای پیاده‌سازی این عملکرد از یک متغیر باینری استفاده کرده‌ایم که در صورت `True` بودن مقدار آن، متد `get_frame` فراخوانی می‌شود. با انتخاب گزینه توقف مقدار این متغیر به `False` تغییر پیدا می‌کند. در نهایت نیز، اگر کاربر گزینه پایان را انتخاب کند، گرفتن فریم متوقف شده و برنامه نیز بسته خواهد شد.

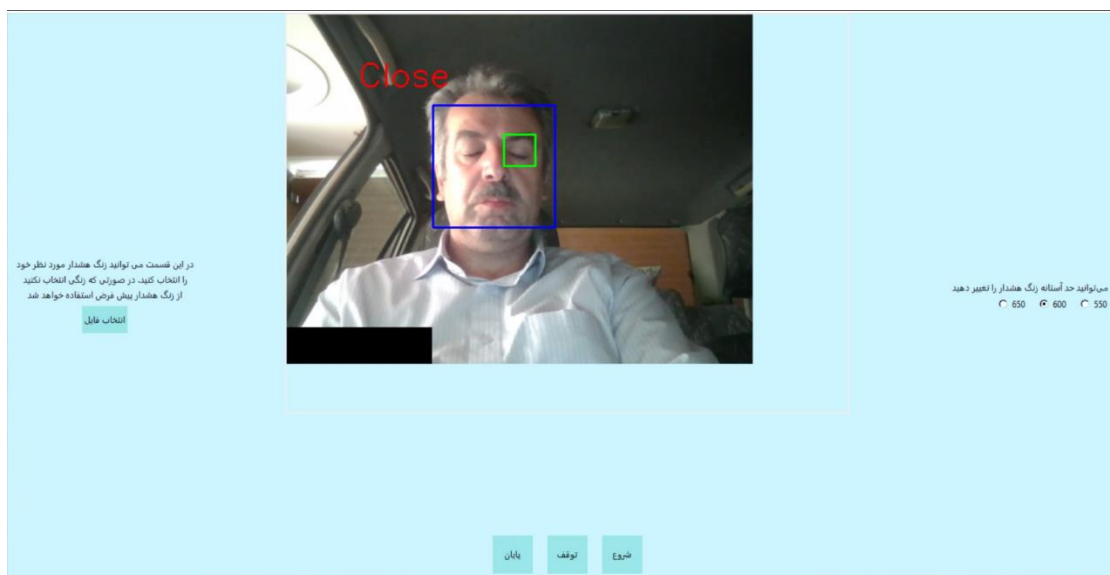
۵-۳- مثال از عملکرد کلی سیستم

در این بخش مثال‌هایی از نحوه عملکرد سیستم را با تصویر نشان می‌دهیم. تصویر ۵-۱۰ مربوط به زمانی است که راننده بدون خواب‌آلودگی در حال رانندگی است.



شکل ۵-۱۰- مثالی از رانندگی غیر خواب‌آلود

شکل ۵-۱۱ مربوط به زمانی است که چشم راننده خواب‌آلود طبقه‌بندی شده‌است. در صورتی که به مدت ۶۰۰ میلی ثانیه (در صورت تغییر ندادن این مقدار توسط کاربر)، چشم خواب‌آلود باشد، زنگ هشدار به صدا در می‌آید.



۵-۱۱- مثالی از رانندگی خواب‌آلود

۴-۵- برنامه قابل اجرا

راه اندازی یک پروژه پایتون برای غیر توسعه‌دهندگان می‌تواند خسته کننده باشد. اغلب، راه‌اندازی با باز کردن یک ترمینال شروع می‌شود که برای گروه بزرگی از کاربران بالقوه ناآشنا است. این مسئله منجر می‌شود بسیاری از کاربران قبل از اینکه شروع به نصب جزئیات محیط‌های مجازی، نسخه‌های پایتون و بی‌شمار وابستگی‌های احتمالی بپردازند، ادامه کار را متوقف کنند.

