פרויקט גמר חמש יחידות לימוד- למידת מכונה

Deep Learning

נושא הפרויקט:

זיהוי סרטן השד לפי תמונות היסטופתולוגיה

Breast cancer detection with histopathology images

שם בית ספר: מקיף י'א ראשונים

שם התלמיד: ליהי שפיר

תעודת זהות: 214398927

שם המנחה: דינה קראוס

תאריך הגשה: 16/6/2022

תוכן

מבוא

השנה נושא ההתמחות שלמדנו בבית הספר הינו Deep Learning- זוהי מחלקה של שיטות למידת מכונה המבוססות על רשתות עצביות מלאכותיות שמאפשרת למידת ייצוגים. הלמידה עצמה יכולה להיות מונחית, מונחית למחצה או ללא הנחיה.

הלמידה העמוקה נעשית באמצעות Deep Learning, שזהו סוג של Machine learning, למידת מכונה. למידה זו מאפשרת למפתחים ׳להכשיר׳ את המכונות (אפליקציות, מערכות, מחשבים וכד׳) לנתח תהליכי לימוד כך שיהיו דומים ככל האפשר ליכולות החשיבה והניתוח של מוח אנושי.

מוח האדם בנוי מרשת נוירונים בהם נקלט המידע. שיתוף הפעולה בין אלפי הנוירונים המורכב מפעולה קטנה של כל אחד מהם, מאפשר למוח לעשות פעולה גדולה ומורכבת. ובעזרת התקשורת שבין כל תא עצב לאילו שבקרבתו נוצרת היכולת ללמוד דברים חדשים גם בהתבסס על מידע קיים ולזכור את מה שלמדנו. על רעיון זה מתבססים הפרויקטים בהתמחות זו בכלל וכן הפרויקט שלי בפרט. רשת נוירונים של "למידה עמוקה" עובדת באופן דומה, שכן היא מורכבת ממשקלים ונוירונים שכל אחד מהם מבצע פעולה מתמטית פשוטה, ובסופו של דבר התוכנה מסוגלת ללמוד באופן דומה למוח האנושי. למשל בפרויקט שלי יש שימוש ברשת קונבולוציה (באנגלית: CNN – Convolutional Neural Network) היא סוג של רשת נוירונים המשתמשת בפעולת הקונבלוציה במקום בכפל מטריצות כללי לפחות באחת מהשכבות שלה. סוג זה של רשת נוירונים משמש בעיקר לעיבוד תמונה וראיה ממוחשבת, אך יש לו שימושים גם במערכות המלצה, עיבוד שפה טבעית וממשק מוח-מחשב.

בפרויקט שלי עסקתי בנושא למידת מכונה (באנגלית: Machine Learning). זהו תת-תחום במדעי המחשב ובבינה מלאכותית המשיק לתחומי הסטטיסטיקה והאופטימיזציה. התחום עוסק בפיתוח אלגוריתמים המיועדים לאפשר למחשב ללמוד מתוך דוגמאות, ופועל במגוון משימות חישוביות בהן התכנות הקלאסי אינו אפשרי.

בפרויקט זה בחרתי לבנות תוכנה שמטרתה לזהות את סרטן השד באמצעות תמונות היסטופתולוגיות. המודל שבניתי לומד באמצעות תמונות מסווגות כיצד לזהות היכן יש והיכן אין סרטן, הפרויקט מתבסס על למידה מונחית (supervised learning). כל דוגמה מגיעה עם תווית סיווג. מטרת האלגוריתם היא לחזות את הסיווג של דוגמאות חדשות שאותן לא פגש בתהליך הלמידה. אימון של רשת עצבית מלאכותית ("רשת נוירונים") מסתמך על אלגוריתמים מסוג זה.

בחרתי בנושא זה משלל הנושאים שהיו פתוחים בפני משום שנושא זה עורר את סקרנותי וגרם לי לרצות לדעת עוד על המקור שלו ואופן טיפולו.

סרטן השד הוא הסרטן הנפוץ בנשים בישראל ובעולם – אחת מכל 8 צפויה לחלות בו במהלך חייה. סרטן השד הוא גידול ממאיר שמקורו ברקמת השד. זהו הגידול הממאיר השכיח ביותר בקרב נשים. ישנם כמה סוגים של סרטן שד, אולם בכל הסוגים ישנה חשיבות מכרעת לאבחון מוקדם, שכן אבחון זה יכול להציל חיים.

קהל היעד שהפרויקט שלי פונה אליו הוא נשים וגברים כאחד, אם כי סרטן השד נפוץ פי 100 בקרב נשים מאשר בקרב גברים. בנוסף, הפרויקט שלי פונה לכל שכבת גיל- על פי נתוני משרד הבריאות, כ- 74% מהחולות בסרטן השד מאובחנות בגילאי 50 ומעלה, כ- 23% מהחולות מאובחנות בגיל 40-49, ורק כ- 3% מהחולות היו בנות פחות מ- 40 שנה בעת האבחון. מחלה קודמת - נשים שחלו בסרטן השד והבריאו, נמצאות בקבוצת הסיכון הגבוהה ביותר לחלות בסרטן שד בצד השני. בנוסף, הגיל הוא גורם סיכון חשוב בהתפתחות סרטן

שד אצל גברים. רוב המקרים מאובחנים בגילאים 60 עד 80 כאשר הגיל הממוצע הוא 65. רמת הסיכון עולה עם התקדמות הגיל.

לפני שניגשתי להתחלת הפרוקיט קראתי על המצב בשוק היום והבנתי שכדי לאבחן את סרטן השד יש צורך בכמה שלבים שגרתיים.

בדיקות שיגרה לגילוי מוקדם מצילות חיים:

- בדיקת רופא/ה מומחה פעם בשנה - מומלץ להיבדק על-ידי רופא/ה המתמחה בבדיקת שד ידנית (כירורג/ית שד).
- בדיקת ממוגרפיה - בדיקה זו עשויה להציל את חייך ואת שדייך! זוהי ללא ספק הבדיקה החשובה ביותר מבין כל הבדיקות המומלצות.
הבדיקה מומלצת לביצוע אחת לשנה, לכל אישה מעל גיל 50, שלא נמצאת בקבוצת סיכון. בדיקת ממוגרפיה היא בדיקת רנטגן פשוטה בעוצמה נמוכה, האורכת מספר דקות, ומטרתה לגלות סרטן שד בשלב מוקדם, בו יש שינויים ממאירים התחלתיים. אמינות הבדיקה היא למעלה מ-90%, בהתאם לגיל הנבדקת.
- בדיקת אולטרה סאונד (U.S.) -כהשלמה לממוגרפיה - סריקה על-קולית של השד אינה הכרחית, אולם היא מהווה בדיקה משלימה לבדיקת הממוגרפיה. בדיקה מתאימה בעיקר לנשים צעירות, או לנשים שהיה קושי בפענוח בדיקת הממוגרפיה שלהן.
- בדיקת הדמיה של השד באמצעות תהודה מגנטית - (MRI) הבדיקה מומלצת לנשים צעירות נשאיות של הגנים BRCA1 ו BRCA2,-לנשים בעלות שד דחוס, לנשים עם סיפור סרטן שד משפחתי, או לפי השתייכות לקבוצות הסיכון למחלה.

לעומת הבדיקות השגרתיות המוצגות לעיל, הפרויקט שלי מציע דרך חדשנית שתייעל את תהליך הבדיקה ותחסוך זמן יקר הן לרופאים והן למטופלים. במקום לבוא לבדיקה שגרתית אצל הרופא המטפל, המטופל יוכל לעבור מספר סריקות(תמונות היסטוגרפיות) שישלחו לבדיקה מיידית ותוך זמן קצר המטופל יקבל תשובה לגבי המשך טיפולו ואם בכלל יש צורך.

האתגרים המרכזיים שהיוו קושי גדול מבחינתי היו חוסר הוודאות בהכנת הפרויקט, במהלך כתיבת הפרויקט הוחלפו לנו מספר מורים דבר אשר היווה קושי גדול בלימוד החומר והבנתו. בעיה זאת הובילה אותנו לתהליך של לימוד עצמי ובקשת עזרה מחברי לכיתה, ניסינו יחדיו להתגבר על קשיים במהלך הדרך ולהבין את החומר הנלמד בצורה הטובה ביותר.

קושי נוסף שאירע במהלך הכנת הפרויקט הוא קריאת החומר ולימודו, מכיוון שנושא זה נחשב די חדש בתחום התכנות והמחשבים רוב החומר עליו כתוב באנגלית, ולכן בנוסף ללימוד העצמי היה צורך גם בהבנת המאמרים והחומר שאנחנו קוראים באנגלית דבר אשר העשיר את אוצר המילים שלנו ולימד אותנו למצוא ולהתמודד עם חומר חדש בצורה טובה יותר ומהירה יותר.

מבנה/ ארכיטקטורה של הפרויקט

<u>שלב איסוף הכנה וניתוח הנתונים (data analyze and prepare, Collect):</u>
ראשית חיפשתי מאגרי נתונים שיכילו מספיק תמונות כדי שהפרויקט יעבוד בצורה היעילה ביותר. לאחר חיפושים מצאתי באתר קאגל (Kaggle) מאגר נתונים של תמונות היסטוגרפיות בהן 198,738(מתויגות כ- 0) תמונות אינן עם סרטן השד ו- 78,786 (מתויגות כ- 1) עם סרטן השד, זאת אומרת שסך הכל יש לי 277,524 תמונות בגודל [50,50,3]. מאגר הנתונים הזה יוצר ממאגר נתונים מקורי שכלל 162 תמונות שקופיות שלמות של דגימות סרטן השד שנסרקו ומשם חולצו 277,524 תמונות בגודל [50,50,3].

בנוסף, מאגר התמונות בנוי מהרבה נתיבים של תעודות זהות כאשר בכל נתיב כזה יש לי שתי תיקיות 0,1 שמכילות מספר תמונות עם סרטן ובלי סרטן.

לכל תמונה יש נתיב תמונה שנראה כך:"10253idx5x1351y1101class0.png ".

10253idx5 - מציין מספר זהות של בן אדם.

x1351- מציין מיקום ציר x של איפה שהתמונה נלקחה.

y1101- מציין מיקום ציר y של איפה שהתמונה נלקחה.

class0 – מציין שאין סרטן.

class1 – מציין שיש סרטן.

לאחר שהבנתי כיצד נראה מבנה הנתונים והמידע בתוכו הייתי צריכה להכין את המידע לאימון. לשם כך הייתי צריכה לבדוק את תקינות המידע, את תקינות התמונות ואת גודל התמונות. ראשית הייתי צריכה להוריד את קובץ הזיפ שהכיל את מאגר התמונות ולחלץ משם את כל התמונות- פעולה שנקראת extract_data עושה זאת. לאחר מכן, בדקתי את תקינות התמונות באמצעות עוד פעולה שמוחקת את התמונות הפגומות- היא שולחת לפעולה שבודקת את תקינות התמונה בכך שהיא מורידה ומנרמלת את התמונה (\255) , בנוסף היא בודקת שהתמונה בגודל המתאים [50,50,3], אם אחת מבדיקות אלו החזירה אמת הפעולה מדפיסה את הטעות, מדפיסה את נתיב התמונה, ומוחקת אותה.

לאחר שסיימנו את הבדיקות של תקינות התמונה אנו עוברים לפיצול מאגר הנתונים ל train, test, validation. כדי שהפרויקט יעבוד בצורה הטובה ביותר ולא יביא לנו תוצאות שגויות נרצה שהפרויקט יתאמן על אותו מספר תמונות בהן קיים סרטן לעומת שלא קיים סרטן, ולכן לפני הפיצול אנחנו בונים שני מערכים של 0= אין סרטן ו- 1=יש סרטן שיכילו אותו מספר תמונות. לאחר שיש לנו את שני המערכים עם אותו מספר התמונות נרצה לפצל את מאגר התמונות כאשר התמונות לtrain יהוו 0.7 מכלל התמונות, התמונות לvalidation יהוו 0.1 מכלל התמונות, והתמונות לtest יהוו 0.2 מכלל התמונות.

החילוק יתבצע בשלוש תיקיות חדשות שיכילו את התמונות שנרצה שהפרויקט יעבוד איתן.

לפני אימון המודל יש צורך להוריד את התמונות לאימון וזה נעשה בפעולה נפרדת אשר מורידה את התמונות ומנרמלת אותן (פיקסלים) מתחום של [0,255] לתחום של [0,1]. נרמול זה מאפשר למודל שלנו להגיע למשקלים טובים יותר ובסופו של דבר לתוצאות ומודל מדויק יותר.

