פרויקט גמר 5 יחידות לימוד התמחות – למידת מכונה

Deep Learning

נושא הפרויקט:

זיהוי אקדחים בתמונות וסרטונים

Handgun detection

בית הספר: מקיף י״א ראשונים

שם התלמיד: עידן משה

תעודת זהות: 325944676

שם המנחה: דינה קראוס

שם החלופה: למידת מכונה

תאריך הגשה: 16.6.2022

תוכן עניינים

תוכן עניינים

מבוא

תחום ה"למידת מכונה" (Machine Learning) ככלל וה- "למידה העמוקה" (Deep Learning) בפרט נחשב לחוד החנית של חקר מדעי המחשב בעולם כולו, זהו נושא ההתמחות שלנו במהלך לימודי מדעי המחשב בתיכון וכן זהו נושא פרוייקטי הגמר שלנו.
תחום הלמידת מכונה עוסק בלימוד מחשב לפתור בעיות מורכבות בתהליך הדומה ללמידה אנושית – דרך ניסוי וטעיה. המכונה היא בעלת יכולת חישוב וזיכרון גבוהים בהרבה מזו של בני אדם, ולכן לעתים מודל טוב יכול לפתור בעיות שאפילו בני אדם מתקשים לפתור.
בתחום של למידת מכונה אנו מנסים לחקות בתוך המחשב את תהליך הלמידה של בני אדם, ולכן גם צורת החשיבה והניתוח של בני האדם מועתקת בקירוב לתוך המחשב. אצל בני האדם, המוח מורכב מכמות רבה מאוד של נוירונים, שכל אחד מהם מקבל מידע מנוירונים אחרים ומעביר מידע מעובד לנוירונים הבאים. בלמידת מכונה, בדומה למוח האנושי, יש נוירונים המקבלים מידע מצד אחד ומעבירים אותו, לאחר עיבוד פשוט, לנוירונים הבאים.

לנושא הפרויקט שלי אני בחרתי בנושא של זיהוי וסימון אקדחים בתוך תמונות וסרטונים, Object Detection. כלומר המודל מתאמן על תמונות שבהן יש אקדחים מסומנים, ולאחר מכן יכול לזהות ולסמן היכן יש אקדחים בתמונות שלא ראה לפני כן.
בחרתי בנושא זה מכיוון שנשיאת נשק למקומות אסורים וכן שימוש בהם היא בעיה גדולה שפתרונה כרוך בהצלת חיי אדם. לעיתים מאבטחים שצופים במצלמות האבטחה מאבדים ריכוז או נרדמים, דבר שיכול לסכן את בטחון באנשים. לכן הפתרון הוא שימוש בתוכנה, שלא כמו בני אדם, לא מאבדת ריכוז ולא צריכה לישון. כלומר התוכנה הזה יכול לקבל סרטונים ממצלמות אבטחה ולזהות אקדחים. כך יהיה ניתן להפחית את השימוש בכוח האדם, ובו זמנית לשפר את רמת ההבטחה במקום.
בעתיד, פיתוח התוכנה יכול לכלול גם זיהוי סכינים וכלי נשק גדולים יותר, תוך כדי שימוש בקלט ממספר חיישנים שונים כמו מצלמות רגילות, מצלמות Xray, גלאי מתכות וכו'.

בסקירת המצב הקיים בשוק מצאתי מספר חברות (https://zeroeyes.com/)
(https://www.omnilert.com/) הנותנות שירות של גילוי כלי נשק במצלמות אבטחה, ורובן מבוססות על A.I. עם זאת, החברות מבקשות תשלום על השירות, ולרוב לא מאפשרות אימון, בחינה ושימוש פרטי במודל.

במהלך כתיבת הפרויקט נתקלתי במספר אתגרים משמעותיים. האתגר המשמעותי ביותר היה שהיה לי ידע מצומצם ביותר בנושא הלמידת מכונה ככלל וObject Detection בפרט, לכן היה עלי להתנסות בהרבה דרכים שבהם אני יכול להשלים את מטרת הפרויקט, כמו גם לקרוא הרבה מאמרים באנגלית שמסבירים על הנושא.
אתגר נוסף שאיתו התמודדתי במהלך הפרויקט הוא מחסור בכמות הנתונים. לנושא המחקר שבחרתי לא היו הרבה מקורות נתונים שפתוחים לשימוש לקהל הרחב, ורוב החברות מעדיפות לשמור את מאגרי הנתונים שלהן לעצמן. לכן, בחרתי להשתמש בAPI שנקרא Tensorflow Object Detection API. דבר זה הוא API (Application Programming Interface) כלומר תוכנה שנוצרה לתת מענה לבעיות של מתכנתים אחרים דרך קוד.

הסבר קצר על Tensorflow Object Detection API

במהלך תהליך המחקר המקדים לפרויקט הבנתי שאני אצטרך לבצע Object Detection, כלומר זיהוי אובייקטים בתמונות (סרטונים הם רצף של תמונות).
בהמשך המחקר התברר שObject Detection הוא אחד מהנושאים המתקדמים והמסובכים ביותר בלמידת מכונה וComputer Vision, ולרוב כדי לאמן מודל מוצלח צריך כמות נתונים של עשרות אלפי תמונות וידע של שנים בנושא של למידת מכונה, שני דברים שאין לנו כתלמידים במסגרת ההתמחות בתיכון.
כדי לאפשר לאנשים כמוני גם להתנסות בעולם של Object Detection, היוצרים של Tensorflow פיתחו framework שבנוי על Tensorflow עצמו, בעל קוד פתוח וקריא לכולם ולו הם קראו Tensorflow Object Detection API.

בעזרת הכלי הזה ניתן לבנות, לאמן ולהשתמש במודל המבצע Object Detection גם על מבני נתונים קטנים ובלי ידע נרחב בנושא.
התהליך שבו הTensorflow Object Detection API משתמש כדי לגרום גם לdataset קטן ליצור מודל הוא בעזרת שיטה שנקראת fine-tuning. כלומר לקחת מודל שאומן על dataset אחר לגמרי בעל מספר רב של אובייקטים ותמונות (לפעמים גם אלפי אובייקטים ומיליוני תמונות), וכבר מומחה בלזהות ולהפריד אובייקטים מתמונות, ורק לאמן אותו בצורה מצומצמת כדי שידע לזהות אובייקטים לפי הdataset החדש.

על דרכי השימוש בכלי והכנת הנתונים באופן המתאים לשימוש בו אפרט בהמשך בחלקים המתאימים.

ארכיטקטורה של הפרויקט/מבנה

שלב איסוף הכנה וניתוח הנתונים

תיאור מבנה הנתונים

ה- dataset שמצאתי מגיע במקור ממחקר שנעשה באוניברסיטת גרנדה
(https://sci2s.ugr.es/weapons-detection#RP) שבו הם ניסו לפתור בעיה דומה לשלי.

את הdataset הורדתי ישירות מהקישורים הנ"ל:
images:
https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/WeaponsDetection/BasesDeDatos/WeaponS.zip

labels:
https://sci2s.ugr.es/sites/default/files/files/TematicWebSites/WeaponsDetection/BasesDeDatos/WeaponS_bbox.zip

מקישורים אלו קיבלתי את הקבצים:
WeaponS.zip – קובץ המכיל את התמונות
WeaponS_bbox.zip – קובץ המכיל את התוויות

תיאור וניתוח הנתונים הגולמיים

הdataset הגולמי מגיע בשני קבצי zip שונים, אחד לתמונות ואחד לתוויות. לכן הייתי צריך לשלב אותם לקובץ אחד ולו קראתי Dataset.zip

כדי לעשות זאת, קודם כל יצרתי תיקייה ראשית ששמה Dataset ובה שני תיקיות:
images – תיקייה לתמונות
annotations – תיקייה לתוויות
אל התיקייה images העתקתי את כל התמונות מהקובץ WeaponS.zip
אל התיקייה annotations העתקתי את כל התמונות מהקובץ WeaponS_bbox.zip

לאחר מכן כיווצתי ל-zip את התיקייה Dataset וזוהי התיקייה שהתוכנה תקבל לאימון המודל.

לכל תמונה יש קובץ תוויות משלה, בפורמט xml.

מבנה תוכן קובץ התוויות:

תוכן קובץ התוויות רשום בפורמט הנתונים Pascal VOC

הפורמט הזה הוא דרך לכתוב את התוויות של Object Detection

הארכיטקטורה של קובץ תווית (ללא התוויות הלא נחוצות):

```
annotation
├── filename
├── size
│   ├── width
│   └── height
├── object
│   ├── name
│   └── bndbox
│       ├── xmin
│       ├── ymin
│       ├── xmax
│       └── ymax
└── object
    ├── name
    └── bndbox
        ├── xmin
        ├── ymin
        ├── xmax
        └── ymax
```

דוגמא לקובץ תווית:


```xml
<annotation>
  <folder>Definitiva</folder>
  <filename>armas (1)</filename>
  <path>C:\Users\Rob\Desktop\Definitiva\armas (1).jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>240</width>
    <height>145</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>pistol</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>3</xmin>
      <ymin>1</ymin>
      <xmax>128</xmax>
      <ymax>100</ymax>
    </bndbox>
  </object>
  <object>
    <name>pistol</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>123</xmin>
      <ymin>47</ymin>
      <xmax>238</xmax>
      <ymax>145</ymax>
    </bndbox>
  </object>
</annotation>
```

תהליך הכנת הDataset לאימון

לפני הכל, הdataset מאוחסן בקובץ zip, ולכן תחילה עלינו לחלץ אותו לתיקייה ריקה.

לאחר מכן, עלינו לוודא שהdataset תקין, כלומר שניתן לקרוא את קובצי התוויות (xml) ולחלץ מהם את כל המידע הנחוץ. בנוסף, צריך לבדוק שהמידע עצמו תקין: גודל התמונה בתווית תואם את גודל התמונה האמיתית, הגבלות הקופסא לא גדולות מגודל התמונה וכו'...

אחרי שבדקנו שהנתונים תקינים צריך לחלק אותם ל train ו-evaluation
כלומר נתונים שעליהם יתאמן המודל, ונתונים שעליהם המודל ייבדק

לבסוף צריך לקודד את הנתונים לפורמט שנשלח לTensorflow Object Detection API. כמו שהזכרתי בתחילת הספר, כדי להשתמש בTensorflow Object Detection API הנתונים צריכים להיות מאורגנים בפורמט מיוחד. פורמט זה נקרא TFRecord, ניתן לקרוא עליו קצת בקישור (https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564)

בקיצור, הפורמט TFRecord הוא פורמט בינארי, כלומר המידע שמור בצורה בינארית, שמייעל זמן ריצה ומקום בדיסק.

כדי שהTensorflow Object Detection API יוכל לקרוא את הנתונים, הם צריכים להיות רשומים בתוך קובץ הTFRecord במבנה מסוים, עבור כל תמונה המידע מקודד כך:

פיסת המידע	סוג המידע	ערך
image/height	int64	גובה התמונה
image/width	int64	רוחב התמונה
image/filename	bytes	שם קובץ התמונה
image/source_id	bytes	שם קובץ התמונה
image/encoded	bytes	התמונה עצמה מקודדת
image/format	bytes	שם פורמט התמונה
image/object/bbox/xmin	float_list	רשימת הגבולות השמאליים של התוויות
image/object/bbox/xmax	float_list	רשימת הגבולות הימניים של התוויות
image/object/bbox/ymin	float_list	רשימת הגבולות התחתונים של התוויות
image/object/bbox/ymax	float_list	רשימת הגבולות העליונים של התוויות
image/object/class/text	bytes_list	רשימת שמות התוויות
image/object/class/label	int64_list	רשימת הid של התוויות

שיטת נרמול הנתונים נעשית בזמן הריצה על ידי הTensorflow Object Detection API, מכיוון שהקוד שלו חופשי לכולם מצאתי את שיטת נרמול הנתונים והיא זו:

תחילה מחסירים מכל ערוץ צבע את הערכים הבאים:

אדום – 123.68

ירוק – 116.779

כחול – 103.939

זוהי שיטת נרמול שבה משתמשים במודלים הבנויים על VGG

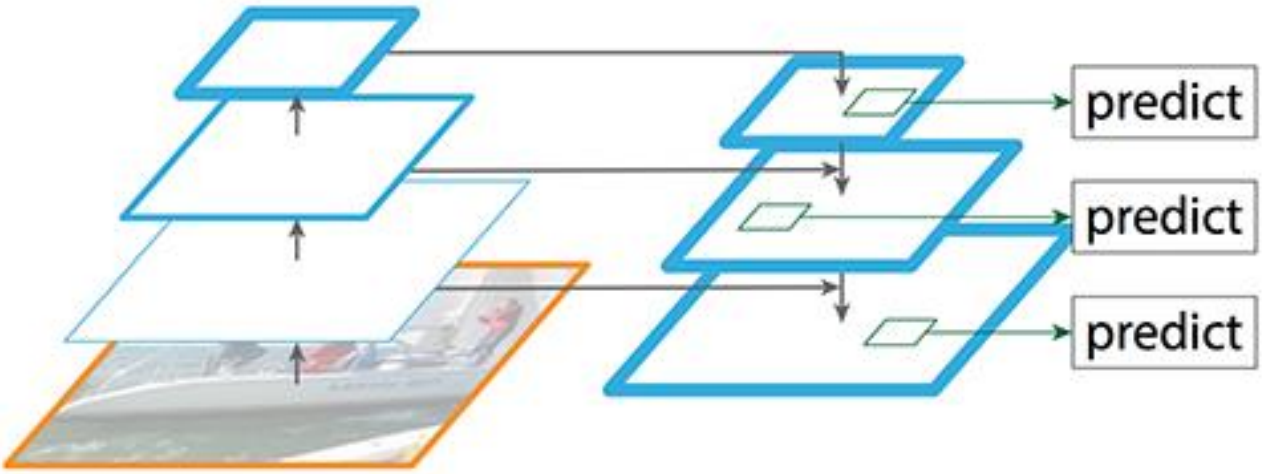
לאחר מכן משנים את הגודל של התמונות לגודל של 640 על 640 פיקסלים

