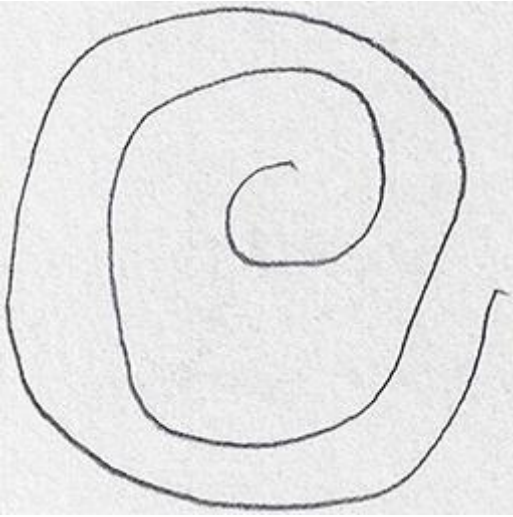
פרויקט גמר 5 יחידות לימוד

התמחות – למידת מכונה Deep Learning

נושא הפרויקט: Parkinson's Disease Detection

Parkinson



healthy

מגישה: יהל דרין

תעודת זהות: 326626850

כיתה: י"ב 4

שם המורה: דינה קראוס

בית ספר: מקיף י"א "ראשונים"

תאריך הבחינה: 3/7/2022

תוכן עניינים

מבוא

השנה במסגרת לימודינו הטכנולוגיים, עסקנו בלמידת מכונה (Machine Learning), בהתמחות של "למידה עמוקה" (Deep Learning Computer Vision). המכונה לומדת לנתח תהליכי לימוד אשר יהיו דומים ככל האפשר ליכולות החשיבה של המוח האנושי. מוח האדם בנוי מרשת נוירונים בהם נקלט המידע ובעזרת התקשורת שבין כל תא עצב לאילו שבקרבתו נוצרת היכולת ללמוד דברים חדשים ולזכור את מה שלמדנו. על רעיון זה מתבסס הפרויקט שלי.

בפרויקט זה בחרתי לבנות תוכנית אשר מטרתה לזהות את מחלת הפרקינסון באמצעות ציור ספירלי. פרקינסון היא מחלה נוירולוגית קשה שגורמת לפגיעה הדרגתית בתנועות הגוף. לאחר תהליך הלמידה של המודל והרצה של אלפי תמונות שונות, התוכנית תוכל לזהות מי הוא אדם בריא ומי הוא אדם החולה במחלת הפרקינסון. קהל היעד הם אנשים מבוגרים אשר מתחילים לזהות סימנים המאפיינים את המחלה או אנשים החוששים ממנה לצורך אבחנה ראשונית. בחרתי בנושא זה הקשור לתחום הרפואה, נושא שמעניין אותי מאוד, ורוצה להתעמק בו בעתיד. מחקירת הנושא הבנתי שאין בדיקה לאבחון מוקדם של מחלת הפרקינסון ולכן יכול להיות קושי בקביעת האבחנה הראשונית, במיוחד בשלבים המוקדמים יותר של המחלה. על כן מצאתי לנכון לבנות תוכנית המזהה את המחלה בשלב מוקדם עם תחילת הופעת התסמינים.

בסקירה של המצב הקיים בשוק מצאתי כי ישנן תוכניות מעטות המנסות לתת מענה לאבחון מחלה זו. תוכנות אלה אף הן מבוססות על Machine Learning ויש להן את היכולת לזהות את המחלה. הפרויקט שלי כולל תהליך של העלאת dataset, אימון המודל ובחינתו.

במהלך כתיבת הפרויקט נתקלתי במספר קשיים. האתגר המשמעותי היה שתחום למידת מכונה הוא תחום חדש בשבילי והיה עליי ללמוד את הנושא באופן עצמאי, להבין אותו וליצור פרויקט המתבסס על נושא זה. קושי מרכזי נוסף שחוויתי הוא, כיצד לכתוב את התוכנית ואת המודל עצמו, ולכן היה עלי לחפש ולמצוא פתרונות שונים דרך האינטרנט. ישנן גרסאות רבות לפתרונות שחיפשתי אשר כל אחד עולה על השני עד לקבלת הגרסה הסופית. כמו כן, מקורות המידע באינטרנט היו כתובים באנגלית ברמה גובהה עם מושגים מקצועיים, אנגלית אינה שפת האם שלי והייתי צריכה להתמודד עם ההבנה של הכתוב. נעזרתי כמובן במורתי דינה ובחברי לכיתה ששמחו לתת יד להתקדמות הפרויקט.

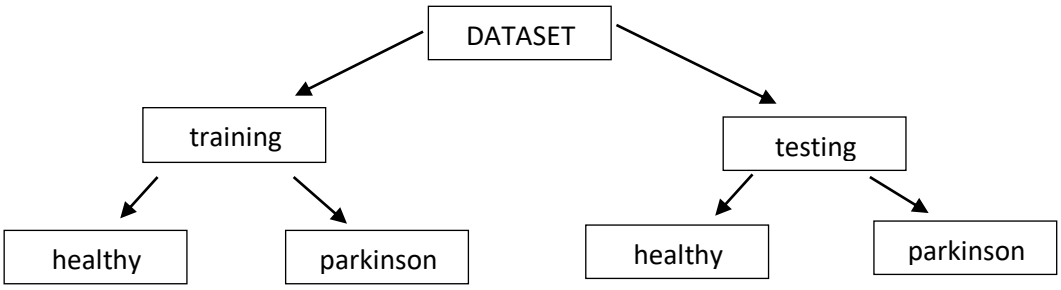
האתגר הגדול היה לכתוב ולהגיע למודל המדמה את שכבות הנוירונים במוח האנושי, אשר יביא לאבחנה המדויקת ביותר.

מבנה הפרויקט

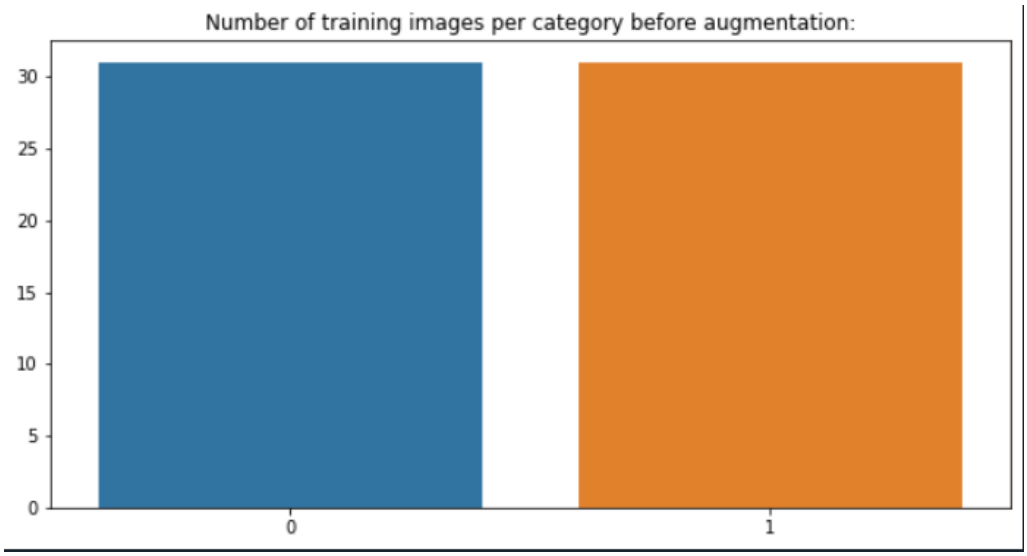
שלב איסוף הכנה וניתוח הנתונים (Collect, prepare and analyze data)