שלב בנייה ואימון המודל (Build and train deep learning model) :

תיאור גרפי של המודל עלי בוצע האימון:

```python
def load_default_model(self) -> None :
    """
    the default model if the user does not want to use
    the pretrained model

    """

    self.model = Sequential()
    self.model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu',
                          input_shape = (50, 50, 3)))
    self.model.add(BatchNormalization())
    self.model.add(MaxPool2D(pool_size=(3, 3)))
    self.model.add(Dropout(0.25))
    self.model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Conv2D(32, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Flatten())
    self.model.add(Dense(2, activation='softmax'))
    adam = keras.optimizers.Adam(learning_rate=0.0001)
    self.model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
    self.model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 50, 50, 32)	896
batch_normalization (BatchNormalization)	(None, 50, 50, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 16)	4624
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 16)	0
conv2d_2 (Conv2D)	(None, 8, 8, 16)	2320
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 16)	0
conv2d_3 (Conv2D)	(None, 4, 4, 32)	4640
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 32)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 2)	258

<u>הסבר על סוגי השכבות השונים ברשת:</u>
מהי בעצם רשת נוירונים?

רשת נוירונים היא מודל מתמטי חישובי, שפותח בהשראת תהליכים מוחיים או קוגניטיביים המתרחשים ברשת עצבית טבעית ומשמש במסגרת למידת מכונה. רשת מסוג זה מכילה בדרך כלל מספר רב של יחידות מידע (קלט ופלט) המקושרות זו לזו, קשרים שלעיתים קרובות עוברים דרך יחידות מידע "חבויות" . צורת הקישור בין היחידות, המכילה מידע על חוזק הקשר, מדמה את אופן חיבור הנוירונים במוח. השימוש ברשתות עצביות מלאכותיות נפוץ בעיקר במדעים קוגניטיביים, ובמערכות תוכנה שונות - בהן: מערכות רבות של אינטליגנציה מלאכותית המבצעות משימות מגוונות - זיהוי תווים, זיהוי פנים, זיהוי כתב יד, חיזוי שוק ההון, מערכת זיהוי דיבור, זיהוי תמונה, ניתוח טקסט וכ'ו.

רשת נוירונים מאופיינת על ידי:

חיבורים - אופן החיבור בין הנוירונים ברשת.

משקלים - השיטה הקובעת את משקלי החיבורים בין הנוירונים.

פונקציית האקטיבציה, העשויה להיות שונה בכל שכבה (פונקציה לא ליניארית, לרוב logsig או sigmoid, relu).

ברשת ישנם שלושה סוגי שכבות:

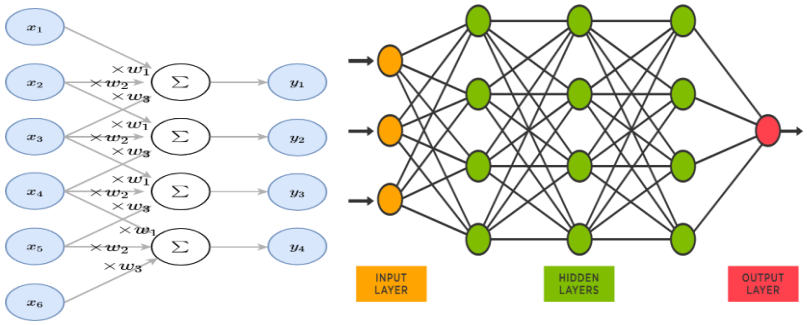
שכבת כניסה (Input Layer) - לכל תא בשכבה זו כניסה אחת. מספר התאים: כמספר המאפיינים (Features). זוהי השכבה הראשונה היא מקבלת את הdata ומשם להמשך הרשת.

שכבות חבויות (Hidden Layers) - לכל תא בשכבה זו מספר כניסות, כמספר תאי הכניסה (Fully Connected). מספר התאים: כמספר התאים בשכבת הכניסה בתוספת תא אחד או שניים. מספר השכבות: מ־0 ועד לאינסוף. בכל השכבות החבויות בדרך כלל אותו מספר תאים. מספר השכבות ומספר התאים מגדירים את גודל הרשת. יש לבחור רשת גדולה מספיק, אך לא גדולה מדי. רשתות רב־שכבתיות אופייניות הן בעלות שכבה חבויה אחת או שתיים, ומספר הנוירונים בשכבות הנסתרות במקרים רבים אינו עולה על 10. בעיות זיהוי תמונה בדרך כלל מאופיינות עם מספר שכבות גבוה מאוד (מאות שכבות). כוונון מספר הנוירונים ומבנה הרשת יעשה לרוב אמפירית (משוואה מתמטית שחוזה תוצאות שנבחנו, אבל אין לה בסיס תאורטי שיכול להסביר מדוע היא עובדת), תוך שימוש בדוגמאות ולידציה (Validation) או באמצעות אימות־צולב (Cross Validation).

(output layer)

שכבת יציאה – לכל תא בשכבה זו מספר כניסות, כמספר תאי השכבה המוסתרת .

שכבה זו מחזיקה בתוצאות הסופיות לאחר העבודה שנעשתה בשכבה השנייה.



המודל שלי:

```
def load_default_model(self) -> None :
    """
    the default model if the user does not want to use
    the pretrained model

    """

    self.model = Sequential()
    self.model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu',
                          input_shape = (50, 50, 3)))
    self.model.add(BatchNormalization())
    self.model.add(MaxPool2D(pool_size=(3, 3)))
    self.model.add(Dropout(0.25))
    self.model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Conv2D(32, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Flatten())
    self.model.add(Dense(2, activation='softmax'))
    adam = keras.optimizers.Adam(learning_rate=0.0001)
    self.model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
    self.model.summary()
```

סוגי השכבות השונות:

Conv2D:

זוהי שכבת קונבולוציה דו ממדית שיוצרת קונבולוציה עם ספריית קרנל, אשר מקשרת בין השכבות שלפניה ואחריה.

Filter:

פילטר זהו גודל / חלק מהתמונה שהמודל יעבוד עליו. כלומר פילטר של 3*3 כפי שיש אצלי במודל אומר שהמודל יעבור על הפיקסלים של כל תמונה עם פילטר (קובייה) בגודל שלוש על שלוש.

BatchNormalization:

זוהי טכניקה לאימון רשתות עצביות עמוקות מאוד המנרמלת הקלטים לשכבה. יש לכך השפעה של רבה על ייצוב תהליך הלמידה והפחתה דרסטית של מספר תקופות האימון הנדרשות לאימון רשתות עצביות עמוקות.

Dropout:

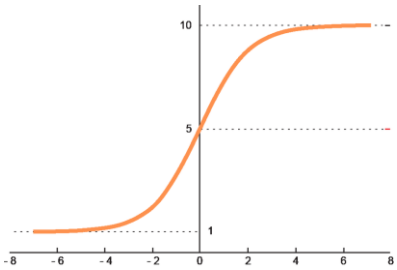
שכבה זו מגדירה באופן אקראי קלטים ל-0 עם תדירות של קצב בכל שלב בזמן האימון מה שעוזר במניעת over fitting.

Activation Function:

Softmax:

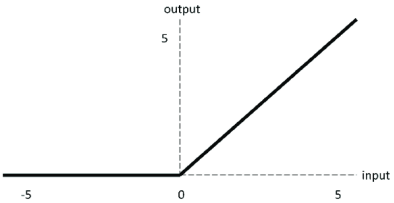
רוב הזמן פונקציה זו קשורה לפונקציית הCross entropy . ברשת קונבולוציה פונקציה זו

משמשת לבדיקת מהימנות המודל תוך כדי שימוש בפונקציית השגיאה, וזאת על מנת למקסם את הביצועים של הרשת העצבית שלנו.



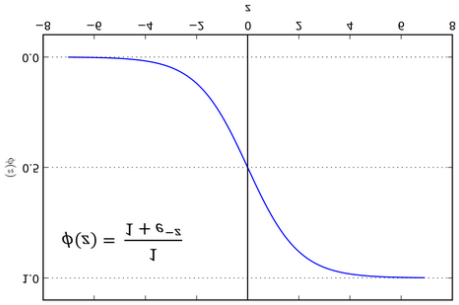
Relu:

פונקציה זו הינה לינארית חלקית שתוציא את הקלט כפלט ישירות אם הוא חיובי, אחרת מחזירה 0.



Sigmoid:

זוהי פונקציה לא לינארית שמחזירה פלט בין 0-1 של סכום המשקלים(קלטים). פונקציה זו מותאמת בעיקר לעבודה של סיווגים בינאריים (0/1).



Dense:

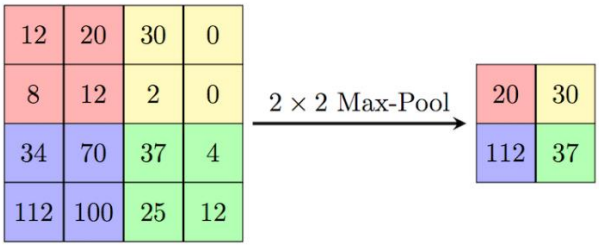
שכבת dense היא שכבה צפופה שמחוברת עם השכבה הקודמת שלה, כלומר הנוירונים של השכבה מחוברים לכל נוירון בשכבה הקודמת. זוהי השכבה הנפוצה ביותר ברשתות עצביות מלאכותיות.

Flatten:

שכבה זו משמשת כדי להמיר את כל המערכים הדו-ממדיים שנוצרו לווקטור ליניארי רציף אחד וארוך. היא משטחת את הערכים מבלי לשנות את הbatch size.

:MaxDool2D

פונקציה אשר מקטינה את ממדי התמונה (אורך ורוחב) על ידי לקיחת הערך המקסימלי של הקלט (בגודל מוגדר מראש pool_size) בכל ממדי התמונה, כלומר מקטין את הפיקסלים של התמונה ולפונקציה זו חשיבות רבה שכן היא משתמשת בפחות משתנים ויכולה להקטין את הסיכוי לover fitting.



<u>מושגים נוספים חשובים:</u>

Data set – מאגר התמונות. המודל צריך ללמוד ולאמן את המודל על תמונות, וכדי לעשות זו אנו נותנים לו מאגר של תמונות המסווגות לקטגוריות. מאגר זה נחלק לשלושה תתי מאגרי מידע: train, test, validation.

Accuracy – אחוז ההצלחה של המודל בחיזוי הקטגוריות של התמונות (1/0).

Loss – מגדיר כמה קרובות היו תוצאות חיזוי התמונות לקטגוריה האמיתית שאליה הן שייכות.

Train data- התמונות אותן המודל לומד בעת תהליך האימון. מאגר זה הוא הגדול ביותר וכולל 70% מכלל התמונות במאגר הממוין.

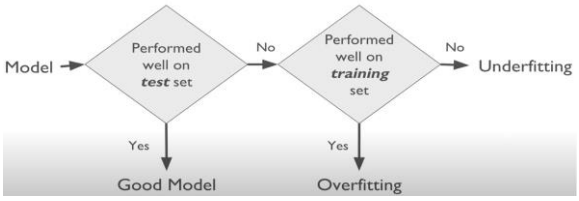
Test data - התמונות אותן המודל אינו לומד, אלא מנסה לזהות בתום תהליך הלמידה. רמת ההצלחה של המודל לזהות נכון את הקטגוריות אליהן שייכות תמונות אלו קובעת כמה המודל טוב וכמה הוא למד לזהות התמונות לפי מאגר ה-train.

validation data – validation הוא דמוי test שנערך במהלך תהליך הלמידה. מטרתו להציג בפני המשתמש את אחוזי ההצלחה של המודל כבר בעת הלמידה שלו. על כן data validation אלו תמונות שהמודל אינו לומד, אלא מנסה לזהות במהלך האימון שלו.

לכל אחד משלושת חלקי המאגר יש ערכי accuracy ו loss. כך ניתן לבחון האם המודל אכן מבצע למידה ואם כן, כמה הוא מצליח.

שתי בעיות שעלולות להיווצר הן: overfitting ו-under fitting:

Overfitting- "התאמת יתר" היא מצב בו המודל מותאם יתר על המידה למאגר אותו הוא לומד, ולכן פחות מצליח בביצוע תחזיות. ניתן לזהות מצב כזה כאשר ה-training accuracy גבוה משמעותית מן ה-validation accuracy או כאשר ה-training loss קטן משמעותית מן ה-validation loss. מצב בו נתקלתי בפרויקט שלי ואפרט עליו יותר בהמשך.



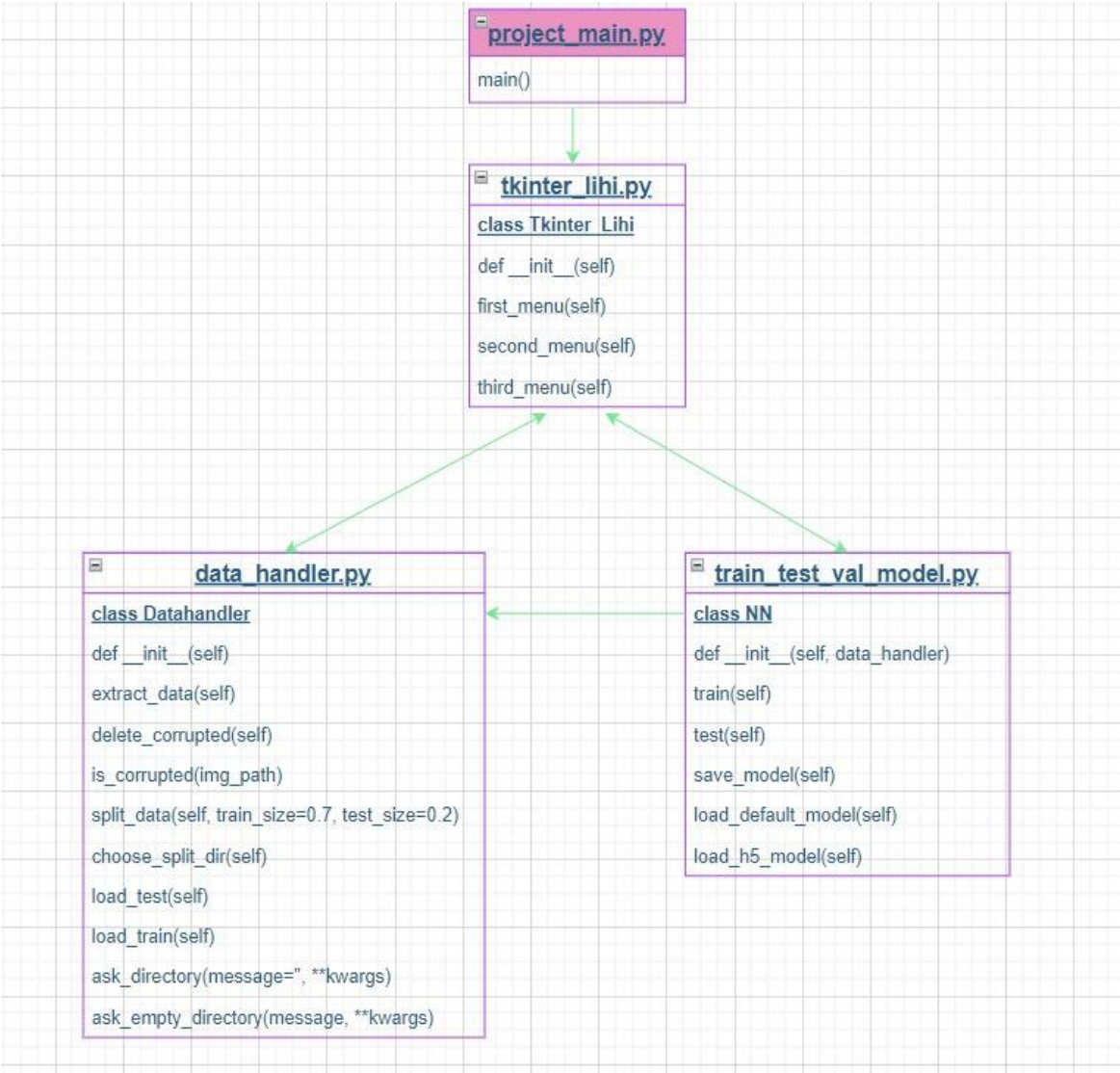
Under fitting – ההפך מ-overfitting, מצב בו מאגר הלמידה פשוט מידי ולא כולל מספיק מגוון של תמונות שונות, או כאשר יש מיעוט בפרמטרים המגדירים את המודל. במצב זה המודל אינו מצליח ללמוד את התמונות. ניתן לזהות מצב כזה כאשר ה-validation accuracy נמוך משמעותית מן ה- accuracy training או כאשר ה-validation loss גבוה משמעותית מן ה-training loss.

epoch- חלוקה ל-epochs הינה הגדרה של מספר פעמים שתהליך הלמידה יתבצע מחדש.

<u>שכבות המודל:</u>

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 50, 50, 32)	896
batch_normalization (BatchNormalization)	(None, 50, 50, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 16)	4624
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 16)	0
conv2d_2 (Conv2D)	(None, 8, 8, 16)	2320
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 16)	0
conv2d_3 (Conv2D)	(None, 4, 4, 32)	4640
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 32)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 2)	258

תיאור UML של המחלקות:

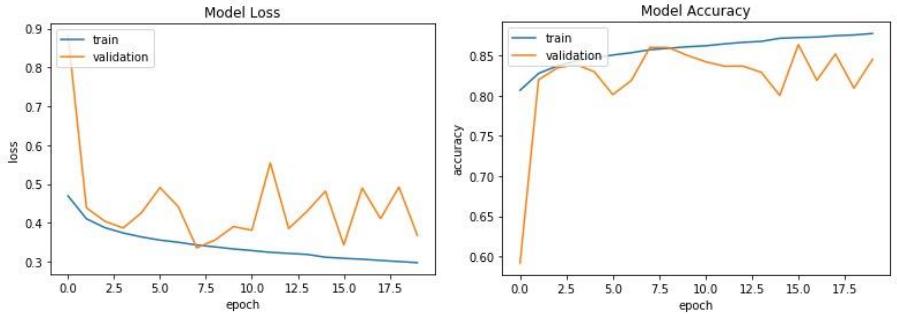


<u>תהליך האימון:</u>