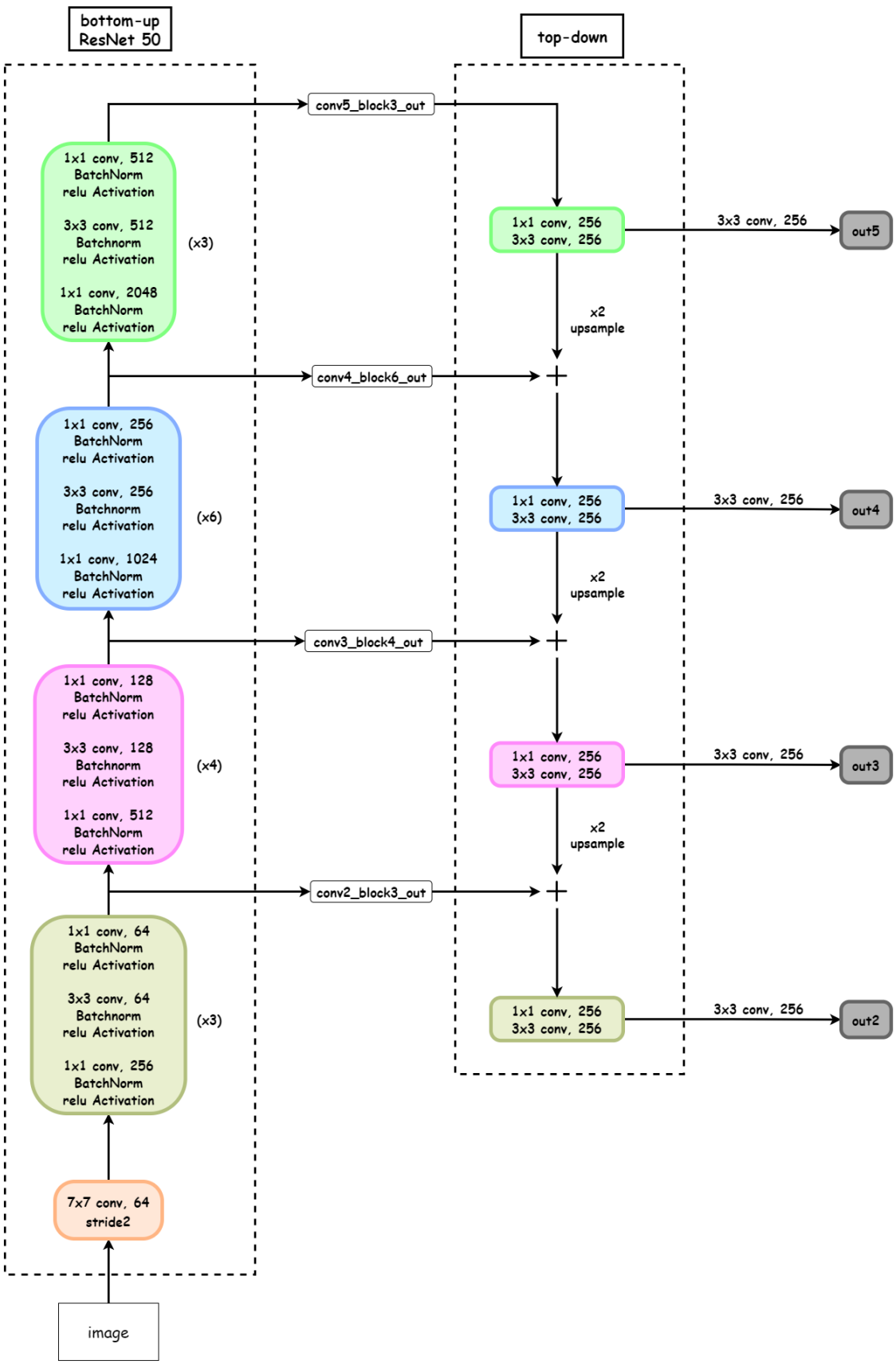
שלב בנייה ואימון המודל

תיאור גרפי של המודל עליו בוצע האימון

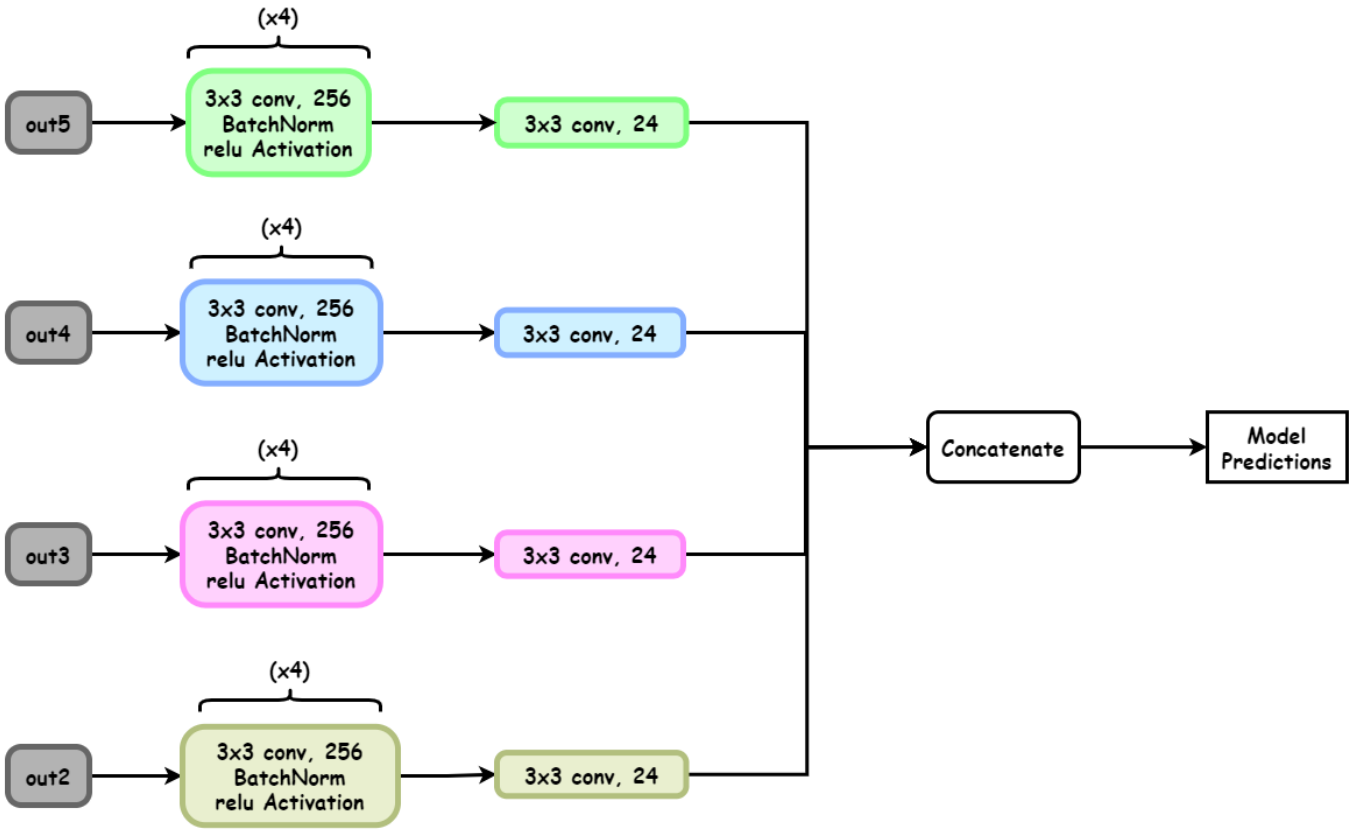
בתוך הTensorflow Object Detection API יש אפשרות לבחור בין מודלים רבים,
אני בחרתי במודל הנקרא *SSD ResNet50 V1 FPN 640x640 (RetinaNet50)* הבנוי כך:

בתחילת המודל יש את ה feature extractor כלומר חלק במודל האחראי למפות בקירוב אובייקטים ואת מיקומם בתמונה. זהו החלק השמאלי בתמונות הבאות.
חלק זה בנוי במבנה של Bottom Up המבוסס על ה ResNet50, שזהו בסיס נפוץ לחלק ה feature extractor, ואז יש את חלק הTop Down. שלוקח את התוצאות מחלק הBottom Up ומוציא את מפת האובייקטים הסופית.




bottom-up
ResNet 50
top-down
conv5_block3_out
1x1 conv, 512
BatchNorm
relu Activation
3x3 conv, 512
Batchnorm
relu Activation
1x1 conv, 2048
BatchNorm
relu Activation
(x3)
1x1 conv, 256
3x3 conv, 256
3x3 conv, 256
out5
x2
upsample
conv4_block6_out
1x1 conv, 256
BatchNorm
relu Activation
3x3 conv, 256
Batchnorm
relu Activation
1x1 conv, 1024
BatchNorm
relu Activation
(x6)
1x1 conv, 256
3x3 conv, 256
3x3 conv, 256
out4
x2
upsample
conv3_block4_out
1x1 conv, 128
BatchNorm
relu Activation
3x3 conv, 128
Batchnorm
relu Activation
1x1 conv, 512
BatchNorm
relu Activation
(x4)
1x1 conv, 256
3x3 conv, 256
3x3 conv, 256
out3
x2
upsample
conv2_block3_out
1x1 conv, 64
BatchNorm
relu Activation
3x3 conv, 64
Batchnorm
relu Activation
1x1 conv, 256
BatchNorm
relu Activation
(x3)
1x1 conv, 256
3x3 conv, 256
3x3 conv, 256
out2
7x7 conv, 64
stride2
image

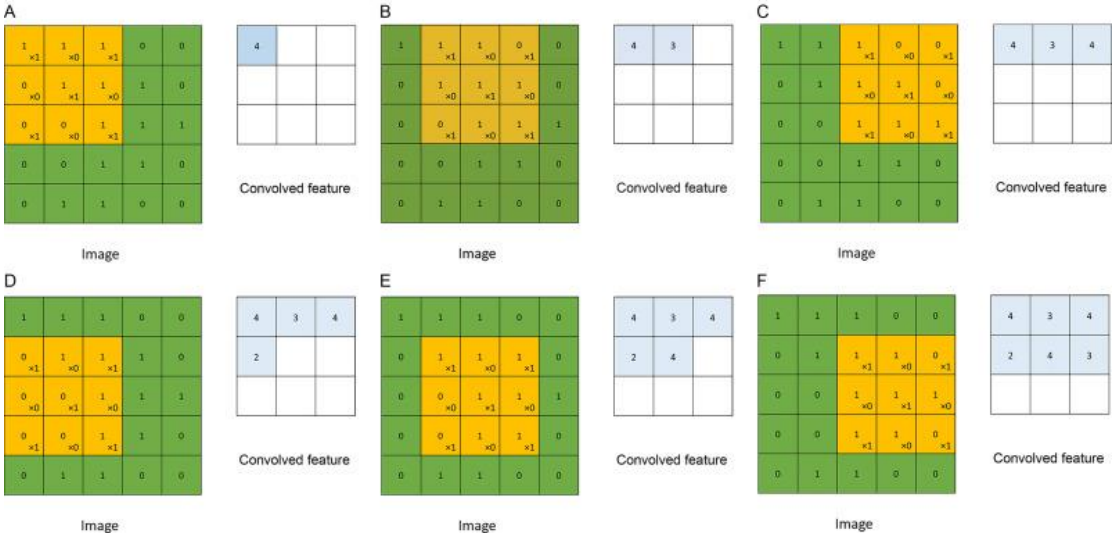
לאחר מכן יש את ה box predictor, זהו החלק במודל המקבל את מפות האובייקטים ומהם מבצע את שלב החיזוי הסופי של האובייקטים.



הסבר על סוגי השכבות השונים ברשת

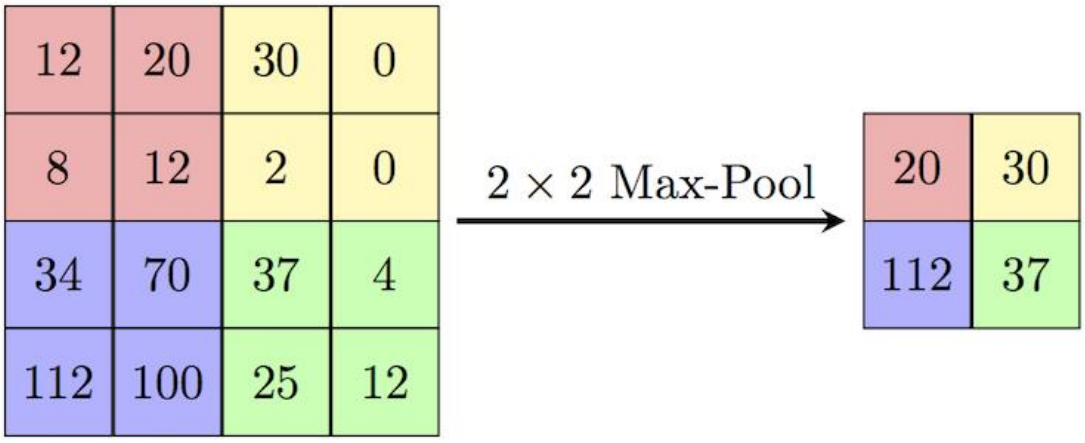
שכבת Conv

שכבה זו היא שכבה שלרוב משומשת כדי לנתח מידע מתמונות.
שכבה זו פועלת בעזרת פילטרים הנקראים kernels. כל פילטר בנוי מריבוע של משקולות שעוברות על התמונה, המשקולות מוכפלות במידע שהריבוע מכסה ומחברת את התוצאות, לאחר מכן הריבוע זז הצידה כמות מסוימת של צעדים וחוזר חלילה.



שכבת MaxPooling

שכבה זו נועדה כדי להקטין את גודל הממדים של התמונה על ידי העברה של ריבוע בגודל מסוים על התמונה, כמו שכבת Conv אבל רק לקחת את הערך הגבוה מבין כל הערכים בריבוע.
בדרך זו גודל הממדים משתנים משמעותית עם פגיעה מינימלית באיכות המידע, דבר זה מאפשר אימון וחישוב מהיר יותר של קלטים דרך המודל.



שכבת BatchNormalization

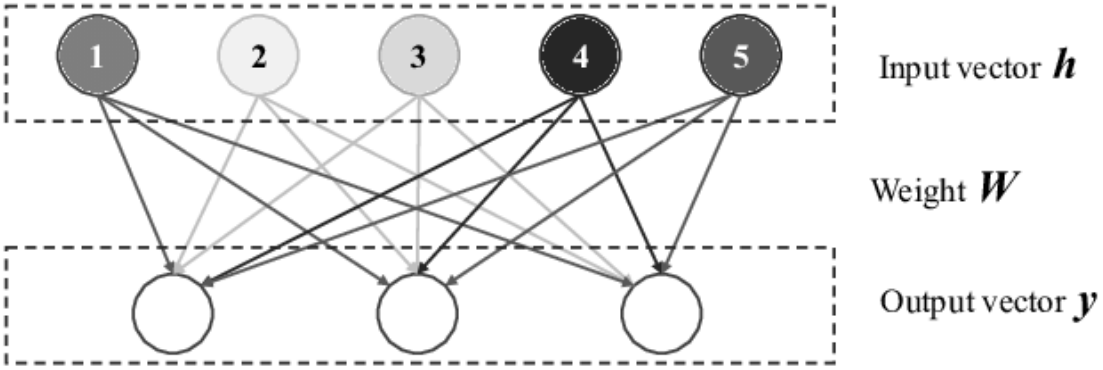
שכבה זו מנרמלת את תוצאות הhidden layers לממוצע סביב ה-0 עם פיזור של 1. דבר זה עוזר להגביר את קצב האימון של המודל ולמנוע את הבעיה של overfitting

שכבת Activation

שכבה זו מבצעת את פונקציית ההפעלה – Activation Function על הערכים שהיא מקבלת. דבר זה מכניס חוסר לינאריות ובכך המודל מסוגל לחזות דברים שלא קשורים בצורה קווית.

שכבת Dense

שכבה זו היא שכבה של נוירונים שבה כל הנוירונים מחוברים לכל הקלטים ולכל הפלטים של השכבה, כלומר שכבה זו היא Fully Connected



שכבת Concatenate

שכבה זו מחברת כמה array שהשכבה מקבלת לarray אחד.
השכבה מחברת את המערכים לפי ציר חיבור - axis.

דוחות וגרפים המתארים את תוצאות שלב האימון

בכל אימון של מודל חדש נוצרים כמה גרפים ודוחות. הגרפים האלו נשמרים בקבצי events בתוך התיקיות eval וtrain בתוך תיקיית אימון המודל וניתנים לקריאה על ידי כלי הוויזואליזציה שנקרא Tensorboard.

תוך כדי אימון המודל Tensorboard נפתח אוטומטית וניתן לראות את הדוחות דרך הלינק שנשלח בתחילת האימון:

```
TensorBoard 2.8.0 at http://localhost:56269/ (Press CTRL+C to quit)
```

לאחר סיום האימון ה Tensorboard נסגר.

כדי לראות מחדש את הדוחות ניתן לפתוח את Tensorboard מחדש דרך שורת הפקודה של אנקונדה על ידי הרצת הפקודה:

```
(test) C:\Users\▯▯▯▯>tensorboard --logdir=<path> --host=localhost --port=0
```

כאשר מחליפים את <path> בנתיב לתיקיית האימון.

ב Tensorboard עצמו ניתן לראות את הגרפים הבאים:

תחת הכרטיסייה Scalars:

ניתן לראות גרפים המתארים את ה mAP, היחס בין כמות החיזויים הנכונים שהמודל ביצע לחלק לסכום הניחושים הכללי של המודל, כלומר גרפים אלו מראים כמה מהחיזויים של המודל היו באמת נכונים.

ניתן לראות גרפים המתארים את ה Recall, היחס בין כמות החיזויים הנכונים חלקי סכום החיזויים הנכונים ועוד חוסר חיזוי במקום שאמור להיות, כלומר גרפים אלו מראים באיזו מידה המודל מצליח למצוא את כל הקופסאות שאמורות להיות.

ניתן לראות גרפים המתארים את הLoss,

ה classification_loss הוא כמה המודל טועה בחישוב הביטחון של המודל בכל חיזוי.

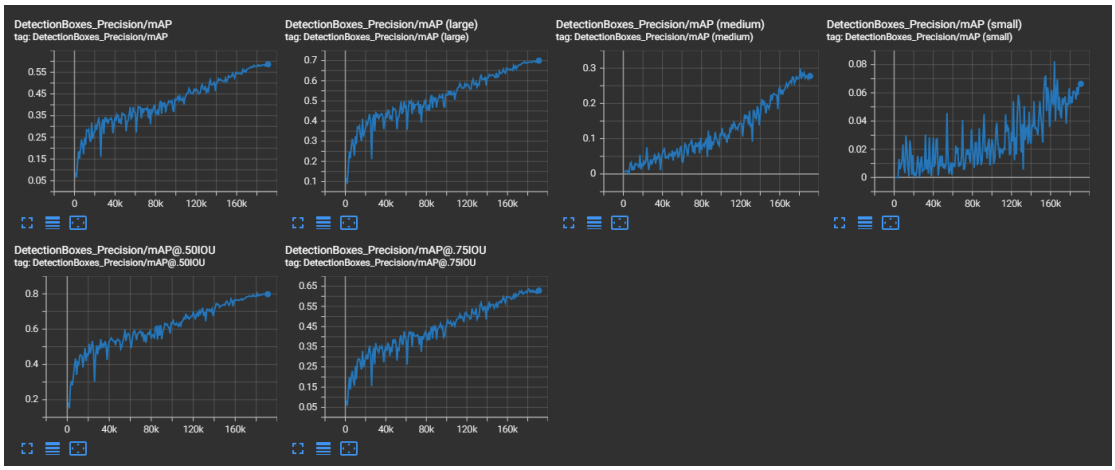
ה localization_loss הוא כמה המודל טועה בחישוב גודל ומיקום החיזוי.