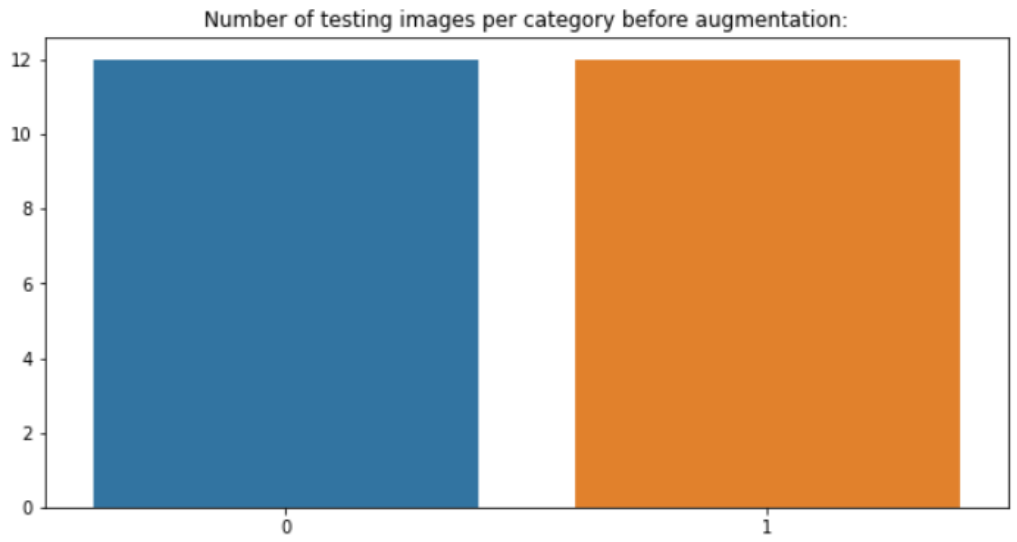
ה-dataset מורכב מקובצי תמונות מסוג png, אותו לקחתי מאתר האינטרנט "Kaggle", ותמונות שציירתי בעצמי בתוכנת "צייר". ה-dataset מכיל ציורי ספירלה וגלים שנעשו על ידי אנשים בריאים ועל ידי אנשים החולים במחלת הפרקינסון. בפרויקט זה השתמשתי רק בציורים הספירלים.

ה-dataset מחולק כבר לספריות של training set ו-testing set, כך שאין צורך לפצל את ה-dataset במהלך ההרצה. הספריות מחולקות באופן הבא –



ניתן להמחיש את חלוקת הנתונים של training set ו-testing set באמצעות התרשים הבאים –





כפי שניתן לראות, ה-dataset כבר מאוזן. עם זאת, הוא מכיל פחות תמונות באופן משמעותי לכל קטגוריה הן ב-training set והן ב-testing set. לכן, כחלק מתהליך הכנת הנתונים לאימון השתמשתי ב-

tensorflow.keras.preprocessing.image.ImageDataGenerator

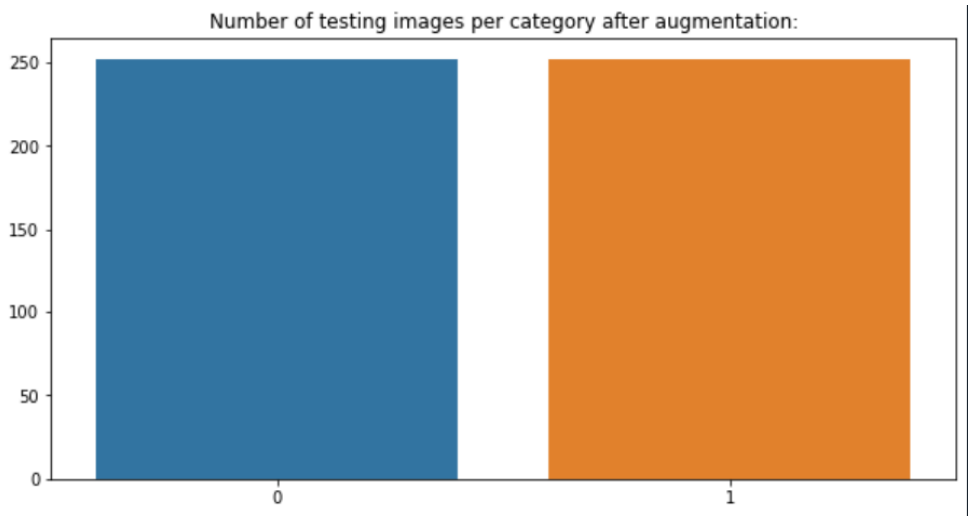
המאפשר לנו לעשות augmentation לתמונות באקראיות ובקלות.

בעזרתו עשיתי על התמונות שלי סיבובים בטווח של 360 מעלות, שיניתי את הבהירות פי 0.5-1.5, והפכתי את התמונות בצורה אנכית ואופקית.

בנוסף, שיניתי את התמונות לגודל (128,128,1), שנחשב לפשוט ונפוץ יותר.

לפני התאמת ה-dataset למודל, התמונות עוברות נורמליזציה.

חלוקת הנתונים לאחר ה-augmentation –



שינויים כאלה לתמונות מאפשר למודל לאמן בצורה מיטבית, מכיוון שנותנים לו מגוון גדול יותר של תמונות שהוא לומד מהן, וכך ישתפר הדיוק של המודל על תמונות חיצוניות.

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

תיאור גרפי של המודל-

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv2D)              (None, 128, 128, 128)     1280

 max_pooling2d_3 (MaxPooling  (None, 62, 62, 128)      0
 2D)

 conv2 (Conv2D)              (None, 62, 62, 64)        73792

 max_pooling2d_4 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 conv2d_1 (Conv2D)           (None, 30, 30, 32)        18464

 max_pooling2d_5 (MaxPooling  (None, 29, 29, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 26912)             0

