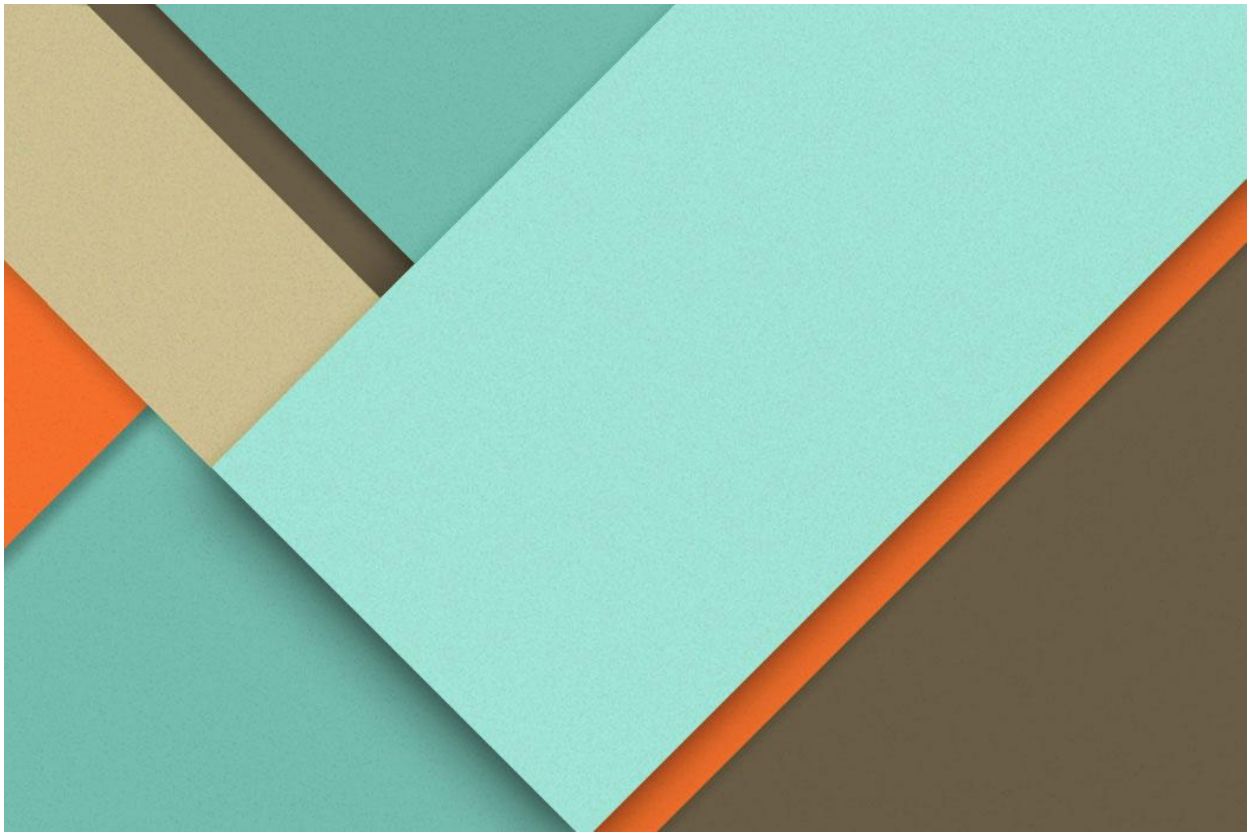


دستیار کشف خواب‌آلودگی راننده

بهار ۱۴۰۱

علی قربان‌پور - ۹۶۱۰۵۹۹۴

امین روانبخش - ۹۶۱۰۹۷۲۵

علی قدیرلی - ۹۶۱۰۵۹۸۳

شروین جهانبخش - ۹۶۱۰۵۶۷۲

گروه ۸

آزمایشگاه سخت‌افزار - دکتر اجلالی

فهرست

مقدمه

با فراگیری علوم کامپیوتر در حوزه‌های مختلف، یکی از مهم‌ترین کاربردهای آن در زمینه افزایش کیفیت زندگی انسان‌ها و افزایش امنیت زندگی آن‌ها می‌باشد. کاربرد کامپیوتر و ابزارهای کامپیوتری به منظور نیل به این اهداف نیز از این امر مستثنی نیست. به دلیل قدرت روزافزون سامانه‌های کامپیوتری و گسترش آن‌ها در زندگی روزمره انسان‌ها می‌توان از این ابزار قدرتمند و در دسترس برای حل بسیاری از مشکلات که پیش از این وجود داشتند ولی قادر به حل آن‌ها نبوده‌ایم استفاده کنیم. با این اوصاف می‌توانیم تعریف پروژه‌ی پیش رو را به گونه‌ای توجیه کنیم که این توجیه در راستای در درجه اول امنیت انسان‌ها و در درجه‌ی بعدی کاهش خسارات و زیان‌های مالی ناشی از تصادفات جاده‌ای می‌باشد. تصادفات جاده ای یکی از مرگبارترین و کشنده ترین اتفاقات ممکن هستند که امروزه می توانیم چنین اتفاقاتی را به شدت از طریق رسانه های اجتماعی و صفحات مجازی مشاهده کنیم که قطعا برای ما اتفاقی خوشایند محسوب نمی شوند و می توانند حتی باعث ایجاد نوعی دغدغه ذهنی در ما شوند. افزایش امنیت خودروها و کاهش تلفات ناشی از تصادفات رانندگی دغدغه‌ی همیشگی نهادهای مختلف اجتماعی اعم از پلیس راهنمایی و رانندگی، شهرداری‌ها، وزارت‌خانه‌های مختلف راه و شهرسازی، کمپانی‌های خودروسازی و غیره بوده است. این گستردگی نیاز و اهمیت بالای مسئله به دلیل درگیری این موضوع با جان انسان‌ها اهمیت این ماجرا را دوچندان می‌کند.

حال که اهمیت و گستردگی موضوع محل بحث آشکار گردید، نگاهی به برخی اطلاعات مرتبط با مسئله می‌اندازیم تا توجیهی بر طرح ارائه‌گردیده باشد. طبق آمار پلیس راهور جاده‌ای، حدود ۴۰ درصد تصادفات ناشی از خواب‌آلودگی رانندگان است که این مورد نشان از نسبت بالای این حالت در آمار تصادفات دارد. طراحی و پیاده‌سازی سامانه‌ای که بتواند این خواب‌آلودگی را تشخیص دهد و به نحوی به راننده هشدار دهد می‌تواند اثر شگرفی در آمار تصادفات جاده‌ای و امنیت و سلامت شهروندان داشته باشد. مورد دیگری که باید به آن اشاره کرد این است که این آمار چهل درصدی از کل تصادفات به وقوع پیوسته در حالی‌ست که عموما تصادفات ناشی از خواب‌آلودگی و عدم هوشیاری راننده از تبعات بسیار بالاتری نسبت به حالات دیگر تصادف برخوردارند و عموما تلفات در این نوع تصادفات بسیار

گسترده‌تر خواهد بود. با این اوصاف ما بر آن شدیم تا در جهت تولید سامانه‌ای با این کارکرد قدم برداریم.

