בס"ד

**פרויקט מובילאי**

במסגרת ההכשרה עשינו פרויקט בשיתוף צוות מחברת מובילאי – מנטורים שליוו אותנו בתהליך, הדריכו, כיוונו, ענו לשאלות ובעצם עבדנו בצוות כשהמנטורים מהחברה משמשים כר"צ. מטרת הפרויקט היתה לזהות רמזורים בקטע וידיאו נתון ולחשב את המרחק אליהם.

את הפרויקט ביצענו בארבעה שלבים:

**שלב 1: זיהוי נקודות אור בתמונה**

( LPF- low path filter מטשטש ניגודים חדים בתמונה ע"י חישוב סכום מכפלת הפיקסלים בקרנל בפיקסלים שמסביב הsource pixel ושמה את התוצאה ב destination pixel.

HPF – high path filter גורם לתמונה להיראות חדה יותר. פילטרים אלה מדגישים פרטים עדינים בתמונה - בדיוק ההפך מהפילטר LPF – עובד כמו הLPF רק עם קרנל שונה.

כל קרנל אוהב את עצמו ולכן היינו רוצים ליצור קרנל בצורת עיגול אור אבל הוא לא נותן תוצאות כ"כ טובות כי יש נקודות עם הילה שמשבשת את התוצאות...(

עבדנו על מאגר תמונות שקיבלנו ממובילאי, ע"מ לזהות את נקודות האור בתמונה השתמשנו בconvolution יצרנו קרנל והרצנו אותו על התמונה. השתמשנו בscipy בשביל ה convolve ובnumpy בשביל הקרנל.

**שלב 2:** **אימון דאטהסט, יצירת רשת והרצתה**

**יצירת הדאטהסט:**

קיבלנו מאגר תמונות ממובילאי כך שכל תמונה קיבלנו בצורת תמונה דו מימדית ואת אותה תמונה בשלש מימדים – תמונה רגילה בצבע רגיל, עברנו על התמונות במקביל, ריפדנו במסגרת כדי למנוע מקרה קצה של גלישה מהתמונה במקרב של חיתוך סביב נקודה בקצה התמונה, בדקנו בתמונה הדו מימדית אלו אינדקסים הם רמזורים לפי הצבע, מתוכם בחרנו אידקסים רנדומלית וחתכנו סביב אותם אינדקסים בתמונה המקורית והופנו תגית מתאימה למערך התגיות. אותו דבר עשינו עבור אינדקסים שאינן רמזורים ותייגנו אותן כלא רמזור.

את התוצאה שמרנו בקבצים בינארים מכיון שהוא חוסך מקום ובעל מהירות גישה גבוהה.

**augmentation** - אחת הדרכים שבאמצעותה ניתן להרחיב את מאגר המידע ממנו יכולה ללמוד הרשת הנוירונית היא באמצעות גישה של augmentation, ולשמחתנו Keras מציעה את הפונקציה ImageDataGenerator שיכולה להפיק תמונות מגוונות ושונות זו מזו על בסיס תמונה מקורית שנספק לה. המגוון בא לידי ביטוי בשינויים ויזואליים דוגמת סיבוב, צביעה, בהירות וחדות. כך נוכל באמצעות פקודה אחת פשוטה להגדיל את כמות התמונות היכולות לשמש לאימון הרשת הנוירונית בסדרי גודל.

**אימון הרשת:**

מושגים:  
Deep learning (למידה עמוקה) הוא תת תחום של Machine learning העושה שימוש ברשתות נוירונים מלאכותיות לשם ביצוע משימות. בסיסו של תחום זה טמון בשאיפה לחקות את הדרך בה המוח האנושי פועל ולרתום את היעילות של מבנה הנוירונים לשם התמודדות עם אתגרים חישוביים מורכבים

המוח בנוי כרשת של נוירונים ( שמקושרים ביניהם. כל 11 רבים ) 10 נוירון הוא תא בודד וטיפש, אך לרשת כולה מתפתחת יכולת מדהימה של פתרון בעיות מורכבות. השאלה שמדענים עסקו בה היא: האם נוכל להציע מקבילה דומה מעולם ההנדסה בעלת יכולות דומות? המוח בנוי כרשת של נוירונים ( שמקושרים ביניהם. כל 11 רבים ) 10 נוירון הוא תא בודד וטיפש, אך לרשת כולה מתפתחת יכולת מדהימה של פתרון בעיות מורכבות שנות השישים בשנות השישים הוצע מענה לשאלה זו, כשאבן הבניין היא הפרספטרון )אנו נקרא לזה גם "נוירון"( המחשבה היתה שרשת של אלמנטים כאלו שמחוברים זה לזה )Forward-Feed )תוכל ליצור מערכת מורכבת ועשירה ביכולות.