 dropout_4 (Dropout)         (None, 26912)             0

 dense_4 (Dense)             (None, 64)                1722432

 dropout_5 (Dropout)         (None, 64)                0

 dense_5 (Dense)             (None, 32)                2080

 dropout_6 (Dropout)         (None, 32)                0

 dense_6 (Dense)             (None, 16)                528

 dropout_7 (Dropout)         (None, 16)                0

 dense_7 (Dense)             (None, 1)                 17

=================================================================
Total params: 1,818,593
Trainable params: 1,818,593
Non-trainable params: 0
_________________________________________________________________
```

הסבר על השכבות-

שכבה 1 – conv2d

Input shape	128x128x1
Filters	128
Kernel size	3x3

שכבה זו עוברת על התמונה עם מטריצה בגודל 5x5 ומחדדת את התמונה ואת האלמנטים בה.

שכבה 2 – maxpooling2d

Input shape	128x128x1
Pool_size	5x5
strides	2x2
Output shape	62x62

שכבה זו עוברת על התמונה עם מטריצה בגודל 5x5 בקפיצות של 2x2 ולוקחת את הפיקסל הגדול מכל מקום שנמצאת בו.

שכבה 3 – conv2d

Input shape	62x62
Filters	64
Kernel size	3x3

שכבה 4 – maxpooling2d

Input shape	62x62
Pool_size	3x3
strides	2x2
Output shape	30x30

שכבה 5 – conv2d

Input shape	30x30
Filters	32
Kernel size	3x3

שכבה 6 – maxpooling2d

Input shape	12x12
Pool_size	2x2
strides	1x1
Output shape	29X29

שכבה 7 – flatten

Input shape	29X29X32
Output shape	26912

משטיחה את התמונה והופכת אותה למערך חד ממדי שאפשר לאמן עליו את המודל

שכבה 8- dense

Input shape	26912
Neurons	64
Output shape	64

השכבה הפשוטה ביותר, מפעילה את ה-activation function על התמונה.

על שכבה זו מופעל dropout של 0.5, מה שגורם לחצי מהנוירונים להיכבות באקראיות במהלך האימון.

שכבה 9 – dense

Input shape	64
Neurons	32
Output shape	32

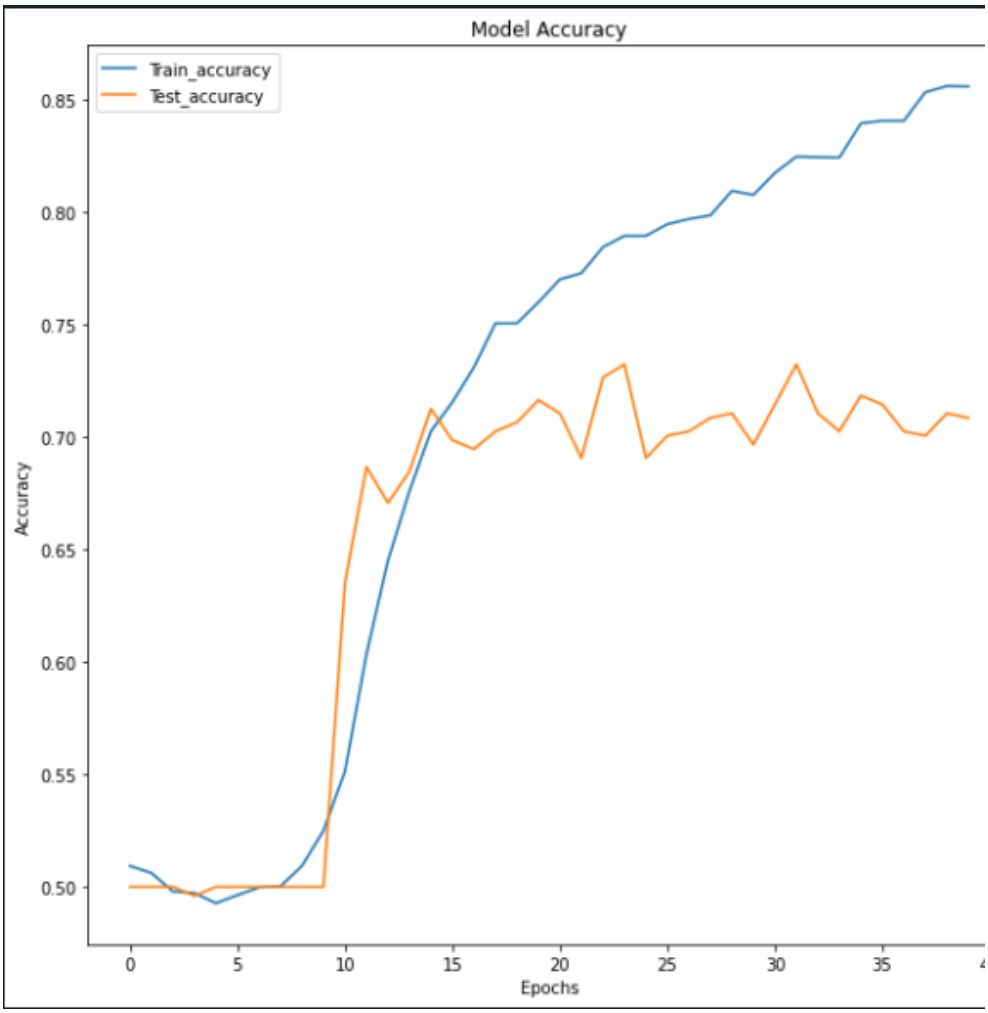
שכבה 10 – dense

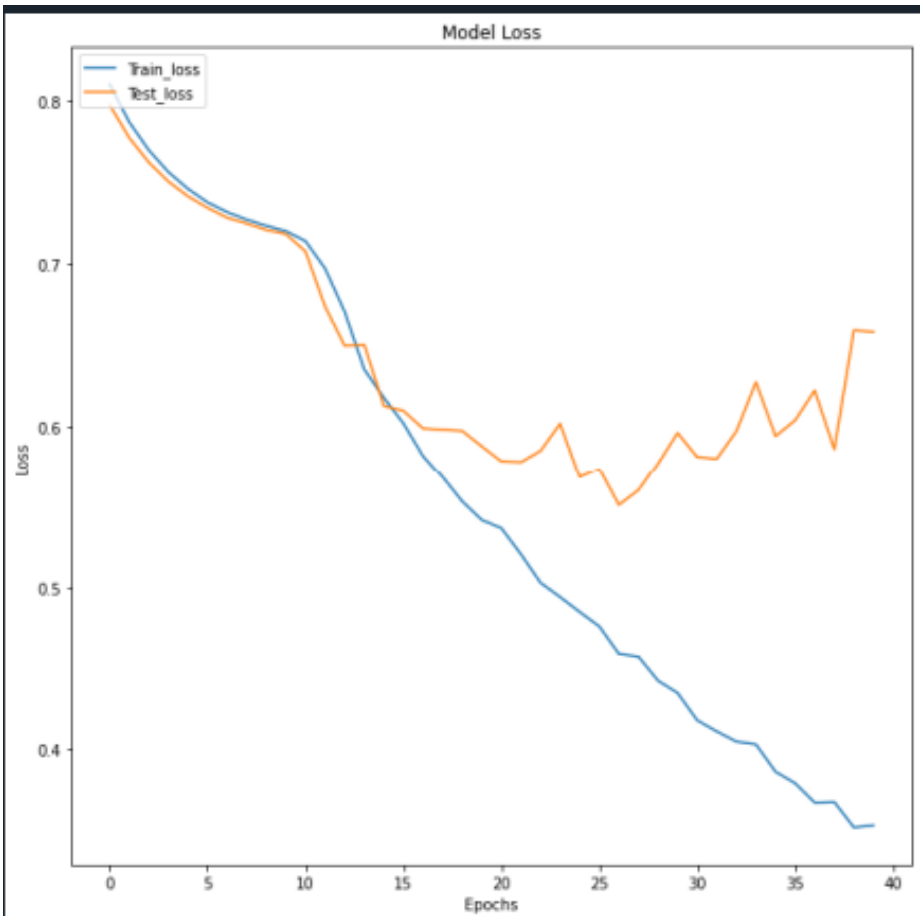
Input shape	32
Neurons	16
Output shape	16

שכבה 11 – dense

Input shape	16
Neurons	1
Output shape	1

תוצאות הרצת המודל-





מהתוצאות ניתן לראות שהשגיאה נמצאת במצב טוב, פחות מ-0.5 אך ה-accuracy מעל 0.85, כך שהמודל נמצא בטווח הרצוי אך לא מושלם.

Hyperparamters-

Number of hidden layers = 11
Dropout = 0.5
Activation function = relu, sigmoid
Weights initialization = random
Learning rate = 0.001
Epoch = 40, iterations and batch size = 20

פונקציית השגיאה -

בחרתי להשתמש בפונקציית השגיאה – categorical cross entropy

פונקציה זו מתאימה לבעיות מסוג classification שמבדילים בין 2 או יותר סוגים.

הפונקציה מכפילה את התוצאה האמיתית ב-log של התוצאה.

כיוון שהתוצאה האמיתית בסוג שאינו הסוג הנכון הוא 0, נקבל בסוף רק את ה - log של התוצאה הנכונה. לכן ככל שהמודל יהיה יותר מדויק, ה-loss יהיה נמוך יותר.

ייעול ההתכנסות –

אני בחרתי ב-optimizer מסוג Adam.

Adam זה ראשי תיבות ל - adaptive moment estimation. היתרון של Adam הוא שבניגוד ל-optimizer אחרים, הוא משנה לא רק את המשקלים של המודל אלא גם את קצב הלמידה ובכך מתאים את עצמו למצב בו נמצא.

היתרון בא לידי ביטוי בקצב הלמידה, שהוא הרבה יותר מהיר, ובכך שהוא מכביד על הזיכרון הרבה פחות.

החיסרון של Adam הוא שהיתרון במהירות, עולה בדיוק של המודל והתוצאות עלולות להיות יותר נמוכות.

שלב היישום (Software deployment)

היישום משתמש במודל במספר דרכים.

הראשונה, המשתמש יכול לבחור לאמן המודל. התוכנית נכנסת לקובץ שבו נמצא המודל, ומאמנת אותו בעזרת התמונות שעשתה להם augmentation לפני כן.

השנייה , המשתמש יכול לבחור לבחון את המודל. התוכנית נכנסת לקובץ Test_the_Model, שם היא מעלה את המודל השמור בתיקיית הקבצים של התוכנית ובודקת את ה-loss וה-accuracy של המודל.

השלישית, המשתמש יכול לבחור לחזות רק תמונה אחת שהוא בוחר. התוכנית נכנסת לקובץ Test_the_Model, שם היא מעלה את המודל השמור בתיקיית הקבצים, ועוברת על הפעולה החוזה את המצב של תמונה אחת שהמשתמש בוחר בעזרת המודל.

היישום כולו מופיע בשורת הפקודה (Command Prompt) CMD.

שורת הפקודה CMD, הינו ממשק מערכת הפעלה של Windows והוא משמש להריץ פקודות טקסטואליות שהוזנו על ידי המשתמש. ממשק שורת הפקודה הוא טקסטואלי, בניגוד לממשקים גרפיים אחרים שהתוכנה או מערכת ההפעלה יכולות לספק. ממשק שורת הפקודה נמצא בהמתנה לפקודות מהמשתמש: במקרה שפקודה הוקשה ונלחץ מקש ה-"ENTER", הוא יעבד אותה מיידית.

המשתמש יכול לבחור להוריד ולהכין את הdataset. התוכנית תיכנס לקובץ Organize_the_Data, שם היא ממירה את ה-dataset שנמצא בקובץ מסוג zip, ועושה לו augmentation. תוך כדי, היא מוסיפה את כל התמונות לרשימה אחת, ויוצרת רשימה נוספת עם התוויות המתאימות לכל תמונה.

מדריך למפתח

הפרויקט שלי מחולק למספר קבצי קוד (מספר מודולים שונים). חלוקה זו בין חלקי הקוד השונים מאפשרת ארגון קוד, ממזערת טעויות ואף גם אפשרה לי לבצע את הפרויקט ביתר קלות.

להלן שמות הקבצים והתפקיד שהקצעתי לכל אחד מהם:

תפקיד הקובץ:	שם הקובץ:
קובץ זה מנהל את כל התוכנית, ממנו נשלחות הפקודות לשאר הקבצים	Main.py
קובץ זה אחראי על התקשורת עם המשתמש	User_App.py
בקובץ זה מתבצעת הכנת ה-dataset לאימון המודל	Organize_the_Data.py
בקובץ זה נמצאות כל פעולות המדפיסות מידע ויזואלי על ה-dataset והתוכנית.	Show_Data.py