فازبندی پروژه

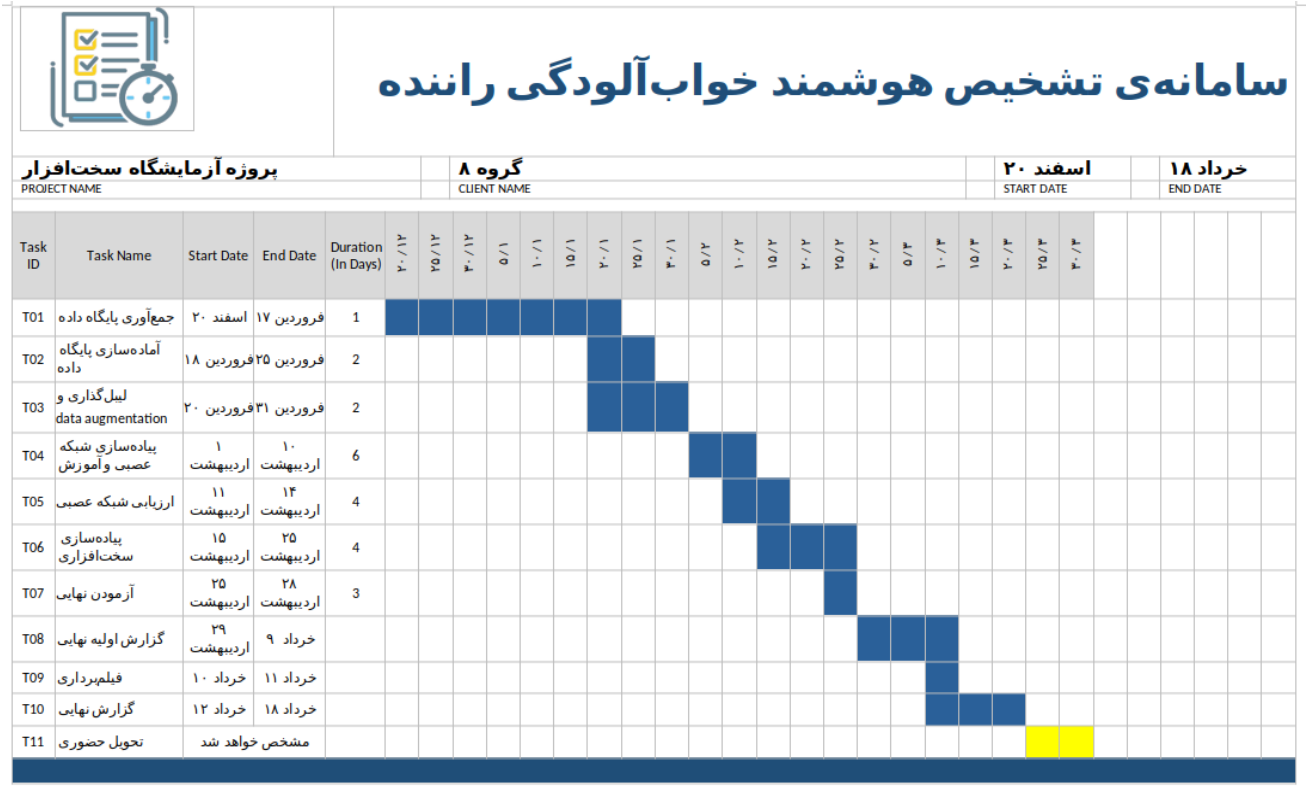
در این بخش می‌خواهیم مراحل پیاده‌سازی پروژه را که به صورت یک برنامه‌ی زمانی مدون درآمده است شرح دهیم و به بررسی جزییات هر فاز بپردازیم. به منظور پیاده‌سازی هر چه بهتر هر پروژه‌ای، نیاز است تا مراحل پیاده‌سازی آن به دقت بررسی شود و در نهایت با فازبندی و اجرای هر فاز، گام به گام به تکمیل پروژه پرداخت. پروژه پیش رو نیز از این امر مستثنی نیست. پروژه‌ی پیش رو از دو بخش کلی نرم‌افزاری و سخت‌افزاری تشکیل شده است که تکمیل فاز نرم‌افزاری آن مقدم بر فاز سخت‌افزاری می‌باشد. بنابراین در ابتدا به تکمیل فاز سخت‌افزاری آن پرداخته و پس از آن به ادغام فازهای پیشین با مرحله‌ی سخت‌افزاری می‌پردازیم. بدین ترتیب پروژه در یک نگاه کلی در ابتدا در فاز نرم‌افزاری تکمیل می‌شود، سپس به تکمیل فاز سخت‌افزاری و ادغام دو گام پیشین می‌پردازیم.

با بررسی‌های انجام شده برای پیاده‌سازی این پروژه، توانستیم تا مراحل پیاده‌سازی هر یک از فازهای سخت‌افزاری و نرم‌افزاری آن را به بخش‌های زیر تقسیم کنیم.

- **فاز نرم‌افزاری:** فاز نرم‌افزاری پروژه که فاز نخست از دو فاز اصلی پروژه می‌باشد خود از ۲ مرحله تشکیل شده است:
  1. تهیه، تکمیل و آماده‌سازی پایگاه داده مناسب به منظور آموزش شبکه‌ی عصبی عمیق به هدف یادگیری حالت خواب‌آلودگی یا عدم خواب‌آلودگی راننده.
  2. پیاده‌سازی یک شبکه‌ی عصبی و آموزش آن با دادگان آماده‌شده در گام پیشین به هدف رسیدن به یک مدل نهایی که قابلیت تشخیص حالت خواب‌آلوده را از غیر این حالات دارا باشد.
- **فاز سخت‌افزاری:** فاز سخت‌افزاری از نیز خود از دو بخش اصلی تشکیل می‌شود:
  1. تهیه، سرهم‌بندی و تست سخت‌افزار مناسب به منظور آماده‌سازی بستر مد نظر برای پیاده کردن نرم‌افزار آماده‌شده در گام پیشین.
  2. ادغام دو فاز سخت‌افزاری و نرم‌افزاری به منظور تکمیل پروژه و تست نهایی.

هر یک از مراحل فوق به تفصیل در بخش مرتبط با آن توضیح داده خواهد شد.

حال که مراحل تکمیل پروژه به وضوح مشخص گردید، لازم است تا یک نمودار زمانی پیاده‌سازی پروژه به منظور در دست داشتن تخمین مناسبی از زمان اجرای پروژه داشته باشیم. نمودار زمانی زیر، برنامه‌ی ارائه‌شده به منظور تکمیل پروژه‌ی در دست می‌باشد. بدین ترتیب می‌توانیم در موعد مناسب به تکمیل این پروژه بپردازیم. شایان توجه است که این برنامه زمانی با توجه به تخمین زمانی پیاده‌سازی هر مرحله، زمان‌ها مقرر شده به منظور تحویل فازهای مختلف پروژه در کلاس و همچنین زمان پایانی ارائه‌ی پروژه‌ها تنظیم گردیده است.



سامانه‌ی تشخیص هوشمند خواب‌آلودگی راننده

پروژه آزمایشگاه سخت‌افزار	گروه ۸	اسفند ۲۰	خرداد ۱۸
PROJECT NAME	CLIENT NAME	START DATE	END DATE

Task ID	Task Name	Start Date	End Date	Duration (In Days)
T01	جمع‌آوری پایگاه داده	اسفند ۲۰	فروردین ۱۷	1
T02	آماده‌سازی پایگاه داده	فروردین ۱۸	فروردین ۲۵	2
T03	لیبل‌گذاری و data augmentation	فروردین ۲۰	فروردین ۳۱	2
T04	پیاده‌سازی شبکه عصبی و آموزش	۱ اردیبهشت	۱۰ اردیبهشت	6
T05	ارزیابی شبکه عصبی	۱۱ اردیبهشت	۱۴ اردیبهشت	4
T06	پیاده‌سازی سخت‌افزاری	۱۵ اردیبهشت	۲۵ اردیبهشت	4
T07	آزمودن نهایی	۲۵ اردیبهشت	۲۸ اردیبهشت	3
T08	گزارش اولیه نهایی	۲۹ اردیبهشت	خرداد ۹	
T09	فیلم‌برداری	خرداد ۱۰	خرداد ۱۱	
T10	گزارش نهایی	خرداد ۱۲	خرداد ۱۸	
T11	تحویل حضوری	مشخص خواهد شد		

جدول زمانی پیاده‌سازی پروژه

شرح پروژه

هدف از این پروژه پیاده سازی یک شبکه عصبی برای تشخیص خواب آلودگی افراد در حین رانندگی بر روی یک سیستم سخت افزاری می‌باشد. با توجه به ساختار بندی ارائه شده، در فاز اول پروژه ما به جمع آوری دیتاست عکس‌های مورد نیاز برای یادگیری شبکه عصبی پرداختیم. همان طور که می‌دانید، برای شبکه‌های عصبی پیشرفته نیاز به جمع آوری حجمی زیادی از داده‌ها برای آموزش شبکه هستیم. همچنین با توجه به اینکه شبکه ما به عنوان ورودی عکس افراد را دریافت می‌کند، اندازه دیتاست حجیم خواهد بود. به همین منظور نیاز بود تا ما عکس‌ها را به گونه‌ای از نظر اندازه کوچک کنیم تا بتوانیم در ادامه فرآیند پیاده سازی این سیستم با سرعت بیشتری کار را پیش ببریم. به همین منظور از روش PCA استفاده کردیم و همچنین عکس‌ها را به صورت سیاه و سفید زخیره سازی کرده‌ایم. همچنین با توجه به تحقیقات ما بیشتر شبکه‌های عصبی برای انجام این عملیات توجه خود را به سمت چشم‌های فرد راننده منعطف می کنند. بنابراین برای کوچک شدن حجم داده تنها عکس چشم‌های افراد را ذخیره کردیم و در زمان اجرا نیز عکس چشم‌های فرد راننده را به شبکه عصبی خواهیم داد. همچنین برای جدا کردن قسمت چشم فرد راننده یک شبکه عصبی جدا پیاده سازی خواهیم کرد.

فاز یک - فاز نرم‌افزاری

۱. تهیه پایگاه داده مناسب:

برای تهیه مجموعه دادگان از منابع متعددی که از طریق شبکه اینترنت در دسترس است بهره بردیم و منابع مختلفی را طرح و بررسی کردیم. نقاط قوت و ضعف‌های متعددی داشتند هر کدام از این منابع مانند تعداد تصاویر در هر مجموعه، یا کیفیت خروجی‌ای که افراد مختلف توانستند روی این مجموعه دادگان بگیرند به نوعی نشانگر کیفیت این مجموعه دادگان نیز است. در نهایت با بررسی منابع متعدد توانستیم به انتخاب نهایی برسیم. هر کدام از مجموعه‌های بررسی شده و نقاط قوت و ضعف آن‌ها را می‌توانید در ادامه مشاهده کنید.