پای‌اینستالر^{۵۷}، یک برنامه پایتون و تمام وابستگی‌های آن را در یک بسته واحد جمع می‌کند. کاربر می‌تواند برنامه بسته‌بندی شده را بدون نصب مفسر پایتون یا هر ماژول دیگر اجرا کند. علاوه بر این، پای‌اینستالر می‌تواند فایل‌های اجرایی برای ویندوز، لینوکس یا مک‌اواس ایجاد کند. این بدان معناست که کاربران ویندوز یک فایل .exe، کاربران لینوکس یک فایل اجرایی معمولی و کاربران مک‌اواس یک بسته نرم افزاری دریافت می‌کنند [۳۰]. برای راحتی استفاده کاربران از این پروژه، به کمک این کتابخانه فایل قابل اجرایی برای نسخه ویندوز تهیه گردیده‌است. کاربران می‌توانند با اجرای این برنامه بدون نصب هیچ پیش‌نیازی، سیستم تشخیص خواب‌آلودگی را روی سیستم‌های شخصی خود اجرا کنند.

برای تهیه این فایل قابل اجرا، از دستور زیر استفاده می‌کنیم. app نام فایلی است که کد پایتون در آن نوشته شده‌است.

```
#pyinstaller -onefile app.py
```



پس از اجرای این دستور موارد زیر ساخته می‌شود.

- یک فایل به نام app.spec
- پوشه build
- پوشه dist

فایل app.spec به پای‌اینستالر می‌گوید که چگونه اسکریپت ما را پردازش کند. پوشه build جایی است که پای‌اینستالر فراداده‌ها^{۵۸} را برای ساخت فایل اجرایی شما قرار می‌دهد. پوشه dist نیز حاوی محصول نهایی است که می‌خواهید برای کاربران خود ارسال کنید. شکل ۵-۱۲ محتویات پوشه dist را نشان می‌دهد.

^{۵۷} PyInstaller

^{۵۸} Metadata

 data	7/22/2022 6:11 PM	File folder	
 app.exe	7/22/2022 6:25 PM	Application	553,761 KB

شکل ۵-۱۲- محتویات پوشه dist

پوشه data نیز حاوی فایل‌های موردنیاز برنامه همانند مدل شبکه عصبی، زنگ هشدار پیش‌فرض و ... می‌باشد.

۵-۵- استفاده از داکر

داکر یک پلتفرم متن‌باز است که بر مبنای سیستم عامل لینوکس راه‌اندازی شده‌است. داکر نوعی ماشین مجازی^{۵۹} است و این امکان را برای برنامه‌ها فراهم می‌کند تا از یک هسته^{۶۰} واحد لینوکس استفاده کرده و از امکاناتی بهره‌مند شوند که در سیستم عامل میزبان ارائه نشده‌است. به این ترتیب برنامه‌ها می‌توانند مستقل از پیش‌نیازها و امکانات مازاد اجرا شوند. این موضوع باعث می‌شود که سرعت و عملکرد برنامه بهبود قابل ملاحظه‌ای پیدا کند و حجم آن نیز کاهش یابد[۳۱].

داکر، کانتینرها^{۶۱} را تهیه و اجرا می‌کند. فناوری کانتینر از طریق سیستم‌عامل در دسترس است. یک کانتینر، سرویس یا عملکرد برنامه را با تمام کتابخانه‌ها، فایل‌های پیکربندی، وابستگی‌ها و پارامترهای لازم بسته‌بندی می‌کند. هر کانتینر داکر به وسیله یک داکر فایل^{۶۲} شروع به کار می‌کند. داکر فایل‌ها، فایل‌های تنظیمات داکر هستند که با استفاده از آن‌ها می‌توانیم به داکر بگوییم که یک کانتینر را چگونه بالا بیاورد و تنظیم کند. سپس به کمک فرمان Docker Build یک ایمج بر اساس محتویات داکر فایل ساخته می‌شود. ایمج یک فایل قابل حمل و شامل یک سری دستورالعمل است که مشخص می‌کند کانتینر کدام کامپوننت‌های نرم‌افزاری را اجرا کند[۳۱]. برای اینکه سایر توسعه‌دهندگان بتوانند به راحتی برنامه را اجرا کنند و تغییرات موردنظر خود را جهت ارتقا برنامه اعمال کنند، تصمیم گرفتیم به کمک داکر منطق این برنامه را پیکربندی کنیم. در گام اول، یک فایل نیازمندی^{۶۳} برای نصب پکیج‌های موردنظر می‌نویسیم. شکل ۵-۱۳ ساختار این فایل را نشان می‌دهد.