קובץ זה אחראי לבניית המודל	The_Model.py
קובץ זה אחראי להעלאת המודל השמור, בחינת המודל על ידי ה-test set ואחראי על חיזוי תמונה שהמשתמש בוחר.	Test_the_Model

הסבר על כל אחד מן הקבצים:

קובץ ראשי – Main.py

קובץ זה, הוא הקובץ המרכזי של הפרויקט. יש בו פעולה אחת (main) ממנה ניתן לפנות אל כל שאר הקבצים בתוכנית. קובץ Main.py אחראי להפעיל את הפונקציות הנמצאות בשאר הקבצים ולשלוח אליהם את המשתנים המתאימים. לפי האופציה שבוחר המשתמש, הפונקציה main יוזמת את תחילת הפונקציות הרצויות.

אם המשתמש בחר באופציה הראשונה – להכין את ה-dataset לאימון המודל. הפונקציה אחראית להוריד את ה-dataset, להכניס אותו לרשימות המתאימות ולנרמל אותן.

אם המשתמש בחר באופציה השנייה – לאמן את המודל. על הפונקציה לקרוא לפונקציה שתיצור את המודל, לאחר מכן לאמן אותו, ולבסוף לשמור אותו על הדיסק.

אם המשתמש בחר באופציה השלישית – לבחון את המודל. הפונקציה מעלה את המודל השמור ולאחר מכן בוחנת אותו עם ה-testing set.

אם המשתמש בחר באופציה הרביעית – לחזות תמונה אחת. על הפונקציה להעלות את המודל מהדיסק, לתת למשתמש להכניס את כתובת התמונה, ולאחר מכן לחזות האם זוהי תמונה של אדם בריא או חולה.

המשתנים הנמצאים בקובץ:

הסבר על המשתנה:	המשתנה:	הפונקציה:
משתנה בוליאני. שווה ל-False כאשר המשתמש עדיין לא בחר להוריד את התמונות ולהכין אותם לקראת אימון המודל. שווה ל-True לאחר שבחר באופציה הראשונה.	flag_option1	main
כאשר המשתנה ירצה להפסיק את התוכנית,	user_want	

המשתמש יקלוט לתוך המשתנה את המילה "Exit".		
מכיל את הכתובת של ה-dataset	dir_path	
רשימה מסוך מערך של numpy. מכילה את כל התמונות של ה- train set.	x_train	
רשימה של התוויות המתאימות לתמונות הנמצאות ברשימה x_train.	y_train	
רשימה מסוך מערך של numpy. מכילה את כל התמונות של ה-test set.	x_test	
רשימה של התוויות המתאימות לתמונות הנמצאות ברשימה x_test.	y_test	
מכיל את המודל	model	
מכיל את המודל המאומן	hist	

קובץ ממשק המשתמש – User_App.py

קובץ האחראי על ממשק המשתמש.

הקובץ מציע למשתמש את אפשרויות התוכנית, קולט את בחירת המשתמש תוך כדי בדיקת האפשרות אותו בחר (כלומר, האם יכול לבחור באפשרות שבחר), ובסופו של דבר מעביר הלאה את בחירת המשתמש.

הפונקציות המצויות בקובץ:

תפקידה:	**שם הפונקציה:**
הפונקציה מקבלת משתנה בוליאני הקובע אם המשתמש כבר בחר באפשרות מספר 1 (True) או לא (False). הפונקציה בודקת האם המשתמש יכול לבחור באופציה שבחר. אם המשתמש בחר באופציה אפשרית, הפונקציה מחזירה את האופציה שבחר, ואם הוא לא יכול לבחור באופציה הזאת, הפונקציה מדפיסה אזהרה מתאימה למשתמש ושולחת אותו לבחור אופציה אחרת. לבסוף הפונקציה מחזירה גם את flag_option1 שוב.	check_option(flag_option1)
הפונקציה מציגה למשתמש את האפשרויות של התוכנית, ומבקשת ממנו לבחור אחד: 1. להכין את ה-dataset 2. לאמן את המודל 3. לבחון את המודל 4. לחזות תמונה אחת אם המשתמש בחר מספר שהוא לא אחד מן האפשרויות, הפונקציה תשלח לו אזהרה מתאימה ותיכנס על הפעולה הזאת שוב. לבסוף תחזיר את האופציה שבחר.	text()

המשתנים הנמצאים בקובץ:

הסבר על המשתנה:	**המשתנה:**	**הפונקציה:**
האופציה שבחר המשתמש מסוג str. יכול להכיל רק את המספרים 1/2/3/4.	option	check_option
האופציה שבחר המשתמש מסוג str. יכול להכיל רק את המספרים 1/2/3/4.	option	text()

קובץ המארגן את הdataset – Organize_the_Data.py

קובץ זה אחראי לכל הפעולות הקשורות להכנת ה-dataset לאימון המודל.

הקובץ ממיר את ה-dataset מקובץ מסוג zip, מוריד את ה-dataset ומגדיל אותו, מסדר את התמונות לגודל ולצבע הרצוי, מכניס את התמונות לרשימות training ו-testing מתאימות והופך אותם ל-numpy array.

הפונקציות המצויות בקובץ:

תפקידה:	שם הפונקציה:
הפונקציה ממירה את קובץ ה-dataset שנמצא בקובץ zip.	extract_zip_file()
הפונקציה מקבלת את הכתובת של ספריית התמונות הרצויה להורדה. הפונקציה מורידה כל תמונה ומקטינה את הגודל שלה. כל תמונה היא הופכת למערך numpy יחיד, ולאחר מכן מצרפת אותה למערך המכיל את כל התמונות מסוג numpy. את המערך של כל התמונות היא מקבצת לקובץ zip מסוג numpy. בנוסף הפונקציה מבצעת סיווג בינארי, לפי שם הקובץ הפונקציה קובעת האם לסווג אותה כ-0 (אדם בריא) או כ-1 (אדם חולה). את תוצאות הסיווג מוסיפה ל-y_list בהתאם לאיברים ב-x_list. הפעולה חוזרת פעמיים. פעם ראשונה לספריית ה-training, ופעם שנייה לספריית ה-testing. לבסוף, מחזירה את הרשימות x_list ו-y_list של ספריית התמונות.	loading_dataset(path_to_files)
הפונקציה מבצעת augmentation על התמונות. היא מרחיבה את ה-dataset על	augmenting_the_dataset(x_list, y_list, num)

ידי סיבוב ושינוי הבהירות של התמונות. ותוך כדי מגדילה גם את רשימת התוויות y_list לפי התמונות שנמצאות ב-x_list. מחזירה את x_list ו-y_list לאחר השינויים.	
הפונקציה מקבלת את הרשימות של ה-training וה-testing. הפונקציה הופכת את כל התמונות לגווני אפור על מנת להפוך את מערך ה-numpy של התמונה לפשוט יותר. ולאחר מכן מנרמלת את הערכים שברשימות. לבסוף הפונקציה מחזירה את הרשימות של ה-training וה-testing לאחר הנירמול.	preprocessing_the_images(x_train, y_train, x_test, y_test)

המשתנים הנמצאים בקובץ:

הסבר על המשתנה:	**המשתנה:**	**הפונקציה:**
הכתובת של קובץ ה-zip של ה-dataset	zip_path	extract_zip_file()
תמונה מסוג png	single_im	loading_dataset
single_image כמערך מסוג numpy	single_array	
מערך מסוג numpy המכיל את המערכים single_array המייצגים את התמונות.	array_of_images	