- Driver Drowsiness Detection Dataset

  Computer Vision Lab, National Tsuing Hua University

  http://cv.cs.nthu.edu.tw/php/callforpaper/datasets/DDD/

برای دسترسی به دادگان این مجموعه نیاز به ارتباط با مالکین این دیتاست و توضیح شرایط استفاده بوده است. همچنین تنوع فردی این پایگاه داده در مجموعه خود پایین بوده و با افراد محدودی سعی کرده‌اند شرایط را شبیه‌سازی کنند. همچنین با توجه به اهمیت پیاده‌سازی پروژه‌ی ما و تاکید آن بر حالت چشم‌ها، اینکه مجموعه‌ی آموزش از افرادی با حالت چشم‌های بادامی باشد و آزمودن این شبکه با افرادی غیر از آن نژاد باشد می‌تواند در نتیجه نهایی موثر واقع شود. به همین دلیل این مجموعه رد شد.

- UTA Real-Life Drowsiness Dataset

  https://sites.google.com/view/utarldd/home

این مجموعه نیز از مجموعه‌های خوب بوده است که بسیار با کیفیت بالا تهیه شده است. منتهی چندین مشکل برای این مجموعه نیز وجود داشته. نخست آنکه حجم دیتاست بسیار بالا بوده که این مورد ما را از کار کردن با این مجموعه منصرف کرده است. مورد دیگر آنکه این مجموعه تصویر نبوده و فیلم است که هر فیلم از وضعیت‌های مختلف برداشت شده است. بدین ترتیب این خود یک مرحله بنسبت زمانبر به کارهای ما اضافه میکرد تا فیلم را به تصویر تبدیل کنیم. بدین ترتیب این مجموعه نیز کنار گذاشته شد.

مجموعه دادگان نهایی

- driver drowsiness

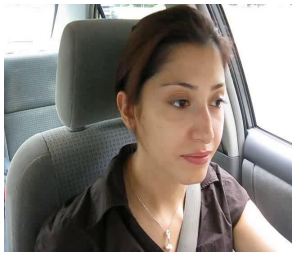
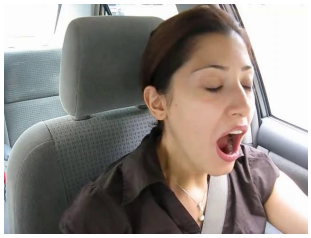
https://www.kaggle.com/code/adinishad/driver-drowsiness-using-keras/data

این مجموعه از بین گزینه‌های موجود بهترین مجموعه بود. تصاویر کم‌حجم و همچنین مجموعه دادگان کاملا تمیز شده و مراحل اولیه preparation آن به صورت کامل انجام شده است که این مورد خود کمک بزرگی در تسریع و کیفیت پیاده‌سازی آن انجام می‌دهد. این مجموعه حدودا ۱۷۰ مگابایت می‌باشد که تصاویری از حالات مختلف صورت و چشم افراد در حالت‌های مختلف به ما ارائه می‌دهد که بسیار مناسب آموزش شبکه عصبی ما است.

دادگان این مجموعه در ۴ دسته‌ی زیر تقسیم می‌شوند:

Closed, Open, No yawn, Yawn



جمع‌بندی

بدین ترتیب با این مجموعه دادگان در فاز بعدی به آماده‌سازی دادگان برای پیاده‌سازی، طراحی و آموزش شبکه عصبی پیچشی می‌پردازیم تا بتوانیم سامانه تشخیص خواب‌آلودگی راننده را در سمت نرم‌افزاری پیش ببریم. بدین منظور با تکنیک‌های data augmentation به افزایش تعداد دادگان و قدرت مجموعه می‌افزاییم.

۲. آموزش شبکه عصبی به منظور تشخیص حالت خواب‌آلودگی:

مسائلی که به جان انسان‌ها وابسته است و بر کیفیت زندگی و سلامت آن‌ها اثر می‌گذارد همیشه مورد توجه خاص حوزه‌های مختلف تکنولوژی بوده است. به همین علت پروژه‌ی جاری که دستیاری‌ست برای هشدار به رانندگان خواب‌آلود می‌تواند در جهت این مهم قدم بردارد. در مراحل پیشین به تفصیل توضیح داده‌ایم که اهمیت این پروژه چیست، چگونه می‌خواهیم این پروژه را به انجام رسانیم و در نهایت گام‌های پروژه را به دقت شرح دادیم. در این مرحله که در ادامه‌ی کارهای مرحله‌ی پیشین است، می‌خواهیم به صورت عملیاتی مدل یادگیری ماشین اصلی را بر روی دادگان انتخاب شده پیاده کنیم تا بدین صورت بخش نرم‌افزاری پروژه تکمیل و آماده‌ی بهره‌برداری گردد.

در این بخش که در ادامه‌ی فاز پیشین می‌باشد، می‌خواهیم ابتدا پردازشی بر روی دادگان انتخاب شده در مرحله‌ی قبل انجام دهیم. سپس بعد از آماده‌سازی این مجموعه دادگان، یک مدل یادگیری ماشین را از پایه پیاده‌سازی کنیم. در مرحله‌ی بعد این مدل را با مجموعه دادگان در دست آموزش می‌دهیم و سعی می‌کنیم با تنظیم بهینه‌ی پارامترهای آن به بالاترین دقت ممکن دست بیابیم. در نهایت نیز مدل آموزش‌داده‌شده را ذخیره می‌کنیم تا بتوانیم در گام نهایی از آن استفاده کرده و بر روی سخت‌افزار مد نظر خود پیاده‌سازی کنیم.

برای پیاده‌سازی شبکه‌های عصبی، یکی از بهترین و قوی‌ترین کتابخانه‌ها tensorflow می‌باشد. با استفاده از این کتابخانه و مجموعه دادگان مدل مورد نظر را پیاده‌سازی کردیم. رویکرد این پیاده‌سازی بدین صورت بود که ابتدا در هر تصویر سعی بر آن است که چشم فرد را در تصویر تشخیص دهیم. سپس از روی تصویر جداشده‌ی چشم به این مورد پی ببریم که چشم فرد باز است یا بسته. در نهایت با بررسی وضعیت چشم در هر تصویر مشخص کنیم که آیا این فرد خواب‌آلود است یا خیر. برای این مهم شبکه‌های عصبی پیچشی از قدرت بالایی در

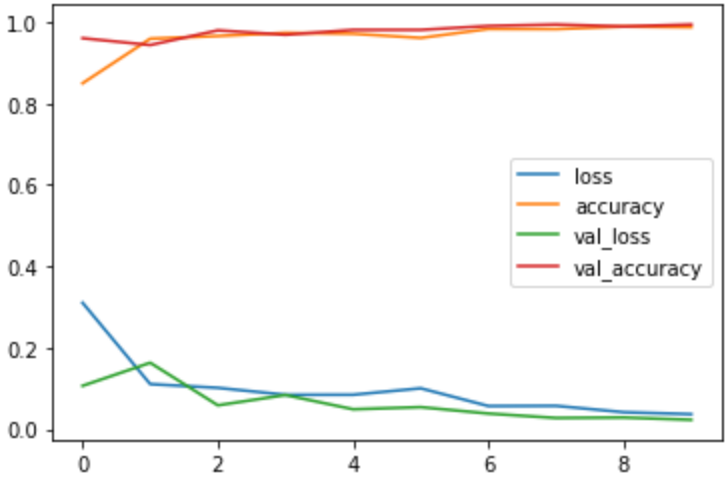
تحلیل تصاویر برخوردار هستند. بعد از بسیاری آزمون و خطا در به نتیجه رسیدن ساختار شبکه‌ی عصبی، به شبکه‌ی زیر رسیدیم که برای ما در نهایت نتایج قابل قبولی را کسب کرد و مدل نهایی ما در این پیاده‌سازی شد. ساختار مدل اولیه که ما طراحی و پیاده‌سازی کرده‌ایم به صورت زیر می‌باشد:

```
tf.keras.Sequential([
tf.keras.layers.Resizing(256,256),
tf.keras.layers.Rescaling(1./255),
tf.keras.layers.RandomFlip(mode="horizontal"),
tf.keras.layers.RandomRotation(0.2,
fill_mode='reflect',interpolation='bilinear'),
tf.keras.layers.Conv2D(32, 3, activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Conv2D(32, 3, activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Conv2D(32, 3, activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(2,activation='softmax')
])
```

با استفاده از ساختار بالا و با الگوریتم بهینه‌سازی Adam، توانستیم در زمان معقولی بر روی مجموعه‌ی دادگان خود مدل را آموزش داده و به مدل نهایی برسیم. تمامی پیاده‌سازی‌ها به صورت عمومی برای مشاهده قابل دسترسی هستند و از لینک می‌توانید جزییات پیاده‌سازی را مشاهده کنید.