^{۵۹} Virtual Machine

^{۶۰} Kernel

^{۶۱} Containers

^{۶۲} Dockerfile

^{۶۳} Requirements

```
tensorflow==2.9.1
numpy==1.21.4
opencv-python==4.1.2.30
```

شکل ۵-۱۳- فایل نیازمندی‌ها

سپس داکر فایل موردنظر خود را می‌نویسیم. تصویر داکر فایل نوشته شده در شکل ۵-۱۴ آورده شده است.

```
FROM python:3.7
WORKDIR /app
RUN apt-get update && apt-get install -y libgl1 libsm6 libxext6 libxrender-dev
COPY requirements.txt /app
RUN pip install -r ./requirements.txt
COPY . .
CMD ["python", "app.py"]
```

۵-۱۴- داکر فایل

داکرفایل حاوی دستورالعمل‌های زیر است:

- FROM: ایمیج پایه برای ساخت (این تصویر به عنوان مثال حاوی یک نصب ساده لینوکس با پایتون ۳.۷ که قبلاً پیکربندی شده است).
- WORKDIR: پوشه کاری داخل فایل سیستم^{۶۴} کانتینر، همه فایل‌های اضافه شده در این دایرکتوری قرار می‌گیرند.
- RUN apt-get: این دستورات وابستگی‌های python-OpenCV را نصب می‌کنند که معمولاً در سیستم محلی وجود دارند، اما ممکن است در کانتینر شما وجود نداشته باشند و باعث ایجاد مشکل شوند.
- COPY <host-file> <container-dir>: فایل را از سیستم فایل میزبان <host-file> در دایرکتوری کانتینر <container-dir> کپی می‌کند.
- RUN pip install: نیازمندی‌های موردنظر را از فایل مربوطه خوانده و به کمک پیپ^{۶۵} آن‌ها را نصب می‌کند.
- CMD ["python","app.py"]: فرمانی که پس از بوت شدن کانتینر شروع می‌شود.

سپس، در یک ترمینال، دستور زیر را برای ساخت ایمیج اجرا می‌کنیم.

```
#docker build -f Dockerfile -t driver-drowsiness-detection .
```

^{۶۴} Filesystem

^{۶۵} Pip

سپس برای اینکه کانتینر بتواند از وب‌کم موجود در سیستم میزبان استفاده کند، دستور زیر را وارد می‌کنیم.

```
#xhost +
```

در نهایت نیز با دستور زیر کانتینر را اجرا می‌کنیم.

```
#sudo docker run --rm -ti --net=host --ipc=host -e DISPLAY=$DISPLAY  
-v /tmp/.X11-unix:/tmp/.X11-unix --env="QT_X11_NO_MITSHM=1" --  
device="/dev/video0:/dev/video0" driver-drowsiness-detection
```

آپشن‌هایی که در کنار دستور run به کار رفته‌اند، جهت اتصال وب‌کم سیستم‌عامل میزبان به کانتینر می‌باشد.

۵-۶- جمع‌بندی

در این فصل به نحوه پیاده‌سازی منطق سیستم تشخیص خواب‌آلودگی و رابط کاربری گرافیکی پرداختیم. در بخش ۵-۲-۱ ابزار موردنیاز برای رابط کاربری گرافیکی معرفی شد و در بخش ۵-۳-۳ مثالی از نحوه خروجی سیستم پس از استفاده‌ی کاربر از خدمات سیستم آورده‌شد. سپس در بخش ۵-۴-۴ نحوه تولید فایل قابل اجرا شرح داده‌شد. در نهایت نیز در بخش ۵-۵-۵ معرفی مختصری از داکر کرده و نحوه پیکربندی منطق برنامه به کمک آن را شرح دادیم. فصل بعدی به ارائه جمع‌بندی کلی از پروژه و پیشنهادات اختصاص داده شده‌است.