ה regularization_loss הוא loss המיועד למנוע התאמה יתרה (overfitting) על הנתונים עליהם הdataset מתאמן, בצורה זו המודל יכול להכליל בצורה יותר טובה ולזהות גם תמונות שלא התאמן עליהן.

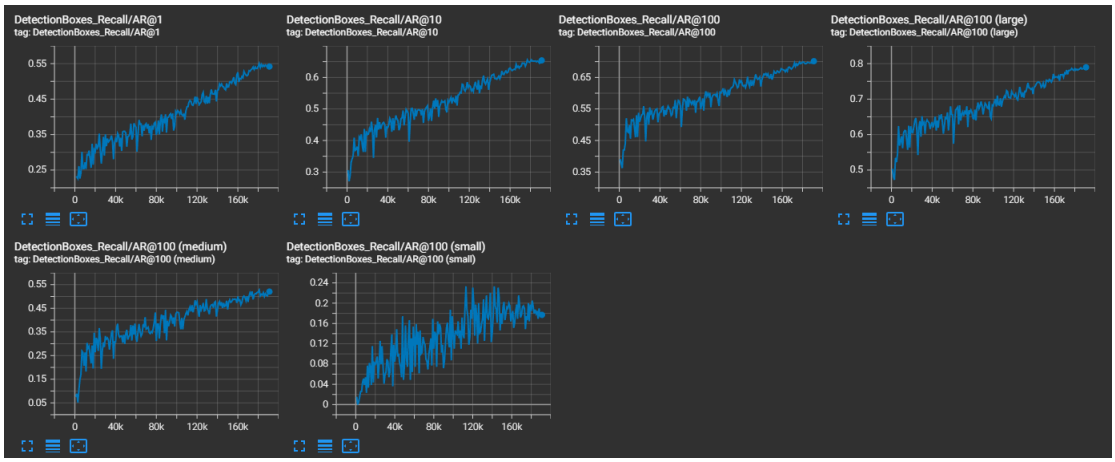
בנוסף ניתן לראות גרפים של ה learning_rate וכמות הsteps בשנייה.

ציר ה-X בכל אחד מהגרפים מתאר את ה-step.

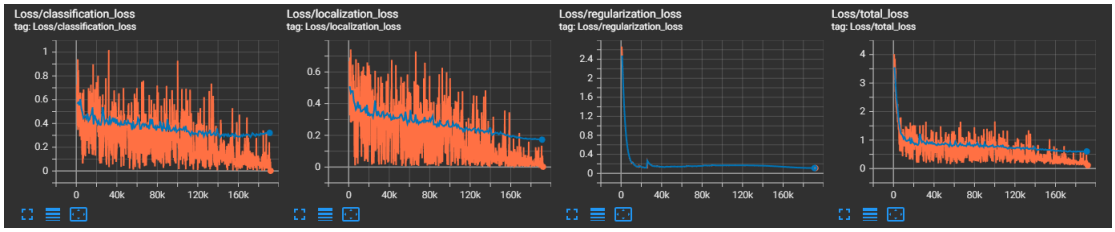
דוגמא לגרפי mAP:



דוגמא לגרפי Recall :

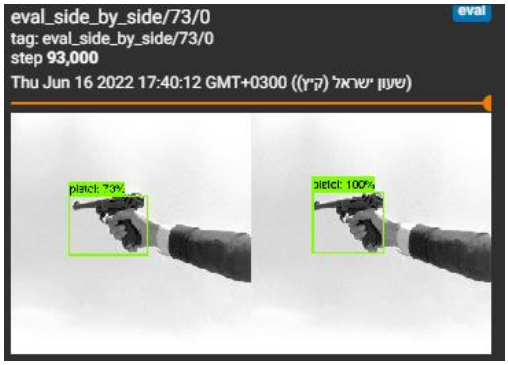


דוגמא לגרפי Loss :
הכתום, גרף Loss של האימון
הכחול, גרף Loss על בדיקת המודל



תחת הכרטיסייה Images:
ניתן לראות חלק מהתמונות שעליו אומן המודל לאחר augmentation.

ניתן לראות חיזוי של המודל על תמונות הevaluation בכל 1000 steps.
בכל אחת מהתמונות ניתן לראות את הstep שעליו נחזה התמונה ובעזרת הסליידר הכתום ניתן לזוז בין הstep.
החלק הימני של התמונה הוא התמונה עם התוויות האמיתית, החלק השמאלי של התמונה הוא התמונה עם החיזוי של המודל.



דוח הכולל ריכוז כל ה Hyper Parameters

כל ה Hyper Parameters הניתנים לשינוי על ידי הTensorflow Object Detection API נמצאים בקובץ הקונפיגורציה בתיקיית האימון הנוכחית.
בקובץ יש מספר רב של Hyper Parameters הניתנים לשינוי.

חלק ה feature extractor של המודל:

```
feature_extractor {
  type: "ssd_resnet50_v1_fpn_keras"
  depth_multiplier: 1.0
  min_depth: 16
  conv_hyperparams {
    regularizer {
      l2_regularizer {
        weight: 0.00039999998989515007
      }
    }
    initializer {
      truncated_normal_initializer {
        mean: 0.0
        stddev: 0.029999999329447746
      }
    }
    activation: RELU_6
    batch_norm {
      decay: 0.996999979019165
      scale: true
      epsilon: 0.0010000000474974513
    }
  }
  override_base_feature_extractor_hyperparams: true
  fpn {
    min_level: 3
    max_level: 7
  }
}
```

ה regularizer של שכבות ה Conv

ה initializer של שכבות ה Conv

ה Activation של שכבות ה Conv

שכבות ה Batch Normalization

חלק ה box predictor של המודל:

```
box_predictor {
  weight_shared_convolutional_box_predictor {
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 0.00039999998989515007
        }
      }
      initializer {
        random_normal_initializer {
          mean: 0.0
          stddev: 0.009999999776482582
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.996999979019165
        scale: true
        epsilon: 0.0010000000474974513
      }
    }
    depth: 256
    num_layers_before_predictor: 4
    kernel_size: 3
    class_prediction_bias_init: -4.599999904632568
  }
}
```

ה regularizer של שכבות ה Conv

ה initializer של שכבות ה Conv

ה Activation של שכבות ה Conv

שכבות ה Batch Normalization

פונקציית הLoss :

```
loss {
  localization_loss {
    weighted_smooth_l1 {
    }
  }
  classification_loss {
    weighted_sigmoid_focal {
      gamma: 2.0
      alpha: 0.25
    }
  }
  classification_weight: 1.0
  localization_weight: 1.0
}
```

ה Loss של הlocalization

ה Loss של הclassification

המשקל של כל אחד מהשגיאות
על השגיאה הכוללת

קונפיגורציית האימון:

```
train_config {
  batch_size: 2
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.04
          total_steps: 25000
          warmup_learning_rate: 0.013333
          warmup_steps: 2000
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "C:/dev/Project/P
  num_steps: 25000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  use_bfloat16: true
  fine_tune_checkpoint_version: V2
}
```

ה batch size

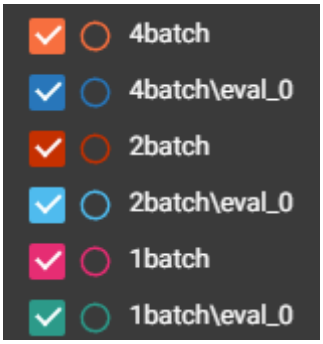
ה augmentations על התמונות המאומנות

ה optimizer

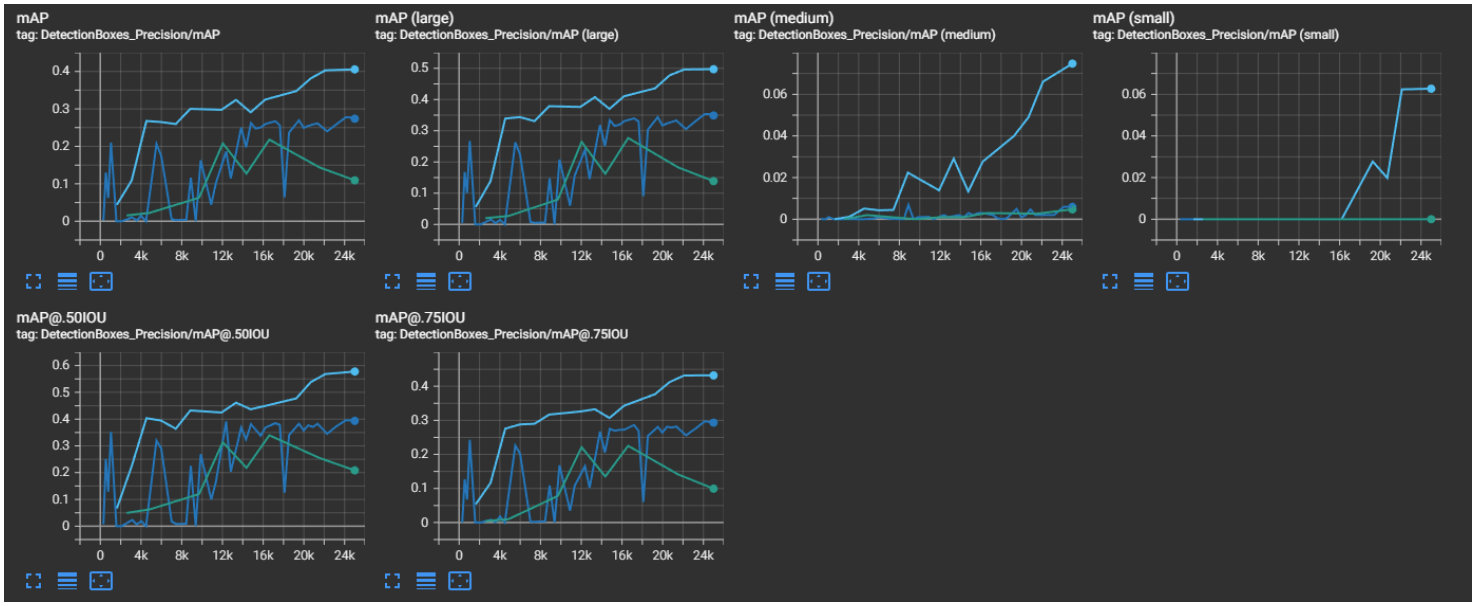
מספר ה steps לאימון

תיעוד על השינוי שנעשו במודל וב Hyper Parameters

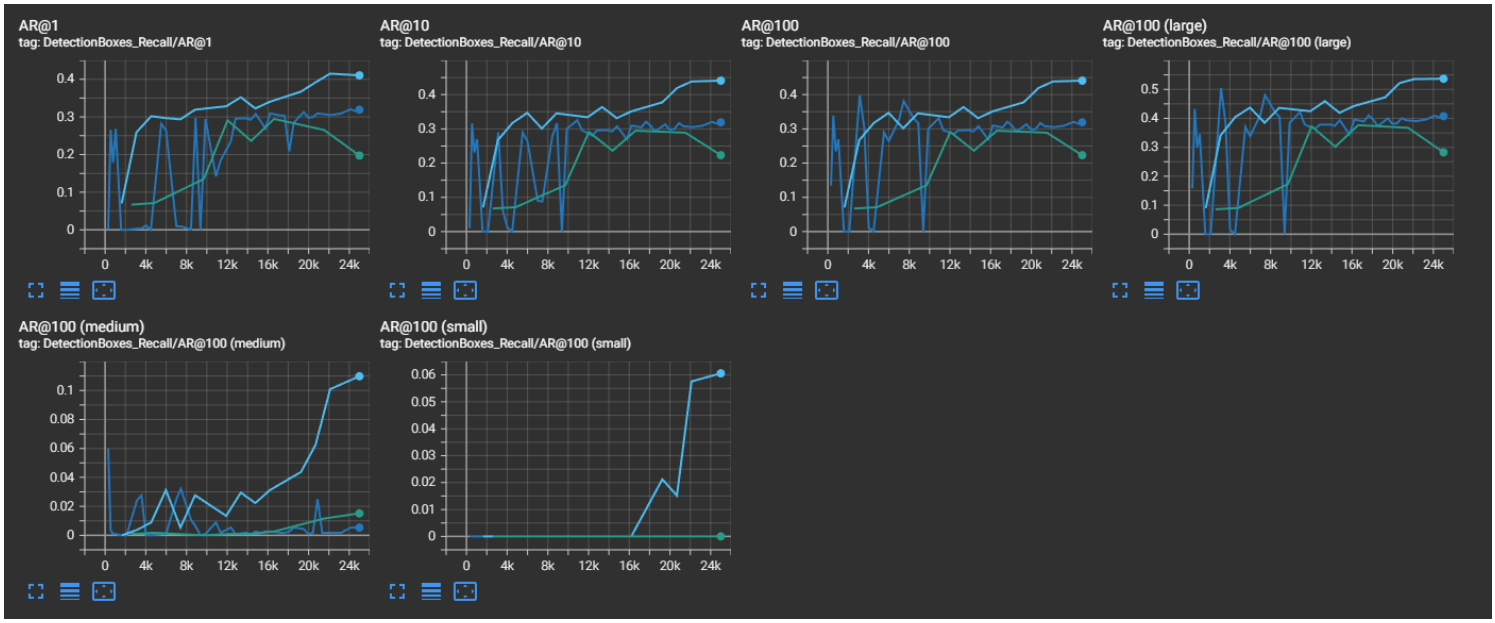
בתחילת תהליך הניסויים של המודל ניסיתי כמות שונה של batch_size במשך 25000 steps של אימונים.
אימנתי את המודל על batch_size של 1, 2, 4
הנה הגרפים של תוצאות האימונים מקודדים בצבעים לפי הטבלה הבאה:



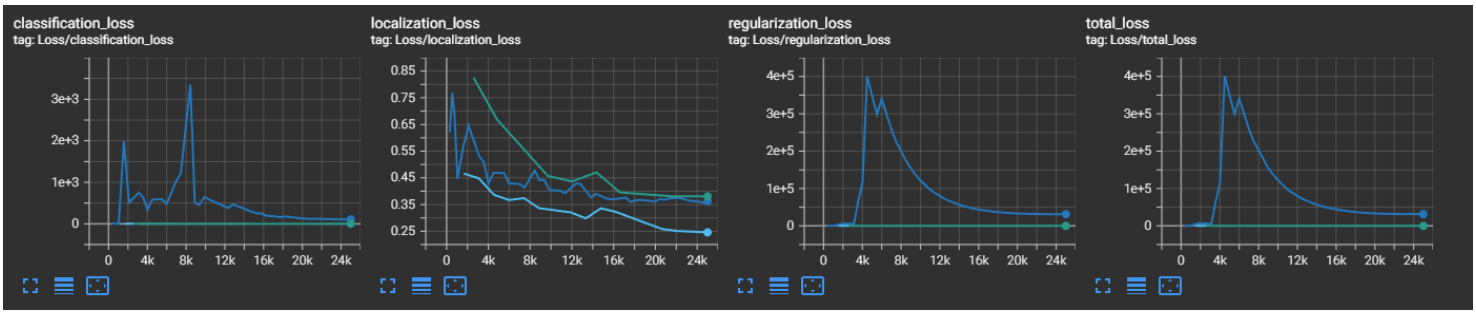
גרפי ה mAP:



גרפי ה Recall :

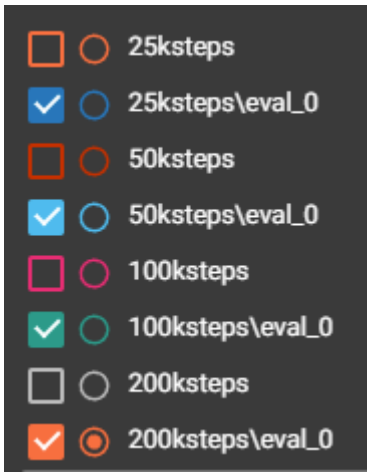


ה Loss :