نتایج:

در ابتدا روند تغییرات دقت یادگیری مدل در مراحل آموزش را مشاهده می‌کنید.



همانطور که در نمودار فوق مشاهده می‌کنید، دقت پیش‌بینی مدل بر روی مجموعه‌ی سنجش (validation) به مرور بهتر شده و مدل ما توانسته است این مجموعه دادگان را یاد بگیرد.

همچنین روند تغییرات دقت تشخیص را نیز در جدول زیر می‌توانید مشاهده کنید:

```
Epoch 1/10
264s 3s/step - loss: 0.3107 - accuracy: 0.8496 - val_loss: 0.1067 -
val_accuracy: 0.9603
Epoch 2/10
246s 2s/step - loss: 0.1109 - accuracy: 0.9592 - val_loss: 0.1633 -
val_accuracy: 0.9435
Epoch 3/10
247s 2s/step - loss: 0.1017 - accuracy: 0.9657 - val_loss: 0.0588 -
val_accuracy: 0.9796
Epoch 4/10
248s 2s/step - loss: 0.0847 - accuracy: 0.9734 - val_loss: 0.0841 -
val_accuracy: 0.9688
Epoch 5/10
247s 2s/step - loss: 0.0848 - accuracy: 0.9712 - val_loss: 0.0491 -
val_accuracy: 0.9808
Epoch 6/10
247s 2s/step - loss: 0.1006 - accuracy: 0.9610 - val_loss: 0.0543 -
val_accuracy: 0.9808
```

```
Epoch 7/10
247s 2s/step - loss: 0.0571 - accuracy: 0.9830 - val_loss: 0.0386 -
val_accuracy: 0.9904
Epoch 8/10
248s 2s/step - loss: 0.0575 - accuracy: 0.9827 - val_loss: 0.0277 -
val_accuracy: 0.9940
Epoch 9/10
255s 3s/step - loss: 0.0419 - accuracy: 0.9889 - val_loss: 0.0288 -
val_accuracy: 0.9892
Epoch 10/10
252s 2s/step - loss: 0.0371 - accuracy: 0.9873 - val_loss: 0.0232 -
val_accuracy: 0.9940
```

در نهایت مدل نهایی ما با دقت ۹۹ درصد بر روی مجموعه‌ی دادگان‌مان می‌تواند بسته یا باز بودن چشم را تشخیص دهد که این دقت، مقدار قابل توجهی است و برای پروژه‌ی ما کفایت می‌کند.

همچین در روند پیاده‌سازی این مدل چالش‌هایی نیز داشتیم که شرح برخی از آن‌ها اینگونه است. برای انجام این فاز که تمرکز اصلی آن بر روی بخش نرم‌افزاری پروژه و علی‌الخصوص بحث یادگیری ماشین بود، چالش‌های متعددی داشتیم که به شرح برخی از آن‌ها در این بخش می‌پردازیم:

1. نخستین چالش این بود که از حالت دهان در یادگیری مدل نهایی استفاده کنیم یا خیر. برای تشخیص این مورد باید در درجه اول به پیچیدگی مدل خود می‌افزودیم که سعی ما بر این است که تا جای امکان از این مورد جلوگیری کنیم. زیرا مدل هر چه پیچیده‌تر شود، منابع محاسباتی برای یادگیری آن و همچنین سرعت آن در پاسخگویی به یک ورودی بیشتر می‌شود که این مورد نهایی با توجه به قالب پروژه که ذاتا یک پروژه real-time محسوب می‌شود منافات دارد. همچنین با افزودن تشخیص حالت دهان که در حال خمیازه است یا خیر، دقت محاسباتمان به میزان قابل توجهی پایین می‌آمد، در حد ۱۰ درصد که این مورد نیز توجیه مناسبی نداشت. به همین علت در نهایت تصمیم بر آن شد که در مدل نهایی فقط از حالت چشم بهره ببریم و با استفاده از حالت چشم به تشخیص خواب‌آلوده بودن یا نبودن راننده بپردازیم.
2. چالش بعدی که از چالش‌های ذاتی پروژه‌های یادگیری ماشین است تنظیم هایپرپارامترها و توابع بهینه‌ساز به گونه‌ی مناسب بود تا بتوانیم دقت مدل را در حد

قابل توجهی به میزان بالا حفظ کنیم. برای این مورد نیز با جست‌وجوی گسترده در گوگل و آزمودن مقادیر مختلف به پارامترهای نهایی دست یافتیم که در مدل نهایی ارائه شده موجود می‌باشند.

چالش‌های نام برده بارزترین چالش‌هایی بودند که با آن‌ها روبرو شدیم. با توجه به چالش‌های نام‌برده برخی محدودیت‌های سخت‌افزاری ما را بر آن داشت تا به پیاده‌سازی مدل دیگری نیز برای این پروژه روی بیاوریم. مدل دومی که پیاده‌سازی گردید به شرح زیر است:

```
face = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_alt.xml')

leye = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_lefteye_2splits.xml')

reye = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_righteye_2splits.xml')
```

در این مدل که کدهای کامل آن به ضمیمه این مستند آمده است، روش ارزیابی به این گونه است که در ابتدا سعی می‌کنیم در تصویر، مکان چشم را تشخیص دهیم. سپس بعد از تشخیص چشم به عنوان مهم‌ترین عضو چهره برای شناسایی حالت خواب‌آلودگی یا عدم آن می‌توانیم به بسته یا باز بودن چشم بپردازیم. برای مثال تشخیص بسته یا باز بودن چشم راست را می‌توانیم به این صورت بررسی کنیم:

```
for (x,y,w,h) in right_eye:
    r_eye=frame[y:y+h,x:x+w]
    count=count+1
    r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
    r_eye = cv2.resize(r_eye,(24,24))
    r_eye= r_eye/255
    r_eye= r_eye.reshape(24,24,-1)
    r_eye = np.expand_dims(r_eye,axis=0)
    predict_x = model.predict(r_eye)
```

```
rpred = np.argmax(predict_x,axis=1)
if(rpred[0]==1):
   lbl='Open'
if(rpred[0]==0):
   lbl='Closed'
break
```

بدین ترتیب می‌توانیم با بررسی حالت چشم هر فرد به تشخیص این مورد بپردازیم که آیا فرد خواب‌آلوده است یا خیر.

گام بعدی، گام نهایی پروژه می‌باشد به این صورت که با در دست داشتن مدل آموزش‌دیده و یک سخت‌افزار مناسب، مدل مد نظر را روی آن پیاده‌سازی می‌کنیم. همچنین ورودی این سخت‌افزار به نحوی که در فاز بعدی تصمیم گرفته می‌شود تصاویر راننده در حال رانندگی خواهد بود که در صورتی که مدل ما تشخیصش بر آن بود که چشمان راننده به مدت طولانی بسته است، با یک چراغ یا صدای بوق این مورد را به راننده اخطار دهد تا از خطرات احتمالی این مورد جلوگیری کند.

فاز دو - فاز سخت‌افزاری

۱. تهیه سخت‌افزار مد نظر برای پیاده‌سازی:

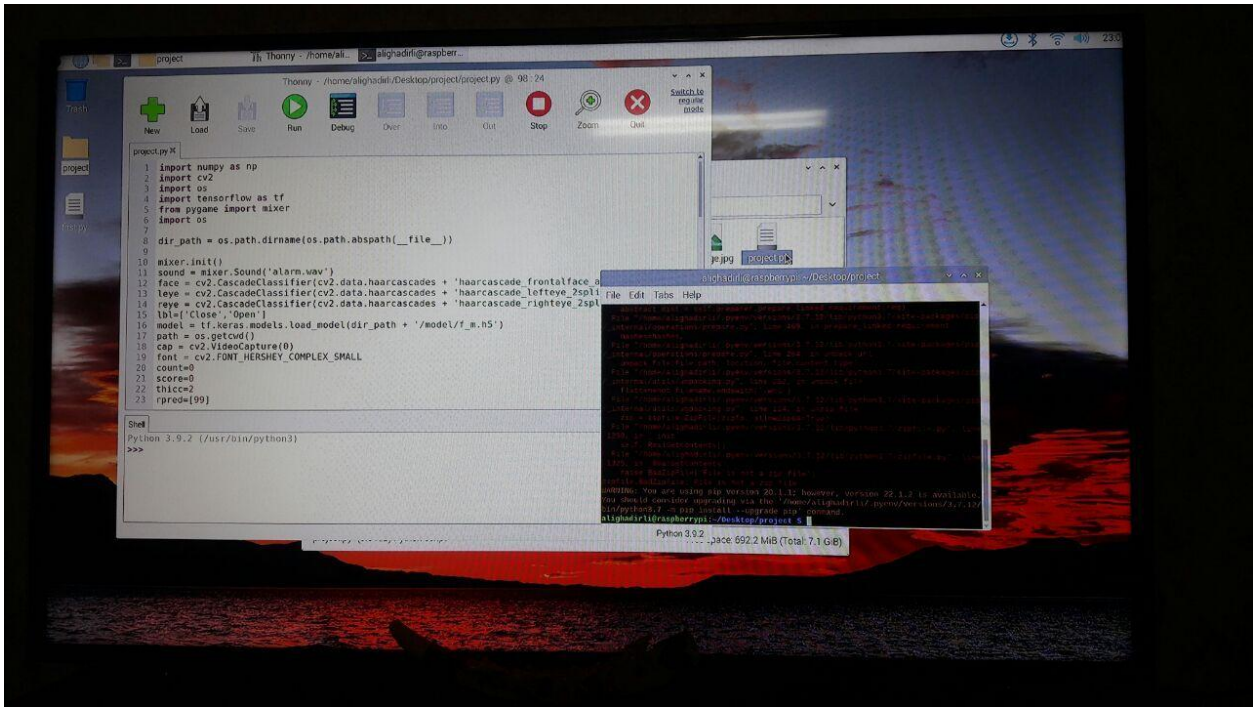
در این فاز در اولین گام نیاز است تا سخت‌افزار مورد نیاز برای پیاده‌سازی پروژه فراهم گردد. به این منظور سخت‌افزارهای موجود در آزمایشگاه بررسی شد و از میان آن‌ها مناسب‌ترین مورد برای پیاده‌سازی پروژه ما آماده گردید. بردهای سخت‌افزاری متنوعی برای چنین پروژه‌های وجود دارد. برای مثال می‌توان به برد رزبری و برد آردوئینو اشاره کرد که در میان سایر بردها به نسبت پرکاربردتر و فراگیرتر می‌باشند. این ویژگی نیز به دلیل کیفیت بالای پیاده‌سازی آن‌ها و تعبیه کردن امکانات متعدد و منابع محاسباتی کافی برای این بردها می‌باشند. در نهایت برد رزبری موجود در آزمایشگاه برای این پروژه انتخاب گردید. به منظور دریافت این برد نیز از طریق فرم رسمی درس و با کمک آقای فصحتی این برد و سایر امکانات سخت‌افزاری منتسب به این برد از قبیل آداپتور و رابط‌های مورد نیاز به گروه ما داده شد.

۲. ادغام دو فاز نرم‌افزاری و سخت‌افزاری:

برای وصل کردن دوربین به برد رزبری از برنامه IP Webcam استفاده شد که با استفاده از آن می‌توانستیم ورودی دوربین را دوربین گوشی بگیریم در صورتی که هم برد و هم گوشی به یک wifi وصل می‌بودند. همچنین برای نصب tensorflow بر روی برد از این گیتهاب (https://github.com/PINTO0309/Tensorflow-bin) فایل‌های wheel دانلود و با استفاده از آن tf را نصب کنیم. با توجه با اینکه نسخه از پیش نصب شده پایتون در رزبری نسخه ۳.۹ می‌بود و طبق این issue) (https://github.com/PINTO0309/Tensorflow-bin/issues/46 در گیتهاب نسخه whl ای برای این نسخه پایتون هنوز وجود ندارد، اقدام به استفاده از pyenv ((https://github.com/pyenv/pyenv برای ایجاد یک نسخه دلخواه در virtual

environment پروژه کردیم. با این اوصاف و با توجه به معماری aarch برد منظور که تعداد whl های مربوط در این گیتهاب برای این معماری کم بود و فقط می‌توانستیم از پایتون نسخه ۳.۷ برای این منظور استفاده کنیم.

با این چالش‌ها، در نهایت امر نیز با ارور badZipFile در هنگام نصب نهایی مواجه می‌شدیم و برای حل این مشکل نیز از سرچ کردن در اینترنت استفاده کردیم که با توجه به معماری برد و نسخه پایتون مورد نظر، فقط به یک جواب (https://github.com/samjabrahams/tensorflow-on-raspberry-pi/issues/76) رسیدیم که راه‌حل مذکور نیز کار نمی‌کرد که در نهایت موفق به نصب tensorflow نشدیم.



توضیحات تکمیلی

شبکه‌های عصبی:

شبکه عصبی (Neural Network)، درونی‌ترین لایه علم شگفت انگیزِ هوش مصنوعی (Artificial Intelligence) است. علمی که دنیای امروز ما را با چند دهه قبل بسیار متفاوت کرده و تکنولوژی بخش اعظمی از پیشرفت خود را مدیون آن است. شبکه‌های عصبی با ساده‌تر کردن زندگی انسان‌ها در زمینه‌های مختلفی مثل علم پزشکی، اقتصاد، مهندسی و... تفاوت‌های زیادی نسبت به شیوه زندگی در چند دهه پیش ایجاد کرده‌اند. شبکه عصبی، بنای علم یادگیری عمیق (Deep Learning) و یادگیری عمیق هم خود، پایه و اساس یادگیری ماشین (Machine Learning) است. این مفاهیم با هم، علمِ هوش مصنوعی را تشکیل می‌دهند. هدف کلی این است که یک سری اطلاعات را طوری به یک ماشین یا همان کامپیوتر بدهیم، که برای او قابل درک باشد و بتواند از آن در راستای اهداف بشر استفاده کند.

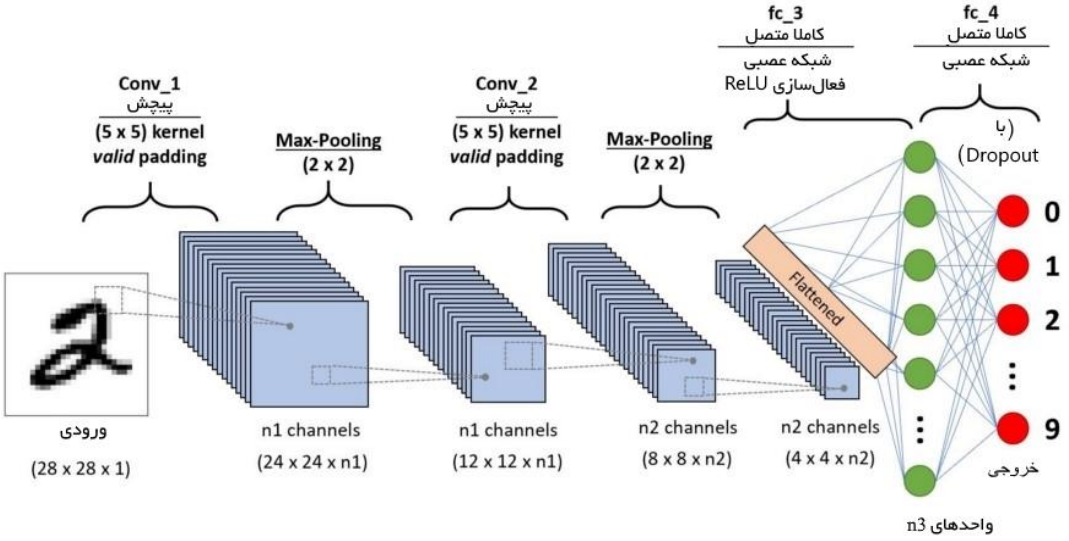
مفهوم شبکه عصبی مثل این است که بخواهیم به یک کودک یاد بدهیم که چگونه از بین اشکال مختلف، شکل دایره را تشخیص دهد. به او چندین عکس از دایره‌ها در ابعاد و رنگ‌های مختلف نشان می‌دهیم. پس از مدتی یاد می‌گیرد که دایره چیست و می‌تواند از میان همه تصاویری که به او نشان داده می‌شود، دایره‌ها را تشخیص دهد. این دقیقا همان کاری‌ست که به کمک شبکه‌های عصبی برای آموزش به یک ماشین انجام می‌دهیم؛ آموزش دادن به ماشین نهایتا باعث ایجاد هوش مصنوعی در آن می‌شود.

شبکه‌های عصبی داده‌ها را دریافت و در لایه‌های مخفی خود آن‌ها را تحلیل می‌کنند تا نهایتا یک خروجی ارائه بدهند. این داده‌ها می‌توانند گروهی از تصاویر، صداها، نوشته‌ها و... باشند که باید ترجمه و برای یک ماشین قابل درک بشوند. به کمک شبکه‌های عصبی، اطلاعات را طبقه‌بندی می‌کنیم؛ اطلاعات مختلف می‌توانند بر اساس شباهت به مثالی مشخص، گروه‌بندی شوند. آن‌ها حتی می‌توانند امکانات و داده‌های لازم برای تغذیه به یک الگوریتم دیگر را هم فراهم و طبقه‌بندی کنند.