קובץ zip מסוג numpy המכיל את כל התמונות שירדו.	data_train	
רשימה של מערכים מסוג numpy. כל מערך מייצג	x_list	

		תמונה.
	y_list	רשימה שבה מסווגת כל תמונה ל-0 (אדם בריא) או ל-1 (אדם חולה), בהתאם לאיברים ב-x_list.
augmenting_the_dataset	x	רשימה המכילה את x_list
	y	רשימה המכילה את y_list
	x_aug	רשימה המכילה את ה-augmented data
	y_aug	רשימה המכילה את התוויות המתאימות של התמונות ברשימה של ה-augmented data
	x_img	מכיל תמונה אחת מה-x_list
	aug_image	מכיל את ה-x_img לאחר ה-augmentation
preprocessing_the_images	train	רשימה המכילה את כל התמונות הנמצאות ב-x_train בגווני אפור
	test	רשימה המכילה את כל התמונות הנמצאות ב-x_test בגווני אפור
	img	כל פעם מכיל תמונה אחת מה-x_train / x_test

קובץ המציג מידע על מהלך ההרצה – Show_Data.py

קובץ זה אחראי להציג מידע ויזואלי על מצב התוכנית וה-dataset במהלך ההרצה.

הקובץ מציג את המידע המתאים לשלב שבחר המשתמש.

הפונקציות המצויות בקובץ:

תפקידה:	שם הפונקציה:
הפונקציה מקבלת את רשימת התוויות של ה-training set וה-testing set, ואת המצב שבו נמצא ה-dataset (לפני או אחרי augmentation). הפונקציה מחלקת כל תוכן של רשימת תוויות לשתי קטגוריות – 0 (בריא) ו-1 (חולה). בעזרת החלוקה הפונקציה מייצרת גרף עמודות המראה את מספר החולים לצד כמות הבריאים באותו ה-set.	data_distribution(y_train, y_test, situation)
הפונקציה מקבלת את המודל המאומן. מטרתה של הפונקציה היא להציג את הגרף של ה-accuracy וה-loss של המודל.	loss_and_accuracy_plot(hist)

המשתנים הנמצאים בקובץ:

הסבר על המשתנה:	המשתנה:	הפונקציה:
מכיל את כותרת הגרף	title	data_distribution
מכיל את המודל המאומן.	hist	loss_and_accuracy_plot

קובץ המכיל את המודל – the_Model.py

קובץ זה אחראי על בניית המודל.

בקובץ זה ישנה פעולה אחת אשר מגדירה את מבנה המודל שעל גביו נבצע את למידת התמונות.

תפקידה:	שם הפונקציה:
הפונקציה מקבלת את ממדי התמונות ואת מספר קטגוריות. הפונקציה אחראית ליצירת המודל. התוכנית תשתמש במודל זה על מנת לסווג את התמונות לאדם בריא ולאדם החולה בפרקינסון. פעולה זו בונה מודל מסוג Sequential בעל ????? שכבות המחוברות ביניהן ומדמות את המוח האנושי. השכבה האחרונה משתמשת בפונקציית sigmoid, וגודל הפלט שלה הינו (None,1), כלומר יש ערך אחד שהמודל מחזיר – בריא/חולה. הפונקציה מאתחלת את המודל על ידי הפעולה compile(). לבסוף הפונקציה מחזירה את המודל שנבנה ואופס.	parkinson_disease_detection_model (input_shape=(128, 128, 1))

המשתנים הנמצאים בקובץ:

הסבר על המשתנה:	המשתנה:	הפונקציה:
המודל מסוג Sequential	model	parkinson_disease_detection_model

קובץ הבוחן את המודל – Test_the_Model.py

קובץ זה אחראי לכל הפעולות הקשורות לבחינת המודל.

הקובץ מעלה את המודל השמור בדיסק, ולפי בחירת המשתמש, הקובץ יכול לבחון את המודל על ידי ה-testing set, או לחזות תמונה אחת שהמשתמש בחר.

הפונקציות המצויות בקובץ:

תפקידה:	שם הפונקציה:
מטרתה של הפונקציה היא להעלות את המודל השמור בדיסק. הפונקציה מחזירה את המודל.	upload_the_model()
הפונקציה מקבלת את המודל ואת ה-testing sets. הפונקציה מודדת את הביצועים של המודל על ידי ה-testing sets, ומדפיסה את התוצאות.	test_model(model, x_test, y_test)
הפונקציה מקבלת את המודל, ולפיו חוזה תמונה אחת שהמשתמש בוחר ומסווגת אותו לבריא/חולה בפרקינסון. לבסוף היא מדפיסה את התמונה ואת תוצאות החיזוי.	testing_model_on_images(model)

המשתנים הנמצאים בקובץ:

הסבר על המשתנה:	המשתנה:	הפונקציה:
מכיל את הכתובת של המודל השמור.	path	upload_the_model
מכיל את המודל השמור.	model	
מכיל את תוצאות הביצועים של המודל.	results	test_model
מכילה את התוויות האפשריות לתמונה.	labels	Testing_Model_on_Images

מכיל את הכתובת לתמונה שהמשתמש בחר.	path	
מכיל את התמונה שהמשתמש בחר.	image	
מכיל את החיזוי של המודל לתמונה שהמשתמש בחר.	ypred	

מדריך למשתמש

לפני שאציג את האופן שבו עובד הפרויקט עובד, אסביר קודם את את התהליך שיש לעשות על מנת שהפרויקט יוכל לעבוד במחשבים אחרים.

הוראות התקנה:

1. יש להוריד למחשב את סביבת העבודה Anaconda.

   לינק להורדת Anaconda - https://www.anaconda.com

2. לאחר הורדת ה-Anaconda, יש להוריד Spyder 5.3.1 על ידי לחיצה על הלחצן Home, שם יופיע Spyder 5.3.1 -

3. יש להוריד מספר ספריית קוד בהן הפרויקט משתמש:

Library Name:	Version:
numpy	1.21.5
cv2	4.6.0.66
tensorflow	2.9.1
PIL	9.0.1
zipfiles	4.0.1
sklearn	1.0.2
matplotlib	3.5.1
seaborn	0.11.2
keras	2.9.0

4. יש להוריד את הקבצים הבאים מחשבון ה-GitHub שלי:
   - קובצי ה-python עליהם מתבסס המודל.
   - את תיקיית התמונות המועלות כקובץ zip בשם Dataset-spiral.
   - את תיקיית התמונות בשם testing images.
   - קובץ המודל השמור parkinson_disease_detection.h5

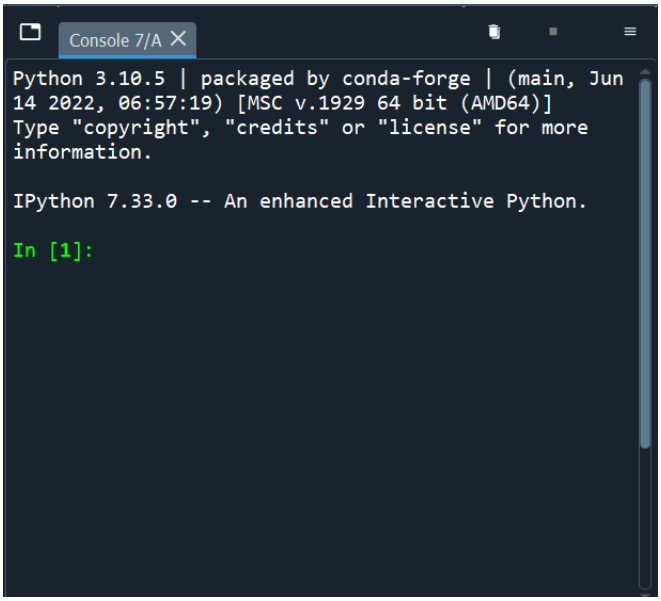
הערה חשובה: **אין לשנות את תוכן הקבצים**.

5. יש לשמור את הקבצים באופן הבא:
   - את קובצי ה-python ואת המודל השמור יש לשמור בתיקייה אחת.
   - התיקייה מסוג zip המכילה את ה-dataset, תישמר באיזה directory שהמשתמש יבחר.

הרצת התוכנית:

בכדי להריץ את התוכנית, יש להיכנס לתוכנה של Spyder 5.3.1, להעביר לשם את קובצי הפרויקט שהורדנו קודם, לעמוד במסך על קובץ ה-main וללחוץ על כפתור ההפעלה שנמצא בסרגל הכלים למעלה – 

הממשק של המשתמש יופיע ב-command של סביבת העבודה ב-Spyder –



ברגע שהמשתמש יפעיל את התוכנית, יוצג עבורו האפשרויות שיש לתוכנית להציע:

```
Hi! welcome to Parkinson's Disease Detection
 Here are the options of the program:
  1) Prepare the Data
  2) Train the Model
  3) Test the Model
  4) Predict an Image


Please enter the number of the option you choose:
  (note:  If you are running the project for the
first time, you must select the first option)
```

1. "Prepare the Data" – בחירה זו תוריד את ה-dataset, תסדר את התמונות לגודל ולצבע המתאים, תבצע על מאגר התמונות augmentation, תכין את רשימת התמונות ואת רשימת התוויות המתאימה לה (גם ל-training set וגם ל- testing set) ולאחר מכן תנרמל אותן.
2. "Train the Model" – בחירה זו תתחיל את אימון המודל תוך שימוש ברשימות x_train, y_train, x_test, y_test. ראשית כל, התוכנית תייצר את המודל, תאתחל אותו באמצעות הפקודה model.compile(), לאחר מכן תאמן את המודל, תציג את הגרפים של ה-accuracy וה-loss של המודל, ולבסוף תשמור אותו.
3. "Test the Model" – בבחירה זו המשתמש בוחר לבחון ת המודל באמצעות התמונות הנמצאות ברשימה x_test.
4. "Predict an Image" – בבחירה זו, המשתמש מכניס תמונה אחת אותה הוא בוחר, והתוכנית תנסה לחזות למי שייך התמונה הציור, לאדם בריא או לאדם החולה בפרקינסון.

אם המשתמש הכניס מספר שהוא לא חלק מהאופציות המוצגות לו, התוכנית תציג למשתמש הודעת שגיאה מתאימה, ותבקש ממנו לבחור שוב -

```
This option doesn't exist. Please try to choose
again
```

אם המשתמש בחר באפשרות השנייה או השלישית, לפני שבחר קודם לכן באפשרות הראשונה, התוכנית תשלח הודעת שגיאה מתאימה למשתמשת ותבקש ממנו לבחור באופציה הראשונה –

```
 In the first time you must to choose in option
number 1. Please try again
```

אם המשתמש כבר בחר באופציה הראשונה פעם אחת, הוא לא יוכל לעשות זאת פעם נוספת. במידה וכן ינסה לעשות זאת, התוכנית תציג למשתמש הודעת שגיאה מתאימה, ותבקש ממנו לבחור שוב -

```
Error: you can't choose this option more than once
try to choose again
```

לאחר שסיים לבצע את כל הפעולות באופציה שהמשתמש בחר, התוכנית תציע למשתמש את האפשרות לצאת מן התוכנית על ידי ההודעה הבאה -

```
If you want to exit the program - please write
'Exit'
Else press ENTER
```

אם המשתמש יבחר לצאת מן התוכנית יקליד – Exit

אם יבחר להמשיך, עליו להקיש על מקש ה-ENTER במקלדת, ולאחר מכן, התוכנית תציג למשתמש שוב את האפשרויות ותבקש ממנו לבחור אחת מהן.

<u>האפשרות הראשונה – "Prepare the Data" –</u>

כאשר המשתמש בחר באופציה זו, הוא החליט להכין את ה-dataset להאימון המודל.

תחילה תוצג לו ההודעה הבאה – `Please Enter the path of the Zip File`

על המשתמש יהיה להכניס את הכתובת של תיקיית ה-zip של ה-dataset.

שים לב: **את הכתובת יש לרשום ללא שימוש בגרשיים**. לדוגמה:

לא נכון – "comp\direc\file_path.zip" , אלא, נכון - "comp\direc\file_path.zip".

לאחר שהתוכנית סיימה לחלץ את ה-dataset מקובץ ה-zip, תוצג על המסך ההודעה הבאה- `Finish to extract the files`

לאחר מכן, תופיע למשתמש ההודעה הבאה - `Please enter the new dataset path`

על המשתמש יהיה להכניס את הכתובת החדשה שיבחר ל-dataset. התוכנית תוריד את התמונות לגודל (128,128,1).

שים לב: **שוב יש לרשום את הכתובת ללא גרשיים**.

*הערה חשובה: התוכנית לא צריכה לחלק את התמונות ל-train ול-test מכיוון שמאגר המידע מגיע עם החלוקה הזאת.

התוכנית יוצרת שתי רשימות, אחת המכילה את הפיקסלים של כל תמונה, והשנייה מכילה את התווית המתאימה לכל תמונה. שתי הרשימות הללו מתואמות ביניהן כך שהמודל יוכל לתאם בין כל תמונה לתשובה שלה. שתי רשימות אלו נבנות פעם ראשונה ל-training set ופעם שנייה ל-testing set. בהמשך, התוכנית משנה את הצבע של התמונות לאפור, ומבצעת על רשימת התמונות augmentation, ומשנה בהתאם את רשימת התוויות. ולאחר מכן התוכנית מבצעת נירמול לרשימות.

לבסוף כשסיימה לבצע את כל השלב הראשון, התוכנית תודיע זאת למשתמש על ידי ההודעה הבאה – `The dataset is prepare`

האפשרות השנייה – "Train the Model" –

בחירה זו תתחיל את אימון המודל.

על מנת לבחור באפשרות זו, על המשתמש לבחור לפני כן באפשרות הראשונה – Prepare the Data. זאת על מנת, שה-dataset יהיה מתאים בגודלו, שמספרו יהיה מספיק בשביל לאמן את המודל, ושיהיה מחולק לרשימת תמונות ולרשימת תוויות. במידה והמשתמש לא בחר באפשרות הראשונה קודם לכן, יוצג הפלט הבא –