```
Epoch 1/20
3447/3447 [==============================] - 227 s 65ms/step - loss: 0.4510 - accuracy: 0.8000 - val_loss: 0.4704 - val_accuracy: 0.7948
Epoch 2/20
3447/3447 [==============================] - 199 s 58ms/step - loss: 0.4029 - accuracy: 0.8252 - val_loss: 0.4121 - val_accuracy: 0.8180
Epoch 3/20
3447/3447 [==============================] - 210 s 61ms/step - loss: 0.3854 - accuracy: 0.8333 - val_loss: 0.3905 - val_accuracy: 0.8334
Epoch 4/20
3447/3447 [==============================] - 200 s 58ms/step - loss: 0.3744 - accuracy: 0.8388 - val_loss: 0.5305 - val_accuracy: 0.7435
Epoch 5/20
3447/3447 [==============================] - 211 s 61ms/step - loss: 0.3672 - accuracy: 0.8430 - val_loss: 0.3765 - val_accuracy: 0.8419
Epoch 6/20
3447/3447 [==============================] - 200 s 58ms/step - loss: 0.3632 - accuracy: 0.8454 - val_loss: 0.3870 - val_accuracy: 0.8350
Epoch 7/20
3447/3447 [==============================] - 238 s 69ms/step - loss: 0.3588 - accuracy: 0.8474 - val_loss: 0.4245 - val_accuracy: 0.8127
Epoch 8/20
3447/3447 [==============================] - 235 s 68ms/step - loss: 0.3561 - accuracy: 0.8481 - val_loss: 0.4067 - val_accuracy: 0.8273
Epoch 9/20
3447/3447 [==============================] - 226 s 66ms/step - loss: 0.3535 - accuracy: 0.8496 - val_loss: 0.3519 - val_accuracy: 0.8541
Epoch 10/20
3447/3447 [==============================] - 217 s 63ms/step - loss: 0.3510 - accuracy: 0.8506 - val_loss: 0.3776 - val_accuracy: 0.8403
Epoch 11/20
3447/3447 [==============================] - 210 s 61ms/step - loss: 0.3479 - accuracy: 0.8526 - val_loss: 0.3556 - val_accuracy: 0.8560
Epoch 12/20
```

```
s 63ms/step - loss: 219 - [==============================] 3447/3447
0.3460 - accuracy: 0.8534 - val_loss: 0.3551 - val_accuracy: 0.8490

Epoch 13/20

s 60ms/step - loss: 208 - [==============================] 3447/3447
0.3442 - accuracy: 0.8538 - val_loss: 0.3438 - val_accuracy: 0.8591

Epoch 14/20

s 62ms/step - loss: 215 - [==============================] 3447/3447
0.3429 - accuracy: 0.8547 - val_loss: 0.4593 - val_accuracy: 0.7972

Epoch 15/20

s 60ms/step - loss: 206 - [==============================] 3447/3447
0.3408 - accuracy: 0.8554 - val_loss: 0.3697 - val_accuracy: 0.8460

Epoch 16/20

s 59ms/step - loss: 205 - [==============================] 3447/3447
0.3393 - accuracy: 0.8571 - val_loss: 0.3620 - val_accuracy: 0.8498

Epoch 17/20

s 61ms/step - loss: 211 - [==============================] 3447/3447
0.3386 - accuracy: 0.8575 - val_loss: 0.3464 - val_accuracy: 0.8584

Epoch 18/20

s 77ms/step - loss: 265 - [==============================] 3447/3447
0.3361 - accuracy: 0.8579 - val_loss: 0.4780 - val_accuracy: 0.8016

Epoch 19/20

s 61ms/step - loss: 211 - [==============================] 3447/3447
0.3350 - accuracy: 0.8585 - val_loss: 0.3548 - val_accuracy: 0.8528

Epoch 20/20

3447/3447 [==============================] - 207s 60ms/step - loss:
0.3342 - accuracy: 0.8590 - val_loss: 0.3383 - val_accuracy: 0.8593
```

דוחות וגרפים המתארים את תוצאות שלב האימון:



מימין ניתן לראות את דיוק האימון והאימות המודל כתלות במספר האיפוצ'ים.

משמאל ניתן לראות את שגיאת האימון והאימות המודל כתלות במספר האיפוצ'ים.

תוצאות שלב האימון:

test loss: 0.330997496843338

test accuracy: 0.8613455891609192

דוח הכולל ריכוז כל ה- Hyper Parameters:

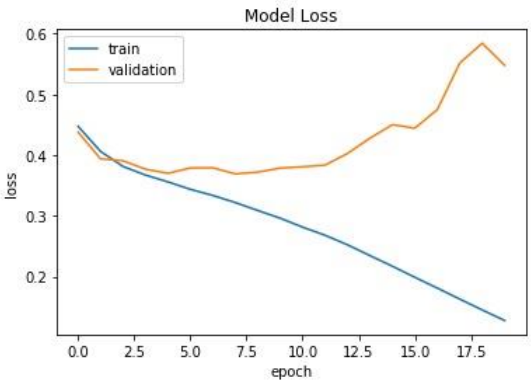
Hyper parameter	Value
Learning rate	0.0001
Epoches	20
Batch size	32
Iterations	3447
Optimizer algorithm	Adam
Activation function	Softmax, relu
Weight initialization	random
Dropout	0.25
Number of hidden layers	5

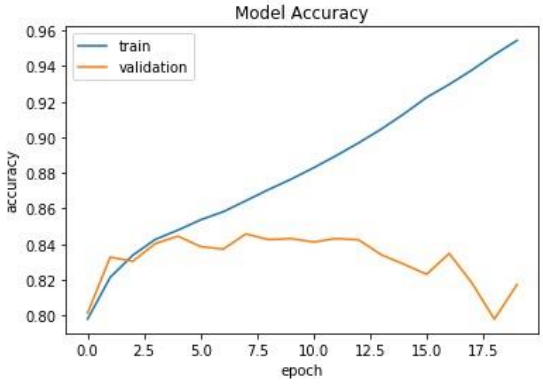
תיעוד השינויים שנעשו במודל+Hyper parameter לשיפור תוצאות האימון:
במהלך אימון המודל אני נתקלתי בבעיית Over fitting בעיה שכיחה בנושא למידת מכונה.

מצב בו המודל מותאם יתר על המידה למאגר אותו הוא לומד, ולכן פחות מצליח בביצוע תחזיות. ניתן לזהות מצב זה כאשר דיוק האימות נמוך בהרבה מדיוק האימון, או כאשר שגיאת האימות גבוהה בהרבה משגיאת האימון. תוצאות האימון מקסימליות(דיוק שואף ל-1 ושגיאה שואפת ל-0).

תחילה לא הבנתי מה גורם לבעיה ולמה אני מקבלת כאלו תוצאות, אך לאחר מחקר באינטרנט והבנה שערך השגיאה של האימות בפרויקט שלי גדול בהרבה מערך השגיאה של המודל המאומן גיליתי שזוהי בעיה שכיחה שהרבה מתמודדים איתה. בין הפתרונות שהוצעו היה לשנות את המודל- להוסיף שכבות ולהוריד את מספר הנוירונים דבר שפתר לי את הבעיה והביא אותי ותוצאות הטובות ביותר שהמודל שלי יכל להפיק.

אציג בצורה גרפית את תוצאות האימון כאשר נתקלתי בבעיה:





ניתן לראות בגרף מצד שמאל ששגיאת האימות גדולה בהרבה משגיאת האימון, וגם מצד ימין דיוק האימות די נמוך בהשוואה לדיוק האימון.

דוגמא לתוצאות שקיבלתי בהרצה :

Epoch 20/20

3447/3447 [==========] - 792s 230ms/step - loss: 0.1276 - accuracy: 0.9544 - val_loss: 0.5479 - val_accuracy: 0.8174

test loss: 0.5411891341209412

test accuracy: 0.818470299243927

ניתן לראות ולהסיק שמדובר בבעיית ה over fitting מכיוון שהמודל לא מניב תוצאות

טובות ומספקות.

לאחר שינוי המודל -הוספת שכבות והקטנת מספר הנוירונים, ושינוי הקצב למידה(הקטנתו) הצלחתי לסדר את הבעיה וקיבלתי תוצאות מספקות וטובות לפרויקט שלי (גרפים ותוצאות צוינו קודם).

מודל שגרם לOver fitting:

```
250
251
252         self.model = Sequential()
253         self.model.add(Conv2D(filters=64,kernel_size=3,input_shape=(50,50,3),activation='relu')) # Changed kernel size to 3x3
254         self.model.add(
255             Conv2D(filters=128, kernel_size=3, input_shape=(50, 50, 3), activation='relu'))  # Changed kernel size to 3x3
256         self.model.add(
257             Conv2D(filters=64, kernel_size=3, input_shape=(50, 50, 3), activation='relu'))  # Changed kernel size to 3x3
258         self.model.add(
259             Conv2D(filters=64, kernel_size=3, input_shape=(50, 50, 3), activation='relu'))  # Changed kernel size to 3x3
260         self.model.add(Conv2D(filters=128, kernel_size=3, input_shape=(50, 50, 3),
261                               activation='relu'))  # Changed kernel size to 3x3
262         self.model.add(Dropout(0.25))
263         self.model.summary()
264
265
266     def load_h5_model(self) -> None :
```

מודל מתוקן:

```
def load_default_model(self) -> None :
    """
    the default model if the user does not want to use
    the pretrained model

    """

    self.model = Sequential()
    self.model.add(Conv2D(32, (3, 3), padding = 'same', activation = 'relu',
                          input_shape = (50, 50, 3)))
    self.model.add(BatchNormalization())
    self.model.add(MaxPool2D(pool_size=(3, 3)))
    self.model.add(Dropout(0.25))
    self.model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Conv2D(16, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Conv2D(32, (3, 3), padding="same", activation="relu"))
    self.model.add(MaxPool2D(pool_size=(2, 2)))