شبکه‌های عصبی پیچشی:

«شبکه عصبی پیچشی» (Convolutional Neural Network | CNN ConvNet) یک الگوریتم یادگیری عمیق است که تصویر ورودی را دریافت می‌کند و به هر یک از

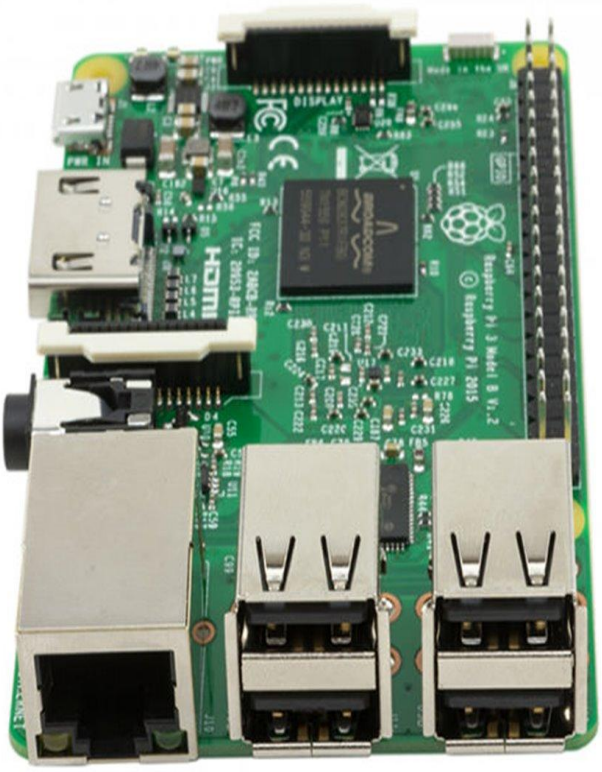
اشیا/جنبه‌های موجود در تصویر میزان اهمیت (وزن‌های قابل یادگیری و بایاس) تخصیص می‌دهد و قادر به متمایزسازی آن‌ها از یکدیگر است. در الگوریتم ConvNet در مقایسه با دیگر الگوریتم‌های دسته‌بندی به «پیش‌پردازش» (Pre Processing) کمتری نیاز است. در حالیکه فیلترهای روش‌های اولیه به صورت دستی مهندسی شده‌اند، شبکه عصبی پیچشی (ConvNets)، با آموزش دیدن به اندازه کافی، توانایی فراگیری این فیلترها/مشخصات را کسب می‌کند. معماری ConvNet مشابه با الگوی اتصال «نورون‌ها» (Neurons) در مغز انسان است و از سازمان‌دهی «قشر بصری» (Visual Cortex) در مغز الهام گرفته شده است. هر نورون به محرک‌ها تنها در منطقه محدودی از میدان بصری که تحت عنوان «میدان تاثیر» (Receptive Field) شناخته شده است پاسخ می‌دهد. یک مجموعه از این میدان‌ها برای پوشش دادن کل ناحیه بصری با یکدیگر هم‌پوشانی دارند.



برد رزبری‌پای:

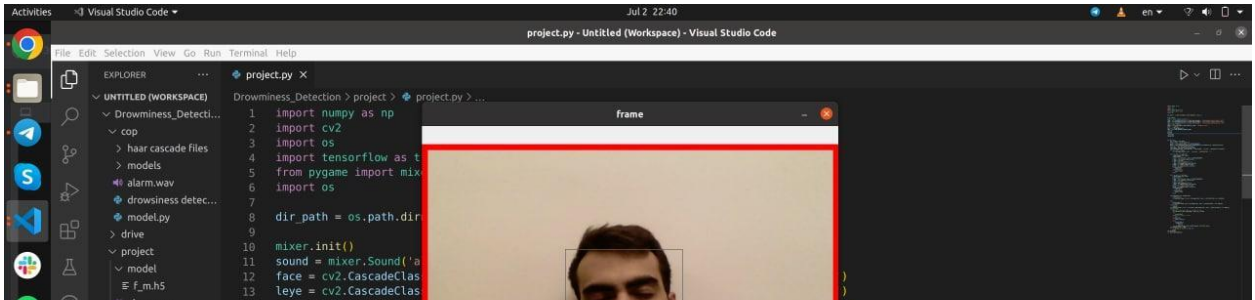
رزبری‌پای رایانه کوچکی است که از سال 2006 در حال توسعه است و قطعات آن روی یک مادربرد به اندازه کارت بانکی سوار شده و Raspbian را اجرا می‌کند که یک نسخه اختصاصی از سیستم عامل لینوکس است که اختصاصاً برای این رایانه طراحی شده است. رزبری پای کاربردهای محاسباتی ابتدایی اداری، بازی‌های سطح پایین، دسترسی به اینترنت و ایمیل، بازپخش ویدئو و بسیاری قابلیت‌های دیگر دارد که به طور معمول از یک رایانه در قرن بیست و یکم انتظار

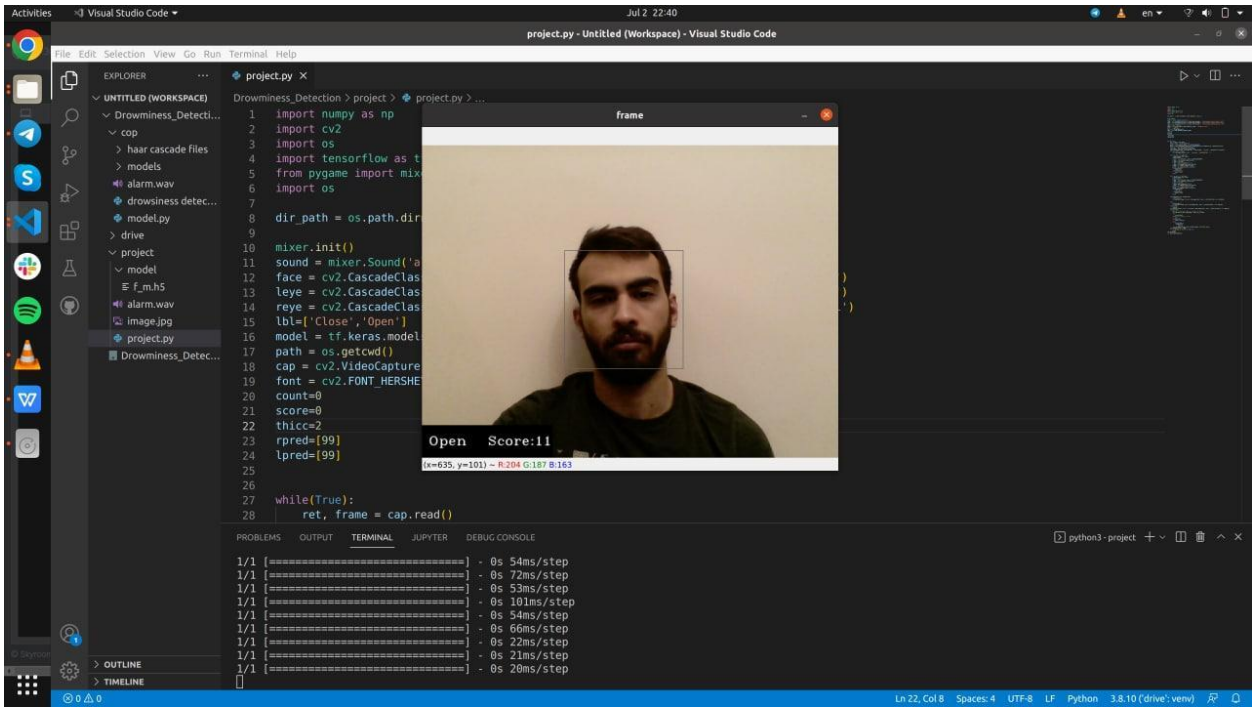
می‌شود. رزبری پای همه این امکانات را با تعداد بسیار کمی از قطعات از جمله یک پردازنده ARM و قیمت بسیار پایین عرضه می‌کند. با حذف کابل‌ها، فضای ذخیره‌سازی و کیس، هزینه ساخت این رایانه پایین نگه داشته شده است. البته کابل‌ها و فضای ذخیره‌سازی برای هر رایانه‌ای ضروری هستند و اگر بخواهید یک کیس داشته باشید، راهکارهای مختلفی وجود دارند که در بخش « گزینه‌های کیس» این راهنما می‌توانید مشاهده کنید.