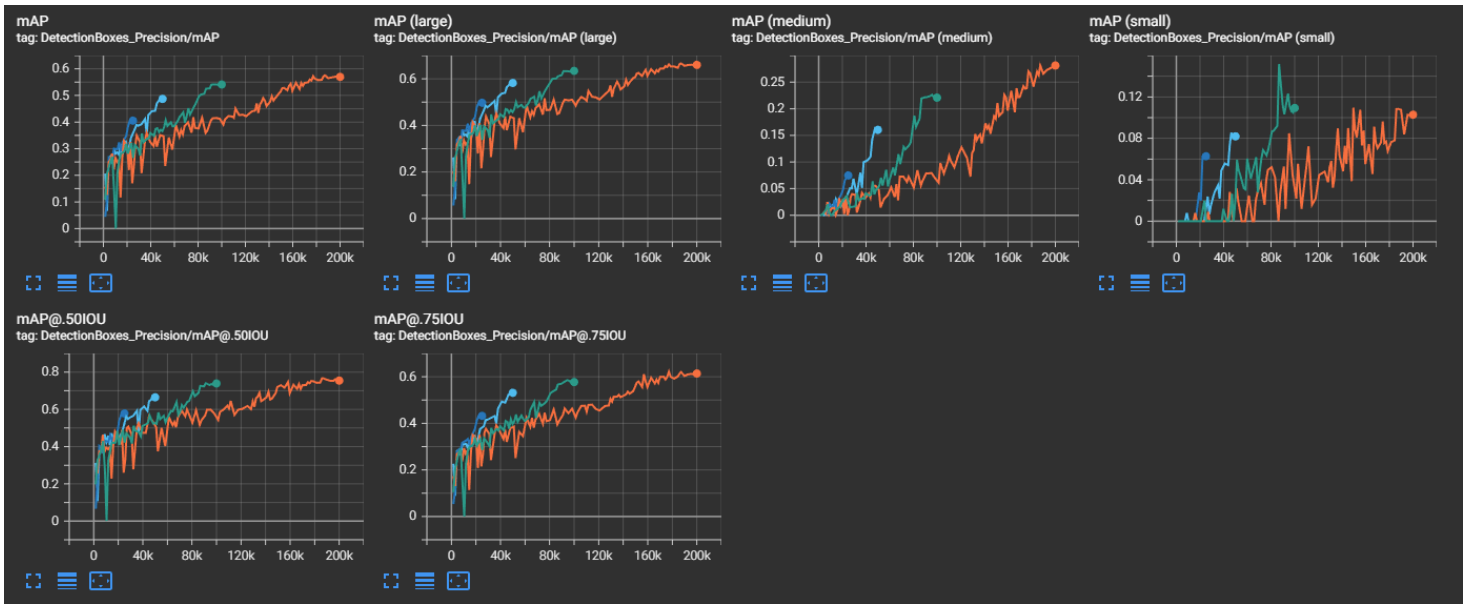


מתוך הגרפים ניתן לראות שכאשר הbatch_size הוא 2 תוצאות האימונים של המודל טובה יותר בצורה משמעותית.

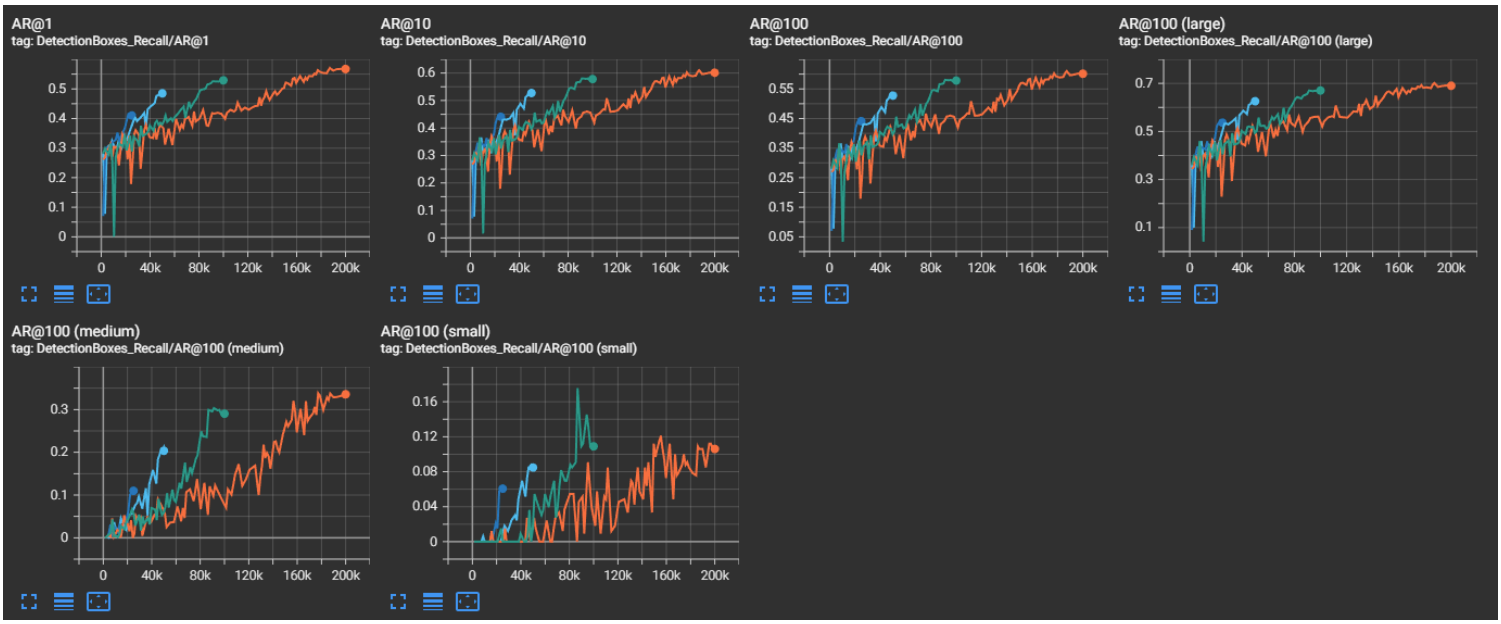
לאחר מכן ניסיתי שלושה ערכים שונים לכמות הsteps של אימון המודל על batch_size של 2.
בהתחלה אימנתי את המודל במשך 25000, 50000, 100000 ולבסוף 200000 steps.
הנה הגרפים של תוצאות האימונים מקודדים בצבעים לפי הטבלה הבאה:



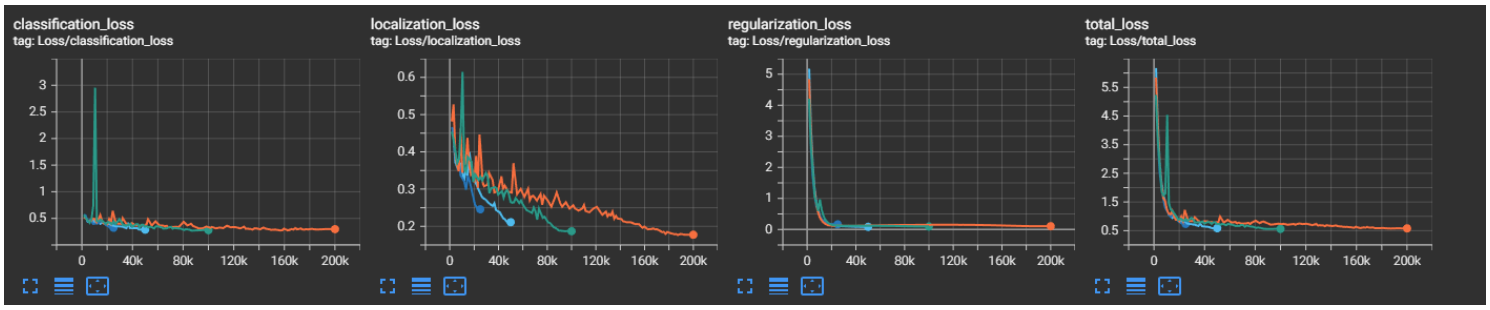
גרף ה mAP:



גרף ה Recall :



ה Loss :



מהתוצאות ניתן לראות שיש שיפור ככל שכמות ה steps עולה, למרות שהשיפור לא גדול הוא עדיין נוכח.

תיעוד והסבר של פונקציית השגיאה

פונקציית השגיאה של המודל מורכבת משלושה חלקים.
החלק הראשון הוא ה localization loss, זוהי שגיאה שנקבעת על פי היכולת של המודל לחזות את גודל ומיקום הקופסא בצורה נכונה.
שגיאה זו מחושבת על ידי פונקציית השגיאה weighted_smooth_l1 זוהי פונקציית שגיאה המשומשת כדי לבצע Object Detection.
פונקציה זו מחושבת על ידי הנוסחה:

$$L_1 := (a, b, N) \rightarrow \sum_{i=1}^{N} |a[i] - b[i]|$$

$$smooth_{L1} := (x) \rightarrow piecewise\left(\text{abs}(x) < 1, 0.5 \cdot x^2, \text{abs}(x) - 0.5\right)$$

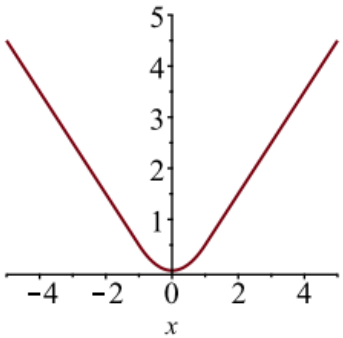
$$x \rightarrow piecewise\left(|x| < 1, 0.5\, x^2, |x| - 0.5\right) \quad (1)$$

Where x will be the L1 distance between 2 vectors.

$$smooth_{L1_{plot}} := piecewise\left(\text{abs}(x) < 1, 0.5 \cdot x^2, \text{abs}(x) - 0.5\right)$$

$$\begin{cases} 0.5\, x^2 & |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (2)$$

→



$$smooth_{L1}\left(L_1(p, q, \text{numelems}(p))\right)$$

החלק השני הוא ה classification loss, זוהי שגיאה הנקבעת על פי היכולת של המודל לזהות איזה אובייקט יש בתמונה וכמה בטוח המודל שיש שם אובייקט.
שגיאה זו מחושבת על ידי פונקציית השגיאה weighted_sigmoid_focal.
פונקציה זו מחושבת על ידי הנוסחה:

$$\boxed{FL = -\sum_{i=1}^{C=2} (1 - s_i)^{\gamma} t_i log(s_i)}$$

החלק השלישי הוא ה regulization loss והיא שגיאה המתווספת לשגיאה הרגילה כדי למנוע התאמה יתרה (overfitting) על הנתונים עליהם המודל מתאמן, בצורה זו המודל יכול להכליל בצורה יותר טובה ולזהות גם תמונות שלא התאמן עליהן.
פונקציה זו מחושבת על ידי הנוסחה:

$$\|\mathbf{w}\|_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}}$$

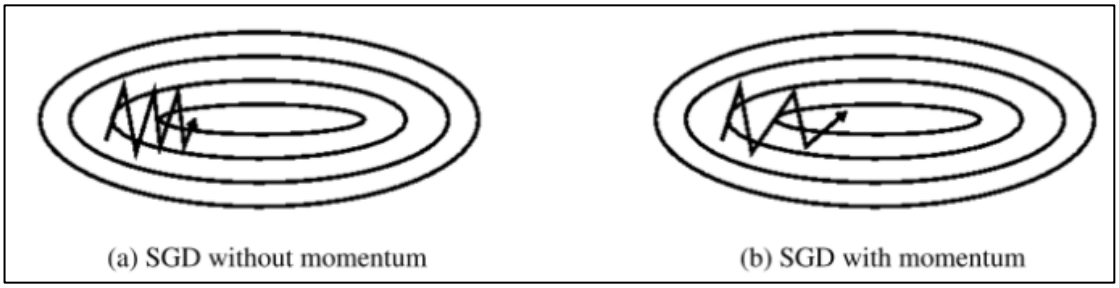
2-norm (also known as L2 norm or Euclidean norm)

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

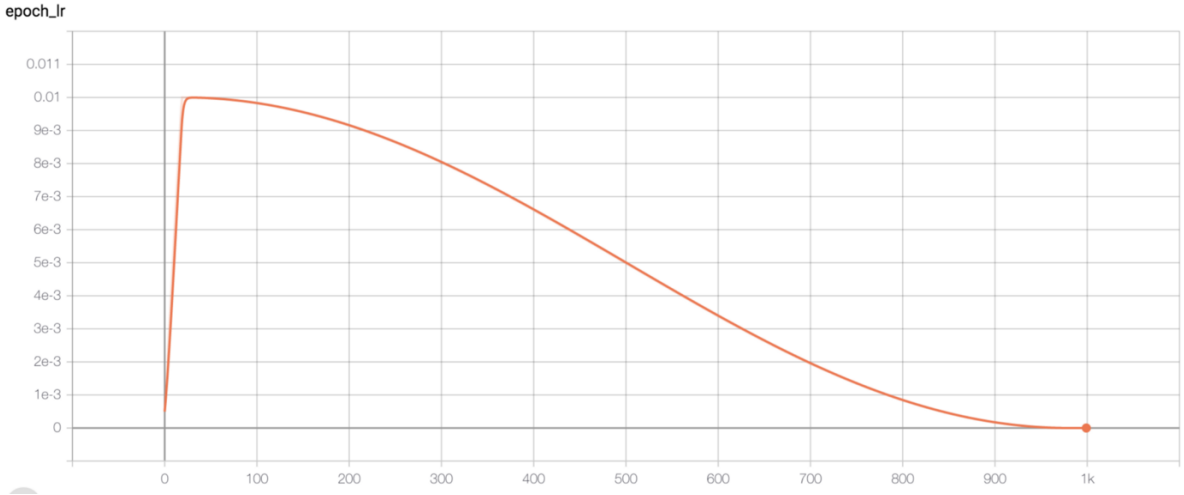
Loss function with L2 regularisation

תיעוד והסבר של ייעול ההתכנסות (Optimization)

ה Optimizer של המודל נקרא Momentum Optimizer.
מייעל זה פועל בשיטה הרגילה של Gradient Decent שבו המודל משתפר באמצעות חישוב השיפוע של פונקציית השגיאה, אבל בנוסף משתמש בMomentum.
Momentum הוא התנע של חישוב השגיאה. כלומר אם השגיאה מתקדמת בכיוון מסוים וממשיכה להתקדם באותו כיוון למשך פרק זמן, השינוי במשקולות יהיה גדול יותר ויותר כדי לעזור למודל להגיע לנקודת המינימום בפונקציית השגיאה יותר מהר. דבר זה מייעל את תהליך אימון המודל ומקצר את מספר הsteps הדרוש לאימון.

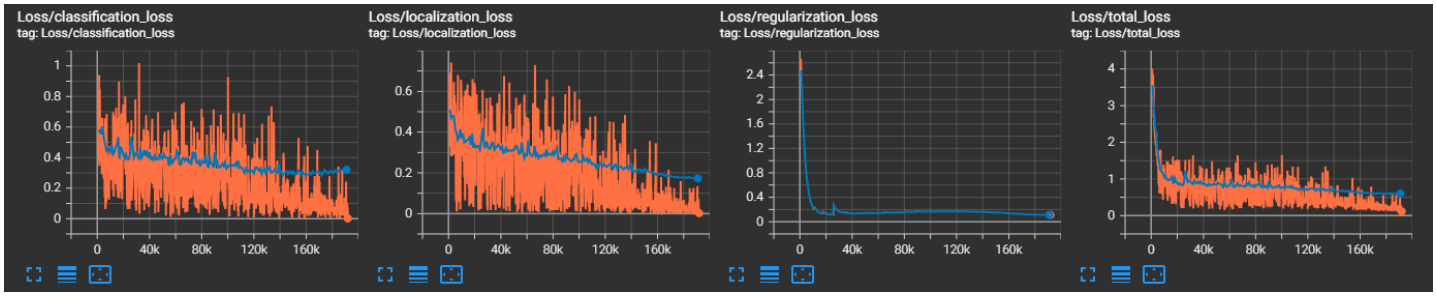


ה learning_rate של המודל נקרא cosine_decay_learning_rate, זהו learning rate המשתנה לאורך אימון המודל.
הנה גרף המציג את צורת הlearning rate:



כמו שניתן לראות בהתחלה הlearning rate עולה לערך מסוים ואז יורד בצורה הדרגתית.
דבר זה מקנה למודל שינוי גדול בהתחלה כדי להגיע למצב בו המודל מוצא נקודת מינימום טובה בפונקציית השגיאה.
לאחר מכן הlearning rate יורד בצורה הדרגתית, דבר שמקנה למודל יכולת להתאמן על הדקויות ולהגיע קרוב יותר ויותר לנקודת המינימום בפונקציית השגיאה.
כך נוצר מצב שבו המודל מתאמן מהר יחסית אבל לא מפספס את נקודת המינימום.

תיעוד ההתמודדות עם הטיה ושונות



הגרף הכתום הוא הגרף של שגיאת האימון, ניתן לראות בכל אחד מסוגי השגיאות שלמרות שהגרף קופץ למעלה ולמטה די הרבה, בטווח הרחוק הוא במגמה יורדת.