    self.model.add(Flatten())
    self.model.add(Dense(2, activation='softmax'))
    adam = keras.optimizers.Adam(learning_rate=0.0001)
    self.model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
    self.model.summary()
```

בנוסף, בתחילת הפרויקט מספר הepoches שלי היה 10 וכדי להגדיל את יעילות ודיוק המודל הגדלתי את המספר ל20.

<u>פונקציית השגיאה:</u>

פונקציית השגיאה מציגה ערך המייצג את סכום השגיאות במודל שלנו. הוא מודד כמה טוב (או רע) המודל שלנו מצליח. אם השגיאות גבוהות, ההפסד יהיה גבוה, מה שאומר שהמודל לא מאומן בצורה טובה. אחרת, ככל שהוא נמוך יותר, כך המודל שלנו עובד טוב יותר.

כדי לחשב את ההפסד, נעשה שימוש בפונקציית הפסד או עלות. ישנן מספר פונקציות הפסד שונות לשימוש. כל אחד מהם מחשב שגיאות בדרכים שונות, והבעיה קובעת באיזו מהן עדיף להשתמש. Cross-Entropy and Mean Squared הם הנפוצים ביותר עבור בעיות סיווג ורגרסיה, בהתאמה.

משמשת לחישוב שגיאה במקרים שיש 2 פלטים.Binary Cross Entropyפונקציית ה-

בפרויקט שלי יש שני פלטים אשר יכולים להתקבל 0/1 יש סרטן או אין סרטן, ולכן השתמשתי בפונקציה זו.

ניתן לחשב פונקציה זו באמצעות לקיחת ההפרש בין מה שהמודל חזה לבין הפלט בפועל. לאחר מכן אנחנו כופלים את התוצאה שקיבלנו בערך הוואי בחזקת הלוגריתם שלו (שמייצג את סיווג התמונה 0/1) .

(שיהיה חיובי), yזה אומר שאנחנו לוקחים מספר שלילי מעלים אותו בחזקת הלוגריתם של

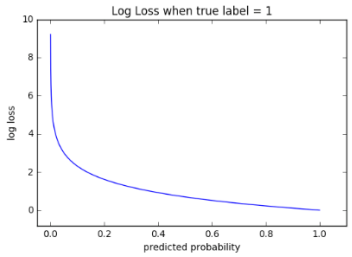
ואז מחסרים את זה מהחישוב המקורי שלנו.

נוסחה לחישוב הפונקציה:

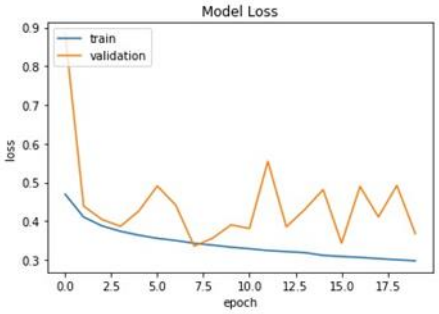
$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

לאחר מכן ערך השגיאה מחושב בסכום כל שגיאות המודל כפול הערך מחישוב הפונקציה.

גרף הפונקציה:



גרף השגיאה של המודל שלי :



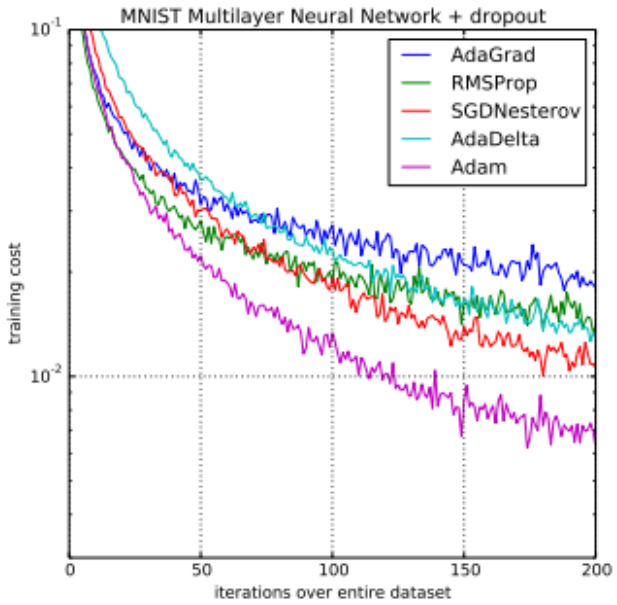
ייעול ההתכנסות (optimization)

אלגוריתמי אופטימיזציה אחראים לצמצום ההפסדים וסיפוק התוצאות המדויקות ביותר שאפשר. המשקל מאותחל באמצעות כמה אסטרטגיות אתחול ומתעדכן עם כל epoch בהתאם למשוואת העדכון.

$$W_{new} = W_{old} - lr * (\nabla_W L)_{W_{old}}$$

המשוואה שלמעלה היא משוואת העדכון באמצעותה מתעדכנים משקלים כדי להגיע לתוצאה המדויקת ביותר. ניתן להשיג את התוצאה הטובה ביותר באמצעות כמה אסטרטגיות אופטימיזציה או אלגוריתמים הנקראים optimizers.

בפרויקט שלי השתמשתי בAdam optimizer - זהו אלגוריתם אופטימיזציה חלופי לירידה בשיפוע לאימון מודלים של למידה עמוקה. אדם משלב את המאפיינים הטובים ביותר של מסר אלגוריתמי אופטימיזציה כדי לספק אלגוריתם אופטימיזציה שיכול להתמודד עם שיפועים דלילים בבעיות קשות, ולכן בחרתי אותו למודל שלי.



שלב היישום (Software deployment):

תיאור והסבר כיצד היישום משתמש במודל:

היישום שלי נבנה באמצעות שימוש ב- Tkinter, למשתמש יוצגו מספר חלונות שיוצרו באמצעות הספרייה Tkinter כאשר כל פעם הוא יצטרך לבחור בכפתור שירצה מכלל הכפתורים בחלון (הסבר מפורט יותר במדריך למשתמש).

המשתמש יכול לבחור להריץ את הפרויקט ממאגר נתונים בקובץ זיפ/ ממאגר נתונים מחולץ/ ממאגר נתונים מחולק לtrain, test, validation (בהרצה ראשונה יהיה חייב מקובץ הזיפ).
לאחר מכן הוא יצטרך לבחור שוב בחלון השני האם הוא רוצה להריץ את הפרויקט על המודל הקיים או על מודל שאומן מראש (קובץ הh5 שהמשתמש יקבל- נמצא בGitHub שלי).
לבסוף בחלון האחרון שיפתח לו המשתמש יבחר האם הוא רוצה לאמן את המודל, לבחון אותו (בנוסף בסוף בחינת המודל תוצג לו תיקיה עם 100 תמונות המסווגות באופן וויזואלי), או לשמור את המודל שאימן, כמו כן הוא גם יכול לחזור לתפריט הקודם במידה ורוצה לבחור מודל אחר.

הסברים מפורטים יותר ניתן לראות במדריך למשתמש (כולל תמונות).

הקוד לתפריט הראשון שיוצג למשתמש:

```
class Tkinter_Lihi:
    def __init__(self):
        self.data_handler = Datahandler() # an object of the class, in order to call diffrent functions in this class.
        self.nn = NN(self.data_handler) # an object of the class, in order to call diffrent functions in this class.
        self.window = None # contains the screen that the user will see in each level.

    def first_menu(self) -> None :
        """
        The first menu that will pop up to the user
        this screen is responsible for the data
        """
        self.window = tk.Tk()
        self.window.configure(bg='pink')
        self.window.geometry('500x500')
        tk.Button(self.window, text='Load from zip', command=lambda: [self.data_handler.extract_data(), self.second_menu()],
                  background='#42eff5').pack(anchor='nw', padx=40, pady=50)
        tk.Button(self.window, text='Load from extracted zip',
                  command=lambda: [self.data_handler.delete_corrupted(), self.second_menu()],
                  background='red').pack(anchor='nw', padx=40, pady=50)
        tk.Button(self.window, text='Load from split data',
                  command=lambda: [self.data_handler.choose_split_dir(), self.second_menu()],
                  background='yellow').pack(anchor='nw', padx=40, pady=50)

        self.window.mainloop()
```

הקוד לתפריט השני שיוצג למשתמש:

```

    def second_menu(self) -> None :
        """
        The second menu that will pop up to the user
        this screen is responsible for the model that we will work with.
        """
        self.window.destroy()

        self.window = tk.Tk()
        self.window.configure(bg='pink')
        self.window.geometry('500x500')
        tk.Button(self.window, text='Load default model',
                  command=lambda: [self.nn.load_default_model(), self.third_menu()]).pack(anchor='nw', padx=40, pady=50)
        tk.Button(self.window, text='Load pretrained model',
                  command=lambda: [self.nn.load_h5_model(), self.third_menu()]).pack(anchor='nw', padx=40, pady=50)
        self.window.mainloop()

```

הקוד לתפריט השלישי שיוצג למשתמש:

```
def third_menu(self) -> None :
    """
    The third and last menu that will pop up to the user
    this screen is responsible for training/ testing/ saving the model.
    we can alse return to the prior screen if we want to load a diffrent model.
    """
    self.window.destroy()
    self.window = tk.Tk()
    self.window.configure(bg='pink')
    self.window.geometry('500x500')
    tk.Button(self.window, text='Train model', command=self.nn.train).pack(anchor='nw', padx=40, pady=50)
    tk.Button(self.window, text='Test model', command=self.nn.test).pack(anchor='nw', padx=40, pady=50)
    tk.Button(self.window, text='save trained model', command=self.nn.save_model).pack(anchor='nw', padx=40,
                                                                                        pady=50)
    tk.Button(self.window, text='Back', command=self.second_menu).pack(anchor='nw', padx=40, pady=50)
    self.window.mainloop()
```

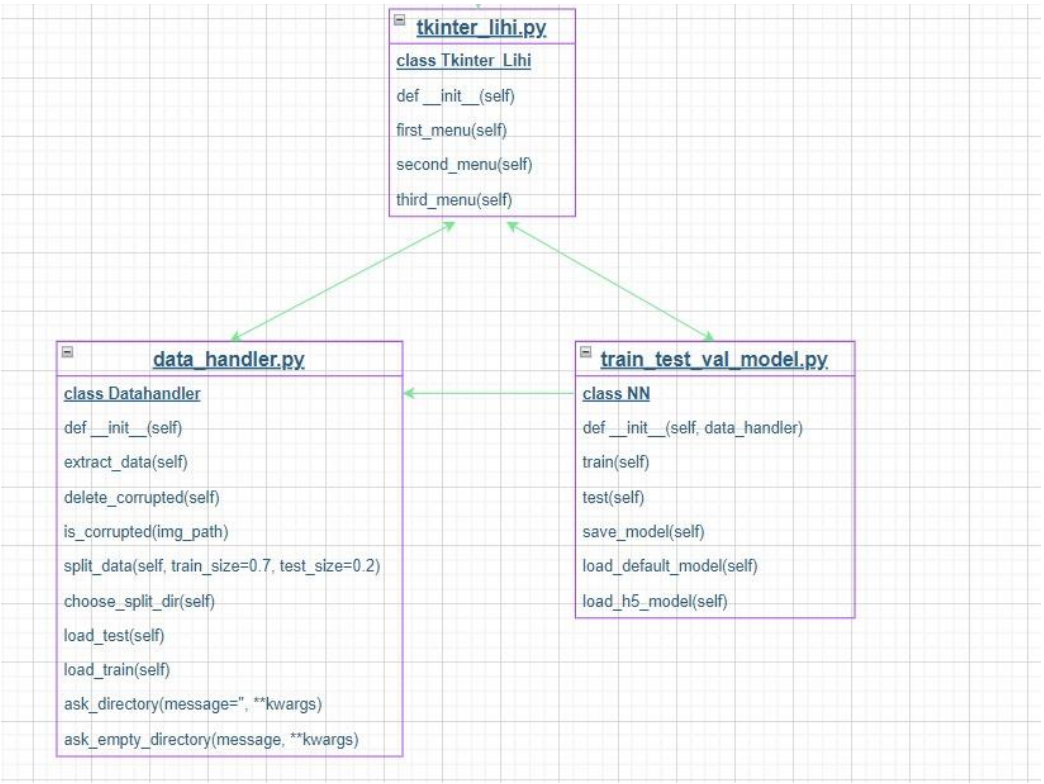
<u>טכנולוגיית Tkinter:</u>

Tkinter הוא ממשק של שפת התכנות פייתון לערכת התצוגה Tk (ספרית קוד חוצה פלטפורמות של כלים גרפיים). כלול בספריה הסטנדרטית של פייתון בהתקנות של פייתון במערכות ההפעלה Linux, Microsoft Windows, ו-Mac OS X.

דוגמא לפקודה בסיסית בספרייה:

()Tk-יצירת החלון עצמו, הבסיס שעליו ייבנו הווידג'טים האחרים.

תרשים UML של המחלקות שמממשות את ממשק המשתמש:



מדריך למפתח

הקבצים בפרויקט:

שם מחלקה/קובץ	היכן נמצא	תפקיד
class Datahandler	בקובץ data_handler.py	המחלקה מטפלת בכל הבעיות/ דרישות הקשורות למאגר הנתונים. סידור מבנה הנתונים, בדיקתו, טעינת הנתונים/ תמונות הנדרשות.
class NN	בקובץ train_test_val_model.py	המחלקה אחראית על בניית המודל, אימון המודל, בדיקת המודל, שמירת המודל המאומן. בנוסף, בתוך המחלקה מודפסים הגרפים שמראים את דיוק המודל (accuracy ו- loss).
class Tkinter_Lihi	בקובץ tkinter_lihi.py	המחלקה אחראית על תקשורת עם המשתמש באמצעות הצגת תפריטים וכפתורים שונים בהם יוכל לנווט בפרויקט.
project_main.py	זהו קובץ בפני עצמו	בקובץ זה נמצאת הפעולה הראשית שממנה נריץ את הפרויקט והיא תחבר בין כלל נתוני הפרויקט.

הקובץ project_main :
פונקציות בקובץ:

שם הפונקציה	תפקידה
def main()	הפעולה הראשית, תפקידה להתחיל את הרצת הפרויקט בכך שתקרא למחלקה class Tkinter_Lihi ותיצור מופע חדש.

משתנים בקובץ:

שם המשתנה	תפקידו
tkinter_lihi	יכיל מופע של המחלקה class Tkinter_Lihi ויזמן את הפעולה של המחלקה אשר מציגה את התפריט הראשוני למשתמש, כדי להתחיל את הרצת הפרויקט.

המחלקה class Datahandler :
פונקציות במחלקה:

שם הפונקציה	תפקידה
def __init__(self)	הפעולה הבונה של המחלקה- בנאי ללא פרמטרים המשמש ליצירת אובייקטים/מופעים של המחלקה. באמצעות פעולה זו יהיה ניתן לקרוא לפעולות אחרות במחלקה.
def extract_data(self)	פעולה שמבקשת מהמשתמש את קובץ הזיפ עם מאגר התמונות, ולאחר מכן מבקשת תיקיה ריקה אליה היא תחלץ את הקבצים שבזיפ. פעולה זו גם מזמנת בסופה את הפעולה שמוחקת קבצים/תמונות לא תקינות.
def delete_corrupted(self)	פעולה זו עובדת עם קובץ הזיפ המחולץ. היא מחלקת את כלל התמונות לשני מערכים של יש סרטן=1, אין סרטן=0, ולאחר מכן שולחת לפעולה def __is_corrupted(img_path) שבודקת האם הקובץ לא תקין, במידה והוא לא תקין (הפעולה מחזירה True) הפעולה מוחקת את הקובץ ממאגר הנתונים. בנוסף, הפעולה מחשבת ומדפיסה כמה תמונות לא תקינות יש בכל קבוצה (0/1), וכמה תמונות תקינות יש בסך כל מאגר הנתונים(לאחר בדיקת המידע ותקינותו). בסוף התוכנית הפעולה קוראת לפונקציה self.split_data()
def __is_corrupted(img_path)	זוהי פעולה סטטית במחלקה המקבלת נתיב של תמונה מהפעולה def

	delete_corrupted(self), היא מורידה את התמונה, מנרמלת אותה (לטווח של בין 0-1) , ובודקת את הגודל שלה (צריך להיות [50,50,3] . אם אחת הבדיקות יצאה לא תקינה הפעולה מחזירה True אחרת מחזירה False.
def split_data(self, train_size=0.7, test_size=0.2)	ראשית, פעולה זו מבקשת מהמשתמש תיקיה ריקה שבה יהיו התמונות שמחולקות ל-train,validation,split. לאחר מכן, הפעולה מחלקת את התמונות לשני מערכים (0=בלי סרטן, 1= עם סרטן) , היא מדפיסה כמה תמונות יש בכל מערך, ולאחר מכן ממיינת אותן בכך שבוחרת את המערך שבו מספר התמונות קטן יותר ומשווה את מספר התמונות במערך השני למספר זה, לאחר פעולות אלה הפעולה תדפיס את מספר התמונות בכל מערך לאחר המיון(יהיה אותו מספר). לאחר מכן, הפעולה בונה 3 תיקיות בנתיב של הקובץ שקיבלה מהמשתמש: "training"- שיכיל 70% מכלל התמונות במאגר (כמובן מספר שווה של תמונות עם וללא סרטן). "validation"- שיכיל 10% מכלל התמונות במאגר (כמובן מספר שווה של תמונות עם וללא סרטן). "testing" – שיכיל 20% מכלל התמונות במאגר (כמובן מספר שווה של תמונות עם וללא סרטן). כאשר הוא בונה את התיקיות יהיה מודפס על המסך איזו תיקיה נבנית עכשיו. מעבר לכך, התיקיות בנויות כך שבכל תיקיה יש 2 תיקיות ל-0= ללא סרטן, ול-1=עם סרטן. בסוף הפונקציה הפעולה תדפיס שהיא סיימה לחלק את מאגר הנתונים.
def choose_split_dir(self)	במידה והמשתמש מריץ יותר מפעם אחת את הפרויקט ויש לו כבר את הקובץ של מאגר הנתונים המחולק, כדי לחסוך זמן המשתמש יכול באמצעות פעולה זו לבחור בקובץ זה במקום לפצל מחדש את המידע.
def load_train(self)	פעולה זו מורידה את התמונות לאימון (train), מנרמלת אותן (בין 0-1) ומחלקת אותן לשני מערכים, כאשר מערך אחד יכיל את התמונות והשני יכיל את "סיווג" התמונה (כלומר 0/1) , הפונקציה תחזיר את שני המערכים לאחר ערבוב המיקומים שלהן.
def load_test(self)	פעולה זו מורידה את התמונות לבדיקה (test), מנרמלת אותן (בין 0-1) ומחלקת אותן לשני מערכים, כאשר מערך אחד יכיל

	את התמונות לבדיקה והשני יכיל את "סיווג" התמונה (כלומר 0/1) , הפונקציה תחזיר את שני המערכים לאחר ערבוב המיקומים שלהן.
def load_val(self)	פעולה זו מורידה את התמונות לאימות (validation), מנרמלת אותן (בין 1-0) ומחלקת אותן לשני מערכים, כאשר מערך אחד יכיל את התמונות לאימות והשני יכיל את "סיווג" התמונה (כלומר 0/1) , הפונקציה תחזיר את שני המערכים לאחר ערבוב המיקומים שלהן.