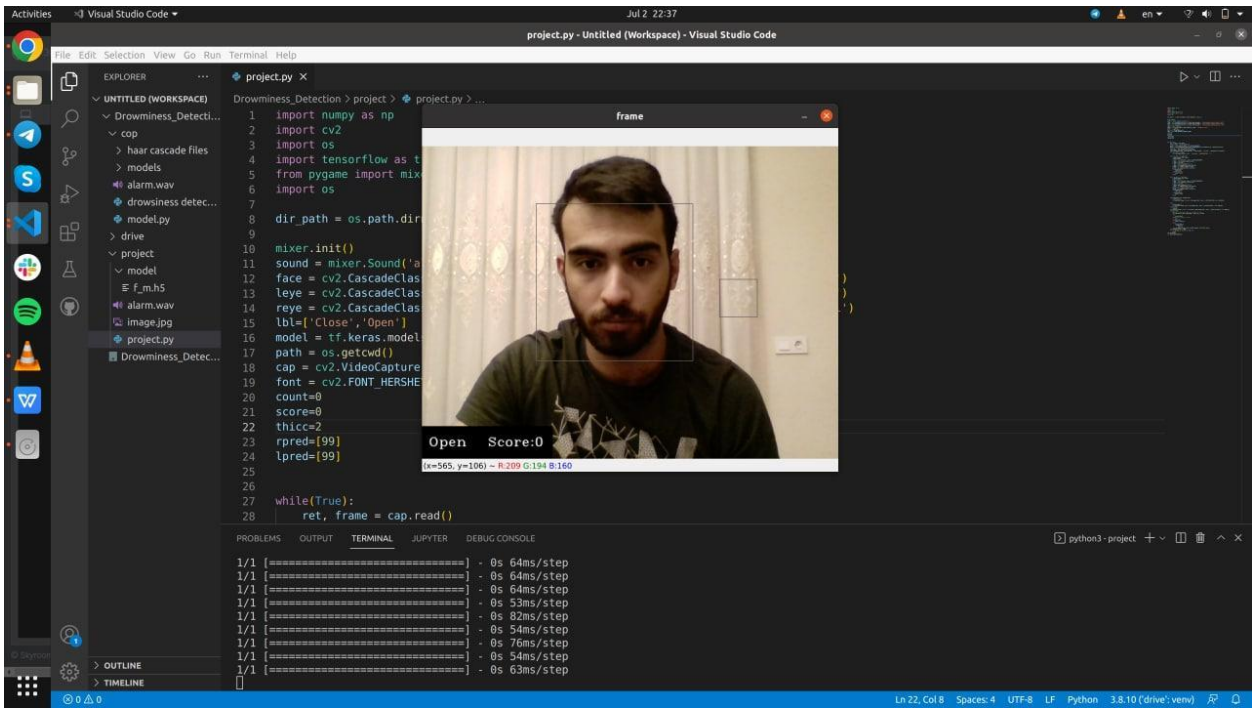


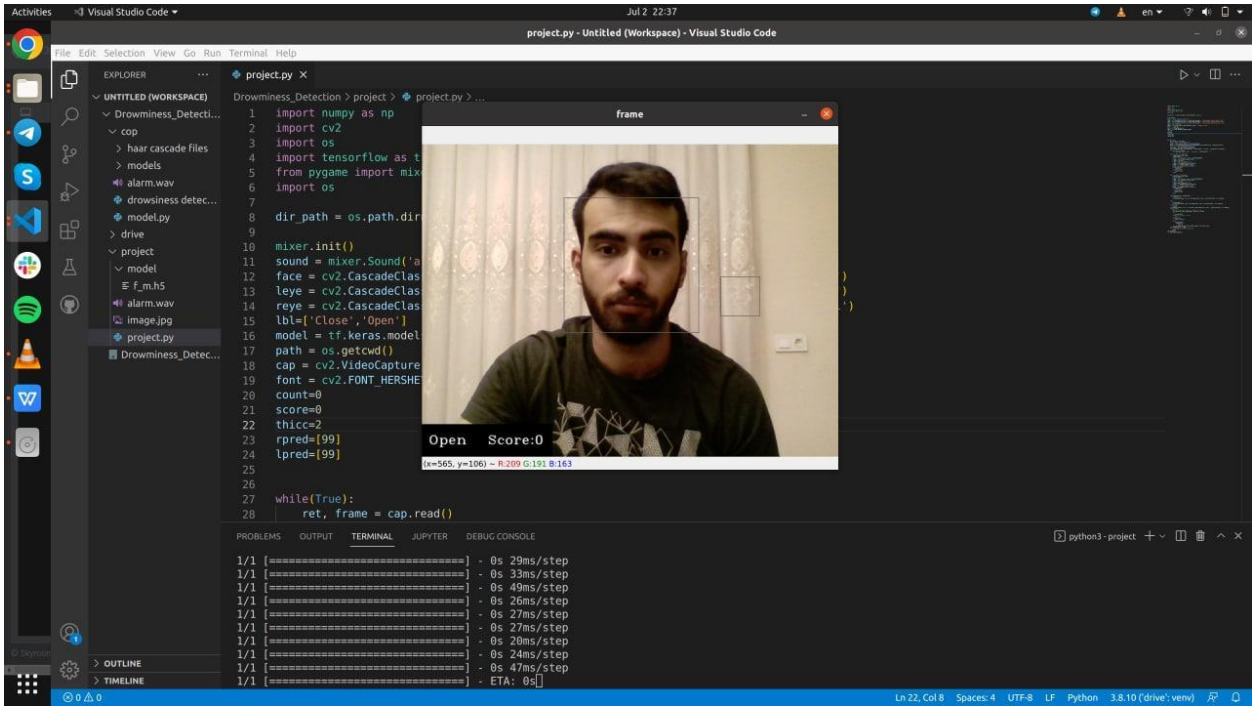
تصاویر اجرا و پیاده‌سازی

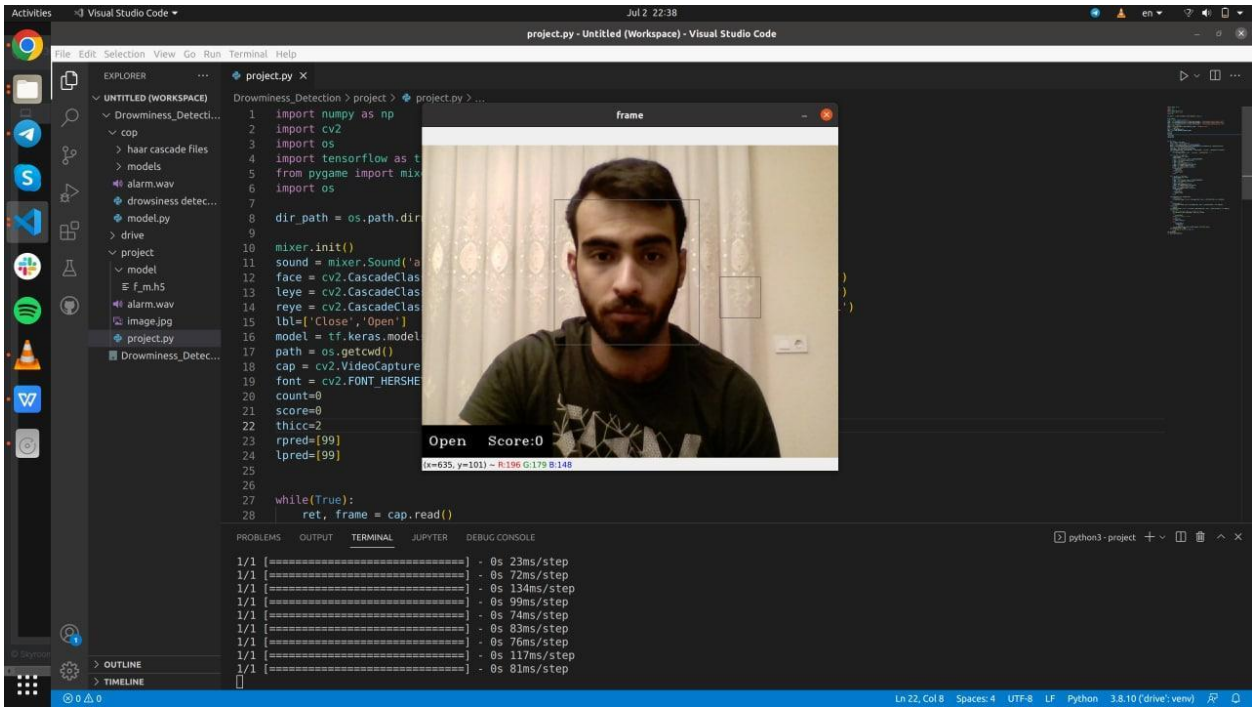
همانطور که در اجراهای زیر مشاهده می‌کنید سامانه طراحی شده قابلیت تشخیص حالت خواب‌آلود را از غیر خواب‌آلود دارد و زمانی که چشمان در حالت خواب‌آلود می‌روند سامانه به حالت اخطار می‌رود. کافیست در این حالت سخت‌افزار مد نظر یک بوق نصب شود که به مانند اینجا صدای بوق به نشانه‌ی هشدار درآید تا راننده را هشیار کند.