فصل ششم

نتیجه‌گیری و پیشنهادات

۶-۱- جمع‌بندی و نتیجه‌گیری

هدف نهایی این پروژه، پیاده‌سازی سیستم تشخیص خواب‌آلودگی رانندگان به کمک شبکه‌های عصبی پیچشی بود. این روش، خواب‌آلود یا غیر خواب‌آلود بودن چشم را مشخص می‌کند و هنگام خواب‌آلودگی، زنگ هشدار برای هشیار کردن راننده به صدا در می‌آید. برای رسیدن به این هدف، در فصل ابتدایی به ارائه تعریف و اهمیت سیستم تشخیص خواب‌آلودگی پرداختیم و روش‌های موجود و سیستم‌های پیاده‌سازی شده را بررسی کردیم. سپس در فصل دوم، روش‌ها و الگوریتم‌های انتخاب‌شده جهت پیاده‌سازی را با مفاهیم پایه‌شان توضیح دادیم. برای آموزش شبکه عصبی یادگیری انتقالی و روش معمولی آموزش شبکه عصبی معرفی شدند. فصل سوم، ابزارهای مورد نیاز جهت این پیاده‌سازی بیان شد و سپس مجموعه داده مورد استفاده در پیاده‌سازی شبکه عصبی پیچشی را معرفی کردیم. در فصل چهارم به ریز جزئیات بخشی از پیاده‌سازی آموزش و ارزیابی هر سه شبکه عصبی پیچشی پرداختیم و پارامترهای ارزیابی را برای هر سه مدل محاسبه کردیم. در نهایت نیز مدل تمام طراحی شده را به عنوان مدل منتخب برای استفاده در سیستم تشخیص خواب‌آلودگی انتخاب کردیم. در فصل پنجم نیز پیاده‌سازی نهایی سیستم تشخیص خواب‌آلودگی را تشریح کردیم. همچنین نحوه پیاده‌سازی رابط کاربری گرافیکی برای ارائه خدمات به کاربر توضیح داده‌شد. علاوه بر این موارد، چند مثال از نحوه استفاده از سیستم تشخیص خواب‌آلودگی مطرح شد و به معرفی ابزاری جهت ساخت فایل قابل اجرا برای این سیستم پرداختیم. در نهایت نیز ابزار داکر برای پیکربندی منطق برنامه معرفی شد.

در این روش پیشنهادی، ابتدا سه شبکه عصبی پیچشی آموزش داده شدند تا وضعیت چشم راننده را بررسی کنند. برای تشخیص چهره و استخراج ناحیه چشم از تصاویر صورت، از الگوریتم تشخیص چهره ویولا و جونز استفاده شد. اولین شبکه عصبی پیچشی، یک شبکه عصبی کاملاً طراحی شده است. از تابع فعالیت سیگموئید نیز برای طبقه‌بندی راننده به عنوان خواب‌آلود یا غیرخواب‌آلود استفاده شد. این مدل که شامل چهار لایه پیچشی و یک لایه کاملاً متصل است به دقت ۹۶/۵۴٪ رسید. در دو شبکه عصبی دیگر، از مزیت یادگیری انتقالی برای آموزش شبکه‌های پیشنهادی در مجموعه‌ی داده آموزشی خود استفاده کردیم. برای یادگیری انتقالی از مدل‌های MobileNet-V2 و VGG-16 استفاده شد که از نظر حافظه و پیچیدگی کارآمدتر هستند. پس از ایجاد تغییراتی در ویژگی‌های سطح بالا، MobileNet-V2 و VGG-16 به ترتیب به دقت ۹۶٪ و ۹۶/۷٪ رسیدند. مقدار صحت نیز در مدل تمام طراحی شده ۹۸/۶۸۸٪ و در دو مدل دیگر به ترتیب ۹۴/۷۱٪ و ۹۵/۵۹٪ است. منحنی ROC نیز برای هر سه مدل رسم شد و مساحت زیر منحنی برای آن‌ها محاسبه شد. این مقدار برای مدل تمام طراحی شده، MobileNet-V2 و VGG-16 به ترتیب ۰/۹۶، ۰/۹۶ و ۰/۹۷ به دست آمد. زمان صرف‌شده مدل تمام طراحی‌شده برای ارزیابی مجموعه داده آموزشی نیز از دو مدل دیگر کم‌تر بود. همچنین مدل تمام طراحی‌شده در

سایر پارامترها نظیر دقت، منحنی ROC، ماتریس سردرگمی و امتیاز F1 عملکرد مطلوبی داشت، لذا این مدل به عنوان مدل منتخب برای استفاده در سیستم تشخیص خواب‌آلودگی انتخاب شد.