def __ask_directory(message='', **kwargs)	פעולה סטטית של המחלקה המקבלת הודעה על בקשה לתיקיה מסוימת, מבקשת מהמשתמש את התיקייה שבהודעה שקיבלה ומחזירה את התיקייה המבוקשת.
def ask_empty_directory(message, **kwargs)	פעולה סטטית של המחלקה המקבלת הודעה/ בקשה לתיקיה מסוימת- כאשר התיקייה חייבת להיות ריקה. כל עוד המשתמש בחר תיקיה לא ריקה הפעולה תמשיך לבקש תיקיה ריקה. הפעולה מחזירה את התיקייה הריקה שהמשתמש בחר.

<u>משתנים במחלקה:</u>

שם המשתנה	תפקידו
self.extracted_dir	תכונה של המחלקה אשר תכיל את מאגר התמונות המחולץ (לאחר פתיחת הזיפ).
self.data0	תכונה של המחלקה. תכיל מערך של נתיבי תמונות ללא סרטן.
self.data1	תכונה של המחלקה. תכיל מערך של נתיבי תמונות עם סרטן.
self.train_paths	יכיל את נתיבי התמונות (גם עם וגם ללא סרטן) שנכניס לתיקיית התמונות לאימון.
self.test_paths	יכיל את נתיבי התמונות (גם עם וגם ללא סרטן) שנכניס לתיקיית התמונות לבדיקה.
self.val_paths	יכיל את נתיבי התמונות (גם עם וגם ללא סרטן) שנכניס לתיקיית התמונות לאימות.
self.split_dir	יכיל נתיב תיקיה ריקה שהמשתמש יכניס, כאשר שם יהיה מאגר הנתונים המחולק לשלוש תיקיות :אימון, בדיקה ואימות.
zip_file	יכיל את קובץ הזיפ הראשוני של מאגר התמונות שהמשתמש ייתן.
imagePatches	משתנה שיכיל את כל נתיבי התמונות ממאגר התמונות המחולץ (גם לפני וגם אחרי בדיקת תקינות התמונות).
corrupted_0	משתנה שיכיל את מספר התמונות הלא תקינות ללא סרטן- על מנת להדפיס הודעה שתעדכן את המשתמש כמה יש.

corrupted_1	משתנה שיכיל את מספר התמונות הלא תקינות עם סרטן- על מנת להדפיס הודעה שתעדכן את המשתמש כמה יש.
uncorrupted_dir	משנה זה יכיל את הקובץ של מאגר הנתונים המחולץ לאחר בדיקת תקינות הנתונים.
sampled_data0	יכיל את סך כל נתיבי התמונות התקינות ללא סרטן- נתיבים שעברו "ערבוב" (כאשר מספר נתיבי התמונות ללא סרטן שווה למספר נתיבי התמונות עם סרטן).
sampled_data1	יכיל את סך כל נתיבי התמונות התקינות עם סרטן- נתיבים שעברו "ערבוב" (כאשר מספר נתיבי התמונות ללא סרטן שווה למספר נתיבי התמונות עם סרטן).
TRAIN_PATH	נתיב לתיקיה שתכיל את כל נתיבי התמונות לאימון.
VAL_PATH	נתיב לתיקיה שתכיל את כל נתיבי התמונות לאימות.
TEST_PATH	נתיב לתיקיה שתכיל את כל נתיבי התמונות לבדיקה.
train_count	יכיל את מספר נתיבי התמונות שצריכים להיות בתיקיה training (70% מכלל התמונות)
val_count	יכיל את מספר נתיבי התמונות שצריכים להיות בתיקיה validation (10% מכלל התמונות)
test_size	יכיל את מספר נתיבי התמונות שצריכים להיות בתיקיה testing (20% מכלל התמונות)
datasets	מכיל את כל התמונות מהאימון, בדיקה ואימות לפני הפיצול.
PATH	יכיל נתיב של כל תמונה כדי שנכניס את נתיב לתיקיה של האימון/ בדיקה / אימות.
label	יכיל את "הסיווג" של כל תמונה (0/1) אשר נכניס למערך "הסיווגים"/ תוויות.
img	יכיל את התמונה המנורמלת אשר נוסיף למערך התמונות.
images_list	יכיל מערך של התמונות לאימון/בדיקה/אימות של המודל.
labels_list	יכיל מערך של "הסיווגים"(0/1) לאימון/בדיקה/אימות של המודל.
count	יכיל את מספר התמונות הנטענות בכל פעם כדי להדפיס למסך כל 1000 תמונות שנטענות.
x	יכיל את מערך images_list בסדר מעורבב. בשביל לאמן/לבדוק/לאמת את המודל.
y	יכיל את מערך labels_list בסדר מעורבב. בשביל לאמן/לבדוק/לאמת את המודל.

idx	יכיל סדר מעורבב של מערך התמונות כדי שהמודל לא יתאמן על תבנית קבועה ולא נקבל תוצאות מדויקות.
dire	יכיל את נתיב התיקייה שהמשתמש בחר בין אם זו תיקייה עם נתונים, או תיקייה ריקה.

המחלקה class NN :

<u>פעולות במחלקה:</u>

שם הפונקציה	תפקידה
def __init__(self, data_handler)	פעולה בונה של המחלקה, מקבלת מופע של המחלקה Datahandler. פעולה זו משמשת ליצירת מופעים של המחלקה NN.
def train(self)	פעולה זו אחראית על אימון המודל באמצעות פונקציית fit, תחילה היא שולחת את התמונות לאימון ולאימות לפעולה שמורידה את התמונות ואז היא שולחת לפונקציית fit את שני המערכים (תמונות ותוויות). כאשר הפעולה מסיימת את האימון היא מדפיסה שני גרפים שמראים את השינוי בדיוק (accuracy) של התמונות ב- train, validation וגרף שני מראה את השינוי בשגיאה (loss) של ה train, validation. לבסוף כאשר הפונקציה סיימה את האימון היא תדפיס למשתמש הודעה שהמודל סיים להתאמן, ושיבחר את הפעולה (כפתור) הבא שברצונות לממש.
def test(self)	פעולה זו אחראית על בחינת המודל בפועל. תחילה הפעולה מבקשת מהמשתמש תיקייה ריקה שתשמש להראות באופן ויזואלי 100 תמונות למשתמש וסיווגן, כלומר היא תראה 100 תמונות כאשר רשום על כל אחת מהן האם יש או אין סרטן ואיך המודל "סיווג" את זה (0/1). פונקציה זו תקרא לפעולה שמורידה את התמונות לבדיקה ומחזירה לה שני מערכים (תמונות ותוויות 0/1), ולאחר מכן הפונקציה מבצעת בחינה של המודל באמצעות הפקודה evaluate שבוחנת את אחוזי ההצלחה של המודל בכך שהוא מנסה לזהות תמונות חדשות שלא למד קודם לכן ומציג את אחוזי ההצלחה שלו. בסופו של דבר הפונקציה מדפיסה את אחוזי ההצלחה של המודל – Loss, Accuracy.
def save_model(self)	פונקציה זו שומרת את המודל המאומן בקובץ במיקום שהמשתמש בחר וכן שומרת את קובץ התוויות הבינארי במיקום

	שהמשתמש בחר. זאת כדי שיהיה ניתן להשתמש בלמידה הנוכחית גם בהרצות הבאות של התוכנית וכן בשביל שיהיה ניתן לבצע predict מבלי להריץ את המודל מחדש.
def load_default_model(self)	פעולה זו אחראית על בניית המודל שאותו נאמן ונבדוק באמצעות הפעולות שהוזכרו לעיל. (פירוט על המודל, שכבות וכיצד הוא פועל יפורט במבנה/ ארכיטקטורה של הפרויקט).
def load_h5_model(self)	פעולה זו טוענת את המודל המאומן שנשמר קודם לכן (במידה ונשמר). היא מבקשת מהמשתמש שיבחר את התיקייה שבה המודל המאומן (h5) נמצא ושולחת את נתיב הקובץ לפעולה קיימת הנמצאת בספריית tensorflow אשר טוענת את המודל.

<u>משתנים במחלקה:</u>

שם המשתנה	תפקידו
self.model	תכונה של המחלקה. מכיל את המודל
self.last_history	תכונה של המחלקה, תכיל את ההיסטוריה של סשן האימונים האחרון.
x_train	מערך התמונות לאימון המודל.
y_train	מערך הסיווגים/ תוויות של התמונות לאימון.
x_val	מערך התמונות לאימות המודל.
y_val	מערך הסיווגים/ תוויות של התמונות לאימות.
history	ישמור את המודל המאומן האחרון
loss	מכיל את פונקציית השגיאה של המודל המאומן
accuracy	מכיל את פונקציית הדיוק של המודל המאומן.
val_loss	יכיל את מידת השגיאה של התמונות לאימות, נראה את זה תוך כדי אימון המודל.