```
 In the first time you must to choose in option
number 1. Please try again
```

המשתמש לא יוכל לבחור באפשרות לאמן את המודל, לפני שבחר באפשרות להכין את ה-dataset.

כאשר בוחרים לאמן את המודל, התוכנית מבצעת כמה פעולות:

1. יוצרת מודל מסוג Sequential
2. מאמנת את המודל
3. שומרת את משקלי המודל המאומן והארכיטקטורה של המודל.

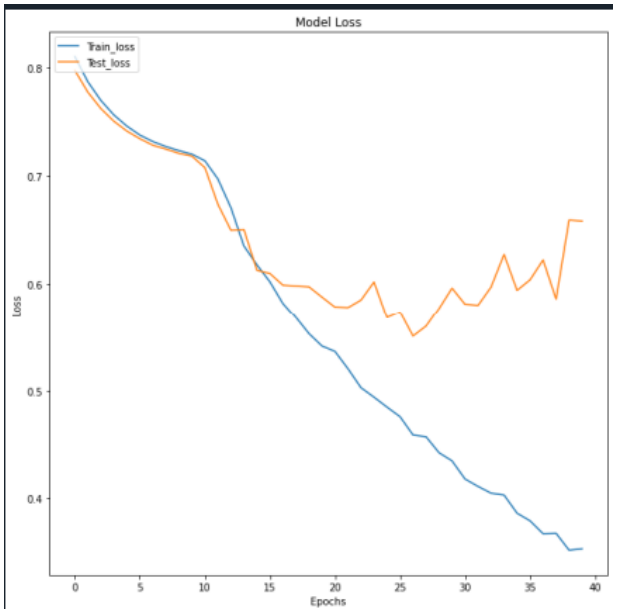
לאחר אימון המודל 40 פעמים, ניתן לראות כי אחוז ההצלחה בערך 85 אחוזים:

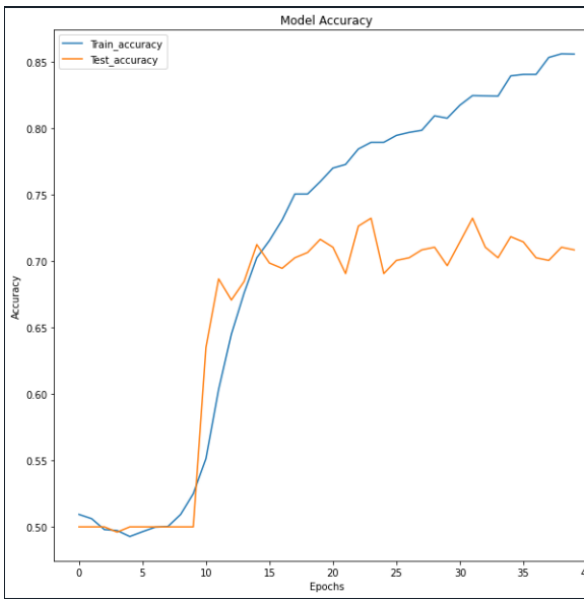
```
Epoch 1/40
221/221 [==============================] - 207s 910ms/step - loss: 0.8101 - accuracy: 0.5093 - val_loss: 0.7974 - val_accuracy: 0.5000
Epoch 2/40
221/221 [==============================] - 194s 880ms/step - loss: 0.7868 - accuracy: 0.5061 - val_loss: 0.7772 - val_accuracy: 0.5000
Epoch 3/40
221/221 [==============================] - 182s 824ms/step - loss: 0.7696 - accuracy: 0.4980 - val_loss: 0.7621 - val_accuracy: 0.5000
Epoch 4/40
221/221 [==============================] - 199s 901ms/step - loss: 0.7561 - accuracy: 0.4973 - val_loss: 0.7503 - val_accuracy: 0.4960
Epoch 5/40
221/221 [==============================] - 211s 956ms/step - loss: 0.7458 - accuracy: 0.4927 - val_loss: 0.7412 - val_accuracy: 0.5000
Epoch 6/40
221/221 [==============================] - 216s 978ms/step - loss: 0.7375 - accuracy: 0.4964 - val_loss: 0.7341 - val_accuracy: 0.5000
Epoch 7/40
221/221 [==============================] - 222s 1s/step - loss: 0.7315 - accuracy: 0.4998 - val_loss: 0.7281 - val_accuracy: 0.5000
Epoch 8/40
221/221 [==============================] - 231s 1s/step - loss: 0.7269 - accuracy: 0.5002 - val_loss: 0.7245 - val_accuracy: 0.5000
Epoch 9/40
221/221 [==============================] - 234s 1s/step - loss: 0.7231 - accuracy: 0.5093 - val_loss: 0.7206 - val_accuracy: 0.5000
Epoch 10/40
221/221 [==============================] - 240s 1s/step - loss: 0.7198 - accuracy: 0.5250 - val_loss: 0.7180 - val_accuracy: 0.5000
Epoch 11/40
221/221 [==============================] - 246s 1s/step - loss: 0.7137 - accuracy: 0.5513 - val_loss: 0.7073 - val_accuracy: 0.6349
Epoch 12/40
221/221 [==============================] - 253s 1s/step - loss: 0.6969 - accuracy: 0.6034 - val_loss: 0.6735 - val_accuracy: 0.6865
Epoch 13/40
221/221 [==============================] - 261s 1s/step - loss: 0.6703 - accuracy: 0.6449 - val_loss: 0.6494 - val_accuracy: 0.6706
```

```
Epoch 14/40
221/221 [==============================] - 266s 1s/step - loss: 0.6349 - accuracy: 0.6758 - val_loss: 0.6501 - val_accuracy: 0.6845
Epoch 15/40
221/221 [==============================] - 267s 1s/step - loss: 0.6174 - accuracy: 0.7024 - val_loss: 0.6124 - val_accuracy: 0.7123
Epoch 16/40
221/221 [==============================] - 274s 1s/step - loss: 0.6016 - accuracy: 0.7154 - val_loss: 0.6094 - val_accuracy: 0.6984
Epoch 17/40
221/221 [==============================] - 285s 1s/step - loss: 0.5815 - accuracy: 0.7308 - val_loss: 0.5985 - val_accuracy: 0.6944
Epoch 18/40
221/221 [==============================] - 40211s 183s/step - loss: 0.5677 - accuracy: 0.7503 - val_loss: 0.5977 - val_accuracy: 0.7024
Epoch 19/40
221/221 [==============================] - 199s 898ms/step - loss: 0.5526 - accuracy: 0.7503 - val_loss: 0.5970 - val_accuracy: 0.7063
Epoch 20/40
221/221 [==============================] - 220s 996ms/step - loss: 0.5411 - accuracy: 0.7597 - val_loss: 0.5875 - val_accuracy: 0.7163
Epoch 21/40
221/221 [==============================] - 219s 991ms/step - loss: 0.5361 - accuracy: 0.7699 - val_loss: 0.5781 - val_accuracy: 0.7103
Epoch 22/40
221/221 [==============================] - 197s 891ms/step - loss: 0.5200 - accuracy: 0.7726 - val_loss: 0.5777 - val_accuracy: 0.6905
Epoch 23/40
221/221 [==============================] - 188s 849ms/step - loss: 0.5025 - accuracy: 0.7842 - val_loss: 0.5847 - val_accuracy: 0.7262
Epoch 24/40
221/221 [==============================] - 188s 849ms/step - loss: 0.4937 - accuracy: 0.7892 - val_loss: 0.6015 - val_accuracy: 0.7321
Epoch 25/40
221/221 [==============================] - 187s 844ms/step - loss: 0.4845 - accuracy: 0.7892 - val_loss: 0.5678 - val_accuracy: 0.6905
Epoch 26/40
221/221 [==============================] - 187s 849ms/step - loss: 0.4757 - accuracy: 0.7944 - val_loss: 0.5731 - val_accuracy: 0.7004
```