سیستم پیشنهادی، به طور مداوم وضعیت راننده را مورد بررسی قرار می‌دهد و زمانی که مدل شبکه عصبی وضعیت چشم راننده را به طور مداوم خواب‌آلود پیش‌بینی کند، با زنگ هشدار به او هشدار می‌دهد. مدت زمانی که چشم باید خواب‌آلود پیش‌بینی شود تا زنگ هشدار به صدا در بیاید، ۶۰۰ میلی ثانیه در نظر گرفته شد. این مقدار به صورت پیش‌فرض استفاده می‌شود و کاربر می‌تواند در رابط کاربری گرافیکی آن را تغییر دهد. همچنین کاربر می‌تواند زنگ هشدار را به دلخواه خود انتخاب کند.

۶-۲- پیشنهادات

جهت پیشرفت در مدل‌های شبکه عصبی می‌توان موارد زیر را به کار گرفت:

- ابرپارامتر: در برخی از ابرپارامترها مانند نرخ یادگیری، تعداد لایه و تعداد نورون‌های شبکه عصبی در لایه کاملاً متصل تغییراتی ایجاد کرد.
- مدل‌های یادگیری عمیق: برای این پروژه از دو مدل MobileNet-V2 و VGG-16 استفاده شد که برای تشخیص اشیا روی پایگاه داده ImageNet آموزش دیده‌اند. مدل یادگیری عمیق دیگری که برای سیستم تشخیص خواب‌آلودگی می‌تواند موثر باشد، EfficientNetB0 است. این مدل علاوه بر دقت مناسب، تعداد پارامترهای کمی نیز دارد.
- کاهش پارامتر و پیچیدگی زمانی: در جهت کاهش پارامترهای قابل آموزش، پیچیدگی زمانی و زمان پیش‌بینی مدل، تغییراتی در مدل شبکه عصبی ایجاد کنیم.

برای هر سه شبکه عصبی پیچشی، افزودن کلاس سوم به کلاس‌های «خواب‌آلود» و «غیر خواب‌آلود» نیز می‌تواند مفید باشد. این کلاس جدید که می‌تواند «کم‌هوشیار» نامیده شود، می‌تواند برای جلوگیری از منفی کاذب استفاده شود. یکی از روش‌هایی که برای انجام این کار می‌توان به کار برد، این است که به جای تقسیم‌بندی خروجی تابع سیگموئید به دو قسمت، آن را به سه بازه تقسیم کنیم و بازه میانی را به عنوان کلاس کم‌هوشیار معرفی کنیم. علاوه بر این، یکی دیگر از پیشرفت‌های احتمالی در سیستم می‌تواند اضافه کردن ویژگی‌های جانبی مانند خمیازه باشد.

این سیستم فقط بر اساس ویژگی‌های رفتاری یک راننده پیاده‌سازی شده است و وسیله نقلیه و اندازه‌گیری‌های فیزیولوژیکی در نظر گرفته نمی‌شوند، زیرا بسیار گران بوده و استفاده از آن‌ها در فضای واقعی

دشوار است و کارآمد نیز نمی‌باشد. اندازه‌گیری‌های فیزیولوژیکی نیازمند حسگرها و تجهیزات بسیار گران هستند. بنابراین، اگر در آینده به دلیل پیشرفت فناوری در زمینه سخت‌افزار، این تجهیزات کم‌هزینه‌تر و کارآمدتر شوند، می‌توان آن‌ها را با داده‌های رفتاری ترکیب کرد تا نتیجه قابل اطمینان‌تری به دست‌آید. علاوه بر این، استفاده از مدل‌های یادگیری عمیق در مجموعه داده‌های متنوع‌تر نیز ویژگی‌های اضافی را استخراج می‌کند و نتیجه رضایت‌بخش‌تری را ارائه می‌دهد.