אלגוריתמים של Machine learning מסווגים לעתים קרובות לפי 2 קטגוריות: מפוקחים (supervised) ולא מפוקחים (unsupervised).

**Supervised** – פותרת בעיות ע"ב מאגר דוגמאות "פתורות" -מתויגות

**Unsupervised** - לא צריכים לעבור תהליך למידה מבוקר. אלגוריתמים אלה מאתרים דפוסים נסתרים בנתוני הקלט.למעשה, הם משתמשים בגישה איטרטיבית הקרויה למידה עמוקה לסקירת נתונים וכך מגיעים למסקנות.

מודל לוקה ב-**overfitting** אם הוא מתאים יותר לסט הדוגמאות שעליהם הוא אומן (training set) מאשר עם דוגמאות אחרות. תת התאמה (**Underfitting**) לעומת זאת מתרחשת כאשר המודל הסטטיסטי פשוט מדי מכדי לייצג כראוי את המבנה הבסיסי של הנתונים

קיימות 5 דרכים עיקריות להתמודד עם בעיית ה- overfitting: כמו: הפיכת המודל לפשוט יותר, הוספת דוגמאות אימון לצורך הגדלת המגוון.

**Training set** - הדאטה שעלי מאמנים את הרשת.

**Validation set** - דאטה שעליו מריצים את הרשת לאחר האימון לקבל אינדיקציה על מידת יעילותה. משמשת למפתחים ללמוד ממנה מה צריך לשפר.

**Test set** - דאטה שעליו מריצים את הרשת לאחר האימון לקבל אינדיקציה על מידת יעילותה.

גרעין: בגרעין עיבוד תמונה הוא מטריצת קונבולציה או מסכות בהן ניתן להשתמש לטשטוש, השחזה, הבלטה, זיהוי קצה ועוד על ידי ביצוע התמרה בין גרעין לתמונה.

אנחנו השתמשנו ברשת CNN המשמשת בד"כ לעיבוד תמונות. הרשת שלנו מורכבת בין היתר מהשכבות הבאות:

**Keras Conv2D** - שכבת קונוולוציה דו-ממדית, שכבה זו יוצרת קרנל קונבולציה שהוא רוח עם קלט שכבות המסייע לייצור טנסור של תפוקות.

**BatchNormalization** - טכניקה שנועדה לתקן באופן אוטומטי את הקלטים לשכבה ברשת נוירונים של למידה עמוקה. לאחר היישום, ל Batch normalization יש השפעה של האצה דרמטית של תהליך האימון של רשת עצבית, ובמקרים מסוימים משפרת את ביצועי המודל באמצעות אפקט רגולציה (-הסדר) מוסתר.

**Activation('relu')** - במילים פשוטות, פונקציית activation  היא פונקציה שמתווספת לרשת עצבית מלאכותית במטרה לעזור לרשת ללמוד דפוסים מורכבים בנתונים. כאשר משווים עם מודל מבוסס נוירון שנמצא במוחנו, פונקציית ההפעלה היא בסופו של דבר מחליטה מה יש לפטר לנוירון הבא.

ברשת עצבית, פונקציית ה activation  אחראית על הפיכת הקלט המשוקלל המסוכם מהצומת ל activation  של הנוד או הפלט עבור קלט זה.

**rectified linear activation function** or **ReLU** בקיצור הוא piecewise ליניארי הפונקציה כי תפוקת רצון הקלט ישירות אם היא חיובית, אחרת, זה יהיה אפס תפוקה. זה הפך להיות פונקציית ההפעלה המוגדרת כברירת מחדל עבור סוגים רבים של רשתות עצביות מכיוון שמודל שמשתמש בה קל יותר להכשרה ולעתים קרובות משיג ביצועים טובים יותר.

reLU מייצג יחידה ליניארית מתוקנת, והיא סוג של פונקציית הפעלה. מתמטית, זה מוגדר כ- y = מקסימום (0, x). מבחינה ויזואלית נראה כך: ReLU היא פונקציית ההפעלה הנפוצה ביותר ברשתות עצביות, במיוחד ברשתות CNN.