הגרף הכחול הוא הגרף של שגיאת המבחן, גם בו ניתן לראות שבכל אחד מסוגי השגיאות, השגיאה היא במגמה יורדת.

שלב היישום

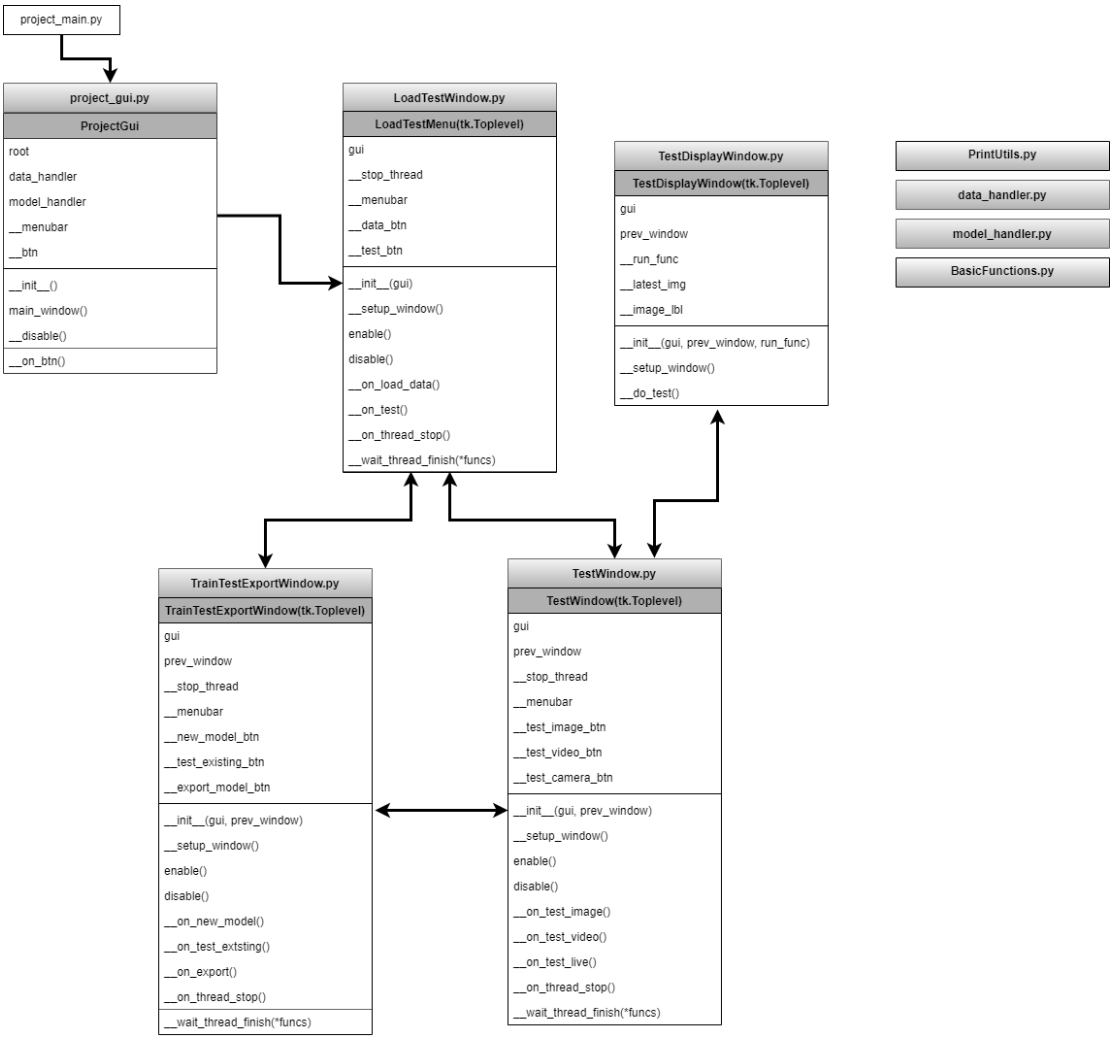
תיאור והסבר כיצד היישום משתמש במודל

דרך אחת היא כאשר רוצים לאמן מודל חדש, במקרה זה היישום מבקש מהמשתמש תיקייה שבה יוכל לאחסן את קבצי המודל, מייצר קובץ קונפיגורציה שאותו המשתמש יכול לשנות, ולאחר שהמשתמש סיים לשנות את הקובץ, פונה לTensorflow Object Detection API כדי לאמן מודל חדש עם הפרמטרים שניתנו בקובץ קונפיגורציה.

דרך שנייה היא כאשר אנו רוצים לייצא מודל שאימנו, במקרה זה היישום מבקש מהמשתמש את התיקייה של המודל שאותו המשתמש רוצה לייצא ואת התיקייה שבה יווצרו הקבצים של המודל המיוצא, לאחר מכן היישום פונה לTensorflow Object Detection API כדי לייצא את המודל.

דרך שלישית היא כאשר אנו רוצים לבדוק מודל שכבר ייוצא בשלב השני, במקרה זה המודל מקבל תמונה, או סרטון המחולק לתמונות ועבור התמונות האלה הוא מחזיר את הקופסאות המנוחשות.

תרשים UML של מחלקות הממשק



הטכנולוגיה שעל פיה מומש הממשק היא הספרייה tkinter.
הספרייה tkinter היא ספרייה סטנדרטית לייצור של ממשקים בפייתון.

תיאור קוד הקולט את הנתונים ומכין אותם לחיזוי

יש כמה דרכים בהן ניתן לבדוק את המודל המאומן:
דרך אחת היא לחזות על תמונה אחת ולהציג אותה למשתמש.
תחילה הממשק מבקש מהמשתמש לבחור תמונה מסוג jpg שעליה המודל יבצע חיזוי. לאחר מכן התמונה נטענת, מועברת לפורמט RGB ואז לnumpy array. לבסוף המערך מוגדל כדי להתאים לגודל הinput של המודל. כלומר (BatchSize, Width, Height, 3) כאשר Batchsize יהיה 1 מכיוון שיש רק תמונה אחת. המערך מוכנס למודל, מהמודל מתקבלות הקופסאות והערכת הניקוד של כל קופסא, כלומר כמה המודל בטוח שיש באותה קופסא אקדח. החיזוי של המודל מוצג על התמונה המקורית וזו מוצגת למשתמש בחלון חדש.

דרך שנייה היא לחזות על סרטון, ולשמור את הסרטון עם החיזוי עליו.
תחילה הממשק מבקש מהמשתמש קובץ סרטון מסוג mp4 וקובץ שאליו יישמר הסרטון לאחר החיזוי. הסרטון המקורי מחולק לפריימים, כלומר הוא מחולק לתמונות שמהן בנוי הסרטון, בעבור כל פריים המודל מציג חיזוי של קופסאות והערכות הניקוד, החיזוי של המודל מוצג על התמונה ונרשם כפריים בסרטון החדש.

דרך שלישית היא לחזות בשידור חי על המצלמה המותקנת על המחשב.
הממשק פותח את המצלמה ומתחיל לקלוט ממנה פריימים, בעבור כל פריים המודל מחזיר חיזוי של קופסאות והערכות הניקוד, החיזוי של המודל מוצג על התמונה ומוצג למשתמש בחלון החדש.

מדריך למפתח

project_main.py

זהו הקובץ הראשי של הפרויקט. זהו הקובץ הראשון שאותו מריצים בהתחלה והוא קורא לחלונות הממשק.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
env	שם סביבת הפייתון הנוכחית
python_version	גרסת הפייתון הנוכחית
projectgui	החלון הראשי שנפתח למשתמש

project_gui.py

זהו הקובץ המכיל את החלון הראשי שקופץ למשתמש.
קובץ זה מכיל את המחלקה ProjectGui האחראית על ניהול החלון המרכזי.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
root	חלון הtkinter הראשי
data_handler	מנהל הdata, דרך המשתנה הזה מתבצעות כל הפעולות על הdataset
model_handler	מנהל המודל, דרך המשתנה הזה מתבצעות על הפעולות על המודל, תוך כדי שימוש בTensorflow Object Detection API
__menubar	הכפתור של יציאה מהפרויקט
__btn	הכפתור של תחילת השימוש בפרויקט

שם הפעולה	תפקיד	משתנים
main_window	הפעולה פותחת את חלון ההתחלה של הפרויקט. הפעולה לא מקבלת ולא מחזירה כלום.	img – שומר את תמונת הפתיחה שמוצגת photo – תמונת הפתיחה מוכנה להצגה בחלון
__disable	משבית את הכפתורים שבחלון. הפעולה לא מקבלת ולא מחזירה כלום.	אין
__on_btn	הפעולה שנקראת כאשר לוחצים על כפתור ההתחלה. הפעולה משביתה את הכפתורים של החלון ופותחת את החלון של טעינת הdata או טעינת מודל קיים לבדיקה. הפעולה לא מקבלת ולא מחזירה כלום.	אין

LoadTestWindow.py

זהו הקובץ המכיל את החלון של בחירת טעינת הdata או טעינת מודל מאומן לבדיקה.
קובץ זה מכיל את המחלקה LoadTestWindow היורשת מהמחלקה tk.Toplevel המנהלת את החלון.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
gui	חלון הtkinter הראשי
__stop_thread	משתנה האומר אם לעצור את הThread שמריץ את הפעולות של הכפתורים
__menubar	הכפתור של יציאה מהפרויקט
__data_btn	הכפתור של טעינת הdata
__test_btn	הכפתור של טעינת מודל קיים ובדיקתו

שם הפעולה	תפקיד	משתנים
__setup_window	הפעולה פותחת וטוענת את החלון עצמו. הפעולה לא מקבלת ולא מחזירה כלום.	אין
enable	הפעולה מפעילה את הכפתורים שבחלון. הפעולה לא מקבלת ולא מחזירה כלום.	אין
disable	הפעולה משביתה את הכפתורים שבחלון. הפעולה לא מקבלת ולא מחזירה כלום.	אין
__on_load_data	הפעולה שנקראת כאשר לוחצים על כפתור ההתחלה. הפעולה משביתה את הכפתורים של החלון ומתחילה את תהליך טעינת הdata. לאחר מכן הפעולה פותחת את חלון האימון, בדיקה או ייצוא מודל. הפעולה לא מקבלת ולא מחזירה כלום.	t – הThread שמבצע את טעינת הdata
__on_test	הפעולה שנקראת כאשר לוחצים על כפתור הבדיקת מודל. הפעולה משביתה את הכפתורים של החלון ומתחילה את תהליך טעינת מודל. לאחר מכן הפעולה פותחת את חלון הבדיקת מודל. אם טעינת המודל כשלה הפעולה מפעילה מחדש את הכפתורים. הפעולה לא מקבלת ולא מחזירה כלום	t – הThread שמבצע את טעינת המודל ואז מריץ את חלון בדיקת המודל.
__on_thread_stop	הפעולה נקראת כאשר הThread מסתיים. הפעולה לא מקבלת ולא מחזירה כלום.	אין

__wait_thread_finish	הפעולה נקראת כאשר Thread מסתיים ומריצה את הפעולות שקיבלה. הפעולה מקבלת פונקציות שאותה תריץ – funcs הפעולה לא מחזירה כלום.	funcs – פונקציות שהפעולה תריץ ברגע שThread מסתיים.

TrainTestExportWindow.py

זהו הקובץ המכיל את החלון של בחירת אימון מודל חדש, בדיקת מודל קיים או ייצוא מודל מאומן.
קובץ זה מכיל את המחלקה TrainTestExportWindow המנהלת את החלון.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
gui	חלון הtkinter הראשי
prev_window	החלון הtkinter הקודם
__stop_thread	משתנה האומר אם לעצור את הThread שמריץ את הפעולות של הכפתורים
__menubar	הכפתור של יציאה מהפרויקט
__new_model_btn	הכפתור של אימון מודל חדש
__test_existing_btn	הכפתור של טעינת מודל קיים ובדיקתו
__export_model_btn	הכפתור של ייצוא מודל מאומן

שם הפעולה	תפקיד	משתנים
__setup_window	הפעולה פותחת וטוענת את החלון עצמו. הפעולה לא מקבלת ולא מחזירה כלום.	אין
enable	הפעולה מפעילה את הכפתורים שבחלון. הפעולה לא מקבלת ולא מחזירה כלום.	אין
disable	הפעולה משביתה את הכפתורים שבחלון. הפעולה לא מקבלת ולא מחזירה כלום.	אין
__on_new_model	הפעולה שנקראת כאשר לוחצים על כפתור האימון מודל חדש. הפעולה משביתה את הכפתורים של החלון ומתחילה את תהליך טעינת האימון. לאחר מכן הפעולה מפעילה את הכפתורים. הפעולה לא מקבלת ולא מחזירה כלום.	t – הThread שמבצע את אימון המודל.
__on_test_existing	הפעולה שנקראת כאשר לוחצים על כפתור הבדיקת מודל קיים. הפעולה משביתה את הכפתורים של החלון ומתחילה את תהליך טעינת מודל. לאחר מכן הפעולה פותחת את חלון הבדיקת מודל. אם טעינת המודל כשלה הפעולה מפעילה מחדש את הכפתורים. הפעולה לא מקבלת ולא מחזירה כלום.	t – הThread שמבצע את טעינת המודל ואז מריץ את חלון בדיקת המודל.

__on_export	הפעולה שנקראת כאשר לוחצים על כפתור הייצוא מודל מאומן. הפעולה משביתה את הכפתורים של החלון ומתחילה את תהליך הייצוא. לאחר מכן הפעולה מפעילה מחדש את הכפתורים. הפעולה לא מקבלת ולא מחזירה כלום.	Thread – שמבצע את ייצוא המודל.
__on_thread_stop	הפעולה נקראת כאשר Thread מסתיים. הפעולה לא מקבלת ולא מחזירה כלום.	אין
__wait_thread_finish	הפעולה נקראת כאשר Thread מסתיים ומריצה את הפעולות שקיבלה. הפעולה מקבלת פונקציות שאותה תריץ – funcs הפעולה לא מחזירה כלום.	funcs – פונקציות שהפעולה תריץ ברגע שThread מסתיים.

TestWindow.py

זהו הקובץ המכיל את החלון של בדיקת מודל.
קובץ זה מכיל את המחלקה TestWindow המנהלת את תהליך בדיקת המודל.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
gui	חלון הtkinter הראשי
prev_window	החלון הtkinter הקודם
__stop_thread	משתנה האומר אם לעצור את הThread שמריץ את הפעולות של הכפתורים
__menubar	הכפתור של יציאה מהפרויקט
__test_image_btn	הכפתור של בדיקת המודל על תמונה
__test_video_btn	הכפתור של בדיקת המודל על סרטון
__test_camera_btn	הכפתור של בדיקת המודל על סרטון חי מהמצלמה

שם הפעולה	תפקיד	משתנים
__setup_window	הפעולה פותחת וטוענת את החלון עצמו. הפעולה לא מקבלת ולא מחזירה כלום.	אין
enable	הפעולה מפעילה את הכפתורים שבחלון. הפעולה לא מקבלת ולא מחזירה כלום.	אין
disable	הפעולה משביתה את הכפתורים שבחלון. הפעולה לא מקבלת ולא מחזירה כלום.	אין
__on_test_image	הפעולה שנקראת כאשר לוחצים על כפתור הבדיקת המודל על תמונה. הפעולה משביתה את הכפתורים של החלון ופותחת את חלון הצגת החיזוי. הפעולה לא מקבלת ולא מחזירה כלום.	אין
__on_test_video	הפעולה שנקראת כאשר לוחצים על כפתור הבדיקת המודל על סרטון. הפעולה משביתה את הכפתורים של החלון ומתחילה את תהליך החיזוי על סרטון. לאחר מכן הפעולה מפעילה מחדש את הכפתורים. הפעולה לא מקבלת ולא מחזירה כלום.	t – הThread שמבצע את החיזוי על הסרטון.
__on_test_live	הפעולה שנקראת כאשר לוחצים על כפתור הבדיקת המודל על סרטון חי. הפעולה משביתה את הכפתורים של החלון ופותחת את חלון הצגת החיזוי. הפעולה לא מקבלת ולא מחזירה כלום.	אין

__on_thread_stop	הפעולה נקראת כאשר הThread מסתיים. הפעולה לא מקבלת ולא מחזירה כלום.	אין

TestDisplayWindow.py

זהו הקובץ המכיל את החלון של הצגת החיזוי של המודל.
קובץ זה מכיל את המחלקה TestDisplayWindow המנהלת את הצגת החיזוי של המודל.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
gui	חלון tkinter הראשי
prev_window	החלון tkinter הקודם
__run_func	הפונקציה שאותה החלון יריץ בעת החיזוי
__latest_img	התמונה האחרונה שהוצגה על ידי החלון
__image_lbl	התוויות שדרכה מוצגת התמונה של החיזוי