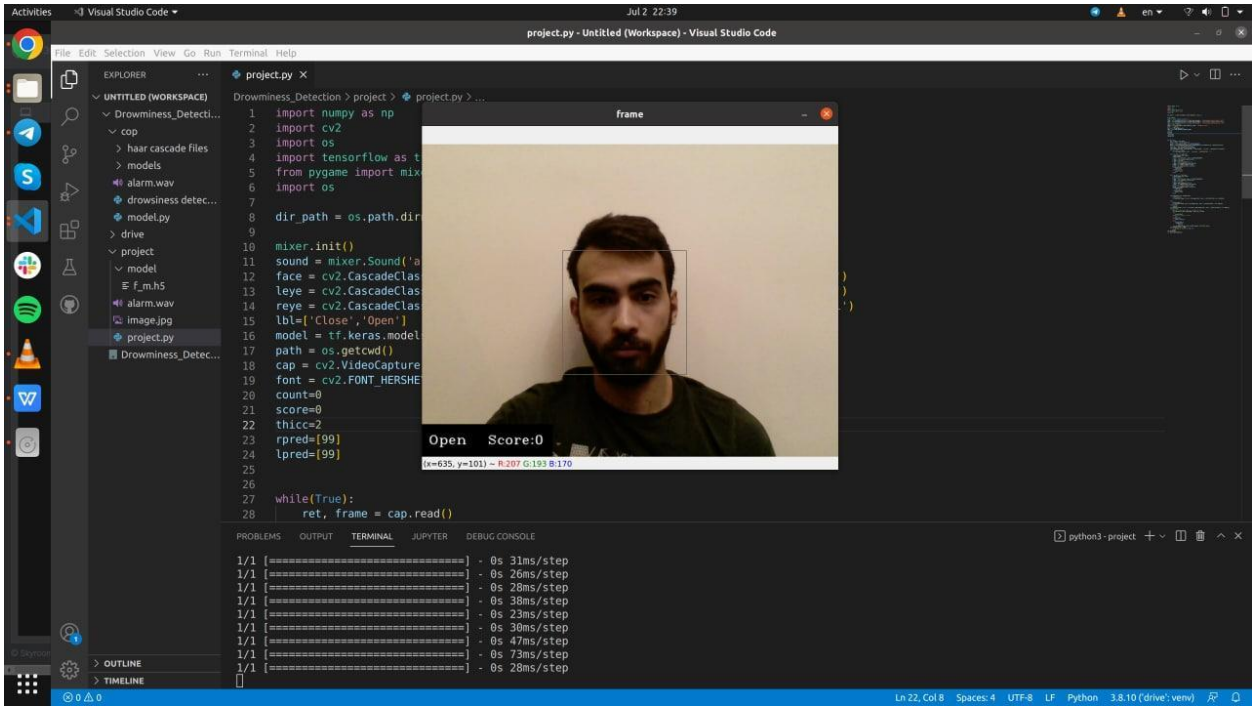


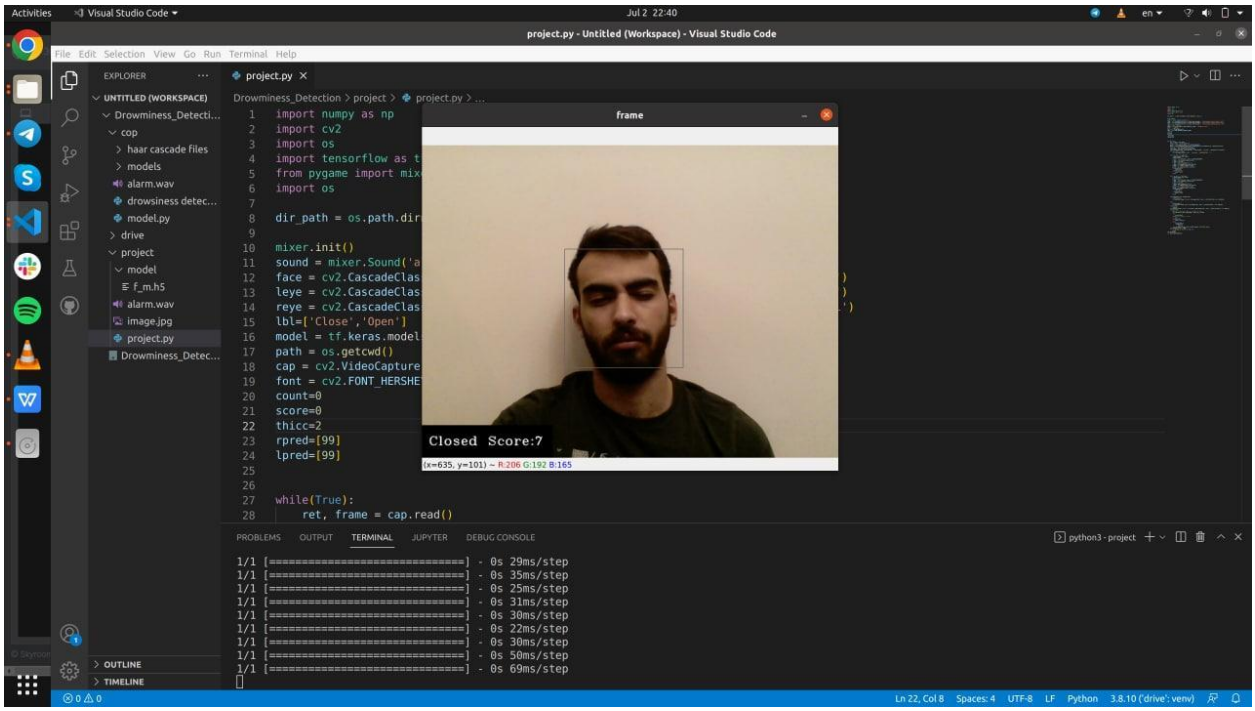

project.py - Untitled (Workspace) - Visual Studio Code
frame
Open Score:11
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step


project.py - Untitled (Workspace) - Visual Studio Code
frame
Open Score:0
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 76ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 63ms/step


project.py - Untitled (Workspace) - Visual Studio Code
frame
Open Score:0
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - ETA: 0s


project.py - Untitled (Workspace) - Visual Studio Code
frame
Open Score:0
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 99ms/step
1/1 [==============================] - 0s 74ms/step
1/1 [==============================] - 0s 83ms/step
1/1 [==============================] - 0s 76ms/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 81ms/step


project.py - Untitled (Workspace) - Visual Studio Code
frame
Open Score:0
project.py
Drowminess_Detection > project > project.py
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 73ms/step
1/1 [==============================] - 0s 28ms/step


project.py - Untitled (Workspace) - Visual Studio Code
frame
Closed Score:7
project.py
Drowminess_Detection > project > project.py
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 69ms/step

جمع‌بندی

با انجام این پروژه قصد داریم در هر دو جنبه‌ی نرم‌افزاری و سخت‌افزاری یک پروژه‌ی مهندسی کامپیوتر در حوزه‌ی سخت‌افزار را پیاده‌سازی کنیم و از صفر تا صد این پروژه را تحلیل، طراحی و پیاده‌سازی کنیم.

در این مستند نیز به ترتیب از ابتدا سعی بر آن داشتیم تا اهمیت پروژه را هر چه بهتر و عمیق‌تر نشان دهیم. سپس به بررسی روال زمانی پیاده‌سازی پروژه پرداختیم. در گام بعدی پروژه را فازبندی کرده و مراحل اجرای هر فاز را به تفکیک و تفضیل بیان کردیم. در نهایت نیز کارهای انجام شده در هر گام و چالش‌های به وجود آمده را توضیح دادیم. در نهایت نیز تصاویر و مستندات را ضمیمه کرده و در بخش توضیحات تکمیلی هر موردی که نیاز به توضیح بیشتر برای فهم بهتر خواننده داشت را نیز افزودیم. امید است این پژوهش برای خواننده مفید بوده باشد.

مراجع

- https://github.com/PINTO0309/Tensorflow-bin
- https://github.com/pyenv/pyenv
- https://github.com/samjabrahams/tensorflow-on-raspberry-pi/issues/76
- https://blog.faradars.org/convolutional-neural-networks/
- https://faranesh.com/blog/what-is-neural-network
- https://blog.faradars.org/%D8%B1%D8%B2%D8%A8%D8%B1%DB%8C-%D9%BE%D8%A7%DB%8C-%DA%86%DB%8C%D8%B3%D8%AA/
- http://cv.cs.nthu.edu.tw/php/callforpaper/datasets/DDD/
- https://sites.google.com/view/utarldd/home
- https://www.kaggle.com/code/adinishad/driver-drowsiness-using-keras/data