**Dense layer** - מזינה את כל הפלטים מהשכבה הקודמת לכל הנוירונים שלה, כל נוירון מספק פלט אחד לשכבה הבאה. זו השכבה הבסיסית ביותר ברשתות עצביות. Dense (10) יש עשרה נוירונים.

**MaxPooling2D** הוא פעולת איגום המחשב את הערך המרבי, או הגדול ביותר, בכל תיקון של כל מפת תכונות. ... ניתן להוסיף את פעולת המאגר המקסימלית לדוגמא שעבדה על ידי הוספת שכבת MaxPooling2D הניתנת על ידי ממשק ה- API של Keras.

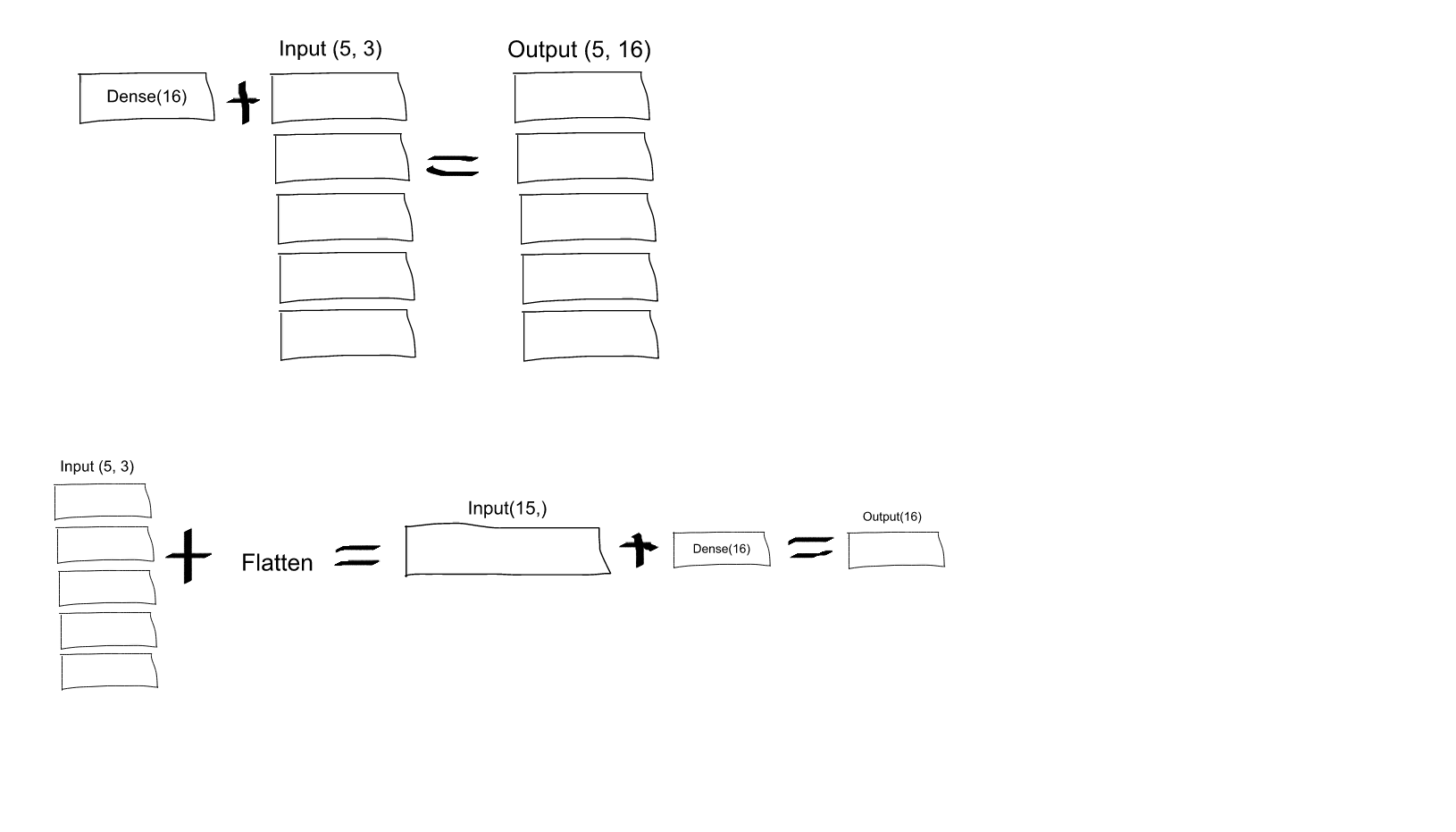
פעולת איגום מרבית לנתונים מרחביים דו-ממדיים. מדגם את ייצוג הקלט על ידי לקיחת הערך המרבי מעל החלון המוגדר על ידי pool\_size עבור כל מימד לאורך ציר התכונות. החלון מוסט על ידי צעדים בכל מימד.