שם הפעולה	תפקיד	משתנים
__setup_window	הפעולה פותחת וטוענת את החלון עצמו. הפעולה לא מקבלת ולא מחזירה כלום.	menubar – הכפתור של החזרה לחלון הקודם t – הThread של חיזוי המודל
__do_test	הפעולה שמבצעת את תהליך החיזוי ומציגה אותו. הפעולה לא מקבלת ולא מחזירה כלום.	אין

data_handler.py

זהו הקובץ שאחראי על כל העבודה עם הdata המתרחשת בפרויקט.
קובץ זה מכיל את המחלקה DataHandler המנהלת את העבודה עם הdata.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
gui	המחלקה של חלון הtkinter הראשי
root	חלון הtkinter הראשי
extracted_dir	הנתיב לתיקייה שבה נפתח הdataset
split_dir	הנתיב לתיקייה שבה מאוחסן הdataset המפוצל לאימון ובדיקה
encoded_dir	הנתיב לתיקייה שבה מאוחסן הdataset לאחר שקודד
train_record_path	הנתיב לקובץ המקודד של הdataset לאימון
eval_record_path	הנתיב לקובץ המקודד של הdataset לבדיקה
label_map_path	הנתיב לקובץ השומר את ההמרה של מספר אובייקט לשם
num_train	מספר התמונות לאימון המודל
num_eval	מספר התמונות לבדיקת המודל

שם הפעולה	תפקיד	משתנים
handle_dataset	הפעולה האחראית על כל השלבים של טעינת הdataset. החל מטעינת הzip המקורי, לבדיקת תקינות המידע וחילוקו, ולבסוף קידודו. הפעולה לא מקבלת ולא מחזירה כלום.	אין
__get_data	הפעולה שמבצעת את תהליך חילוץ הdataset מקובץ הzip. הפעולה קולטת מהמשתמש את קובץ הzip ואת התיקייה הריקה שאליה יחולץ. הפעולה לא מקבלת ולא מחזירה כלום	data_zip – הקובץ zip של הdataset

__validate_data	הפעולה שמבצעת את תהליך בדיקת התקינות של הdataset. הפעולה מוחקת את הקבצים הלא תקינים. הפעולה לא מקבלת ולא מחזירה כלום	images_dir – התיקייה של התמונות anns_dir – התיקייה של התוויות לתמונות cnt – כמות הקבצים התקינים cnt_err – כמות הקבצים הלא תקינים error_files – רשימת הנתיבים לקבצים הלא תקינים error – האם יש בעיה בקובץ הנוכחי xml_tree – עץ ההיררכיה של הקובץ הנוכחי xml_root – האלמנט הראשי של הקובץ הנוכחי filename – שם הקובץ, נלקח מקובץ התווית img_width – רוחב התמונה, נלקח מקובץ התווית xml_objects – האלמנטים של הקופסאות objects – הקופסאות בקובץ הנוכחי obj_err – האם יש בעיה בקופסא הנוכחית clazz – שם האובייקט בקופסא הנוכחית, נלקח מקובץ התווית xmin – הגבול השמאלי בקופסא הנוכחית, נלקח מקובץ התווית ymin – הגבול התחתון בקופסא הנוכחית, נלקח מקובץ התווית xmax – הגבול הימני בקופסא הנוכחית, נלקח מקובץ התווית ymax – הגבול העליון בקופסא הנוכחית, נלקח מקובץ התווית img_path – הנתיב לקובץ התמונה לפי הקובץ img – התמונה שנטענה org_height – הגובה האמיתי של התמונה org_width – הרוחב האמיתי של התמונה

__split_data	הפעולה שמחלקת את הdataset לtrain וtest. הפעולה קולטת מהמשתמש את התיקייה שאליה יחולק הdata. הפעולה לא מקבלת ולא מחזירה כלום.	anns_dir – התיקייה של קבצי התוויות train_dir – התיקייה שאליה יועברו תוויות האימון eval_dir – התיקייה שאליה יועברו תוויות הבדיקה files – רשימת הנתיבים לתוויות train – רשימת הנתיבים לתוויות האימון eval – רשימת הנתיבים לתוויות הבדיקה
__encode_data	הפעולה שמקודדת את הdataset להכנה לשליחה ל Tensorflow Object Detection API. הפעולה לא מקבלת ולא מחזירה כלום.	train_dir – התיקייה של קבצי התוויות לאימון eval_dir – התיקייה של קבצי התוויות לבדיקה train_df – הDataFrame המכיל את תוויות האימון classes_train – רשימה המכילה את שמות האובייקטים הקיימים בתיקיית האימון eval_df – הDataFrame המכיל את תוויות הבדיקה classes_eval – רשימה המכילה את שמות האובייקטים הקיימים בתיקיית הבדיקה classes – רשימת שמות האובייקטים בכל הdataset pbtxt_content – התוכן של קובץ האובייקטים label_to_id – תרגום שם התוויות למספר אובייקט gb – הdataset מחולק לפי הקבצים data – רשימה המכילה את הdataset המחולק לפי שם קובץ ותוויות tf_example – התוויות של התמונה מקודדת כהכנה ל Tensorflow Object Detection API

model_handler.py

זהו הקובץ שאחראי על כל העבודה עם המודל המתרחשת בפרויקט.
קובץ זה מכיל את המחלקה ModelHandler המנהלת את העבודה עם המודל.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
gui	המחלקה של חלון הtkinter הראשי
root	חלון הtkinter הראשי
data_handler	מנהל הdata, דרכו מתנהלת על העבודה עם הdataset
training_dir	הנתיב לתיקייה בשה יישמרו קבצי האימון של המודל
category_index	הממיר של מספר אובייקטים לשמם
loaded_model	המודל הטעון ומוכן לבדיקה

שם הפעולה	תפקיד	משתנים
train_model	הפעולה אחראית על אימון מודל חדש. הפעולה קולטת מהמשתמש את תיקיית האימון. הפעולה לא מקבלת ולא מחזירה כלום.	tb_proc – התהליך שמריץ את Tensorboard start_time – הזמן שבו התחיל האימון של המודל end_time – הזמן שבו נגמר האימון של המודל
export_model	הפעולה שמייצאת מודל שכבר אומן ומכווצת אותו לקובץ zip. הפעולה קולטת מהמשתמש את תיקיית המודל המאומן. הפעולה קולטת מהמשתמש את תיקיית הייצוא. הפעולה לא מקבלת ולא מחזירה כלום	model_dir – הנתיב לתיקייה של המודל המאומן export_path – הנתיב לתיקייה של המודל המיוצא proc – התהליך שמריץ את אימון המודל
load_model	הפעולה שטוענת מודל מיוצא כדי לבצע עליו test. הפעולה קולטת מהמשתמש את קובץ הzip של המודל. הפעולה קולטת מהמשתמש את התיקייה שאליה ייפתח המודל. הפעולה מחזירה True אם המודל נטען בהצלחה, אחרת False.	model_zip – קובץ הzip של המודל extract_dir – התיקייה שאליה יחולץ המודל PATH_TO_LABEL_MAP – הנתיב לקובץ label_map label_map – הקובץ label_map הטעון categories – ההמרה של קובץ הlabel_map להמרת קטגוריות
test_on_image	הפעולה שמבצעת את החיזוי על תמונה. הפעולה קולטת מהמשתמש קובץ תמונה. הפעולה מקבלת את תווית הtkinter שאליה תציג את התמונה - label. הפעולה מחזירה את התמונה בפורמט הtkinter.	image_file – קובץ התמונה לחיזוי image_np – מערך הnumpy שמייצג את התמונה img – התמונה בפורמט PIL photo – התמונה בפורמט tkinter

test_on_video	הפעולה שמבצעת את החיזוי על סרטון. הפעולה קולטת מהמשתמש את קובץ הסרטון לחיזוי. הפעולה קולטת מהמשתמש את הנתיב לקובץ הסרטון לאחר החיזוי. הפעולה לא מקבלת ולא מחזירה כלום.	video_file – נתיב קובץ הסרטון לחיזוי out_video_file – נתיב קובץ הסרטון לאחר החיזוי vidcap – קורא הסרטון לחיזוי length – מספר הפריימים בסרטון width – רוחב הפריים בסרטון height – אורך הפריים בסרטון fps – מספר הפריימים בשנייה בסרטון success – האם הפריים נראה בהצלחה frame – המערך הnumpy המייצג את הפריים count – מספר הפריים הנוכחי writer – כותב הסרטון לקובץ החיזוי
test_on_live	הפעולה שמבצעת חיזוי על שידור חי מהמצלמה. הפעולה מקבלת את תווית הtkinter שאליה תוצג החיזוי – label. הפעולה לא מחזירה כלום.	vidcap – קורא את השידור החי מהמצלמה success – האם קריאת הפריים הצליחה frame – הפריים הנוכחי image_np – מערך הnumpy המייצג את התמונה לאחר החיזוי img – התמונה בפורמט PIL photo – התמונה בפורמט tkinter
__pred_img	הפעולה מבצעת חיזוי על תמונה. הפעולה מקבלת מערך numpy המייצג את התמונה לחיזוי – image_np הפעולה מחזירה את מערך התמונה לאחר החיזוי	image_np_expanded – מערך הnumpy מוכן לשליחה למודל output – החיזוי הישיר של המודל boxes – חיזוי הקופסאות של המודל scores – הביטחון של המודל בכל אחד מהקופסאות classes – האובייקטים שחוזו על ידי המודל
__prepare_config	הפעולה מכינה קובץ קונפיגורציה לאימון ושומרת אותו בתיקיית האימון הנוכחית. הפעולה לא מקבלת כלום ולא מחזירה כלום.	pipeline – קובץ הקונפיגורציה הבסיסי config_text – מחרוזת המתארת את קובץ הקונפיגורציה המוכן לאימון.

BasicFunctions.py

קובץ זה מכיל פעולות בסיסיות לשימוש בקבצים אחרים.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם הפעולה	תפקיד	משתנים
load_image_into_numpy_array	טוענת תמונה מנתיב למערך numpy. הפעולה מקבלת נתיב לקובץ תמונה – path. הפעולה מחזירה את המערך המכיל את התמונה.	image – התמונה בפורמט PIL
ask_directory	הפעולה מבקשת מהמשתמש לבחור תיקייה. הפעולה מקבלת האם להכריח את המשתמש לבחור תיקייה – force ועוד משתנים להכניס לפעולת הדיאלוג – kwargs הפעולה מחזירה את נתיב התיקייה שנבחרה.	output_dir – הנתיב לתיקייה שהמשתמש בחר
ask_empty_directory	הפעולה מבקשת מהמשתמש לבחור תיקייה ריקה. הפעולה מקבלת האם להכריח את המשתמש לבחור תיקייה – force ועוד משתנים להכניס לפעולת הדיאלוג – kwargs הפעולה מחזירה את נתיב התיקייה שנבחרה.	dire – הנתיב לתיקייה שהמשתמש בחר
ask_for_file	הפעולה מבקשת מהמשתמש לבחור קובץ. הפעולה מקבלת האם להכריח את המשתמש לבחור קובץ – force ועוד משתנים להכניס לפעולת הדיאלוג – kwargs הפעולה מחזירה את נתיב הקובץ שנבחר.	output – הנתיב לקובץ שהמשתמש בחר
ask_save_file	הפעולה מבקשת מהמשתמש לבחור קובץ לשמירה. הפעולה מקבלת האם להכריח את המשתמש לבחור קובץ – force ועוד משתנים להכניס לפעולת הדיאלוג – kwargs הפעולה מחזירה את נתיב הקובץ שנבחר.	output – הנתיב לקובץ שהמשתמש בחר
extract_zip	הפעולה פותחת קובץ zip. הפעולה מקבלת את נתיב הקובץ - path ונתיב לתיקייה שאליה ייוצא הקובץ – output_path הפעולה מחזירה True אם הקובץ נפתח בהצלחה, אחרת False	ext – הסיומת של הקובץ

to_zip	הפעולה מכווצת תיקייה לקובץ zip. הפעולה מקבלת את הנתיב לתיקייה – path והנתיב לקובץ הzip – output_file הפעולה לא מחזירה כלום.	root – הנתיב המקורי לתיקייה dirs – התיקיות בתיקייה המקורית files – הקבצים בתיקייה המקורית
xml_to_csv	הפעולה לוקחת את כל קבצי הxml שמייצגים תוויות והופכת אותם לcsv. הפעולה מקבלת את הנתיב לתיקיית התוויות – path ופילטר לשמות האובייקטים התקניים – classes_filter הפעולה מחזירה DataFrame המכיל את תוכן הקבצים ורשימה בעלת שמות האובייקטים שבקבצים	images_extension – הסיומת לקבצי התמונות classes_names – השמות של האובייקטים הנמצאים בקבצי התוויות xml_list – כל התוויות שנמצאות בכל הקבצים בתיקייה xml_file – קובץ התוויות הנוכחי xml_ tree – עץ ההיררכיה של הקובץ הנוכחי xml_root – האלמנט הראשי של הקובץ הנוכחי filename – שם הקובץ, נלקח מקובץ התווית img_width – רוחב התמונה, נלקח מקובץ התווית xml_objects – האלמנטים של הקופסאות objects – הקופסאות בקובץ הנוכחי obj_err – האם יש בעיה בקופסא הנוכחית clazz – שם האובייקט בקופסא הנוכחית, נלקח מקובץ התווית xmin – הגבול השמאלי בקופסא הנוכחית, נלקח מקובץ התווית ymin – הגבול התחתון בקופסא הנוכחית, נלקח מקובץ התווית xmax – הגבול הימני בקופסא הנוכחית, נלקח מקובץ התווית ymax – הגבול העליון בקופסא הנוכחית, נלקח מקובץ התווית column_names – שם הטורים של DataFrame התוויות xml_df – הDataFrame של הקובץ הנוכחי

create_pbtxt	הפעולה יוצרת קובץ pbtxt משמות האובייקטים. הפעולה מקבלת רשימה של שמות האובייקטים – classes הפעולה מחזירה את תוכן קובץ הpbtxt ומילון המרה של שם אובייקט למספר	
create_tf_example	הפעולה מקודדת קובץ תוויות לפומט הניתן לקריאה על ידי Tensorflow Object Detection API הפעולה מקבלת את הנתונים group – נתיב לתיקיית התמונות – image_dir מילון המרה של שם אובייקט למספר – label_to_id הפעולה מחזירה את הנתונים המקודדים	encoded_jpg_io – התמונה מקודדת לביטים image – התמונה בפורמט PIL width – רוחב התמונה height – גובה התמונה image_format – סיומת התמונה מקודדת לביטים xmins – הגבולות השמאליים של התוויות xmaxs – הגבולות הימניים של התוויות ymins – הגבולות התחתונים של התוויות ymaxs – הגבולות העליונים של התוויות classes_text – השמות של האובייקטים בתמונה classes – הid של האובייקטים בתמונה tf_example – התמונה והתוויות מקודדות ל Tensorflow Object Detection API
combine_functions	הפעולה מחברת כמה פונקציות כדי שיוכלו להיקרא ביחד אחר כך. הפעולה מקבלת פונקציות – funcs הפעולה מחזירה פונקציה הקוראת לכל הפונקציות	אין

PrintUtils.py

קובץ זה מכיל פעולות בסיסיות לתקשורת עם המשתמש דרך שורת הפקודה של אנקונדה.
מיקומו של הקובץ הוא בתיקייה הראשית.