مراجع و منابع

- [١] V. R. R. Chirra, S. ReddyUyyala, and V. K. K. Kolli, "Deep CNN: A machine learning approach for driver drowsiness detection based on eye state," *Rev. d'Intelligence Artif.*, vol. 33, no. 6, pp. 461-466, 2019.
- [٢] "World Health Organization The top ten causes of death." <https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death> (accessed 22-March, 2022).
- [٣] M. Poursadeghiyan, A. Mazloumi, G. N. Saraji, M. M. Baneshi, A. Khammar, and M. H. Ebrahimi, "Using image processing in the proposed drowsiness detection system design," *Iranian journal of public health*, vol. 47, no. 9, p. 1371, 2018.
- [٤] A. Čolić, O. Marques, and B. Furht, *Driver drowsiness detection: Systems and solutions*. Springer, 2014.
- [٥] S. Park, F. Pan, S. Kang, and C. D. Yoo, "Driver drowsiness detection system based on feature representation learning using various deep networks," in *Asian Conference on Computer Vision*, 2016: Springer, pp. 154-164.
- [٦] I.-H. Choi, S. K. Hong, and Y.-G. Kim, "Real-time categorization of driver's gaze zone using the deep learning techniques," in *2016 International conference on big data and smart computing (BigComp)*, 2016: IEEE, pp. 143-148.
- [٧] R. Jabbar, K. Al-Khalifa, M. Kharbeche, W. Alhajyaseen, M. Jafari, and S. Jiang, "Real-time driver drowsiness detection for android application using deep neural networks techniques," *Procedia computer science*, vol. 130, pp. 400-407, 2018.
- [٨] A.-C. Phan, N.-H.-Q. Nguyen, T.-N. Trieu, and T.-C. Phan, "An efficient approach for detecting driver drowsiness based on deep learning," *Applied Sciences*, vol. 11, no. 18, p. 8441, 2021.
- [٩] M. Dua, R. Singla, S. Raj, and A. Jangra, "Deep CNN models-based ensemble approach to driver drowsiness detection," *Neural Computing and Applications*, vol. 33, no. 8, pp. 3155-3168, 2021.
- [١٠] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, 2001, vol. 1: IEEE, pp. 3.

-
- [١١] S. Anber, W. Alsaggaf, and W. Shalash, "A hybrid driver fatigue and distraction detection model using AlexNet based on facial features," *Electronics*, vol. 11, no. 2, p. 285, 2022.
- [١٢] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, 2017: Ieee, pp. 1-6.
- [١٣] "شبكة عصبية كائولوشن." <https://howsam.org/convolutional-neural-network/> (accessed 6-July, 2022).
- [١٤] S. Tammina, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 10, pp. 143-150, 2019.
- [١٥] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [١٦] "Tensorflow." <https://www.tensorflow.org/> (accessed 13-July, 2022).
- [١٧] "Colaboratory." <https://research.google.com/colaboratory/faq.html> (accessed 20-March, 2022).
- [١٨] "How to finally install tensorflow GPU on Windows 10 in 2022." <https://towardsdatascience.com/how-to-finally-install-tensorflow-gpu-on-windows-10-63527910f255> (accessed 20-March, 2022).
- [١٩] "OpenCV." <https://docs.opencv.org/4.x/d1/dfb/intro.html> (accessed 20-March, 2022).
- [٢٠] "scikit-learn." <https://scikit-learn.org/stable/> (accessed 20-March, 2022).
- [٢١] "NumPy." <https://numpy.org/> (accessed 20-March, 2022).
- [٢٢] "Pillow." <https://pillow.readthedocs.io/en/stable/> (accessed 21-March, 2022).
- [٢٣] "MRL Eye Dataset|MRL." <http://mrl.cs.vsb.cz/eyedataset> (accessed 4-March, 2022).
- [٢٤] "Training Data vs. Validation Data vs. Test Data for ML Algorithms." <https://www.applause.com/blog/training-data-validation-data-vs-test-data> (accessed 10-July, 2022).

- [۲۵] M. Hashemi, A. Mirrashid, and A. Beheshti Shirazi, "Driver safety development: Real-time driver drowsiness detection system based on convolutional neural network," SN Computer Science, vol. 1, no. 5, pp. 1-10, 2020.
- [۲۶] E. Tuba, N. Bačani, I. Strumberger, and M. Tuba, "Convolutional neural networks hyperparameters tuning," in Artificial intelligence: theory and applications: Springer, 2021, pp. 65-84.
- [۲۷] E. Magán, M. P. Sesmero, J. M. Alonso-Weber, and A. Sanchis, "Driver drowsiness detection by applying deep learning techniques to sequences of images," Applied Sciences, vol. 12, no. 3, p. 1145, 2022.
- [۲۸] V. E. Wilkinson et al., "The accuracy of eyelid movement parameters for drowsiness detection," Journal of clinical sleep medicine, vol. 9, no. 12, pp. 1315-1324, 2013.
- [۲۹] "TKinter." <https://docs.python.org/3/library/tkinter.html> (accessed 21-March, 2022).
- [۳۰] "Pyinstaller." <https://pyinstaller.org/en/stable/> (accessed 21-March, 2022).
- [۳۱] J. Nickoloff and S. Kuenzli, Docker in action. Simon and Schuster, 2019.