**Flatten()** - תפקידה של שכבת השטוח בקרס הוא סופר פשוט: פעולה שטוחה על טנסור מעצבת מחדש את הטנסור כך שתהיה לו הצורה השווה למספר האלמנטים הכלולים בטנסור שאינו כולל מידת הbatch.

**Dense**

Dense(16, input\_shape=(5,3))

יביא Denseלרשת עם 3 כניסות ו -16 יציאות אשר יושמו באופן עצמאי לכל אחד מ -5 הצעדים. לכן, אם D(x)הופך וקטור תלת מימדי לווקטור 16-ד, מה שתקבל כפלט מהשכבה שלך יהיה רצף של וקטורים: [D(x[0,:]), D(x[1,:]),..., D(x[4,:])]עם צורה (5, 16). על מנת לקבל את ההתנהגות שתציין, תחילה תוכל Flattenלהזין את הקלט שלך לווקטור 15-ד ואז להחיל Dense:



**שלב שלישי: חישוב המרחק מהרכב למכונית**

**שלב רביעי: אינטגרציה, חיבור כל החלקים למערכת שלמה שעובדת**

יוצרים קונטרולר, מפעילים פונקציה controller.proccess בתוכה יוצרים את הקלאסים Data ProcessData שמחזיק את הקישור לpls\_file ו TFLManager.

עוברים בלולאה על כל תמונה – שולחים אותה ל process\_dataשקוראת את התמונות בפלייליסט וקובץ פיקל, מאתחלת שדות של קלאס Data, עוברת בלולאה על כל תמונה – לכל תמונה יוצרת FrameContainer, מאתחלת את השדות שלו ומחזירה את הID של התמונה הנוכחית (ע"י דקורטור yeald כך שבאיטרציה הבאה בלולאה בקונטרולר הוא יחזור לאיטרציה הבאה בלולאה הנוכחית – התמונה הבאה) על התמונה שחזרה מפעילים את

manager.run(data, frame) שמקבלת גם את הדאטה שאתחלנו.

הפונקציה יוצרת Adapter, מוצאת נקודות חשודות לרמזורים ( ע"י הפונקציה מחלק אחד) ושולחת ל: adapter.adapt\_part\_1\_to\_part\_2(find\_tfl\_lights(image)) שהיא מחזירה מערך של נקודות לאחר סינון נקודות סמוכות ומערך auxiliary של סימון אדום או ירוק לכל נקודה. מרפדת את התמונה וחותכת מסביב לנקודות, מיבאת את המודל שיצרנו בחלק השני – הרשת המאומנת ומפעילה אותה על התמונות החתוכות. מפעילה את adapt\_part\_2\_to\_part\_3(cropped\_imgs\_predicts, candidates)

הפונקציה עוברת על כל האינדקסים שחזרו מהמודל באחוזים גבוהים לזיהוי רמזור ושומרת בדיקט לכל נקודה כמה פעמים היא מופיעה, -**לא כ"כ הבנתי מה שומרים לכל נקודה...** לבסוף הפונקציה מחזירה את הנקודות עם המספר הגבוה שהן בעצם הרמזורים. לאחר מכן הפונקציה עוברת על כל נקודה ומחשבת את המרחק ממנה למכונית (ע"י הפונקציה מהשלב השלישי) ומציגה את התמונה הקודמת והנוכחית כשעליה מסומנים הרמזורים והמרחק אליהם ע"י הפונקציה .Output.visualize