שם המשתנה	תפקיד
__TIME_FORMAT	הפורמט לשליחת הודעה עם זמן השליחה
__FORMAT	הפורמט לשליחת הודעה בלי זמן שליחה
__TIMEFORMAT	הפורמט להפיכת זמן ממספר למחרוזת

שם הפעולה	תפקיד	משתנים
printmsg	הפעולה מדפיסה את ההודעה עם התחלית המתאימה. הפעולה מקבלת את ההודעה לשליחה – message הפעולה מקבלת את התחילית לשליחה – prefix הפעולה מקבלת האם להדפיס את זמן השליחה – show_time הפעולה לא מחזירה כלום.	אין
getinput	הפעולה מדפיסה את הודעת הקלט ומקבלת קלט מהמשתמש. הפעולה מקבלת את ההודעה לשלוח למשתמש – message	אין
chooseinputs	הפעולה נותנת למשתמש לבחור מבין כמה אפשרויות. הפעולה מקבלת את ההודעה לשלוח למשתמש – message הפעולה מקבלת את רשימת האפשרויות – options הפעולה מחזירה את האופציה שנבחרה.	inp – הקלט שהמשתמש בחר
inputmsg	הפעולה מדפיסה הודעה של קלט מהמשתמש. הפעולה מקבלת את ההודעה לשליחה – message הפעולה לא מחזירה כלום.	אין
debug	הפעולה מדפיסה הודעה בסימן debug. הפעולה מקבלת את ההודעה לשליחה – message הפעולה מקבלת האם לשלוח את זמן ההודעה – show_time הפעולה לא מחזירה כלום.	אין
info	הפעולה מדפיסה הודעה בסימן info. הפעולה מקבלת את ההודעה לשליחה – message הפעולה מקבלת האם לשלוח את זמן ההודעה – show_time הפעולה לא מחזירה כלום.	אין

warning	הפעולה מדפיסה הודעה בסימן warning. הפעולה מקבלת את ההודעה לשליחה – message הפעולה מקבלת האם לשלוח את זמן ההודעה – show_time הפעולה לא מחזירה כלום.	אין
error	הפעולה מדפיסה הודעה בסימן error. הפעולה מקבלת את ההודעה לשליחה – message הפעולה מקבלת האם לשלוח את זמן ההודעה – show_time הפעולה לא מחזירה כלום.	אין
critical	הפעולה מדפיסה הודעה בסימן critical. הפעולה מקבלת את ההודעה לשליחה – message הפעולה מקבלת האם לשלוח את זמן ההודעה – show_time הפעולה לא מחזירה כלום.	אין

base_pipeline.config

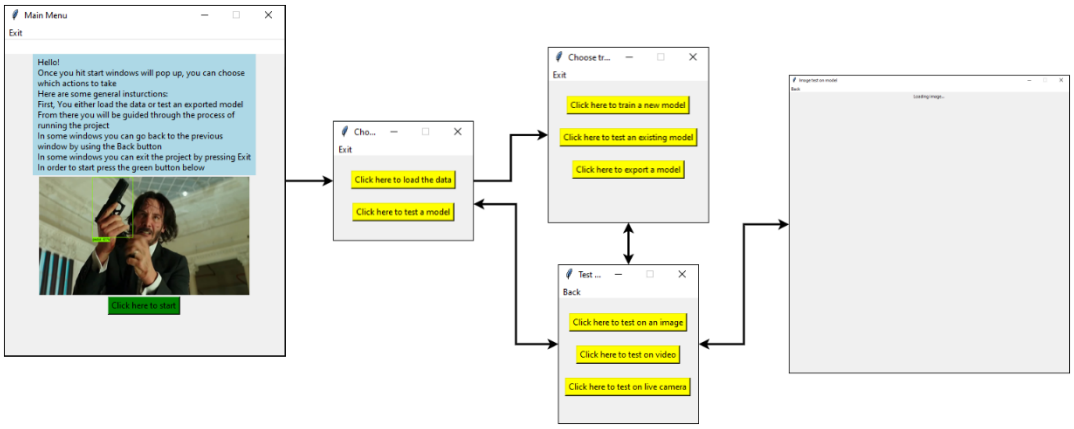
קובץ זה מכיל את הקונפיגורציה הבסיסית לאימון מודל חדש.
קובץ זה ממוקם בתוך התיקייה resources.
קובץ זה מועתק כאשר מאמנים מודל חדש, הערכים בקובץ זה מייצגים את את ה hyper parameters של המודל.

התיקייה models

התיקייה המכילה את קבצי Tensorflow Object Detection API.
התיקייה נמצאת בתוך התיקייה resources.

מדריך למשתמש

תרשים מסכים - Screen flow diagram



הוראות התקנה

יש להתקין במחשב את סביבת העבודה Anaconda
(https://docs.anaconda.com/anaconda/install/)
בתוך Anaconda יש ליצור סביבה (Environment) חדשה המבוססת על פייתון 3.7.13
על המשתמש להריץ את הפרויקט דרך סביבת העבודה בדרך שתפורט בהמשך.

כדי ליצור סביבה חדשה המשתמש פותח את שורת הפקודה של אנקונדה (Anaconda Prompt)
ומקליד :

```
>conda create --name <Name> python=3.7.13
```

את המילה <Name> המשתמש מחליף בשם הסביבה כרצונו. ההתקנה תבקש מהמשתמש ללחוץ
על כפתור האנטר כדי להתקין את הספריות הבסיסיות.

כעת על המשתמש להפעיל את הסביבה החדשה באמצעות

```
conda activate <Name>
```

ולאחר מכן המשתמש יראה את שם הסביבה בסוגריים בתחילת שורת הפקודה.

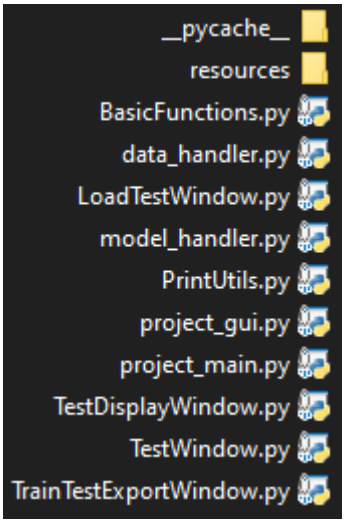
```
(test) C:\Users\BBBB>
```

נעבור להתקנת הספריות הנדרשות,
על המשתמש להוריד את הספריות הנדרשות לפי הסדר הנ"ל:

שם ספרייה	פקודת הורדה	לינק למקור
colorama	pip install colorama	https://pypi.org/project/colorama/
Tensorflow	pip install tensorflow==2.8.0	https://pypi.org/project/tensorflow/
pandas	pip install pandas	https://pypi.org/project/pandas/
protobuf	pip install protobuf== 3.14.0	https://pypi.org/project/protobuf/
PIL	pip install pillow	https://pypi.org/project/Pillow/
cv2	pip install opencv-python	https://pypi.org/project/opencv-python/
sklearn	pip install scikit-learn	https://pypi.org/project/scikit-learn/
matplotlib	pip install matplotlib	https://pypi.org/project/matplotlib/
tf_slim	pip install tf_slim	https://pypi.org/project/tf-slim/
pycocotools	pip install pycocotools	https://pypi.org/project/pycocotools/
lvis	pip install lvis	https://pypi.org/project/lvis/
tensorflow-io	pip install tensorflow_io	https://pypi.org/project/tensorflow-io/
pyyaml	pip install pyyaml	https://pypi.org/project/PyYAML/
gin	pip install gin-config	https://pypi.org/project/gin-config/
tensorflow-addons	pip install tensorflow_addons	https://pypi.org/project/tensorflow-addons/

לאחר התקנת הספריות יש להוריד את הפרויקט...

ראשית, יש להוריד את הפרויקט מחשבון ה GitHub שלי.
לאחר מכן יש לפתוח את קובץ הzip, אמורים להתקבל הקבצים הבאים:



זוהי כעת התיקייה הראשית של הפרוייקט.

הרצת התוכנית ותפקידו של כל מסך

לפני הרצת התוכנית יש להכיר את סוגי ההודעות השונות שהמשתמש יכול לקבל דרך שורת הפקודה של אנקונדה:

```
[DEBUG] This is a debug message
[INFO] This is an info message
[WARNING] This is a warning message
[ERROR] This is an error message
[CRITICAL] This is a critical error message
```

```
This is an input message
Enter:
```

נעבור להרצת הפרויקט עצמו

תחילה על המשתמש לפתוח את שורת הפקודה של אנקונדה, ה(Anaconda Prompt) ולהפעיל את הסביבה שיצר בהתקנת הפרויקט באמצעות:

```
conda activate <Name>
```

ולאחר מכן המשתמש יראה את שם הסביבה בסוגריים בתחילת שורת הפקודה.
את המילה <Name> המשתמש מחליף בשם הסביבה שיצר

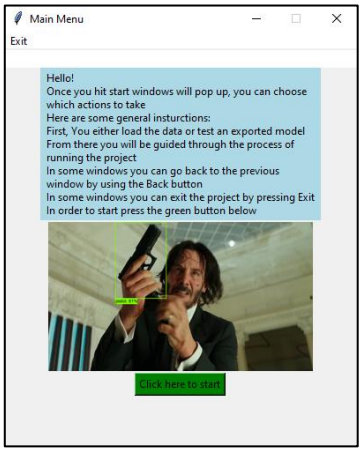
לאחר מכן המשתמש יריץ את הפרויקט באמצעות הרצת הקובץ project_main.py
כדי לעשות זאת יש לכתוב בAnaconda Prompt, python רווח ואז את הpath למיקום בו שמרתם את קובץ ה Python: project_main.py – ניתן לגרור את הקובץ ישירות לשורת הפקודה.
בסוף שלב זה שורת הפקודה אמורה להיראות כך:

```
(test) C:\Users\□□□□>python C:\dev\Project\ProjectFinal\project_main.py
```

ישר עם הרצת הפקודה, יודפס למשתמש כמה פרטים על הסביבה הנוכחית

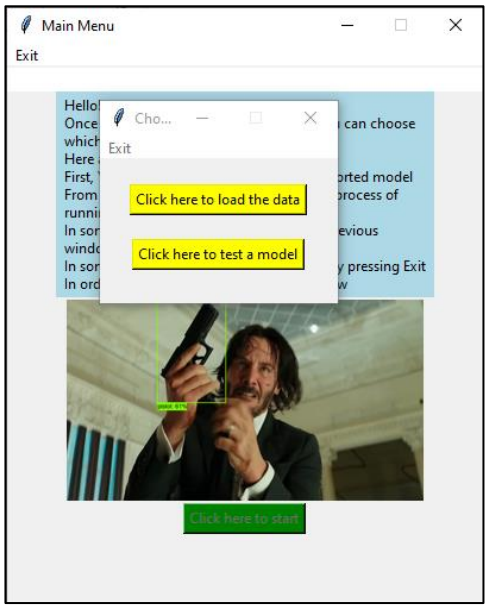
```
[INFO] Current conda environment: test
[INFO] Current python version: 3.7.13
```

וייפתח **חלון הפתיחה**:



תפריט זה מציג מידע כללי על הרצת הפרויקט.
כאשר המשתמש רוצה להתחיל את השימוש, ילחץ על הכפתור הירוק.

ייפתח **חלון load או test** :



כעת המשתמש יוכל לבחור בין לטעון את הdata הדרוש לאימון המודל או ישירות לtest של מודל שכבר אומן.

הסבר test על מודל שכבר אומן יופיע בהמשך המדריך

כעת נסביר על תהליך טעינת הdataset:

כאשר המשתמש ילחץ על הכפתור Click here to load the data **הוא יקבל הוראות דרך שורת הפקודה של האנקונדה וייפתחו לו מסכים של בחירת קבצים ותיקיות. על המשתמש למלא את ההוראות.**

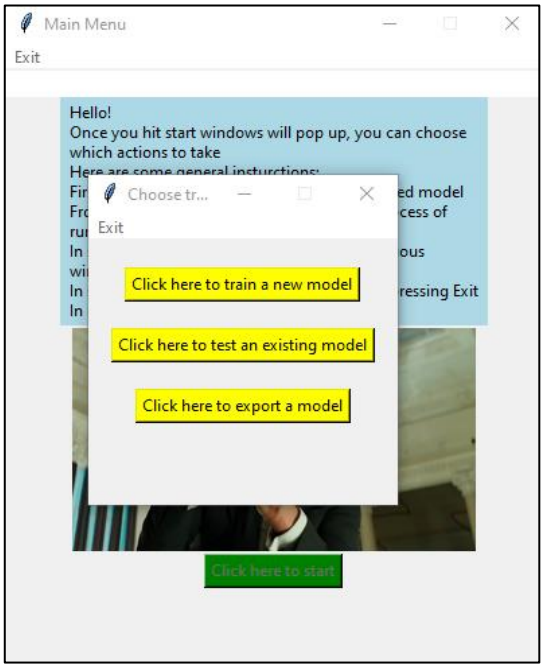
תחילה המשתמש יתבקש לבחור את קובץ הzip המכיל את הdataset מהמחשב שלו, ותיקייה שלתוכה יחולץ הdataset עצמו. לאחר הבחירה הdataset יחולץ וייבדק לתקינות של הנתונים, אם יש קובץ נתונים לא תקין הוא יימחק ויירשם בשורת הפקודה.

לאחר מכן המשתמש יתבקש לבחור תיקייה ריקה שבה יחולק הdataset לtrain ול-eval לפי יחס חלוקה של train=0.8 ו-eval=0.2.

לבסוף המשתמש יתבקש לבחור תיקייה שבה יאוחסנו הנתונים המקודדים, כלומר הנתונים שמוכנים להכנסה ולאימון המודל.

תהליך טעינת הdataset נגמר וכעת הdataset מוכן לאימון המודל.

כעת ייפתח חלון **train, test או export:**



בחלון זה המשתמש יוכל לבחור האם לאמן מודל חדש, לבדוק מודל קיים או לייצא מודל שאומן לקובץ zip.

תהליך אימון מודל חדש:

כאשר המשתמש ילחץ על הכפתור Click here to train a new model יתבקש המשתמש לבחור תיקייה ריקה שתכיל את קבצי האימון.
קובץ קונפיגורציה של מודל ייווצר אוטומטית בתיקייה והמשתמש יוכל לשנות אותה כרצונו.

בתוך קובץ הקונפיגורציה המשתמש יכול לשנות כמה Hyper Parameters. **הערה: שינוי ערכי קונפיגורציה ללא ידע קודם יכול לפגוע בתקינות המודל ולגרום לשגיאה.**

הנה כמה ערכים שהמשתמש יכול לשנות:

הערך **batch_size** – זהו כמות התמונות המאומנות על מודל בכל step.

הערך **num_steps** – זהו משך האימון של המודל על הdataset, כל step מייצג אימון אחד של המודל על batch_size תמונות. חייב להיות לפחות 5000

```
train_config {
  batch_size: 2
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.04
          total_steps: 25000
          warmup_learning_rate: 0.013333
          warmup_steps: 2000
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  fine_tune_checkpoint: "C:/dev/Project/P
  num_steps: 25000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  use_bfloat16: true
  fine_tune_checkpoint_version: V2
}
```

משתנה הbatch_size

משתנה הnum_steps

לאחר שסיים לשנות את הקובץ, ילחץ אנטר בשורת הפקודה של אנקונדה.
למשתמש יודפס כתובת אינטרנט של TensorBoard שאליה הוא יכול להתחבר כדי לראות את התקדמות המודל בזמן אמת:

```
TensorBoard 2.8.0 at http://localhost:56269/ (Press CTRL+C to quit)
```

כמו כן יודפסו למשתמש הודעות על אימון המודל והתקדמותו דרך שורת הפקודה של אנקונדה.
לא ניתן לעצור את אימון המודל באמצע.

קצת על Tensorboard:

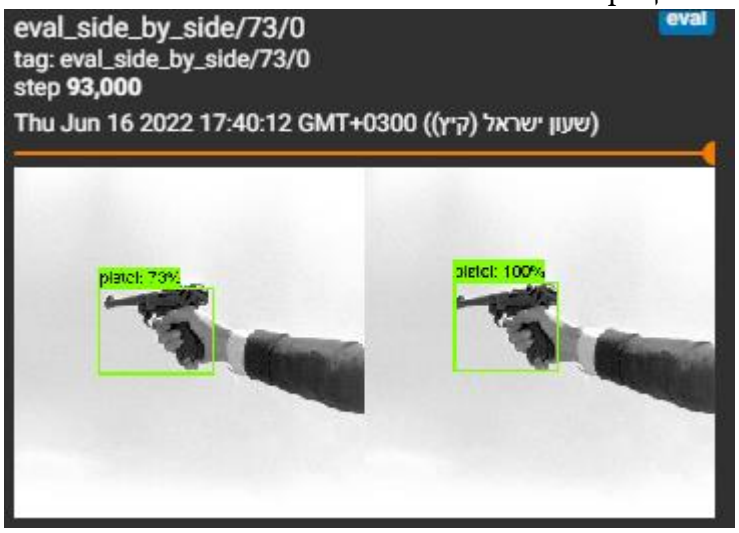
Tensorboard הוא כלי הנותן דרך להראות בצורה ויזואלית גרפים תוך כדי אימון המודל, כך ניתן לראות את התקדמות והשתפרות המודל בצורה יותר נוחה, דרך הדפדפן.

בדף האינטרנט הנפתח על ידי Tensorflow יש כמה סוגים של מידע מוצג:

תחת הכרטיסייה Scalars:
ניתן לראות גרפים המתארים את ה mAP, היחס בין כמות החיזויים הנכונים שהמודל ביצע לחלק לסכום הניחושים הכללי של המודל, כלומר גרפים אלו מראים כמה מהחיזויים של המודל היו באמת נכונים.
ניתן לראות גרפים המתארים את ה Recall, היחס בין כמות החיזויים הנכונים חלקי סכום החיזויים הנכונים ועוד חוסר חיזוי במקום שאמור להיות, כלומר גרפים אלו מראים באיזו מידה המודל מצליח למצוא את כל הקופסאות שאמורות להיות.
ניתן לראות גרפים המתארים את הLoss,
ה classification_loss הוא כמה המודל טועה בחישוב הביטחון של המודל בכל חיזוי.
ה localization_loss הוא כמה המודל טועה בחישוב גודל ומיקום החיזוי.
ה regularization_loss הוא loss המיועד למנוע התאמה יתרה (overfitting) על הנתונים עליהם המודל מתאמן, בצורה זו המודל יכול להכליל בצורה יותר טובה ולזהות גם תמונות שלא התאמן עליהן.
בנוסף ניתן לראות גרפים של ה learning_rate וכמות הsteps בשנייה.
ציר ה-X בכל אחד מהגרפים מתאר את ה-step.