```
Epoch 27/40
221/221 [==============================] - 189s 855ms/step - loss: 0.4587 - accuracy: 0.7967 - val_loss: 0.5504 - val_accuracy: 0.7024
Epoch 28/40
221/221 [==============================] - 187s 845ms/step - loss: 0.4571 - accuracy: 0.7983 - val_loss: 0.5596 - val_accuracy: 0.7083
Epoch 29/40
221/221 [==============================] - 188s 852ms/step - loss: 0.4423 - accuracy: 0.8092 - val_loss: 0.5767 - val_accuracy: 0.7103
Epoch 30/40
221/221 [==============================] - 202s 915ms/step - loss: 0.4348 - accuracy: 0.8074 - val_loss: 0.5958 - val_accuracy: 0.6964
Epoch 31/40
221/221 [==============================] - 289s 1s/step - loss: 0.4180 - accuracy: 0.8171 - val_loss: 0.5808 - val_accuracy: 0.7143
Epoch 32/40
221/221 [==============================] - 294s 1s/step - loss: 0.4111 - accuracy: 0.8244 - val_loss: 0.5798 - val_accuracy: 0.7321
Epoch 33/40
221/221 [==============================] - 303s 1s/step - loss: 0.4049 - accuracy: 0.8242 - val_loss: 0.5968 - val_accuracy: 0.7103
Epoch 34/40
221/221 [==============================] - 296s 1s/step - loss: 0.4031 - accuracy: 0.8239 - val_loss: 0.6270 - val_accuracy: 0.7024
Epoch 35/40
221/221 [==============================] - 293s 1s/step - loss: 0.3864 - accuracy: 0.8392 - val_loss: 0.5938 - val_accuracy: 0.7183
Epoch 36/40
221/221 [==============================] - 309s 1s/step - loss: 0.3793 - accuracy: 0.8403 - val_loss: 0.6036 - val_accuracy: 0.7143
Epoch 37/40
221/221 [==============================] - 217s 979ms/step - loss: 0.3674 - accuracy: 0.8403 - val_loss: 0.6220 - val_accuracy: 0.7024
Epoch 38/40
221/221 [==============================] - 192s 871ms/step - loss: 0.3678 - accuracy: 0.8530 - val_loss: 0.5857 - val_accuracy: 0.7004
Epoch 39/40
221/221 [==============================] - 190s 859ms/step - loss: 0.3521 - accuracy: 0.8557 - val_loss: 0.6589 - val_accuracy: 0.7103
Epoch 40/40
221/221 [==============================] - 189s 854ms/step - loss: 0.3535 - accuracy: 0.8555 - val_loss: 0.6579 - val_accuracy: 0.7083
```

לאחר שמסיימת לאמן את המודל, התוכנית מדפיסה את גרף ה-accuracy וגרף ה-loss של המודל.





אם המשתמש העלה את קובצי המודל השמור מחשבון ה-GitHub שלי, כשהמודל מסיים להתאמן התוכנית שומרת את המודל החדש ואת משקליו על קובצי המודל השמורים מחשבון ה-GitHub. כלומר, המודל השמור יימחק, ובמקומו יישמר המודל החדש אותו המשתמש אימן כעת.

לבסוף לאחר שהתוכנית סיימה לשמור את המודל היא מציגה למשתמש הודעה מתאימה –

```
The model done to fit and saved
```

<u>האפשרות השלישית – "Test the Model" –</u>

בחירה זו תבחן את המודל.

על מנת לבחון את המודל, המשתמש צריך לפני כן לבחור באפשרות הראשונה – Prepare the Data, על מנת שלתוכנית תהיה גישה לרשימות ה-test שהתוכנית מכינה.

במידה ולא בחר באופציה הראשונה קודם, יוצג לפניו ההודעה הבאה –

```
 In the first time you must to choose in option
number 1. Please try again
```

התוכנית תחזיר את המשתמש למסך האפשרויות, ויהיה עליו לבחור באפשרות הראשונה קודם.

במידה והמשתמש לא העלה את קובצי המודל השמור מחשבון ה-GitHub, או לא אימן את המודל לפני כן, לתוכנית לא יהיה מודל שמור לעבוד איתו, ולכן עליו לעשות אחד משתי האפשרויות הללו לפני שבוחר באפשרות זו.

לבסוף, למשתמש יוצג את ערכי ה-accuracy וה-loss של המודל שנשמר –

```
16/16 [==============================] - 8s 233ms/step - loss: 0.6579 - accuracy: 0.7083
test loss: 0.657905638217926, test acc: 0.7083333134651184
The model test completed
```

האפשרות הרביעית – "Predict an Image"

בחירה זו תאפשר למשתמש לבחן בעצמו את המודל השמור.

תחילה, התוכנית תבקש מהמשתמש להכניס את הכתובת של קובץ ה-zip של תמונות המבחן –

```
Please Enter the path of the Zip File
```

שים לב: **יש לרשום את הכתובת ללא גרשיים**.

התוכנית תחלץ את תמונות המבחן מקובץ ה-zip, וכשתסיים תשלח הודעה מתאימה-

```
Finish to extract the files
```

לאחר מכן, התוכנית תוריד את המודל השמור, זה שמחשבון ה-GitHub שלי, או זה ששמרה לאחר שהמשתמש בחר לאמן את המודל.

בהמשך, התוכנית תבקש מהמשתמש את הכתובת של התמונה שבחר מקובץ תמונות המבחן.

שים לב: **יש לרשום את הכתובת ללא גרשיים**.

לבסוף התוכנית תדפיס את התמונה ואת תוצאת החיזוי של הפונקציה –

Prediction by the model: Healthy

Prediction by the model: Parkinson

סיכום אישי / רפלקציה

ביצוע הפרויקט היה משימה מאתגרת ומשמעותית עבורי. ברמה האישית הפרויקט לימד אותי על עצמי, על דרך הלמידה שלי, היכולות וההתמודדות מול קשיים שלא תמיד יכולה למצוא מענה מיידי ובאופן עצמאי .

בחירת הנושא הייתה מהירה וקלה עבורי, רציתי נושא שיעניין אותי, במיוחד שידעתי שעליי יהיה לעבוד זמן ממושך על הפרויקט, ולכן בחרתי בנושא הרפואה.

תהליך העבודה וכתיבת התוכנית כלל שעות רבות של עבודה, הבנת הנושא, חיפוש אחר dataset מתאים, ומציאת פתרונות לשיפור המודל. היו פעמים רבות שנתקלתי בשגיאות בפרויקט שהיוו לי אתגרים חשיבתיים, שהיה עליי לנסות לפתור בכוחות עצמי, ובחיפוש אחר פתרונות אפשריים.

במבט לאחור, השקעתי זמן רב ובחיפוש ובהרחבת ה-dataset, דבר אשר מנע ממני להתקדם בכתיבת התוכנית. הייתי צריכה להיעזר יותר במקורות מידע שיהוו רעיון כיצד לרשום את התוכנית היעילה והטובה ביותר.

הפרויקט לא היווה מטלה פשוטה עבורי, נדרשו שעות עבודה רבות, והבנה רחבה ועמוקה של חומר חדש. אך לאחר סיום כתיבת הפרויקט, אני יכולה לסכם ולומר שרמת ההתעמקות וההיכרות שלי עם הנושא רבה. וחושבת שעמדתי במשימה בהצלחה.

ביבליוגרפיה

הסבר על המחלה -

https://www.mayoclinic.org/diseases-conditions/parkinsons-disease/symptoms-causes/syc-20376055

מידע על חומר הלימוד מהאינטרנט–

https://medium.datadriveninvestor.com/practical-guide-to-keras-deep-neural-networks-nn-f21a7715124f

https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c

ספרים –

file:///C:/Users/Tami/Downloads/Deep%20Learning%20with%20Keras%20by%20Antonio%20Gulli.pdf

file:///C:/Users/Tami/Downloads/deep_learning_for_computer_vision.pdf