val_accuracy	יכיל את מידת הדיוק של התמונות לאימות, נראה את זה תוך כדי אימון המודל.
vis_dir	תיקייה ריקה שהמשתמש יבחר שתשמש להצגת התמונות שסווגו באופן וויזואלי למשתמש.
num_vis	מספר התמונות שהמודל סיווג בבדיקה שיוצגו באופן וויזואלי למשתמש (ערך נקבע ל100).
x_test	מערך התמונות לבדיקת המודל.
y_test	מערך הסיווגים/ תוויות של התמונות לבידקה.

result	רשימה המכילה את תוצאות בדיקת המודל, כלומר accuracy, loss של בדיקת המודל.
idx	יכיל רשימה מעורבבת של מספרים באורך של מערך התמונות לבדיקת המודל.
preds	יכיל את התוצאות שהמודל המאומן ייתן לתמונות לבדיקה. שימוש במשתנה זה בתיקיית התמונות המסווגות באופן וויזואלי.
image	יכיל את התמונות שנכניס לתיקייה שמציגה למשתמש מספר תמונות (100) באופן וויזואלי. ואת הסיווג שלהן.
font	קובע את הפונט של מה שרשום על התמונות המוצגות באופן וויזואלי למשתמש.
label	יכיל את התווית/ סיווג המקורית של התמונה שמציגים באופן וויזואלי למשתמש.
pred	יכיל את התווית/ סיווג שהמודל קבע לתמונה שמציגים באופן וויזואלי.
cv2_im_processed	יכיל את התמונה המוצגת באופן וויזואלי למשתמש בסדר צבעים שונה BGR במקום RGB, כדי שopencv יוכל לקרוא את התמונה הסדר צבעים צריך להיות כך.
save_name	שם התיקייה שהמשתמש יבחר שבה ישמר המודל המאומן.
model_path	יכיל את הנתיב של התיקייה שבה שמור המודל המאומן.

Tkinter_Lihi המחלקה:

<u>פונקציות במחלקה:</u>

שם הפונקציה	תפקידה
def __init__(self)	פעולה בונה של המחלקה. ללא פרמטרים. משמשת ליצירת מופעים / אובייקטים של המחלקה.
def first_menu(self)	פעולה שאחראית על התפריט הראשון שיוצג למשתמש, פעולה זו מציגה למשתמש 3 כפתורים שאחראים על הכנת מאגר הנתונים לפרויקט. לא מקבלת כלום לא מחזירה כלום.
def second_menu(self)	פעולה שאחראית על התפריט השני שיוצג למשתמש, פעולה זו מציגה למשתמש 2 כפתורים שאחראים על בחירת המודל שהפרויקט יעבוד איתו (מאומן או לא מאומן). לא מקבלת כלום לא מחזירה כלום.
def third_menu(self)	פעולה שאחראית על התפריט השלישי שיוצג למשתמש, פעולה זו מציגה למשתמש 4 כפתורים שאחראים על אימון/ בחינת/ שימרת המודל, כמו כן יש עוד

	כפתור שממנו ניתן לחזור לתפריט הקודם- תפריט 2. לא מקבלת כלום לא מחזירה כלום.

<u>משתנים במחלקה:</u>

שם המשתנה	תפקידו
self.data_handler	תכונה של המחלקה הנתונה, תכיל אובייקט/ מופע של המחלקה Datahandler. משתנה זה מאפשר קריאה לפעולות המחלקה Datahandler.
self.nn	תכונה של המחלקה הנתונה, תכיל אובייקט/ מופע של המחלקה NN משתנה זה מאפשר קריאה לפעולות המחלקה NN.
self.window	תכונה של המחלקה, מכילה את המסך/ תפריט שיוצג למשתמש בכל שלב בהרצה.

מדריך למשתמש

ראשית יש להתקין על המחשב את סביבת העבודה אנקונדה (שנה 2019) שעובדת עם פייתון גרסה 3.7, בפרויקט שלי סביבת העבודה היא ספיידר(באנקונדה). ניתן גם לעבוד עם פייתון במידה והגרסה היא 3.7. אפשר לבדוק מהי הגרסה על המחשב שלך באמצעות הפקודה:

python -V או python --version

קישור להורדת סביבת העבודה אנקונדה:

https://drive.google.com/file/d/1XibUZAVECTf3yxcNPJOErJay2xwZ7vZa/view?usp=sharing

הפרויקט משתמש במספר ספריות, ולכן נרצה להוריד אותן באמצעות הprompt (אנקונדה או פייתון).

library name	Installation command	Link
Keras	pip install keras	https://pypi.org/project/Keras/
Tensorflow	pip install tensorflow	https://pypi.org/project/tensorflow/
Matplotlib	pip install matplotlib	https://pypi.org/project/matplotlib/
Numpy	pip install numpy	https://pypi.org/project/numpy/
opencv	pip install opencv-python	opencv-python · PyPI
glob	Pip3 install glob2	https://pypi.org/project/glob2/
Pillow	pip install pillow	https://pypi.org/project/Pillow/

<u>הקבצים שנדרשים לפרויקט:</u>

data_handler.py

project_main.py

tkinter_lihi.py

train_test_val.py

trained_model2.h5

קובץ זיפ של מאגר הנתונים uncorrupted data / archive_final (מאגר הנתונים לאחר מחיקת התמונות הלא תקינות) שניתן להוריד באמצעות הקישור הנתון מטה.

קישור לתיקייה –

https://drive.google.com/drive/folders/1a7y4LwFvX_j0jCoBPEMxIPutN2kYP2fM?usp=sharing

ניתן למצוא את קבצי הפרויקט בGitHub שלי ולהוריד אותם משם- ללחוץ על כפתור Code הירוק וללחוץ על הורד זיפ, אין לשנות שום דבר מתוכני הקבצים ויש לשמור את כולם תחת אותה תיקיה(נשמר אוטומטית תחת אותה תיקייה).

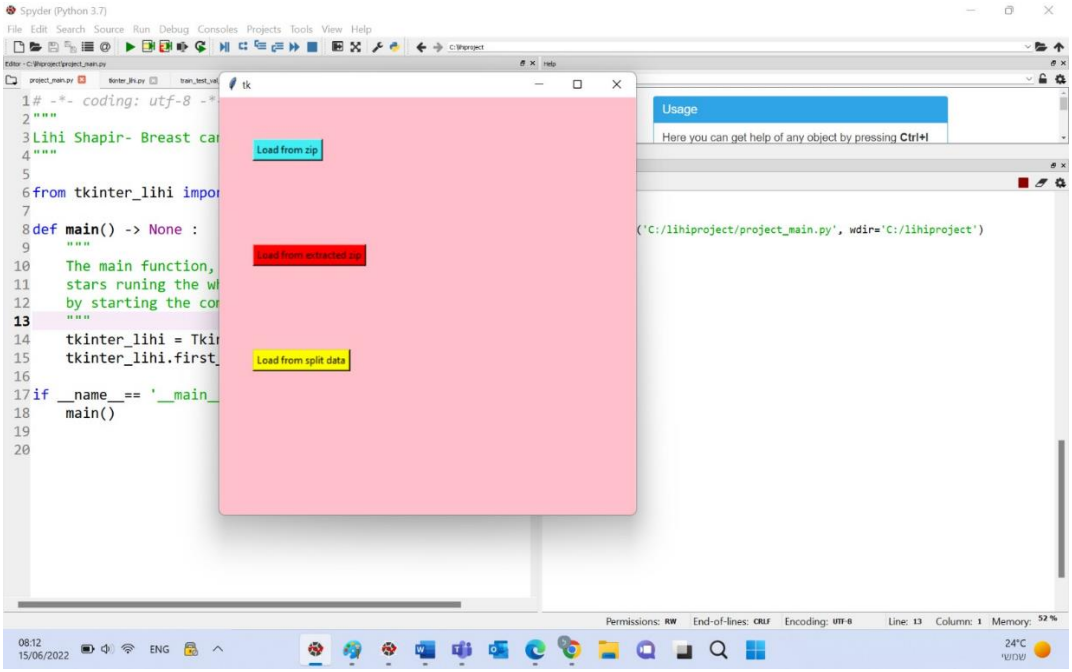
קישור לGitHub שלי:

https://github.com/LihiShapir/breast_cancer_project.git

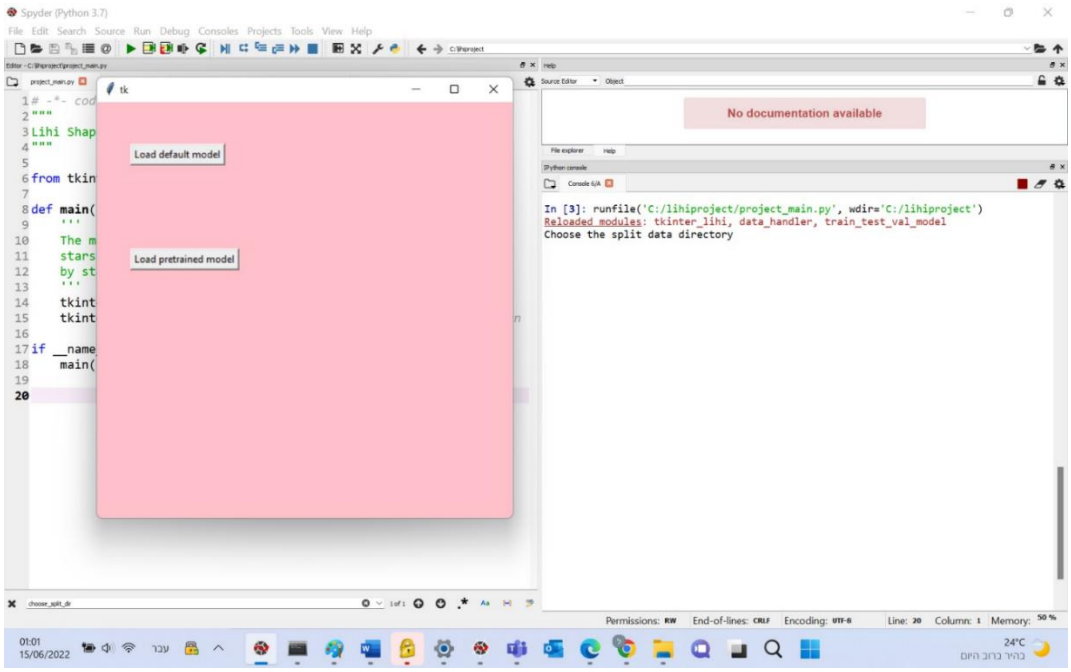
הרצת הפרויקט- לאחר הורדת כל הקבצים – יש לחלץ את הקבצים שהורדו מGitHub באמצעות מקש ימני על הקובץ ולחיצה על extract all/here . לאחר שכל הקבצים חולצו יש לפתוח את spyder באנקונדה, ולפתוח שם את כל הקבצים בעלי סיומת py . כדי להתחיל את הרצת הפרויקט נלך לקובץ project_main.py ונלחץ על כפתור החץ הירוק שבסרגל הכלים למעלה.

<u>היררכיית המסכים:</u>

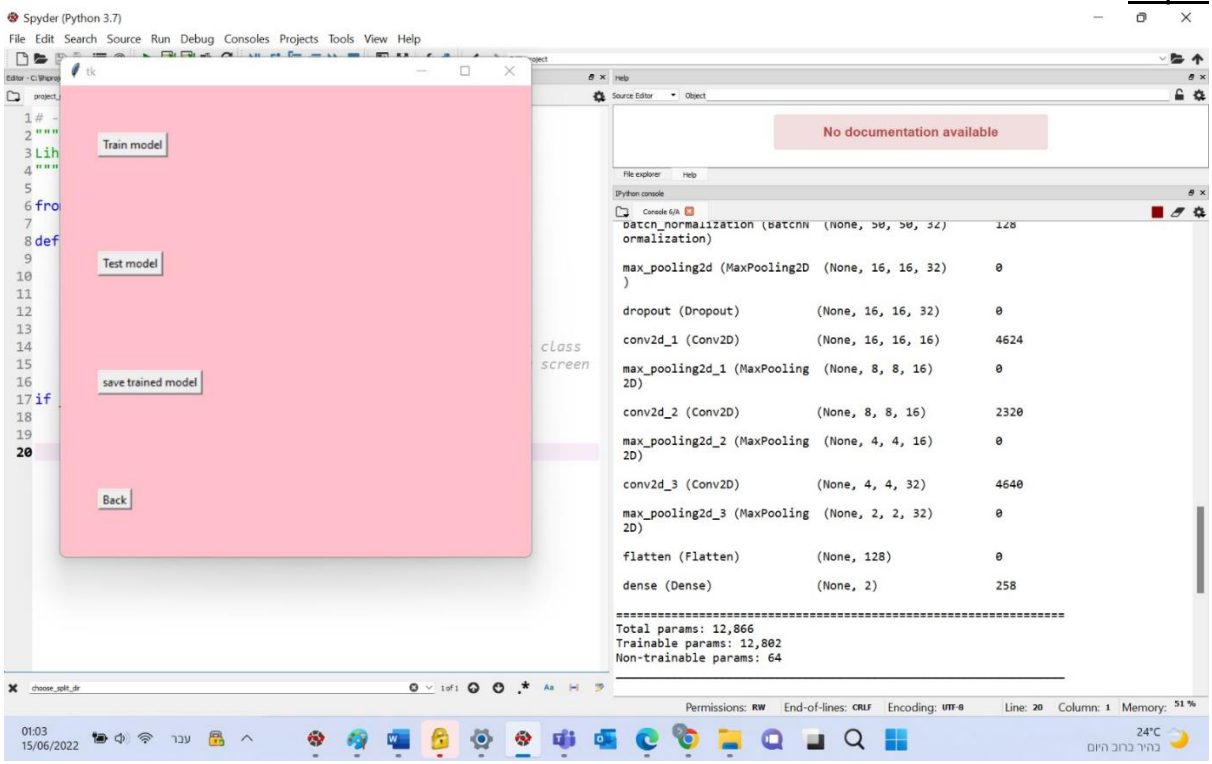
<u>מסך 1</u> שיוצג למשתמש- עם תחילת הרצת הפרויקט



<u>מסך 2</u> שיוצג למשתמש

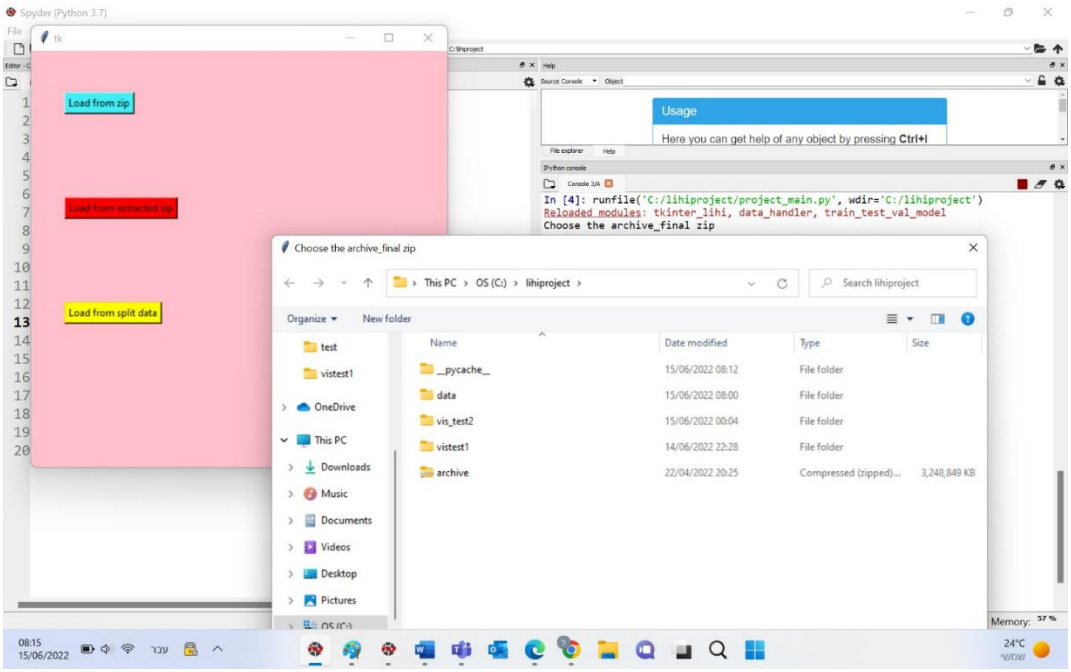


מסך 3 שיוצג למשתמש



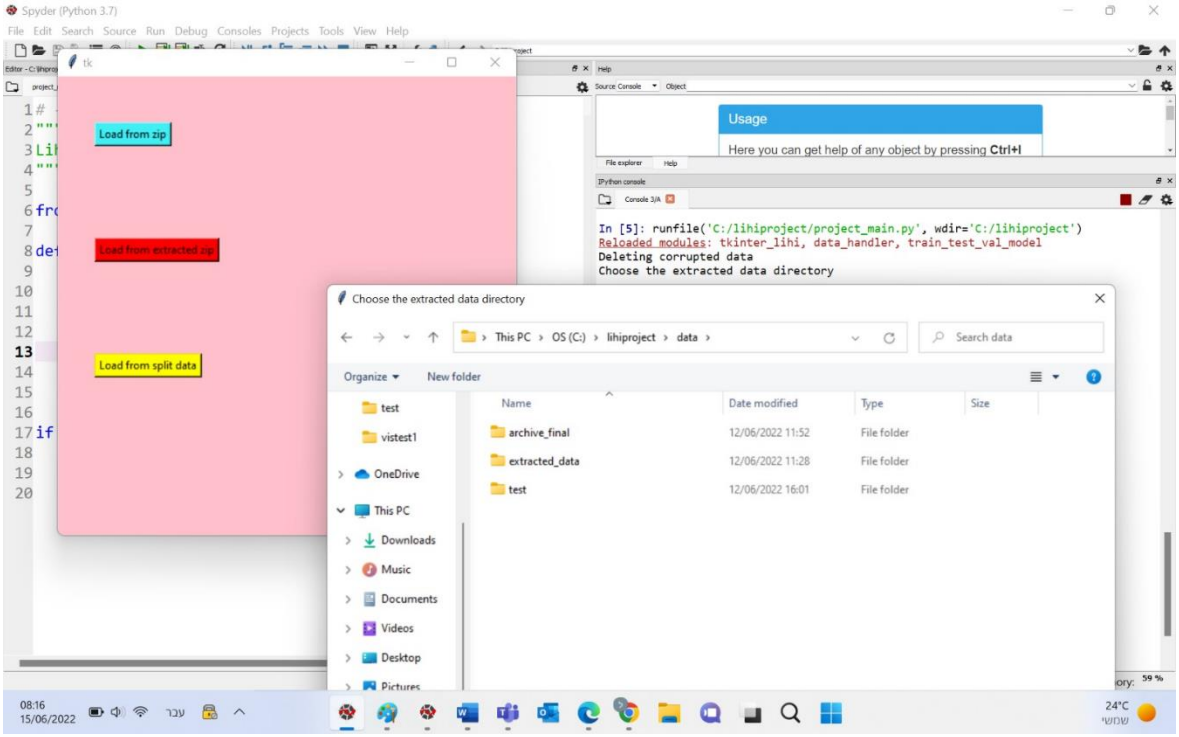
מסך הפתיחה – תפריט 1 שיוצג למשתמש מכיל 3 כפתורים והוא משמש כנקודת ניווט להכנת מאגר הנתונים הנדרש לפרויקט.

במידה ונלחץ על הכפתור הראשון – תכלת שאומר לטעון מקובץ הזיפ תיפתח לנו חלונית חדשה של המסמכים ונצטרך לבחור בקובץ הזיפ שהורדנו מבעוד מועד.



במידה ונלחץ על הכפתור השני בתפריט 1- כפתור אדום שאומר לטעון ממאגר נתונים מחולץ, תיפתח לנו חלונית נוספת שם נצטרך לבחור את קובץ extracted_data .

כפתור זה יכול לשמש אותנו במידה והרצנו פעם אחת את הפרויקט מההתחלה- מקובץ הזיפ, ואנחנו רוצים ליעל את הזמן ולא לחלץ שוב את כל הקבצים מהזיפ.



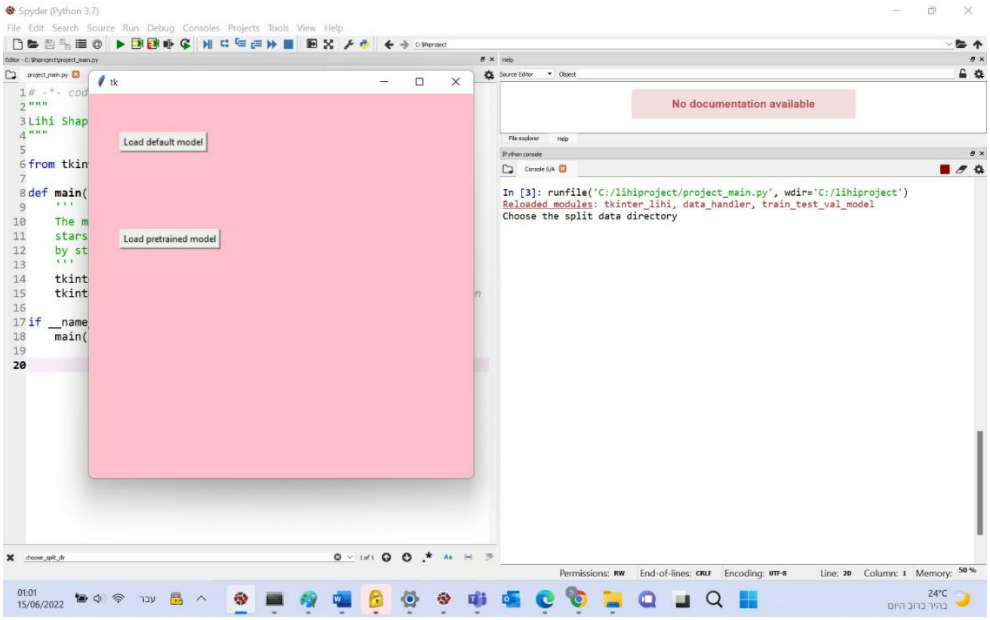
במידה ונלחץ על הכפתור השלישי בתפריט 1- כפתור צהוב שאמר לטעון ממאגר הנתונים המחולק (train, test, validation), תיפתח לנו חלונית נוספת ונצטרך לבחור בקובץ מאגר הנתונים המחולק (split_data).

כפתור זה יכול לשמש אותנו במידה והרצנו פעם אחת את הפרויקט - מקובץ הזיפ ואנחנו רוצים ליעל את הזמן ולא לחלק שוב את כל מאגר הנתונים.

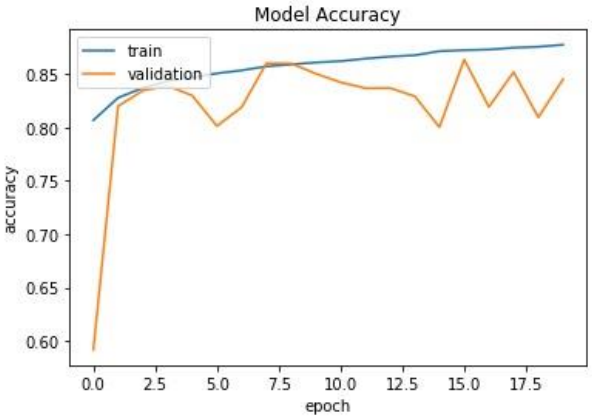
<u>מסך 2</u>- תפריט 2 שיוצג למשתמש מכיל 2 כפתורים והוא אחראי על המודל שהפרויקט עובד איתו.