תחת הכרטיסייה Images:
ניתן לראות חלק מהתמונות שעליו אומן המודל לאחר augmentation.
ניתן לראות חיזוי של המודל על תמונות הevaluation בכל 1000 steps.

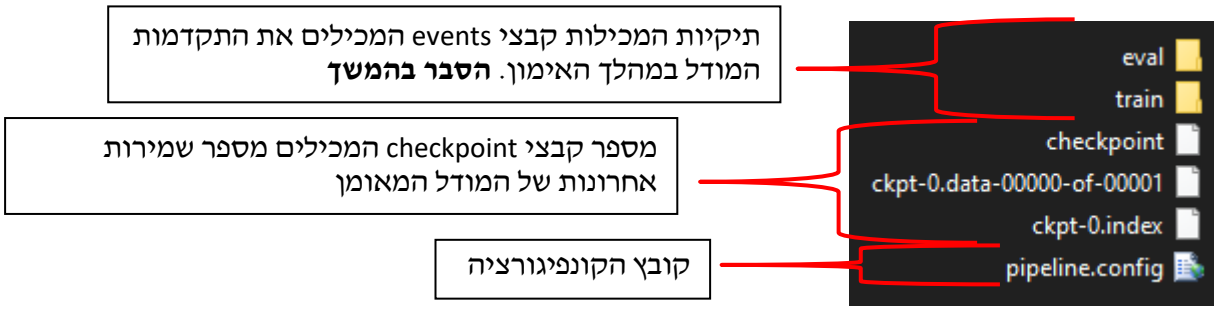
בכל אחת מהתמונות ניתן לראות את הstep שעליו נחזה התמונה ובעזרת הסליידר הכתום ניתן לזוז בין הstep.



לאחר סיום אימון המודל תודפס למשתמש הודעה ובה משך אימון המודל:

```
[INFO] Training finished! Time taken: 5.16 minutes
```

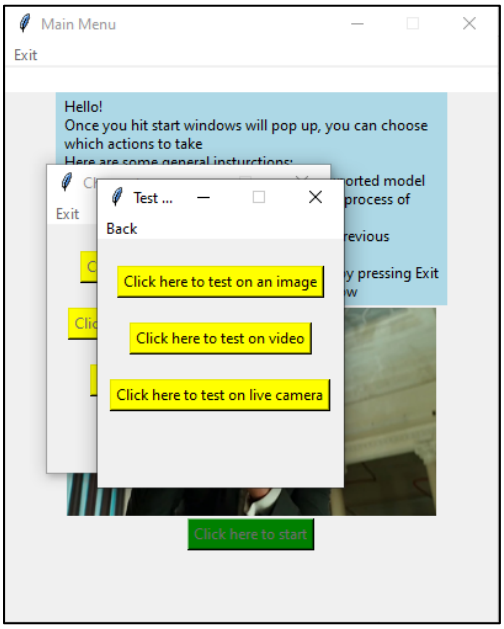
בתיקיית האימון המשתמש יכול לראות מספר קבצים:



כעת המתשמש יחזור לחלון הקודם להמשך השימוש בתכנית.

תהליך הtest של מודל מוכן:

כאשר המשתמש ילחץ על הכפתור של בדיקת מודל מוכן, יתבקש המשתמש לבחור את תיקיית הzip של המודל לאחר export. בנוסף המשתמש יתבקש לבחור תיקייה ריקה שבה יחולץ המודל.
בסיום טעינת המודל ייפתח חלון **בחירת test על תמונה, סרטון או לייב מהמצלמה:**

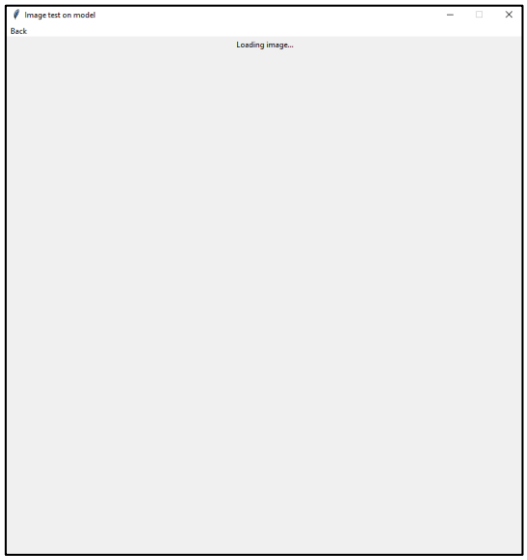


כעת למשתמש יש שלוש אפשרויות:

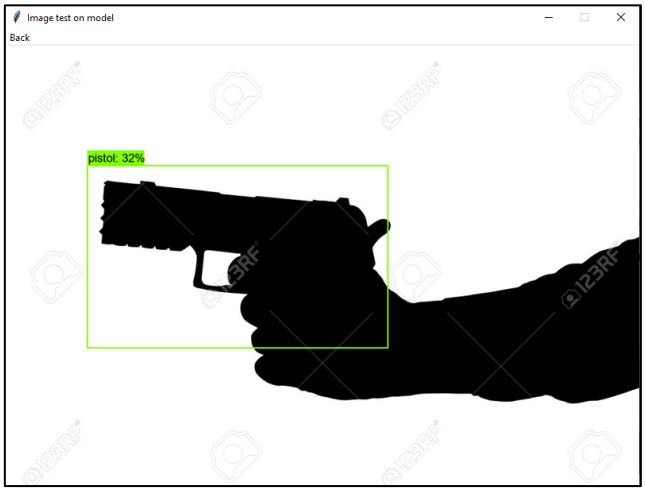
- בדיקת המודל על תמונה
- בדיקת המודל על סרטון
- בדיקת המודל על שידור חי מהמצלמה של המחשב

בדיקת המודל על תמונה:

כאשר המשתמש ילחץ על הכפתור Click here to test on an image המשתמש יתבקש לבחור תמונה מסוג jpg שעליה המודל הטעון יבצע חיזוי. תוך כדי בחירת התמונה ייפתח למשתמש חלון הצגת החיזוי שנראה כך:



לאחר שהמשתמש בחר תמונה לחיזוי, כעבור כמה שניות תוצג התמונה עם קופסאות מצוירות המייצגות את תוצאות המודל.
דוגמא לחיזוי של מודל:



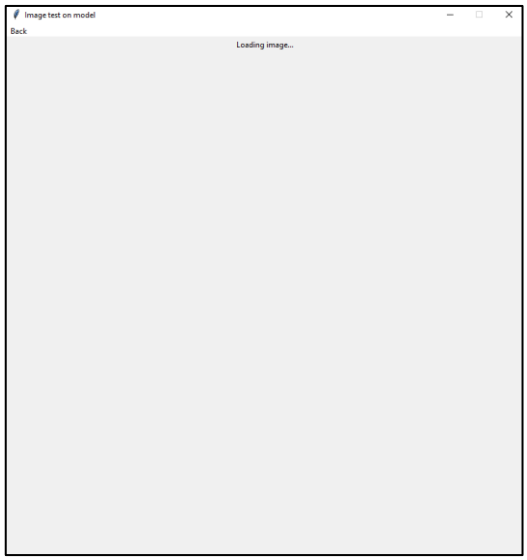
בדיקת המודל על סרטון:

כאשר המשתמש ילחץ על הכפתור Click here to test on video המשתמש יתבקש לבחור קובץ mp4 עליו המודל יבצע חיזוי. בנוסף המשתמש יתבקש לבחור קובץ mp4 חדש שאליו יישמר הסרטון עם החיזוי.
במהלך החיזוי יודפס בשורת הפקודה של האנקונדה את התקדמות החיזוי:

```
[INFO] Frame 0 out of 182
[INFO] Frame 100 out of 182
[INFO] Finished predicting on video
```

בדיקת המודל על תצלום לייב מהמצלמה:

אם למחשב מחוברת מצלמה, ניתן לבדוק את המודל על שידור חיי מהמצלמה.
כאשר המשתמש ילחץ על הכפתור Click here to test on live camera ייפתח למשתמש חלון הצגת החיזוי:

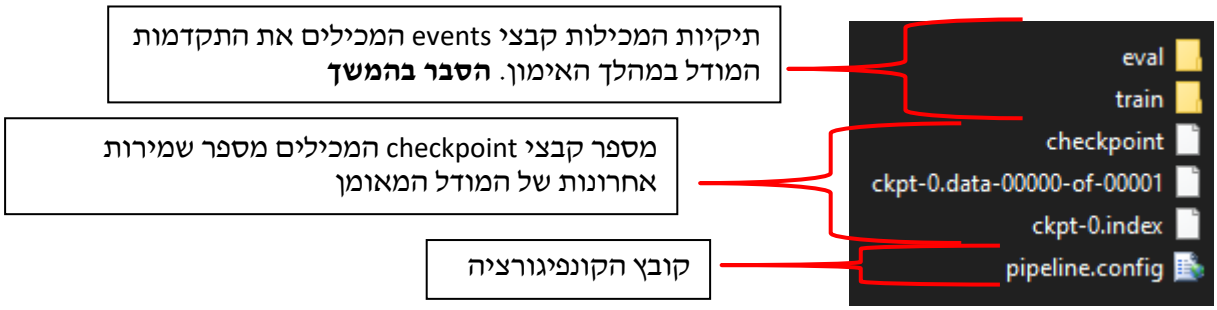


ולאחר כמה שניות יוצג הלייב מהמצלמה לאחר חיזוי של המודל.

תהליך הexport של מודל מאומן:

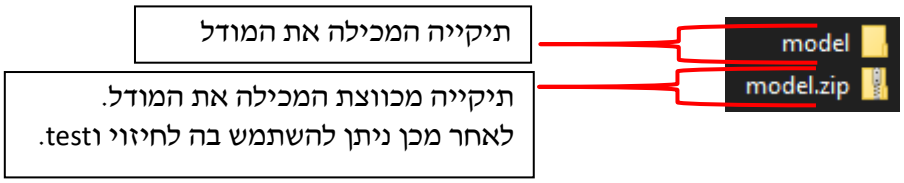
כאשר המשתמש לוחץ על הכפתור Click here to export a model יתבקש המשתמש לבחור תיקיית אימון שבה אומן מודל.

תיקיית האימון אמורה להיראות כך בקירוב:



בנוסף על המשתמש לבחור תיקייה ריקה שאליה ייוצא המודל המאומן.

לאחר כמה שניות בתוך תיקיית הייצוא יתקבלו הקבצים הבאים:



רפלקציה

העבודה על הפרויקט הייתה מאתגרת. בחרתי לעשות את הפרויקט על נושא שמעניין אותי ויאתגר אותי בו זמנית, דבר שהקל עליי במהלך עשייתו. היה לי ידע מוקדם בנושא של למידת מכונה, אך במהלך הפרויקט למדתי והתמקצעתי בנושא, במיוחד בחלק של Object Detection ועבודה עם הTensorflow Object Detection API. למדתי המון דברים חדשים, ובניהם, למדתי איך לעבוד עם סרטונים, למדתי איך ליצור ממשק למשתמש ותקשורת איתו באמצעות tkinter ולמדתי איך להשתמש בכלים של האינטרנט ולחפש מאמרים על הנושאים שלא ידעתי. כלים אלו יעזרו לי הרבה במהלך הדרך העתידית שלי בעולם המחשבים והתכנות ככלל ובעולם הלמידת מכונה בפרט.

במהלך הפרויקט נתקלתי במספר קשיים. ראשית, התקשיתי בלמצוא מאגר נתונים מספיק כדי לאמן מודל בצורה טובה, התקשיתי בלמצוא סביבת עבודה נכונה ויעילה והתקשיתי בבניית המודל. למרות זאת, המשכתי לנסות ולחקור, כדי שאצליח בפרויקט ובמשימות שהצבתי לעצמי.

המסקנות שלי מהפרויקט הן שתחום הלמידת מכונה הוא תחום קשה, אך עם זאת מעניין וחשוב מאוד. אני מעריך שהתחום רק ילך ויתפתח, וישמש אותנו יותר בעתיד. בנוסף, למדתי על עצמי שלמרות שהיה קשה ומאתגר, לא וויתרתי ובסוף גיליתי שמאוד סופקתי אחרי שמאוד התאמצתי ולבסוף הצלחתי.

לו הייתי יכול להתחיל מחדש את הפרויקט, הייתי בוחר להשקיע יותר כבר מההתחלה, ולא להעמיס על עצמי בעיקר בסוף, הייתי חוקר קצת על הנושא של למידת מכונה לפני תחילת כתיבת הקוד. בנוסף, הייתי נעזר באנשי מקצוע או אנשים שמבינים בנושא, כמו לדוגמה בת דודה שלי, שגם היא למדה את הנושא של למידת מכונה וגם מבינה בנושא.

העבודה על הפרויקט הייתה יעילה יותר אילו המחשבים בבית הספר היו יותר חזקים ובעלי GPU. במהלך השנה מצאתי את עצמי פעמים רבות מעביר קבצים וחומר הלוך ושוב בין המחשב בביתי ובין המחשב בבית הספר כדי שאוכל לאמן ולבדוק את הקוד שכתבתי. נאלצתי לעבוד על הקוד גם בבית ובבית הספר, ולאחר בדיקה של המודל בבית, להעביר הכל שוב לבית הספר כדי שאוכל להיות יעיל גם במהלך השיעורים.

לסיכום, הפרויקט היה מלווה קשיים רבים, אך אני שמח שלא וויתרתי לעצמי, ולמדתי הרבה מאוד על השנה. אני שמח שיצא לי ללמוד על הנושא ולחקור עליו יותר.

ביבליוגרפיה

Tensorflow Object Detection API. Retrieved from:
https://github.com/tensorflow/models/tree/master/research/object_detection

Baskaran, V. (2020) *Momentum optimizer !!!*. Retrieved from:
https://medium.com/@vinodhb95/momentum-optimizer-6023aa445e18

Correa, S. (2019) *Cosine Learning rate decay*. Retrieved from:
https://scorrea92.medium.com/cosine-learning-rate-decay-e8b50aa455b

Convolutional neural network, Wikipedia. Retrieved from:
https://en.wikipedia.org/wiki/Convolutional_neural_network#Convolutional_layers

Hui, J. *Understanding Feature Pyramid Networks for object detection (FPN)*. Retrieved from:
https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

University of Granada, *Weapons detection for security and video surveillance*. Dataset. Retrieved from: https://sci2s.ugr.es/weapons-detection#RP

Tkinter filedialog. Retrieved from:
https://docs.python.org/3/library/dialog.html

נספחים

Tensorflow Object Detection API

במהלך תהליך המחקר המקדים לפרויקט הבנתי שאני אצטרך לבצע Object Detection, כלומר זיהוי אובייקטים בתמונות (סרטונים הם רצף של תמונות).
בהמשך המחקר התברר שObject Detection הוא אחד מהנושאים המתקדמים והמסובכים ביותר בלמידת מכונה וComputer Vision, ולרוב כדי לאמן מודל מוצלח צריך כמות נתונים של עשרות אלפי תמונות וידע של שנים בנושא של למידת מכונה, שני דברים שאין לנו כתלמידים במסגרת ההתמחות בתיכון.
כדי לאפשר לאנשים כמוני גם להתנסות בעולם של Object Detection, היוצרים של Tensorflow פיתחו framework שבנוי על Tensorflow עצמו, בעל קוד פתוח וקריא לכולם ולו הם קראו Tensorflow Object Detection API.

בעזרת הכלי הזה ניתן לבנות, לאמן ולהשתמש במודל המבצע Object Detection גם על מבני נתונים קטנים ובלי ידע נרחב בנושא.
התהליך שבו הTensorflow Object Detection API משתמש כדי לגרום גם לdataset קטן ליצור מודל הוא בעזרת שיטה שנקראת fine-tuning. כלומר לקחת מודל שאומן על dataset אחר לגמרי בעל מספר רב של אובייקטים ותמונות (לפעמים גם אלפי אובייקטים ומיליוני תמונות), וכבר מומחה בלזהות ולהפריד אובייקטים מתמונות, ורק לאמן אותו בצורה מצומצמת כדי שידע לזהות אובייקטים לפי הdataset החדש.