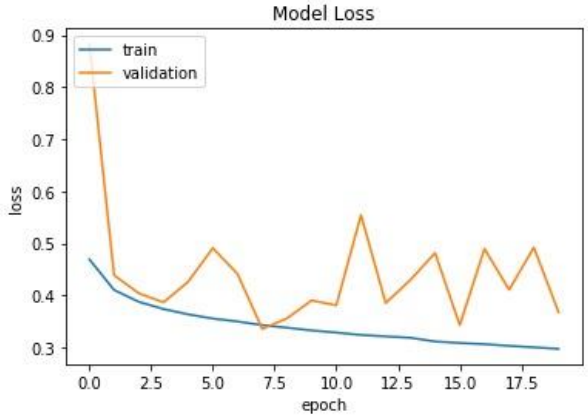
כפתור 1- כפתור שאומר לטעון מודל ברירת מחדל, כלומר המודל שכתוב בקוד הפרויקט.

כאשר נלחץ על הכפתור הזה יפתח לנו תפריט נוסף- תפריט שלוש.



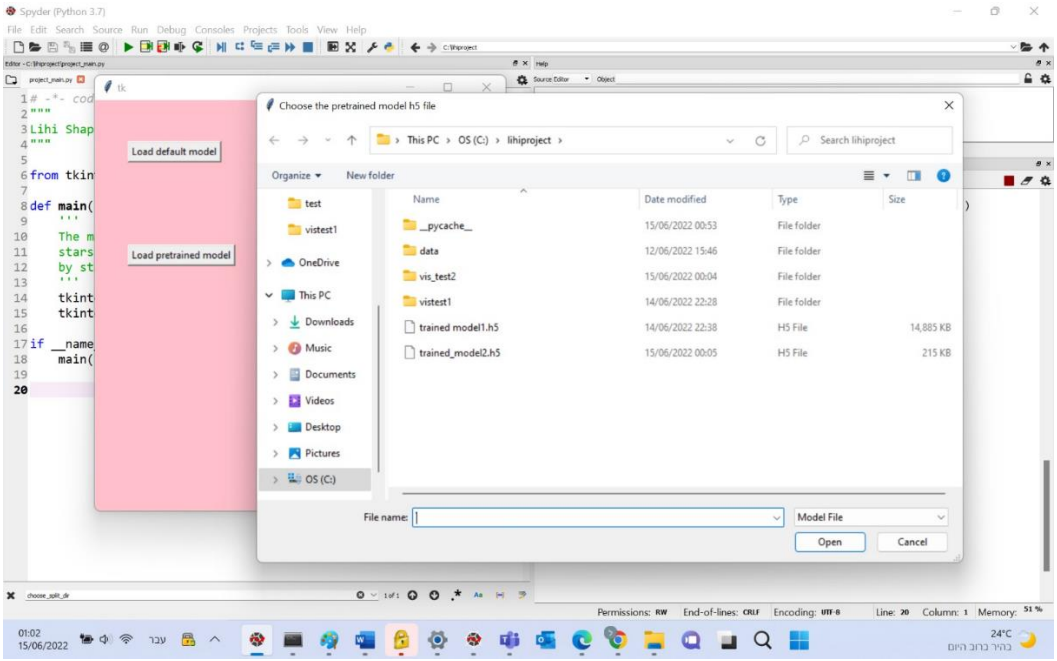
בסוף אימון המודל נוכל לראות שני גרפים שיראו את מידת הaccuracy, loss של המודל באימון (דוגמא לשני גרפים שיוצגו).





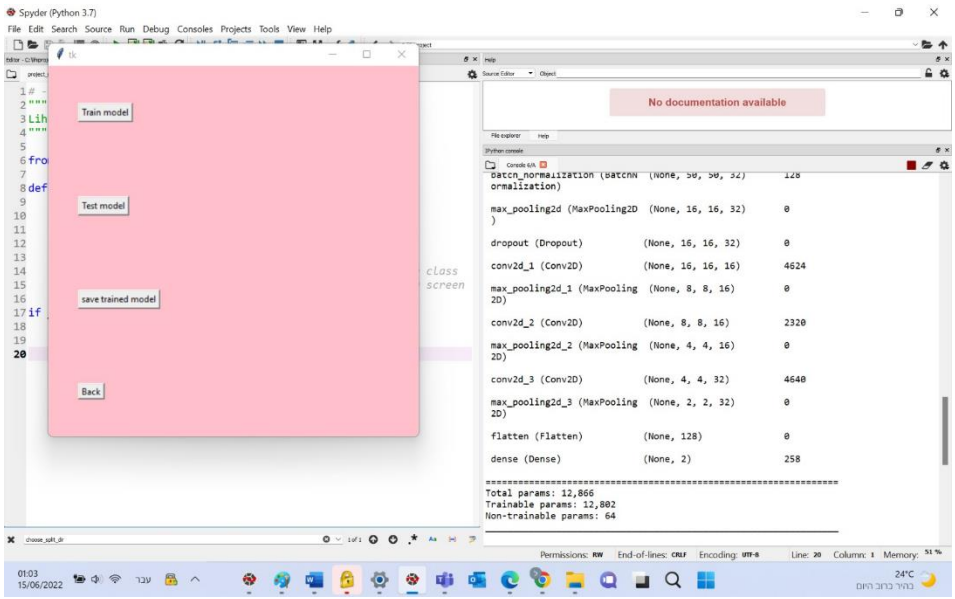
כפתור 2- כפתור שאומר לטעון מודל שאומן קודם לכן, כלומר קובץ h5 של מודל מאומן.

כאשר נלחץ על כפתור זה תיפתח חלונית נוספת שבה המשתמש יצטרך לבחור את הקובץ של המודל המאומן (trained_model2.h5) . כפתור זה יכול לשמש את המשתמש במידה והוא כבר הריץ את המודל פעם אחת ואין ברצונו לאמן את המודל שוב, או במידה והמשתמש הוריד את קובץ h5 של המודל המאומן שקיבל מבעוד מועד ורוצה לחסוך בזמנים ורק לבדוק את המודל.



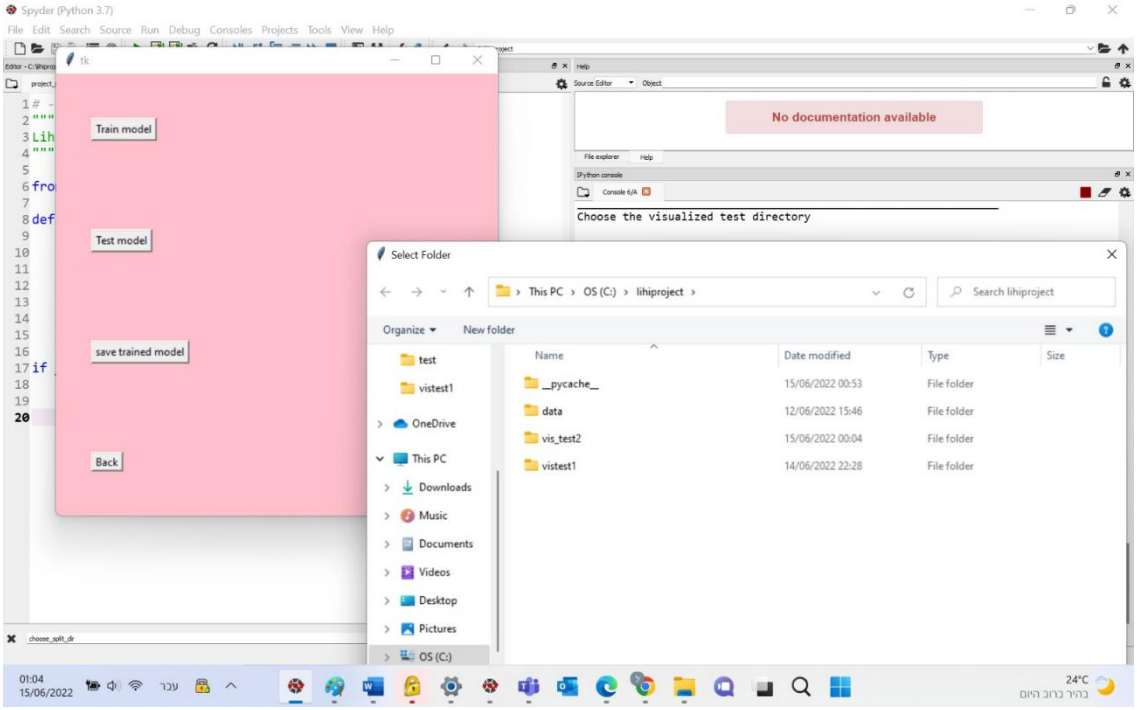
<u>מסך 3-</u> תפריט 3 שיוצג למשתמש מכיל 4 כפתורים והוא אחראי על אימון/ בדיקת / שמירת המודל, כמו כן הוא גם יכול חזור לתפריט הקודם במידה והמשתמש ירצה לבחור מודל אחר.

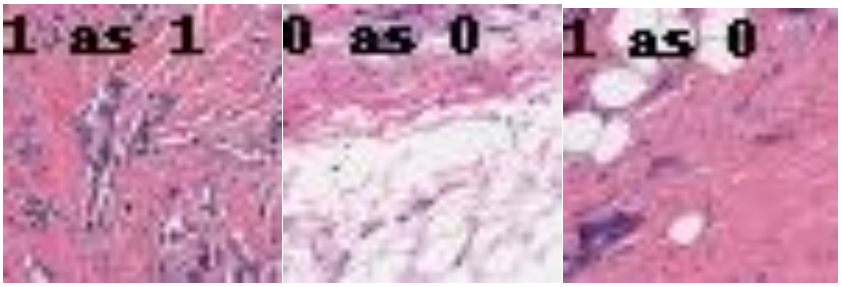
כפתור 1- כפתור שאומר אימון המודל, נוכל לראות בזמן ההרצה את התקדמות אימון המודל.



כפתור 2- כפתור שאומר בחינת המודל, כאשר נלחץ על כפתור זה תיפתח לנו חלונית חדשה שבה נצטרך לבחור תיקיה ריקה/ ליצור אחת ששם לאחר בחינת המודל נוכל לראות באופן ויזואלי "סיווג" של 100 תמונות .

לאחר שיסיים להריץ נוכל לראות מידת דיוק ושגיאה של המודל המאומן.

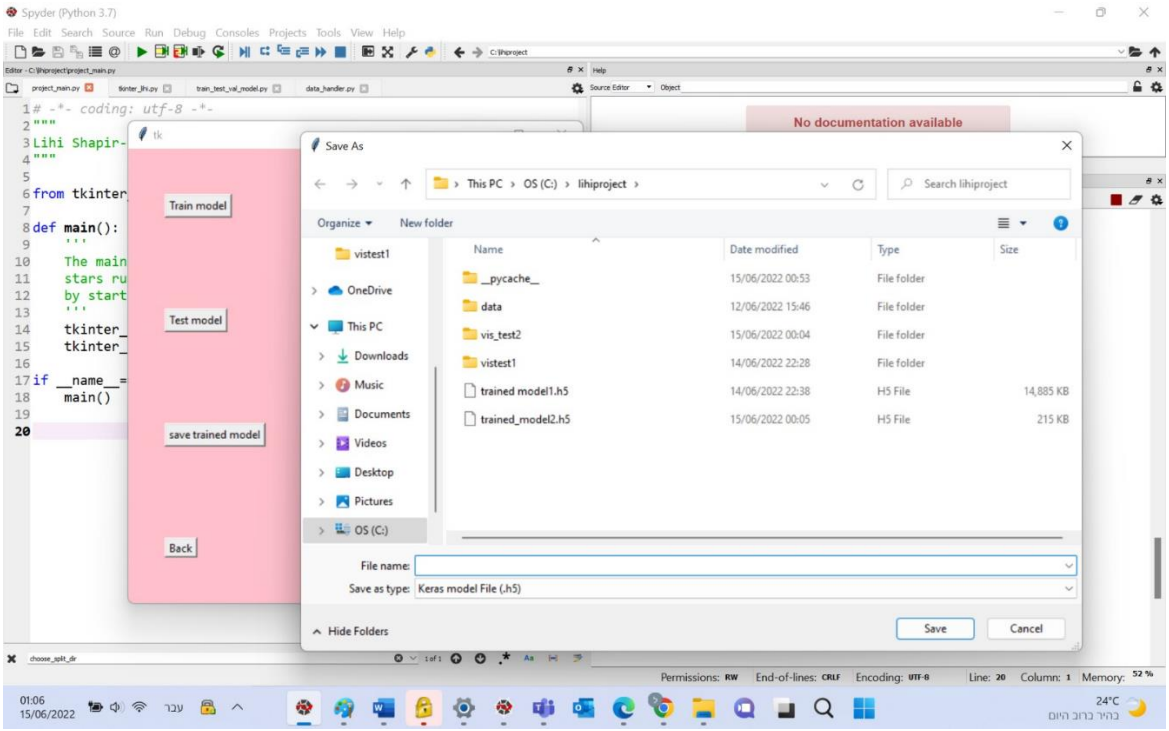




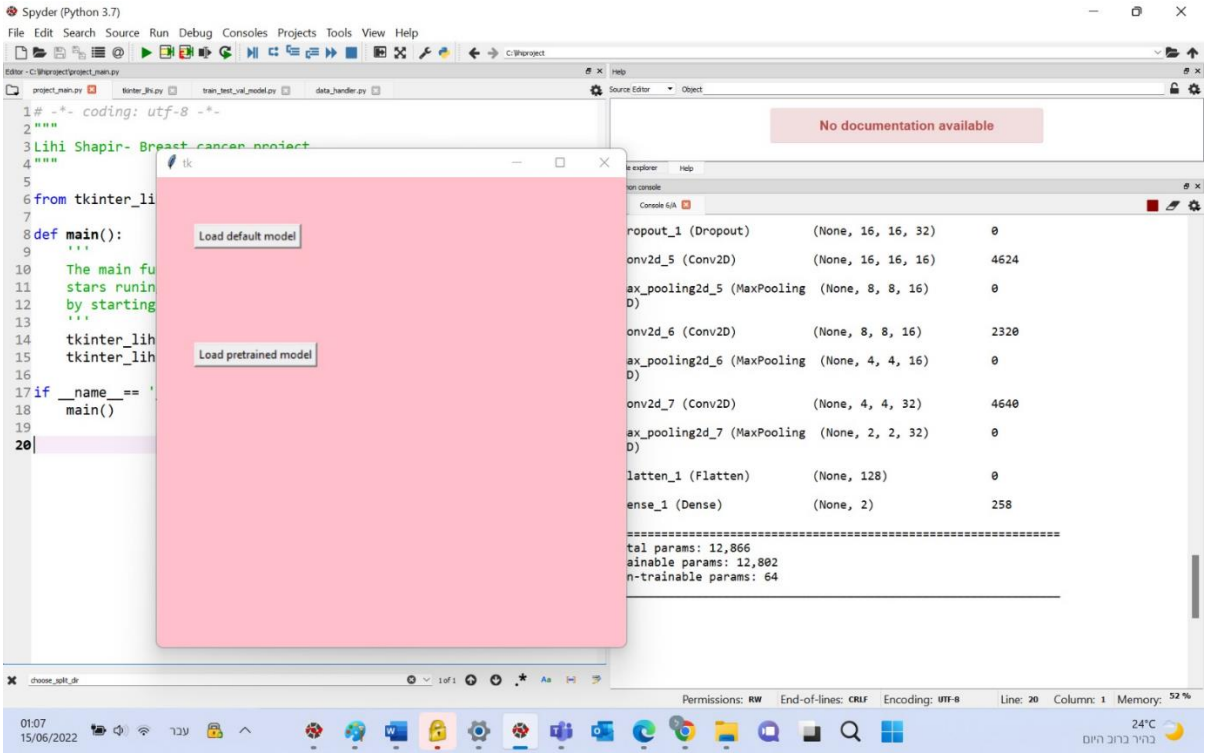
כאן למעלה נוכל לראות 3 תמונות מקובץ vis_test2 – תמונות שמסווגות באופן ויזואלי במהלך בחינת המודל. משמאל ניתן לראות תמונה עם סרטן שסווגה כסרטן, באמצע ניתן לראות תמונה בלי סרטן שסווגה בלי סרטן, ומימין נוכל לראות תמונה עם סרטן שסווגה ללא סרטן. לתמונות נוספות יש להיכנס לקובץ הvis_test2 בGithub שלי.

כפתור 3- כפתור שאומר שמירת מודל מאומן, כלומר לאחר אימון המודל ובדיקתו אם אנחנו מרוצים מהתוצאות שקיבלנו (accuracy, loss) נוכל לשמור את המודל המאומן כדי שאם נרצה להריץ שוב נוכל רק לבחון את המודל על המודל שאומן מראש.

כאשר נלחץ על כפתור זה תיפתח לנו חלונית חדשה שם נצטרך לבחור שם לקובץ המודל המאומן.



כפתור 4- כפתור שאומר לחזור אחורה, כלומר יחזור לתפריט הקודם תפריט 2. כפתור זה יכול לשמש במידה ורוצים לטעון מודל אחר.



רפלקציה

עבורי העבודה על הפרויקט הייתה מאתגרת מאוד מפאת לחץ הזמנים שהייתי צריכה לעמוד בהם, וכן התקלות שהיו במהלך הרצות הפרויקט ואתגרים שהייתי צריכה לפתור ולהתגבר תוך כדי התהליך. אך עם זאת הפרויקט היווה עבורי נקודת דרך חשובה במהלך הלימודים התיכוניים שלי ולימד אותי רבות על סביבת העבודה ב"עולם האמיתי" ועל איך להתמודד עם קשיים ובעיות שאין דרך אחת נכונה לפתור אותן(למידת מכונה).

מהכנת הפרויקט קיבלתי הרבה ידע לחיים כמו איך ללמד את עצמי חומר רב במיוחד כשרובו הוא לא בשפת האם שלי, למדתי מתי לבקש עזרה מחברי ומתי להתעקש על להבין לבד, כי ככה החומר ילמד בצורה הטובה ביותר, ובנוסף למדתי איך לתכנן את הזמנים שלי נכון כך שאספיק את כל המשימות שלי בזמן.

התמודדתי עם קשיים במהלך הכנת הפרויקט כמו למידת חומר ממאמרים ארוכים כשלא תמיד יש לך הנחיה(החליפו לנו מורים במהלך השנה ולכן הייתה תקופה שהיינו ברשות עצמנו), היה אתגר גדול בהתמודדות עם החוסר וודאות שהפרויקט בנושא זה מביא לפנינו, אך משלל האתגרים שהתמודדתי איתם מסקנתי היא שלמידה עצמית היא כלי חשוב מאוד שאלמלא רכשתי אותו היה לי יותר קשה בעתיד, וכמובן שזה משפיע גם על מקצועות אחרים בהם הייתה לי למידה עצמית יעילה בזכות זה, למדתי על חשיבות הזמן שאנחנו מנהלים ועל חשיבות הלמידה מאנשים אחרים שבהרבה מקרים יכולים ללמד אתכם דברים חדשים שיעזרו לכם בשלב כלשהו גם אם זה לא תמיד נראה כך.

אני חושבת שאם הייתי מתחילה את הפרויקט היום הייתי משנה את הזמן שבו התחלתי לעבוד על הספר פרויקט לשלב מוקדם יותר השנה, ומבינה את חשיבותו ואת הזמן שהוא יצרוך ממני, כי הוא מצריך שעות רבות של עבודה והשקעה. במידה וזה היה קורה העבודה שלי הייתה הרבה יותר יעילה ואפקטיבית.

לסיכום, אני אסכם ואגיד שהפרויקט היה מאתגר ומהנה בו זמנית, אני לא מתחרטת על השעות הרבות והעבודה הקשה שהשקעתי בו מפני שלמדתי ממנו רבות ואני מקווה שהתחום הזה יתפתח ויגדל בארץ. זהו תחום שיכול להוביל להרבה דברים, רעיונות ופרויקטים טובים ומשני עולמות, גם בעולם של תחום המחשבים וגם בעולם המודרני שאנחנו חיים בו כיום.

ביבליוגרפיה

קישור לאתר Kaggle ממנו נלקח מאגר הנתונים:

https://www.kaggle.com/datasets/paultimothymooney/breast-histopathology-images

מאמר על זיהוי סרטן השד באמצעות למידת מכונה:

https://www.atlantis-press.com/article/125960864.pdf

הסבר על רשתות קונבולוציה:

https://he.wikipedia.org/wiki/%D7%A8%D7%A9%D7%AA_%D7%A7%D7%95%D7%A0%D7%91%D7%95%D7%9C%D7%95%D7%A6%D7%99%D7%94

שימוש בספריית glob:

https://www.codegrepper.com/code-examples/python/glob+find+all+files+recursively

הסבר על אימון ובחינת המודל:

https://www.tensorflow.org/guide/keras/train_and_evaluate

אתר להורדת ספריות שונות:

https://pypi.org/search/?q=

הסברים על הסוגים השונים של optimizers:

https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

אתרים שמהם קיבלתי מידע על under fitting ו over fitting:

https://www.baeldung.com/cs/training-validation-loss-deep-learning

https://deeplizard.com/learn/video/0h8lAm5Ki5g

https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning

אתר שממנו למדתי איך לסווג תמונות באופן וויזואלי:

https://pillow.readthedocs.io/en/stable/reference/ImageDraw.html

שימוש ב Tkinter :

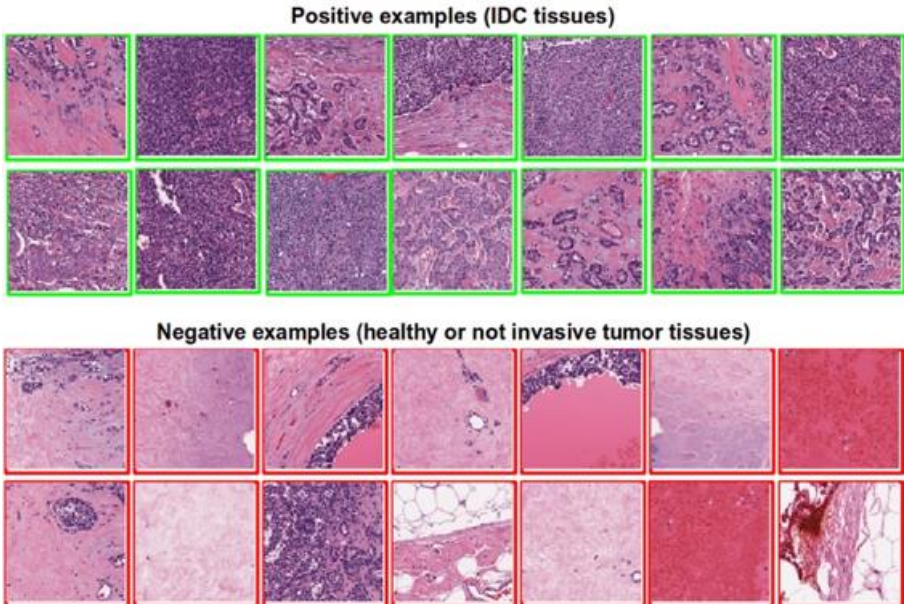
https://docs.python.org/3/library/tkinter.html

כמו כן, הרבה מהחומר הנלמד קראתי ולמדתי ממסמכים שנשלחו על ידי המורה (דינה קראוס).

אתר שמציג דוגמא לתוצאות של מודל אחר העובד עם אותו מאגר נתונים:

https://www.kaggle.com/code/paultimothymooney/predicting-idc-in-breast-cancer-histology-images/notebook

נספחים



מספר תמונות שהמודל סיווג נכון:

מספר תמונות שהמודל סיווג לא נכון: