

פרויקט גמר 5 ייחדות לימוד בהתחמות תכנון ותוכנות מערכות Deep Learning

**Multi-Class Classification of
Synthetically generated images of English digits
embedded on random backgrounds**

זיהוי ספרות המוטבעות על רקע אקראי

מגישה: מיה יוסף

בית ספר: מקיף י"א ראשונים

כיתה: י"ב4

מורה: דינה קראוס

תאריך בחינה: 4.7.2021



תוכן עניינים

5.....	מבנה / ארכיטקטורה של הפרויקט
5.....	תוכולת הפרויקט:
6.....	תרשים זרימה של כל קבצי הפיתון בפרויקט:
7.....	תרשים UML Actors Diagram :Use Case
8.....	תרשים המחלקות: UML Class Diagram
8.....	ראה פירוט על קבצי הפיתון בפרק "מדריך למפתח"
8.....	ראה פירוט על מאגרי התמונות בפרק "מאגרי התמונות"
9.....	מדריך למשתמש
9.....	הוראות התקנה:
10.....	הרצת הסקריפט:
13.....	לחיצה על כפתור Extract zip files
14.....	לחיצה על כפתור Train the model
18.....	לחיצה על כפתור Test the model
19.....	לחיצה על כפתור Predict an image
20.....	מאגר התמונות
20.....	מאגר התמונות המלא dataset
20.....	מאגר התמונות המיועדת לחיזוי predictions
21.....	מדריך למפתח
21.....	ראה תרשימים בפרק "מבנה / ארכיטקטורה של הפרויקט"
21.....	הסבר קבצי החסוק בפרויקט:
23.....	הסבר ופירוט על כל קובץ python בפרויקט:
23.....	הקובץ הראשי- menu_tkinter.py
29.....	קובץ התצורה- config.py
31.....	קובץ חילוץ התמונות- extractzipfiles.py
33.....	קובץ הגדרת מבנה המודל- themodel.py
36.....	הקובץ האחראי על יצירת ושמירת הגרפים- makegraphs.py
40.....	הקובץ האחראי על אימון ובדיקה המודל- train_test_model.py
49.....	הקובץ האחראי על ביצוע חיזוי לתמונה- classifyimage.py
54.....	הקובץ האחראי על בדיקת תקינות- checkdirectorys.py
58.....	הקובץ האחראי על הדפסות- printmessage.py
59.....	מסקנות הרצת המודל
59.....	הסבר מושגים:
61.....	מסקנות מהרצות קודמות של מודלים:

62.....	מסקנות מהרצת המודל הנוכחיית:
64.....	ערכיהם סופיים של המזדים להצלחתו של המודל ומסקנות:
65.....	סיכום אישי / רפלקציה
67.....	ביבליוגרפיה
68.....	נספחים

מבוא

השנה, במסגרת התמחות Deep Learning Computer Vision שמלמדת זו השנה השנייה בבית הספר שלי, נדרשו לבצע את פרויקט הגמר שלנו בנושא זה. ספר זה סוקר את הפרויקט שלי בנושא, כולל בין היתר את הרקע לפרויקט, הוראות הרצה, הסברים ותרשיים על תכלתו עבור משתמשים ומפתחים, מיליון מושגים, מסקנות מההרצה וסיכום אישי שלי על התהילך כולו.

בפרויקט שלי בחרתי לבצע סיוג של ספרות, עם טויסט: מדובר בספרות שנוצרו באופן מלאכותי בצבעים וגופנים שונים, המסובבות בזווית שונות ומוטבעות על גבי רקעים אקראיים שונים. או באנגלית: Multi-class Classification of synthetically generated images of English digits embedded on random backgrounds.

בחימם האמיתיים המידע אינם תמיד שחור-לבן, ישר, אחד ומסודר. לכן רציתי לעסוק בזיהוי של מידע "משמעותי" יותר. במקרה זהה, כל תמונה שונה לגמרי מהשניה מבחינת הסגנון של הספרה והרקע, וענין אותו לראות איך תוכנה מובוססת למידה عمוקה תتمודד עם השוני. לבסוף, גיליתי עניין מיוחד בתחום החסוציאציה Classification של הלימודים ההז, ורציתי מאוד להגיע לתוצר סופי שיכל לסייע סוגים שונים של אובייקט כלשהו.

לפני שאצלול אל הרקע והאתגרים בפרויקט הספציפי שלי, אסביר את מעט על תחום הלמידה העמוקה. למידת מכונה (Machine Learning) היא תת-תחום במדעי המחשב (Computer science) ובבינה מלאכותית (Artificial intelligence) העוסק בפיתוח אלגוריתמים המאפשרים לאחסן למחשב ללמידה מתוך דוגמאות, ופועל במגוון משימות חישוביות בהן התוכנות הקלאסី אינם אפשרי. הלמידה העמוקה (Deep Learning), היא תת-תחום Machine Learning אשר מtabסס על רשותות נירונים בעלות מספר רב של שכבות. בתחום זה אנו מtabסים על ההנחה שהמחשב יכול ללמידה ולמד את עצמו בדומה למוח האנושי, ולמעשה מנסים לחקות אותו. מהותו של תחום DL היא ליצור חיקוי ממוחשב של פעולות המוח האנושי. בפרק "מסקנות מהרצת המודול" הכתני מיליון מושגים בסיסיים בתחום. אני ממליצה לכל מי שקורא את הספר זהה לקפוץ למלון על מנת לקבל קצת רקע חינמי בנושא.

את מאגר התמונות שבו השתמשתי בפרויקט מצאתי באתר Kaggle, אך לא היו שם פתרונות מספקים לבעיית סיוג הספרות הללו. התוצר הסופי שלי מהווה פתרון לבעה זו, על ידי שימוש CNN (מוסבר במלון המושגים). דבר שkn ראייתי באינטרנט הוא דוגמאות לסיווג של כתב יד. למרות השוני הבורר בין תמונות של כתב יד רגיל על רקע חלק לבין מאגר המידע בו בחרתי להשתמש, הנחתתי כי ניתן יהיה להיעזר במודלים המותאמים לכתב יד, או לפחות לקבל קצת רקע, קודם כל ממשום שבשני המקרים מדובר בסיווג רב כיתתי, וכן בזכות קווי הדמיון בין שני המקרים, העוסקים בתמונות בהן אובייקט הבולט על רקע השונה ממנו.

האתגר הראשון שלי היה להבין כיצד למשהו פרויקט DL צריך להראות והיכן אני מתחילה. לאחר מכן נוצרו אתגרים חדשים, המפורטים לאורך הספר ובסיכום בסוףו, שעם כולם התמודדתי, כל אחד בעיתו. בין היתר התפקידים שהוא מלא, כמו סיפוק רקע והסבירים בתחום, הסברים מפורטים על הפרויקט, תרשימים, הוראות הרצה ואף מילימ' מזווית אישית, ספר זה מהווה גם את התזכורת הנחמדה שלי לניצחון על כל האתגרים הללו, וזריקת מוטיבציה לאתגרים שעוד יבואו. ☺

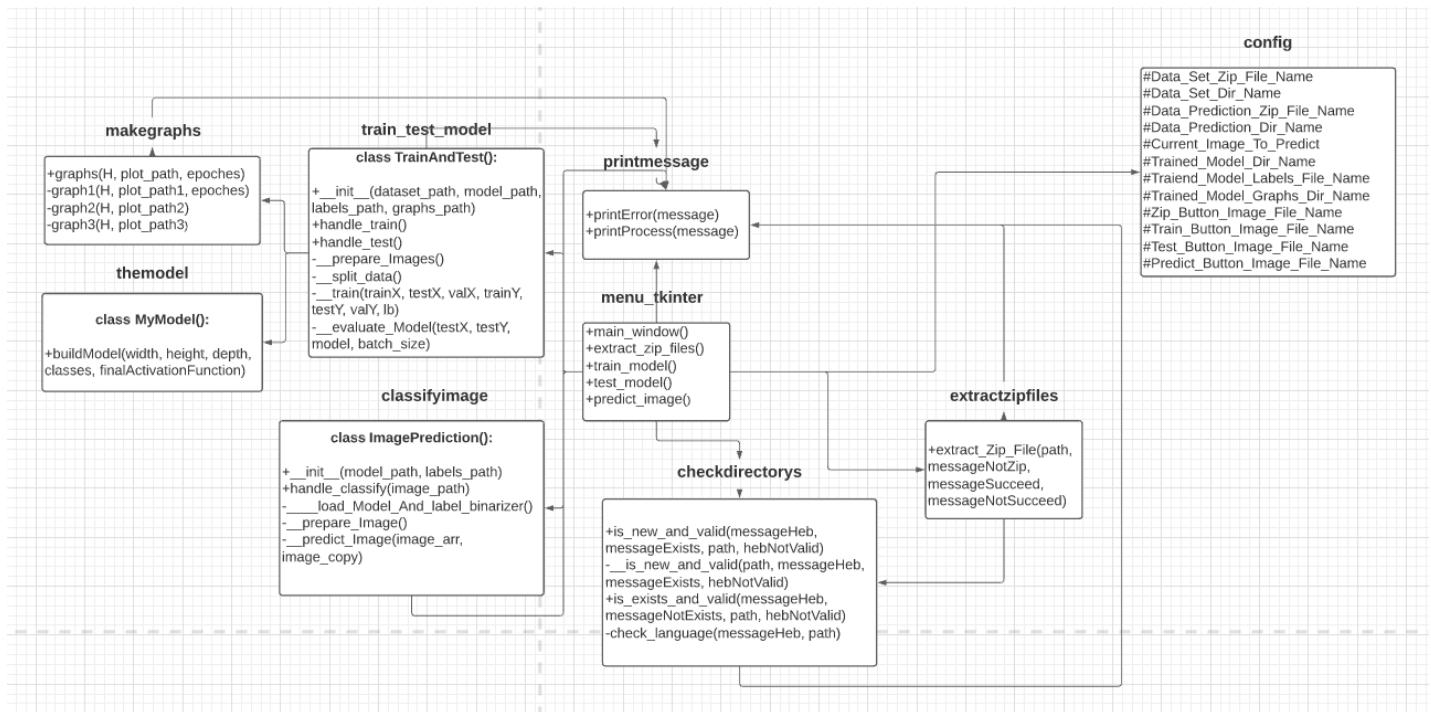
מבנה / ארכיטקטורה של הפרויקט

הפרויקט מכיל מספר קבצים ותיקיות:

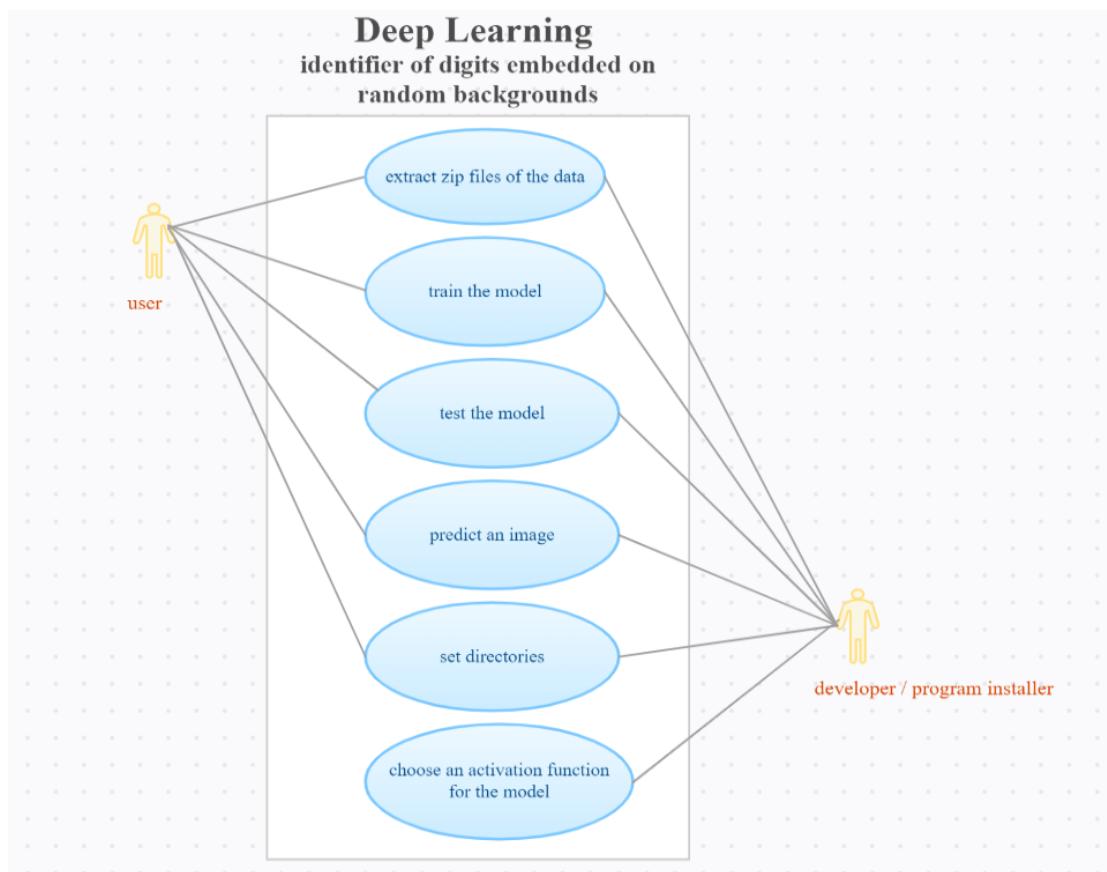
תכולת הפרויקט:

- תיakit python המכילה את קבצי החסוקים של הפרויקט:
-tkinter_menu-config-train_test_model-printmessage-theModel-classifyimage-extractzipfiles-checkdirectorys-makegraphs-האימון-הקובץ האחראי על ביצוע סיווג prediction לתרמונה
- תיikit Data המכילה את קבצי הקובץ של מאגרי התמונות בהם נעשה שימוש בפרויקט:
-קובץ zip המכיל את תמונות dataset בהן ישמש המודל לאחר חלוקה עברית:
train, validation, test
-קובץ zip המכיל תמונות predictions המיועדות לחיזוי (prediction)
-קובץ README המכיל הוראות קצרות להרצת הפרויקט
- תיikit trained_model המכיל המודל המאמון
- קובץ binair labels של תוכיות התמונות לאחר אימון המודל
- תיikit graphs בה שלושה גרפים המתארים את תהליכי האימון של המודל
- תיikit buttons המכילה את תמונות הכפתורים שיוצגו למשתמש במסך:
- תמונה מסוג png של הכפתור הראשון במסך שיוצג בהרצת התוכנית
- תמונה מסוג png של הכפתור השני במסך שיוצג למשתמש בהרצת התוכנית
- תמונה מסוג png של הכפתור השלישי במסך שיוצג למשתמש בהרצת התוכנית
- תמונה מסוג png של הכפתור הרביעי במסך שיוצג למשתמש בהרצת התוכנית
- תיikit diagrams המכילה את התרשיים המתארים את הפרויקט:
-תרשימים זרימה של כל קבצי הפיקט בפרויקט
-תרשימים Use Case
-תרשימים המחלקות
-תרשימים model_layers המציג את שכבות המודל

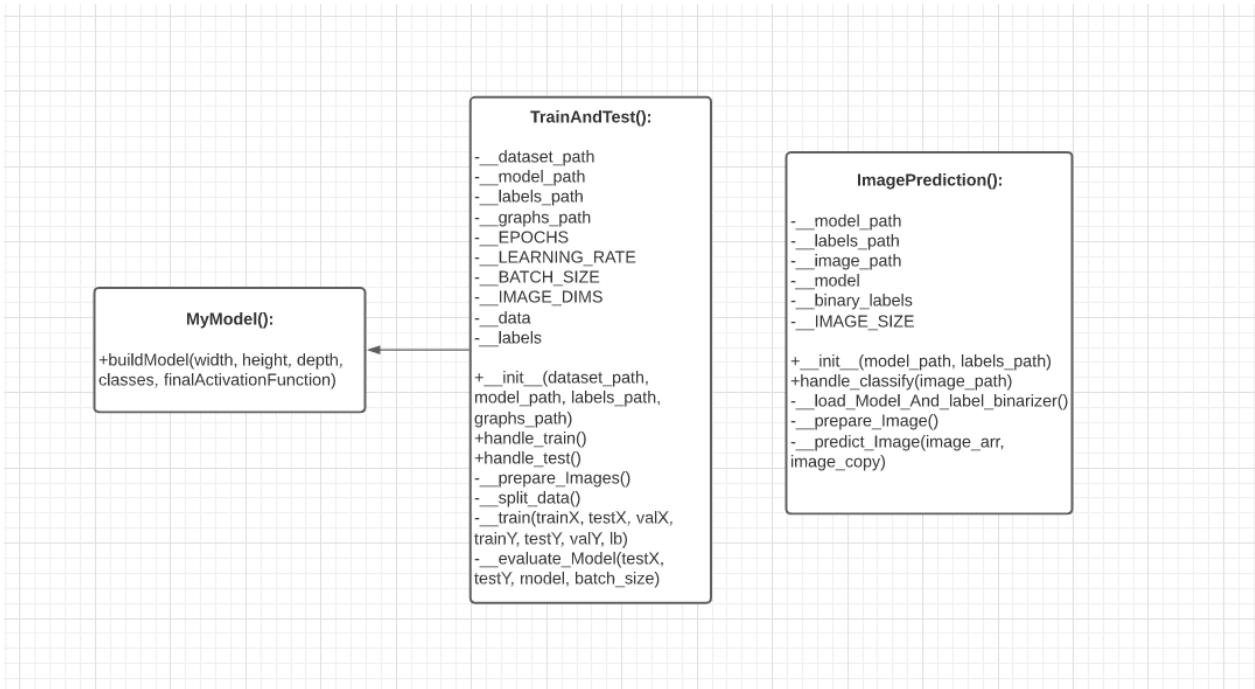
תרשים זרימה של כל קבצי הפיתון בפרויקט:



תרשים UML Actors Diagram :Use Case



תרשים המחלקות: UML Class Diagram



ראה פירוט על קבועי הפיתון בפרק "מדריך למפתח"

ראה פירוט על מאגרי התמונות בפרק "מאגרי התמונות"

מדריך למשתמש

הוראות התקנה:

בפרק זה אסביר על הפרויקט כאשר הנמען הוא המשתמש. ראשית, עבור על הדרישות להרכז התוכנית, הוראות ההתקנה והקבצים הנדרשים.

יש להוריד <https://www.python.org/downloads/> -Python 3.7

אם יש לך Python על המחשב אתה לא בטוח איזו גרסה מותקנת, תוכל לבדוק מהי גרסתו באמצעות הפקודה: V - python

יש להוריד מספר ספירות קוד בהן הפרויקט משתמש:

שם הספרייה	פקודת התקנה	קישור
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/
imutils	pip install imutils	https://pypi.org/project/imutils/
numpy	pip install numpy	https://pypi.org/project/numpy/
opencv	pip install opencv-python	https://pypi.org/project/opencv-python/
PIL	pip install Pillow	https://pypi.org/project/Pillow/
sklearn	pip install -U scikit-learn	https://scikit-learn.org/stable/install.html
colorama	pip install colorama	https://pypi.org/project/colorama/

יש להוריד מחשבון GitHub של איזה פרויקט, עליהם ניתן למצוא פירוט בפרק "מבנה / ארכיטקטורה של הפרויקט".

יש לארגן את הקבצים בהתאם אחד שם הנתיב לה מרכיב מאותיות באנגלית בלבד. על שמות נתיבי התמונות להכיל אותיות באנגלית בלבד.

יש לפתוח את קובץ config.py באמצעות כל עורך טקסט ולהתאים את הנתיבים המוגדרים שם לאופן בו שמרת אותם במחשב שלך, ועל פי הטבלה בה סיכמתי את הדרישות והסביר לכל נתיב המופיע בעוד מספר עמודים.

הפרויקט הורץ על מחשבים בהם מותקנת מערכת הפעלה Windows 10 בגרסת 64 bit עם הנתונים הבאים:

معالד:	Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz 2.71 GHz	
זיכרון מותקן (RAM):	8.00 GB	
סוג מערכת:	מערכת הפעלה של 64 סיביות, מעבד מבוסס 64x	

معالד:	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz	
זיכרון מותקן (RAM):	16.0 GB (15.9 GB ניתנים לשימוש)	
סוג מערכת:	מערכת הפעלה של 64 סיביות, מעבד מבוסס 64x	

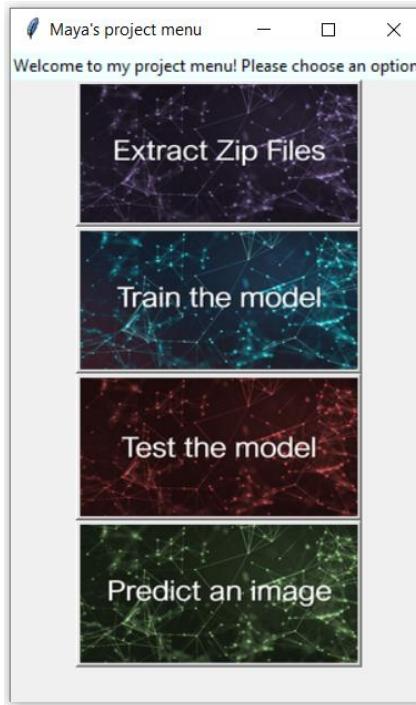
הרצת סקריפט:

כעת, אסביר כיצד להריץ את התוכנית.

ראשית, יש לפתח את קובץ config.py:python config.py:python config.py וכאמור לוודא שכל הנתיבים המוגדרים בו מתאימים וחוקיים.

לאחר מכן, יש להריץ Command Line ב-`python menu_tkinter.py`:python menu_tkinter.py (ניתן להלופין להריצו ב-Command Line של סביבת העבודה Anaconda על פייתון 3.8.8).

הרצת קובץ זה, המזמן מethods מקבץ `shutil` נוספים בפרויקט, תריצ' למשה את כל הפרויקט. לאחר הריצת התוכנית יפתח החלון הבא:



כפי שניתן לראות, על גבי החלון מופיעים 4 כפתורים, שליחיצה על כל אחד מהם תביא לביצוע משימה אחרת.

כפתרו 1 - חילוץ התמונות מקבץ הקז' של מאגר התמונות המלא וכן של מאגר התמונות המיועדות לחיזוי, לתיקיות רגילים עם שם זהה (ללא הסימנת `zip`). זאת, על מנת שהתוכנית תוכל לגשת לתמונות.

כפתרו 2 - הריצת תהליך האימון למודל (train)

כפתרו 3 - ביצוע בדיקה למודל (test)

כפתרו 4 - חיזוי לתמונה ספציפית (solution). בעת לחיצה על כפתרו זה התוכנית מעניקה חיזוי לתמונה שניתנה מוגדר בקובץ `config.py`, colum מחייב לאיזו קטgorיה היא שייכת.

יש לשם לב במספר נקודות חשובות, שאי הבנה שלהן עלולה לגרום לך לא להצליח להריץ את הפרויקט נכון.

בהריצה הראשונה של הפרויקט, יש ללחוץ קודם כל על הכפתרו הראשון בתפריט על מנת שההתמונות יחולצו, אחרת התוכנית לא תוכל לגשת אליהן. לאחר מכן בהרצה זו ובഹרצות

הבות אין סיבה ללחוץ עליו, והדבר יגרור הودעת שגיאה, כיוון שהתמונה כבר חולצו. (כמובן שאם ברצונך לבצע את החילוץ ידנית ולשים לב שהנתיבים מתאימים, אין לחוץ על הcptor). בכל מקרה, יש לשים לב שהנתיבים המוגדרים למאגר התמונות קיימים וחוקיים.

אם ברצונך לבצע אימון, עליך לשנות את הנתיבים למודל, קובץ התווiotות ותיקית הגրפים לנתייבים חדשים שעדיין לא קיימים משום שבכל תהליך אימון נוצרים תיקיות וקבצים אלו מחדש בהתאם לתהליכי האימון הנוכחי.

אם ברצונך לבצע חיזוי, עליך לשים לב שהנתיבים למודל ולקובץ התווiotות הם נתיבים קיימים משום שעבור אפשרות זו יש צורך במודל שמור ובקובץ תווiotות שמור.

אם ברצונך לבצע טסט מבלי שביצעת אימון לפני כן בהרצה הנוכחית, עליך לשים לב שהנתיב למודל הוא נתיב קיים משום שעבור אפשרות זו יש צורך במודל שמור.

אם בהרצה הנוכחית כבר ביצעת אימון וכעת אתה רוצה לבצע טסט או חיזוי, הנתיבים שהיו חדשים באימוןicut כבר קיימים لكن אין בעיה לחוץ cptor הטסט מיד לאחר סיום ביצוע האימון.

אוסף בטבלה את הנתיבים,משמעותם ודרישותיהם:

נתיב	משמעות	דרישה
Data_Set_Zip_File_Name	נתיב לקובץ zip של מאגר התמונות המלא	קובץ קובץ קיימ.
Data_Set_Dir_Name	נתיב לתיקייה המחולצת של מאגר התמונות המלא	הנתיב אליו חולץ קובץ הקובץ של מאגר התמונות המלא- ציריך להיות בעל שם זהה ללא הסיומת .zip.
Data_Prediction_Zip_File_Name	נתיב לקובץ zip של התמונות המיועדות לחיזוי	קובץ קובץ קיימ.
Data_Prediction_Dir_Name	נתיב לתיקייה המחולצת של התמונות המיועדות לחיזוי	הנתיב אליו חולץ קובץ הקובץ של התמונות המיועדות לחיזוי- ציריך להיות בעל שם זהה ללא הסיומת .zip.
Current_Image_To_Predict	שם התמונה המיועדת לחיזוי כולל הסיומת jpg או png).	שם קיים של תמונה עם סיומת נכונה (jpg או png).
Trained_Model_Dir_Name	נתיב למודל המאמן	אם ברצונך לחוץ על cptor 2 (train)- נתיב חוקי חדש (עדין לא קיימ) אם ברצונך לחוץ על cptor 3 (test) או cptor 4 (predict) ללא ביצוע train בהרצה הנוכחית- נתיב קיימ

אם ברצונך ללחוץ על כפתור 2 (train)- נתיב חוקי חדש (עדין לא קיים) אם ברצונך ללחוץ על כפתור 3 (test) או כפתור 4 (predict) ללא ביצוע train בהרצה הנוכחיית- נתיב קיים	נתיב לקובץ התוויות שנשמר במהלך אימון המודל השמור	Traiend_Model_Labels_File_Name
אם ברצונך ללחוץ על כפתור 2 (train)- נתיב חוקי חדש (עדין לא קיים)	נתיב לתיקיית הגרפים שנוצרו במהלך אימון של המודל	Trained_Model_Graphs_Dir_Name
הנתיב האמיתי והחוקי של תמונה הcpfotor.	הנתיב של תמונה הcpfotor Extract zip files	Zip_Button_Image_File_Name
הנתיב האמיתי והחוקי של תמונה הcpfotor.	הנתיב של תמונה הcpfotor Train the model	Train_Button_Image_File_Name
הנתיב האמיתי והחוקי של תמונה הcpfotor.	הנתיב של תמונה הcpfotor Test the model	Test_Button_Image_File_Name
הנתיב האמיתי והחוקי של תמונה הcpfotor.	הנתיב של תמונה הcpfotor Predict an image	Predict_Button_Image_File_Name

כמו כן, חשוב לשים לב שהשמות המוגדרות בנתיבי התמונות נכונות (png או jpg).

אם אחד הנתיבים המוגדרים בקובץ config אינו תקין, תודפס למשתמש הודעה שגיאה.

למעשה, עבור כל סוג של שגיאה בכל אחד מהנתיבים תודפס הודעה אחרת. הודעות אלו

מפורטות באיזה נתיב קיימת הבעיה ומהי הבעיה הספציפית שמנעה מהתוכנית לרוץ כהלה.

בעמוד הבא אתחל לפרט מה יקרה ומה יוצג למשתמש עבור כל לחיצה על אחד הcpfotorim.

יש לשים לב כי הדפסות בצבע כחול מספקות אינפורמציה על התהיליכים המתרחשים, ואילו

הדפסות בצבע אדום מספקות הודעות שגיאה על נתיבים לא תקינים.

לחיצה על בפטור Extract zip files:

כאמור, בהרצה הראשונה של התוכנית יש ללחוץ קודם כל ובאופן חד פעמי על כפתור זה (לא שוב בשאר ההצלחות, אלא אם כן הchlפטה את הנתיב שוב לקובץ zip).

במידה שהנתיב המגדיר את קובץ מאגר התמונות המלא או הנתיב המגדיר את קובץ מאגר התמונות המיועדת לחיזוי אינם נתבי zip, תודפס הודעה שגיאת המציגת זאת:

the dataset path is not a zip file

במידה שנתיב אינם חוקי והחילוץ לא הצליח מכל סיבה שהיא, תודפס הודעה שגיאת:

Could not finish to extract dataset dirs

במידה שנתיב חוקי, תודפס הודעה המبشرת על הצלחת החילוץ:

Finished to extract predictions dirs

כשמסתיים תהליך זה המשתמש חופשישוב ללחוץ על כפתורים אחרים.

לחיצה על בפטור Train the model

אם הנטיב למאגר התמונות אינו נתיב חוקי, כלומר אינו נתיב קיימ, מחולץ וחוקי, יודפסו הודעות שגיאת המסבירות שנטיב זה אינו חוקי ואת הסיבה לכך.

נשים לב כי הימצאות אותיות עבריות בנתיב זה נחשבת לשגיאת. זאת מכיוון שבשלב טיענת תמונה באמצעות opencv ישנה שגיאה כאשר הרשימה של התמונה מכיל אותיות עבריות.

```
(base) C:\Users\mayay\MyMaya>python menu_tkinter.py
Error! the images path contains hebrew letters
```

```
(base) C:\Users\mayay\MyMaya>python menu_tkinter.py
Error! the images path does not exist
```

אם הנטיב לתקייה בה ישמר המודל, לקובץ התווות או לתקיית הגрафים (שיצרו בהרצה), אינו נתיב חוקי, כלומר אינו נתיב חדש וחוקי, יודפסו הודעות שגיאת המצביאות על הנטיב הספציפי שאינו חוקי ואת הסיבה לכך, למשל:

```
(base) C:\Users\mayay\MyMaya>python menu_tkinter.py
Error! the labels path is already exists
```

```
(base) C:\Users\mayay\MyMaya>python menu_tkinter.py
Error! the graphs path is already exists
```

במידה שככל הנטיבים המוגדרים מתאימים:

תהליך האימון יתחיל לרוב, ונראה זאת על גבי המסך. למעשה, על גבי המסך יוצגו לפנינו פלטי אינפורמציה על התהליכי המתרחשים אחד אחרי השני כחלק מביצוע תהליך האימון.

ראשית התוכנית טעונה את התמונות מהirectory של מאגר התמונות. לאחר מכן יוצגו הודעות אינפורמציה על הגודל בMB של רשימת כל התמונות מערכיים. אחר כך תודפס הודעה על כך שהמודל התקםפל.

```
(base) C:\Users\mayay\MyMaya>python menu_tkinter.py
INFO: Loading images...
INFO data matrix: 233.20MB
INFO: Compiling model...
```

לאחר מכן, יודפס סיכום טקסטואלי של המודל הכללי מידע אודות: השכבות וסדרן במודל, צורת הפלט של כל שכבה, מספר הפרמטרים (משקלים) בכל שכבה, והמספר הכללי של פרמטרים (משקלים) במודל.

```

Model: "sequential"
-----  

Layer (type)          Output Shape       Param #
-----  

conv2d (Conv2D)        (None, 50, 50, 32)    320  

activation (Activation) (None, 50, 50, 32)    0  

batch_normalization (BatchNo (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, 64)    18496  

activation_1 (Activation) (None, 16, 16, 64)    0  

batch_normalization_1 (Batch (None, 16, 16, 64)    256  

conv2d_2 (Conv2D)      (None, 16, 16, 64)    36928  

activation_2 (Activation) (None, 16, 16, 64)    0  

batch_normalization_2 (Batch (None, 16, 16, 64)    256  

max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)    0  

dropout_1 (Dropout)     (None, 8, 8, 64)    0  

conv2d_3 (Conv2D)      (None, 8, 8, 128)    73856  

activation_3 (Activation) (None, 8, 8, 128)    0  

batch_normalization_3 (Batch (None, 8, 8, 128)    512  

conv2d_4 (Conv2D)      (None, 8, 8, 128)    147584  

activation_4 (Activation) (None, 8, 8, 128)    0  

batch_normalization_4 (Batch (None, 8, 8, 128)    512  

max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)    0  

dropout_2 (Dropout)     (None, 4, 4, 128)    0  

flatten (Flatten)      (None, 2048)    0  

dense (Dense)          (None, 1024)    2098176  

activation_5 (Activation) (None, 1024)    0  

batch_normalization_5 (Batch (None, 1024)    4096  

dropout_3 (Dropout)     (None, 1024)    0  

dense_1 (Dense)         (None, 10)    10250  

activation_6 (Activation) (None, 10)    0
-----  

Total params: 2,391,370  

Trainable params: 2,388,490  

Non-trainable params: 2,880  

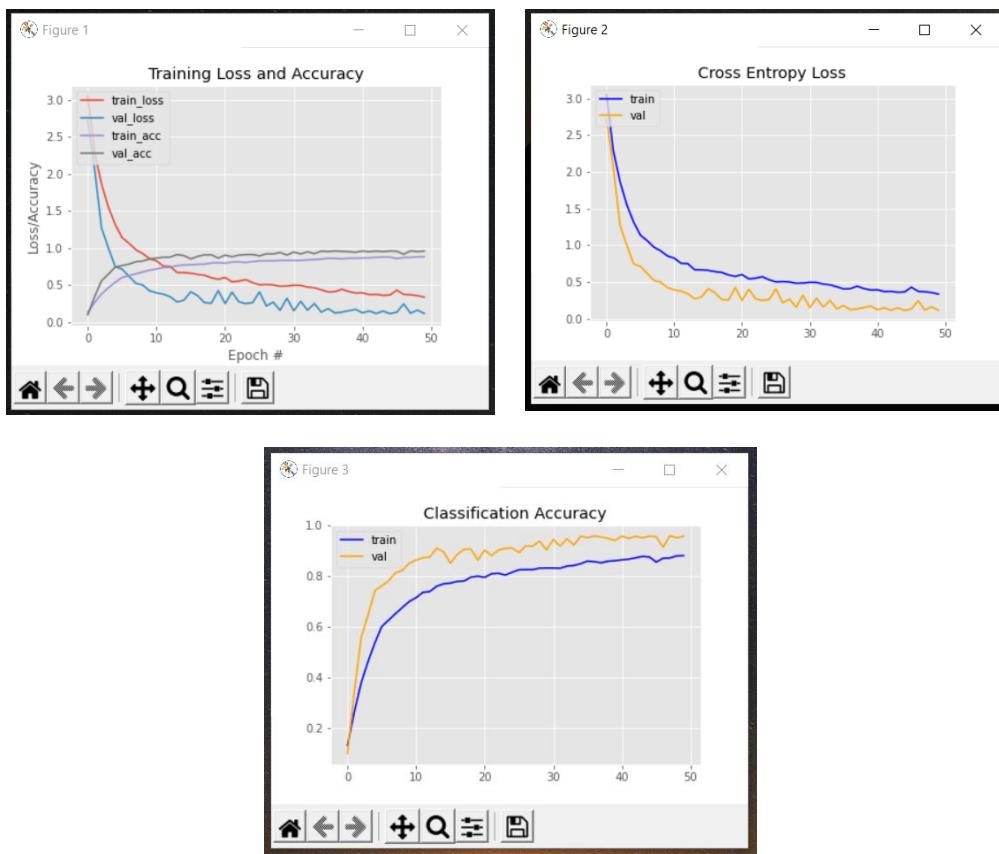
INFO: Training neural network...

```

לאחר מכן נראה את האימון מתרחש בפועל כאשר ערכי accuracy וloss של train data משתנים בכל ריצה חדש, עד ערכים סופיים:

```
326/326 [=====] - 36s 109ms/step - loss: 0.3836 - accuracy: 0.8674 - val_loss: 0.1085 - val_accuracy: 0.9665
Epoch 37/50
326/326 [=====] - 36s 110ms/step - loss: 0.3692 - accuracy: 0.8729 - val_loss: 0.0987 - val_accuracy: 0.9672
Epoch 38/50
326/326 [=====] - 35s 109ms/step - loss: 0.3836 - accuracy: 0.8695 - val_loss: 0.0841 - val_accuracy: 0.9712
Epoch 39/50
326/326 [=====] - 37s 112ms/step - loss: 0.3735 - accuracy: 0.8711 - val_loss: 0.1197 - val_accuracy: 0.9545
Epoch 40/50
326/326 [=====] - 40s 123ms/step - loss: 0.3474 - accuracy: 0.8787 - val_loss: 0.2139 - val_accuracy: 0.9364
Epoch 41/50
326/326 [=====] - 37s 113ms/step - loss: 0.3351 - accuracy: 0.8866 - val_loss: 0.0891 - val_accuracy: 0.9692
Epoch 42/50
326/326 [=====] - 36s 111ms/step - loss: 0.3423 - accuracy: 0.8813 - val_loss: 0.1226 - val_accuracy: 0.9632
Epoch 43/50
326/326 [=====] - 37s 112ms/step - loss: 0.3320 - accuracy: 0.8827 - val_loss: 0.0832 - val_accuracy: 0.9739
Epoch 44/50
326/326 [=====] - 40s 122ms/step - loss: 0.3242 - accuracy: 0.8898 - val_loss: 0.0798 - val_accuracy: 0.9732
Epoch 45/50
326/326 [=====] - 39s 121ms/step - loss: 0.3217 - accuracy: 0.8932 - val_loss: 0.1353 - val_accuracy: 0.9531
Epoch 46/50
326/326 [=====] - 38s 116ms/step - loss: 0.3277 - accuracy: 0.8876 - val_loss: 0.0905 - val_accuracy: 0.9685
Epoch 47/50
326/326 [=====] - 39s 119ms/step - loss: 0.3267 - accuracy: 0.8862 - val_loss: 0.0982 - val_accuracy: 0.9705
Epoch 48/50
326/326 [=====] - 36s 111ms/step - loss: 0.3287 - accuracy: 0.8873 - val_loss: 0.0978 - val_accuracy: 0.9678
Epoch 49/50
326/326 [=====] - 38s 116ms/step - loss: 0.3065 - accuracy: 0.8943 - val_loss: 0.1079 - val_accuracy: 0.9658
Epoch 50/50
326/326 [=====] - 38s 115ms/step - loss: 0.3165 - accuracy: 0.8910 - val_loss: 0.1114 - val_accuracy: 0.9591
```

עם תום תהליכי האימון יפתח למשתמש מסך עם גרפים המתארים את תהליכי האימון. כאשר יסגורו את המסך, יפתח לו מסך חדש עם גרפים חדשים, וכך גם פעמיinus (במשך הכל 3 מסכים).

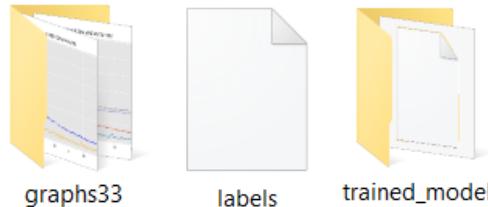


בסוף האימון תודפס הודעה למשתמש על הנטיב בו נשמרו הגרפים שהתוכנית זה עתה יקרה, אשר מתארים את תהליך האימון.

Graphs of the training process are in graphs33

המשתמש יוכל לבדוק בנטיבי הגרפים, המודל והתוויות, שאכן נוצרו קבצים במקומות אלה:

C:\Users\mayay\MyMaya



כשמסתיים תהליך אימון המודל המשתמש חופשישוב ללחוץ על כפתורים אחרים.

תהליך זה לוקח כ-45 דקות.

לחיצה על כפתור Test the model:

במקרה שהנתיב למאגר התמונות או למודל אינו חוקי מהסיבות שפירטתי בתחילת מדריך זה, יופיעו הודעות מתאימות למשתמש, למשל:

Error! the images path contains hebrew letters

Error! the model path does not exist

במקרה שהנתיבים תקינים:

בדיקת המודל (test) תתחיל, בהתאם הלוקח בדקה.

ראשית התוכנית טעונה את התמונות מהirectory של מאגר התמונות. לאחר מכן תודפס הودעות אינפורמציה על הגודל בMB של רשימת כל התמונות כמערכות.

INFO: Loading images...
INFO data matrix: 233.20MB

אחר לכך תודפס ההודעה "evaluate on test data" וירlösו ערכי ה`accuracy` ו`loss`.

```
# Evaluate on test data
75/75 [=====] - 1s 15ms/step - loss: 0.0978 - accuracy: 0.9627
test loss 0.09783092141151428 , test acc 0.9627302885055542
```

כמשמעותם תחילך זה המשתמש חופשי שוב ללחוץ על כפתורים אחרים.

לחיצה על בפטור :Predict an image

אם הנטיב לתמונה המיועדת לחיזוי אינו חוקי, למשל אינו קיים או חוקי, תודפס הודעה שגיאה מתאימה. גם כאן, מכיוון שמדובר בתמונה שיש לטען עם הספרייה opencv, השגיאה מוגדרת בו שמו התמונה חייב להיות באנגלית בלבד.

```
(base) C:\Users\mayay\MyMaya>python menu_tkinter.py
Error! the image path does not exist
```

```
(base) C:\Users\mayay\MyMaya>python menu_tkinter.py
Error! the image path contains hebrew letters
```

באופן דומה מודפסות הודעות שגיאה גם עבור اي תקינות בנטיב למודל או לקובץ התוויות:

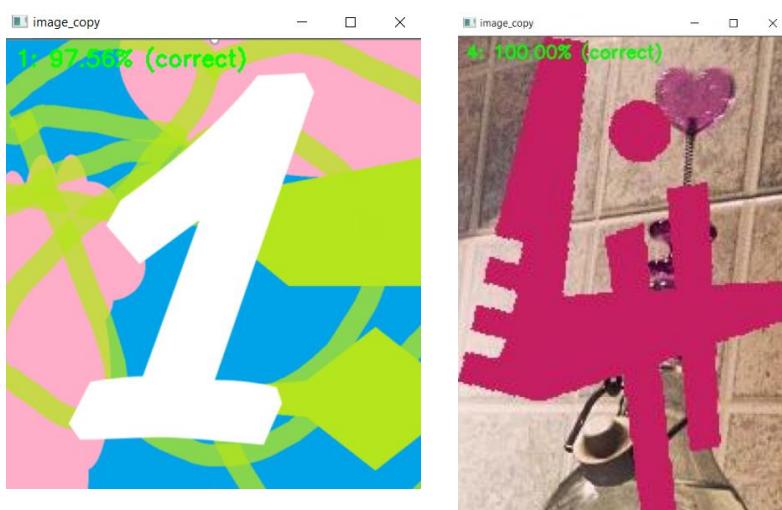
```
Error! the model path does not exist
Error! the labels path does not exist
```

```
INFO: Loading network...
2021-06-16 22:08:45.753328
h oneAPI Deep Neural Network
AVX AVX2
To enable them in other openvino tools, run
INFO: Loading image...
INFO: Classifying image...
INFO: 1: 97.56% (correct)
```

במידה שהנתביים תקינים:

ראשית, תודפס הודעה על כך שהמודל השמור נתען בידי התוכנית. לאחריה תודפס הודעה נוספת נסופה על כך שהתמונה המיועדת לשיזוג נתענת גם היא. הודעה שלישית תסביר שתהליך הסיזוג מתרחש ברגעים אלה.

לאחר סיום תהליך הסיזוג עצמו על ידי התוכנית, תודפס הודעה שמצינית את הקטגוריה (הספרה) שהמודל חזה שהתמונה שייכת אליה, את הסיכוי שצדק על פי חישובי התוכנית באחוזים, וכן בסוגרים האם אכן צדק.



לבסוף, יפתח מסך חדש עליון התמונה ועל גביה תווית עם תוצאה החיזוי, הסיכוי שהתמונה שהתוארכה נכונה והאם התוארכה נכון או לא. אם התוארכה נכון, המידע על גבי התמונה יהיה בצבע ירוק. אם התוארכה לא נכון, המידע יהיה בצבע אדום. תהליך זה לוקח כדקה.

מאג'רי התמונות

לאחר שמתבצע התיילוץ של התמונות מקובץ הקז', מאג'רי התמונות הם למעשה תיקיות רגילות המכילות תמונות מסווג jpg. אסביר על כל אחד ממאגרי התמונות.

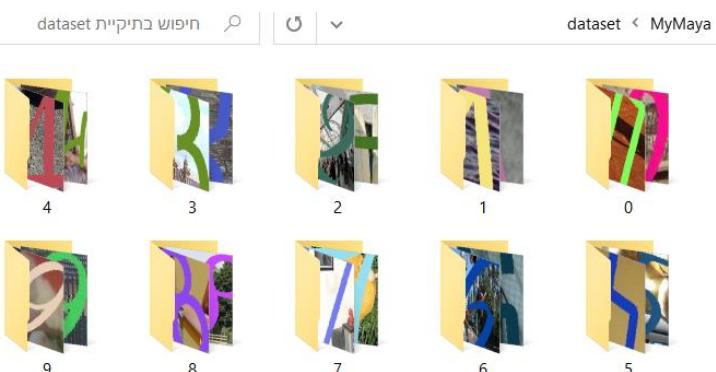
מאגר התמונות המלא dataset

dataset

0
1
2
3
4
5
6
7
8
9

Dataset בו אני עושה שימוש בפרויקט שלי הוא תיקייה בה כ-12,000 תמונות בסך הכל. בהתאם לתיקייה ישן 10 תת-תיקיות שבכל אחת מהן כ-1,200 תמונות של ספרה אחת. התמונות הן וריאציות שונות של הספרות: צבע שונה, רקע שונה, צוית שונה, רקע שונה, עובי שונה וכו'. מדובר בספרות שנוצרו באופן מלאכותי בצבעים וגוונים שונים, המסוכבות בחווית שוננות ומוטבעות על גבי רקעים אקראיים שונים. או באנגלית: *Synthetically generated images of English digits embedded on random backgrounds.*

אצין שבמהלך תהליכי האימון הdata מוגדל (מפורט בפרק "מדריך למפתח").



מאגר התמונות המיעודות לחיזוי predictions

תיקית התמונות השנייה בפרויקט שלי היא תיקייה המכילה כ-70 תמונות המיעודות לביצוע predict למונה של ספרה כלשהי על ידי המודל. תיקייה זו מורכבת מתמונות שהערתתי מכל תה תיקייה במאגר התמונות המלא (6 עבורי כל ספרה, 54 בסה"כ) ואף מתמונות שיצרתתי בעצמי בעדרת תוכנת הציר. כאשר ברצונך לבצע predict למונה כלשהי, עליך לבחור באחת מתמונות בתיקייה זו ולשנות את נתיב התמונה המיעודת לחיזוי בקובץ config לנטייב של התמונה שברחצת. יש לשים לב אם סוג התמונה הוא jpg (תמונות רגילות) או jpg (תמונות שיצרתתי我自己).



חשוב לשמר על שמות קבצי התמונות בתיקייה זו לפי הכלל הבא: התו הראשון של כל שם הוא למשנה הספרה שאותה תמונה מייצגת. זאת, על מנת שבביצוע predict המודל יוכל להשווות בין זו זה בין התווית של התמונה שהוא חושב שהיא מתאימה לה, וכך לדעת ולהדפיס למשתמש האם תוצאות החיזוי שלו הייתה נכונה או לא.

מדריך למפתח

ראה תרשימים בפרק "מבנה / ארכיטקטורה של הפרויקט" הסבר קבצי החומר python בפרויקט:

הפרויקט שלי מכיל מספר קבצי python, כל אחד בעל תחום אחריות שונה. חלוקת הקוד של הפרויקט באופן הגיוני מביאה לוגית שימושית מאוד. היא מסייעת לי וכל מי שמנסה להבין את הפרויקט בארגון הקוד, הבנה טוביה יותר שלו, הפחתתאגים ויעילות.

ראשית, אסביר בקצרה על כל אחד מהקבצים:

קובץ	תפקיד
menu_tkinter.py	קובץ הראשי של הפרויקט, אשר אחראי על ניהול התוכנית ועל התקשרות עם המשתמש. בקובץ זה מצויה הפונקציה הראשית ()main_window המהווה פונקציית שירות עבור המשתמש. כאשר משתמש מרים את קובץ פיתון זה, נפתח בפניו חלון עם מספר כפתורים. על מנת לבצע את תפקיד הכption עליו המשתמש לחץ, קובץ זה מזמן מתודות משאר הקבצים.
config.py	בקובץ זה מוגדרים הנתיבים השונים הנמצאים בשימוש בפרויקט. על מנת להתאים את הרצת הפרויקט למחשב האישי שלו (המיקומים בהם המידע הנוחוו כמו מאגר התמונות שמור) ולמטרה של (למשל, אם ברצונך ללחוץ על הכption של test עלייך לספק בקובץ זה נתיב למודל שמור, לעומת זאת היפתור של החוזר נתיב חדש שמודל מאומן חדש ישמר בו במהלך האימון- הסבר מפורט בפירוט על קובץ זה בהמשך ובמדריך למשתמש).
extractzipfiles.py	קובץ זה אחראי על חילוץ התמונות מקובצי הקז' של מאגר התמונות המלא ושל מאגר התמונות המיועדות לחיזוי, לתיקיות ריגולות עם שם זהה (ללא הסיומת zip).
themodel.py	קובץ זה אחראי על בניית שכבות המודל. המודל הוא למעשה מהותה הרכבת בפרויקט. ביצועים טובים של המודל משמעוותם הצלחת מטרת הפרויקט וחיזויים נכונים של התמונות. תהליך למידת התמונות מבוצע על גבי המודל וביצוע test והחיזוי נעשה באמצעות מודל שמור.

קובץ זה אחראי על ביצוע אימון המודל (למיצית התמונות) והtest (בדיקות המודל). קובץ זה הוא למעשה אחד החלקים המרכזיים של הפרויקט.	train_test_model.py
קובץ זה אחראי על ביצוע החיזוי לתמונה כלשהי, קלומר, סיווג התמונה לקטגוריה המתאימה לה.	classifyimage.py
קובץ זה אחראי על בדיקת תקינות הנתיבים שהוגדרו בעק.config ועל הדפסת הודעות שגיאה מתאימות במקרה הצורך (גם בקבצים אחרים מודפסות הודעות אינפורמציה ושגיאה).	checkdirectorys.py
קובץ זה אחראי על ביצוע ההדפסות השונות למשתמש.	printmessage.py

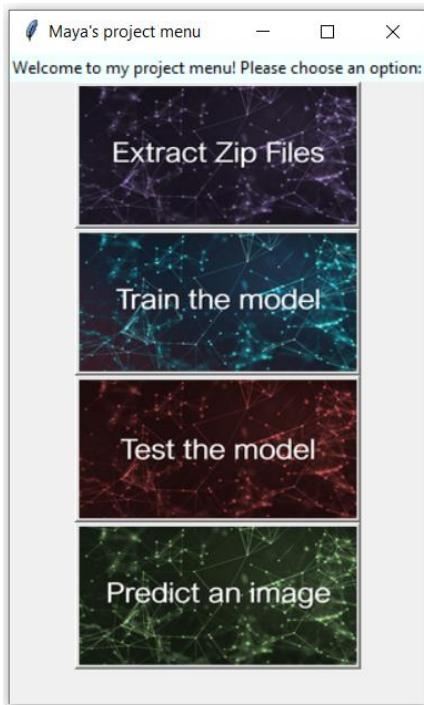
בעמוד הבא אתחליל לפרט על כל אחד מהקבצים.

הסבר ופירוט על כל קובץ `python` בפרויקט:

הקובץ הראשי - `menu_tkinter.py`

קובץ זה אחראי על ניהול התוכנית ועל התקשרות עם המשתמש. על מנת לבצע את תפקיד הcapeטור עליו המשתמש לחץ, קובץ זה מזמן מתוודות משאר הקבצים. בקובץ זה מצויים הפונקציה הראשית בשם `(main_window()`, נפתח בפניו פונקציית שירות עבור המשתמש כאשר המשתמש מריץ את הקובץ הזה, נפתח בפניו חלון עם מספר אפשרויות בהן הוא יכול לבחור.

כפי שניתן לראות משמאל, למשתמש מוצעות 4 אפשרויות. לחיצה על אחד מהcapeטורים



תביא לביצוע המשימה שכתובה על גביו:

1- חילוץ התמונות מקובץ הקיז של מאגר התמונות המלא וכן של מאגר התמונות המיועדות לחיזוי, לתיקיות רגילות עם שם זהה (לא הסימט `zip`). זאת, על מנת שהתוכנית תוכל לגשת אליהן.

בהרצתה הראשונה של הפרויקט, יש ללחוץ קודם כל על הcapeטור הראשון בתפריט על מנת שהtamונות יחולצו, אחרת התוכנית לא תוכל לגשת אליהן. לאחר מכן בהרצתה זו ובהרצות הבאות אין סיבה ללחוץ עלייו, והדבר יגרור הודעה שהגיאה, כיון שהtamונות כבר חולצו.

2- אימון מודל (training)

3- בדיקת המודל (test)

4- ביצוע חיזוי לתמונה ספציפית (prediction)

יש לשם כך למספור נקודות חשובות, שאי הבנה שלhn עלולה לגרום לך לא להצליח להריץ את הפרויקט כראוי:

אם ברצונך לבצע אימון, עליך לשנות את הנתיבים למודל, קובץ התווiotות ותיקיות הגרפים לנטיים חדשים שבעל תהליך אימון נוצרים תיקיות וקובצים אלו מחדר בהתאם לתהליכי האימון הנוכחיים.

אם ברצונך לבצע חיזוי, עליך לשים לב שהנתיבים למודל ולקובץ התווiotות הם נתיבים קיימים משומש עבור אפשרות זו יש צורך במודול שמור ובקובץ תווiotות שמור.

אם ברצונך לבצע טסט מבלי שביצעת אימון לפני כן בהרצתה הנוכחית, עליך לשים לב שהנתיב למודל הוא נתיב קיים משומש עבור אפשרות זו יש צורך במודול שמור.

אם בהרצתה הנוכחית כבר ביצעת אימון וכעת אתה רוצה לבצע טסט או חיזוי, הנתיבים שהיינו חדשניים באימון כתוב קיימים אך אין בעיה ללחוץ כפתור הטסט מיד לאחר סיוע ביצוע האימון.

הסבר המתואdot מפורט בעמוד הבא.

תפקיד	הmethod
זהי המתודה הראשית אשר יוצרת את המסר והכפتورים על גבי שהמשתמש בוחר. במתודה זו מוגדרים ומופעלים ארבעת הcptorim ומחזנות מתוך cptorim ארבע הפונקציות האחראיות לביצוע תפקיך כל כפטור.	main_window()
מתודה זו מגדרה את הנתיבים של מאגר התמונות ומצמנת את המתודה extract_Zip_Files(path, messageNotZip) המוגדרת (היכן messageSucceed, messageNotSucceed) dataset.py על מנת לחץ את התמונות של extractzipfile. והתמונה המיועדת לחיזוי מקובץ הקובץ.	extract_zip_files()
מתודה זו מגדרה את הנתיבים dataset_path (למאגר התמונות המחלוצת), model_path (היכן שיישמר המודל המאומן) graphs_path (היכן שיישמר קובץ התוויות הבינארית) path (היכן שיישמרו הגרפים). המתודה בודקת אם הנתיבים תקינים בעזרת המתודות is_new_and_valid(messageHeb, messageExists, path, hebNotValid = False) is_exists_and_valid(messageHeb, messageNotExists, path, hebNotValid = False) מהקובץ sys.checkdirectorys. בנוסף, היא בודקת האם יש כפילות בתתיבים חדשים (למשל, אותו נתיב עברו המודול ועברו הגרפים). אם יש כפילות כזו, היא מדפסה הודעה שגיאה מתאימה בעזרת המתודה printmessage(). המתודה יוצרת אובייקט חדש בשם training של המחלקה TrainAndTest שבקובץ py.train_test_model ומעבירה את הפרמטרים לבניית המחלקה. לאחר מכן היא מפעילה את המתודה הפומבית () handle_train על גבי אובייקט זה על מנת לבצע את תהליך האימון.	train_model()
מתודה זו מגדרה את הנתיבים model_path, dataset_path ומעבירה אותן כפרמטרים לבניית המחלקה train_test_model שבקובץ py.train_test_model על גבי אובייקט חדש מסוג המחלקה testing. לאחר מכן היא מפעילה את המתודה הפומבית () handle_test על גבי אובייקט זה על מנת לבצע את test(). path, graphs_path, labels_path אין רלוונטיים לtest אלא לתהליכי האימון. כמו כן, המתודה בודקת אם הנתיבים תקינים בעזרת המתודה is_exists_and_valid(messageHeb, messageNotExists, .checkdirectorys(path, hebNotValid = False))	test_model()
מתודה זו מגדרה את הנתיבים model_path, labels_path, ImagePrediction ומעבירה אותן כפרמטרים לבניית המחלקה classifyimage. המתודה יוצרת את הנתיב image_path לתמונה המיועדת לשיזוג על ידי חיבור של נתיב מאגר התמונות לחיזוי ושם התמונה לחיזוי. לאחר מכן היא מפעילה את המתודה הפומבית () handle_classify על גבי האובייקט ומעבירה לה את path.image כפרמטר על מנת לבצע את prediction(). כמו כן, המתודה בודקת אם הנתיבים תקינים בעזרת המתודה is_exists_and_valid(messageHeb, messageNotExists, .checkdirectorys(path, hebNotValid = False))	predict_image()

תדפיו הקיימים:

```
"""
this python file is the main file that runs the project
"""

import tkinter as tk
from PIL import ImageTk, Image

import train_test_model
import checkdirectorys
import printmessage
import classifyimage
import extractzipfiles
import config

def main_window():
    """
        The main method that runs the whole project. creates the screen and buttons on it that the user
        selects.
        defines and activates the four buttons. responsible for performing the function of each button
        (each function is called from her button).
    """

    #The dimensions of the buttons
    WIDTH = 200
    HEIGHT = 100

    main_window = tk.Tk()
    main_window.title("Maya's project menu")
    main_window.geometry("300x470") #The dimensions of the window

    main_lbl = tk.Label(main_window,
                        text="Welcome to my project menu! Please choose an option:",
                        foreground="black",
                        background="azure")
    main_lbl.pack()

    #Button 1- Extract Zip Files
    zip_photo = Image.open(config.Zip_Button_Image_File_Name)
    zip_photo = zip_photo.resize((WIDTH, HEIGHT), Image.ANTIALIAS)
    zip_photo = ImageTk.PhotoImage(zip_photo)
    btn_zip = tk.Button(main_window, image=zip_photo, command=extract_zip_files)
    btn_zip.pack()

    #Button 2- Train the model
    train_photo = Image.open(config.Train_Button_Image_File_Name)
    train_photo = train_photo.resize((WIDTH, HEIGHT), Image.ANTIALIAS)
    train_photo = ImageTk.PhotoImage(train_photo)
    btn_train = tk.Button(main_window, image=train_photo, command=train_model)
    btn_train.pack()

    #Button 3- Test the model
    test_photo = Image.open(config.Test_Button_Image_File_Name)
```

```

test_photo = test_photo.resize((WIDTH, HEIGHT), Image.ANTIALIAS)
test_photo = ImageTk.PhotoImage(test_photo)
btn_test = tk.Button(main_window, image=test_photo, command=test_model)
btn_test.pack()

#Button 4- Predict an Image
predict_photo = Image.open(config.Predict_Button_Image_File_Name)
predict_photo = predict_photo.resize((WIDTH, HEIGHT), Image.ANTIALIAS)
predict_photo = ImageTk.PhotoImage(predict_photo)
btn_predict = tk.Button(main_window, image=predict_photo, command=predict_image)
btn_predict.pack()

main_window.mainloop()

def extract_zip_files():
    """
    responsible for extracting the zip files of the images
    """

    dataset_zip_path = config.Data_Set_Zip_File_Name
    data_predictions_path = config.Data_Prediction_Zip_File_Name
    extractzipfiles.extract_Zip_File(dataset_zip_path,
        "the dataset path is not a zip file",
        "Finished to extract dataset dirs", "Could not finish to extract dataset dirs")
    extractzipfiles.extract_Zip_File(data_predictions_path, "the predictions path is not a zip file",
        "Finished to extract predictions dirs",
        "Could not finish to extract predictions dirs")

def train_model():
    """
    responsible for training the model
    """

    dataset_path = config.Data_Set_Dir_Name
    checkdirectorys.is_exists_and_valid("Error! the images path contains hebrew letters", "Error! the
    images path does not exist", dataset_path, True)

    model_path = config.Trained_Model_Dir_Name
    checkdirectorys.is_new_and_valid("", "Error! the model path is already exists", model_path, False)

    labels_path = config.Traiendo_Model_Labels_File_Name
    checkdirectorys.is_new_and_valid("", "Error! the labels path is already exists", labels_path, False)

    graphs_path = config.Trained_Model_Graphs_Dir_Name
    checkdirectorys.is_new_and_valid("", "Error! the graphs path is already exists", graphs_path, False)

    if model_path == labels_path:
        printmessage.printError("Error! model path defined is the same as the labels path defined. file
        will be override")
        return

```

```

if model_path == graphs_path:
    printmessage.printError("Error! model path defined is the same as the graphs path defined. file
will be override")
    return

if graphs_path == labels_path:
    printmessage.printError("Error! graphs path defined is the same as the labels path defined. file
will be override")
    return

training = train_test_model.TrainAndTest(dataset_path, model_path, labels_path, graphs_path)
training.handle_train()

def test_model():
    """
    responsible for testing the model
    """

    dataset_path = config.Data_Set_Dir_Name
    checkdirectorys.is_exists_and_valid("Error! the images path contains hebrew letters", "Error! the
images path does not exist", dataset_path, True)

    model_path = config.Trained_Model_Dir_Name
    checkdirectorys.is_exists_and_valid("", "Error! the model path does not exist", model_path, False)

    testing = train_test_model.TrainAndTest(dataset_path, model_path, "", "")
    testing.handle_test()

def predict_image():
    """
    responsible for predicting an image
    """

    model_path = config.Trained_Model_Dir_Name
    checkdirectorys.is_exists_and_valid("", "Error! the model path does not exist", model_path, False)

    labels_path = config.Traiend_Model_Labels_File_Name
    checkdirectorys.is_exists_and_valid("", "Error! the labels path does not exist", labels_path, False)

    image_path = config.Data_Prediction_Dir_Name+"/"+config.Current_Image_To_Predict
    checkdirectorys.is_exists_and_valid("Error! the image path contains hebrew letters", "Error! the
image path does not exist", image_path, True)

    predicting = classifyimage.ImagePrediction(model_path, labels_path)
    predicting.handle_classify(image_path)

if __name__ == '__main__':
    main_window()

```

```

if model_path == graphs_path:
    printmessage.printError("Error! model path defined is the same as the graphs path defined. file
will be override")
    return

if graphs_path == labels_path:
    printmessage.printError("Error! graphs path defined is the same as the labels path defined. file
will be override")
    return

training = train_test_model.TrainAndTest(dataset_path, model_path, labels_path, graphs_path)
training.handle_train()

def test_model():
    """
    responsible for testing the model
    """

    dataset_path = config.Data_Set_Dir_Name
    checkdirectory.is_exists_and_valid("Error! the images path contains hebrew letters", "Error! the
images path does not exist", dataset_path, True)

    model_path = config.Trained_Model_Dir_Name
    checkdirectory.is_exists_and_valid("", "Error! the model path does not exist", model_path, False)

    testing = train_test_model.TrainAndTest(dataset_path, model_path, "", "")
    testing.handle_test()

def predict_image():
    """
    responsible for predicting an image
    """

    model_path = config.Trained_Model_Dir_Name
    checkdirectory.is_exists_and_valid("", "Error! the model path does not exist", model_path, False)

    labels_path = config.Traiened_Model_Labels_File_Name
    checkdirectory.is_exists_and_valid("", "Error! the labels path does not exist", labels_path, False)

    image_path = config.Data_Prediction_Dir_Name+"/"+Current_Image_To_Predict"
    checkdirectory.is_exists_and_valid("Error! the image path contains hebrew letters", "Error! the
image path does not exist", image_path, True)

    predicting = classifyimage.ImagePrediction(model_path, labels_path)
    predicting.handle_classify(image_path)

if __name__ == '__main__':
    main_window()

```

קובץ התצורה- config.py

בקובץ זה מוגדרים הנתיבים השונים הנמצאים בשימוש בפרויקט:

הסבר	שם הנתיב
נתיב לקובץ zip של מאגר התמונות המלא	Data_Set_Zip_File_Name
נתיב לתיקייה רגילה ומחולצת של מאגר התמונות המלא	Data_Set_Dir_Name
נתיב לקובץ zip של התמונות המיועדות לחיזוי	Data_Prediction_Zip_File_Name
נתיב לתיקייה רגילה ומחולצת של התמונות המיועדות לחיזוי	Data_Prediction_Dir_Name
שם התמונה המיועדת לחיזוי כולל הסימות	Current_Image_To_Predict
נתיב למודל האימון	Trained_Model_Dir_Name
נתיב לקובץ התוויות שנשמר במהלך אימון המודל השמור	Traiend_Model_Labels_File_Name
נתיב לתיקיית הגרפים שנוצרו במהלך אימון של המודל	Trained_Model_Graphs_Dir_Name
נתיב לתמונה של הכפטור הראשון	Zip_Button_Image_File_Name
נתיב לתמונה של הכפטור השני	Train_Button_Image_File_Name
נתיב לתמונה של הכפטור השלישי	Test_Button_Image_File_Name
נתיב לתמונה של הכפטור הרביעי	Predict_Button_Image_File_Name

באפשרות לשנות את הגדרות הנתיבים בקובץ זה על מנת להתאים את הרצת הפרויקט לאופן בו המידע שמור במחשב האישי שלך.

אם ברצונך לבצע אימון, עליך לשנות את הנתיבים למודל, קובץ התוויות ותיקית הגרפים לנטייבים חדשים שבכל תהליך אימון נוצרים תיקיות וקובציים אלו מוחדר בתאום לתהליך האימון הנוכחי.

אם ברצונך לבצע חיזוי, עליך לשים לב שהנתיבים למודל ולקובץ התוויות הם נתיבים קיימים משומש שעבור אפשרות זו יש צורך במודל שמור ובקובץ תוויות שמור.

אם ברצונך לבצע טסט מוביל שביצעת אימון לפני כן בהרצה הנוכחית, עליך לשים לב שהנתיב למודל הוא נתיב קיים משומש שעבור אפשרות זו יש צורך במודל שמור.

אם בהרצה הנוכחית כבר ביצעת אימון וכעת אתה רוצה לבצע טסט או חיזוי, הנתיבים שהו חדשים באימון כתם כבר קיימים لكن אין בעיה ללחוץ כפתור הטסט מיד לאחר סיום ביצוע האימון.

יש לשים לב שלפני ההרצה הראשונית צריך ללחוץ פעמיים אחת (ואהחרונה) על כפתור חילוץ קבצי הקז על מנת שהתוכנית תוכל לגשת לתמונות.

בקובץ הראשי בפרויקט config.py ישן פונקציית `config` שבודקת אם נתונים המוגדרים בקובץ config.py. השימוש בקובץ תצורה במקום הגדרה של נתיבים בקובציים השונים בפרויקט או בקש תלת הוא מוצע, מסודר, ולפעמים גם נוח יותר. במקום לחפש את המקומות בהם יש לבצע שינויים בהגדירות הדיפולטיביות של הנתיבים או להתעסק עם הזנת קלטים בכל הרצה של הפרויקט, ביכולתך לגשת לקובץ אחד ויחיד המסדר את העניין הזה ולבצע שם את עדכוני או שינויי הנתיבים במידת הצורך.

תדPIO הקוד:

```
"""
this file defines the paths required to run the project
"""

Data_Set_Zip_File_Name = "dataset.zip"
Data_Set_Dir_Name = "dataset"

Data_Prediction_Zip_File_Name = "predictions.zip"
Data_Prediction_Dir_Name = "predictions"

Current_Image_To_Predict = "5_00001.jpg"

Trained_Model_Dir_Name = "trained_model"
Traiend_Model_Labels_File_Name = "labels"
Trained_Model_Graphs_Dir_Name = "graphs"

Zip_Button_Image_File_Name = "zip_btn.png"
Train_Button_Image_File_Name = "train_btn.png"
Test_Button_Image_File_Name = "test_btn.png"
Predict_Button_Image_File_Name = "predict_btn.png"
```

אם שיר לפרט על כל אחד מהקבצים תוך התייחסות לבפתרים המפעילים אותם:**כפתרו ראשון****קובץ חילוץ התמונות- zipfiles.py**

קובץ זה אחראי על חילוץ התמונות מקובץ הקובץ של מאגר התמונות המלא וכן של מאגר התמונות המיועדות לחיזוי, לתיקיות רגילים עם שם זהה (לא הסימנת zip). זאת, על מנת שהתוכנית תוכל לגשת אליהן. הקובץ מכיל מתודה אחת האחראית לחילוץ.

המתודה () extract_Zip_Files() מזמנת מתוך המתודה extract_zip_files() שבקובץ .menu_tkinter.py

שמריצים את הפרויקט, יש ללחוץ קודם כל על הכפתור הראשון בתפריט על מנת שהתמונות יחולצו. לאחר מכן אין סיבה ללחוץ עליו, והדבר יגרור הודעה שגיאה, כיון שהתמונות כבר חולצו.

תפקיד	המתודה
<p>המתודה מבצעת את החילוץ עצמו, במידה ואכן מדובר בקובץ zip. ראשית, היא בודקת תנאי זה. אם הוא מתקיים, בעזרת מתודה בשם <code>is_new_and_valid</code> בקובץ <code>checkdirectorys</code> בודקת האם הנתיב שהועבר אליה חדש וחוקן. אם כן, היא מבצעת את החילוץ לתיקייה רגילה עם שם זהה לשם קובץ הקובץ לא הסימנת zip. כמו כן, המתודה אחראית להדפסת הודעות מתאימות אודות תהליך החילוץ או הודעות שגיאה במקרה הצורך, וונזרת לשם כך במתודה <code>is_new_and_valid(messageHeb, messageExists, path, hebNotValid = False)</code> מהמחלקה שבקובץ <code>checkdirectorys</code> מקבלת ממנה הודעות שונות להדפסה ומדפיסה את ההודעות המתאימות. חלק מההודעות להדפסה מודפסות על ידי המתודה <code>extract_Zip_File(path, messageNotZip, messageSucceed, .messageNotSucceed)</code></p>	extract_Zip_File(path, messageNotZip, messageSucceed, messageNotSucceed)

תדPIO הקוד:

```
#####
this python file is responsible for extracting the zip files
#####

from zipfile import ZipFile

import checkdirectorys
import printmessage


def extract_Zip_File(path, messageNotZip, messageSucceed, messageNotSucceed):
    #####
    # preforms the actual extracting or print a message if it faild (because the paths were not valid)

    param path: a path to a zip file for extracting
    param messageNotZip: a message for printing in case the path is not a directory to a zip file
    param messageSucceed: a message for printing in case the extracting process succeed
    param messageNotSucceed: a message for printing in case the extracting process did not succeed
    #####
    path_and_extension = path.split(".")
    if path_and_extension[len(path_and_extension)-1] != "zip":
        printmessage.printError(messageNotZip)
    else:
        extracted_dir_path = path_and_extension[0]
        with ZipFile(path, 'r') as zipObj:
            # Extract all the contents of zip file in different directory
            is_succeed = checkdirectorys.is_new_and_valid("", "", extracted_dir_path, True)
            if is_succeed:
                zipObj.extractall(extracted_dir_path)
                printmessage.printProcess(messageSucceed)
            if not is_succeed:
                printmessage.printError(messageNotSucceed) #####
```

כפתרו שני

קובץ הגדרת מבנה המודל - themodel.py

בקובץ זה ישנה מחלקה בשם My Model אשר מגדירה את מבנה המודל שלו גביו מבוצע תהליך האימון ולמידת התמונות. המחלקה מכילה מתודה סטטית אחת בשם buildModel.

מתודה זו מקבלת את מדדי התמונות, מספר הקטגוריות וההction function שתתבצע בשכבה האחרון של המודל (softmax function מוגדרת כברירת מחדל), ובונה את שכבות המודל בעדרתם. שכבות המודל מהוות למעשה מושג מלאכותי, רשת נירונים מלאכותית אשר לומדת את התמונות. בכל אחת מהשכבות מספר שונה של נירונים מלאכותיים, אשר יכולים "لتקשר" עם הנירונים שבשכבות הסמוכות. זאת, בעזרת משקלים שמהווים חישוב סטטיסטי אשר יניב את אחוז ההצלחה הגבוה ביותר. בעת ציון המתודה buildModel ערכי המשקלים הללו רנדומליים, ובעת אימון המודל הם משתנים בהתאם למערכות התמונות שרצים על גביהם. לבסוף המתודה מחזירה את המודל שבנתה.

המתודה מזמנת בקובץ py.train_test_model על ידי המתודה __train__(trainX, testX, trainY, testY, valX, valY, config) על מנת לאתחל את המודל כדי להשתמש בו בתהליך האימון. לאחר תהליכי האימון המתודה שומרת את המודל בנתיב המוגדר בקובץ config.prediction. בעזרתו ניתן לשאזרת המודל השמור מtbody prediction.

השכבה הראשונה במודל היא שכבת הקלט input_shape המגדירה את מדדי התמונה - (50,1). השכבה الأخيرة במודל היא שכבת הפלט, אשר משתמשת בפונקציית softmax במידה אין-קלט אחר. זהה למעשה שכבת הסיגוג. בשכבה שלפניה מוגדר מספר הקטגוריות האפשריות, למשל מספר נירוני הפלט. המתודה למעשה בוחרת את נירון הפלט המאומן היא צפואה להציגו אל אחד מנירוני הפלט. המתודה בוחרת בוחרת את החיזוי בעל הסיכוי הגבוה ביותר להיות נכון. ניתן למצוא את תרשימים שכבות המודל בפרק "נספחים".

הסביר על פונקציית האקטיבציה softmax - זהה פונקציה מתמטית המmirה וקטור מספרים לוקטור הסתברויות, כאשר ההסתברויות של כל ערך פרופורציונליות לsolem היחס של כל ערך בוקטור. פונקציית softmax משמשת activation function בשכבת הפלט של רשתות נירונים מלאכותיות עבור בעיות classification multi-class בהן יש יותר משלתי תוויות (קטגוריות).

בחратי במודל זה לאחר תהליכי ארכ'ר של ניסוי וטעייה, שבסופה הגיעו למסקנה שמודל זה הוא המתאים ביותר לפROYKT שלו, בזכות התוצאות הטובות שהוביל ביחס למודלים האחרים בהם התנסית.

תדפיו הקיימים:

```
"""
this python file contains the model
"""

from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K

class MyModel:

    @staticmethod
    def buildModel(width, height, depth, classes, finalActivationFunction = "softmax"):
        """
        The model is built within this static method.
        The method initializes the model along with the input shape to be "channels last" and the
        channels dimension itself.

        params width, height, depth, classes: the image's dimensions
        param classes: the categories
        param finalAct: the final activation function- "softmax" by default

        return: the constructed network architecture.
        """

        model = Sequential()
        inputShape = (height, width, depth)
        chanDim = -1

        #if we are using "channels first", update the input shape and channels dimension.
        #image_data_format() Returns the default image data format convention ('channels_first' or
        'channels_last'). Returns a string, either 'channels_first' or 'channels_last'
        #For "tensorflow" or "cntk" backends, it should be "channels_last". For "theano", it should be
        "channels_first".

        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width)
            chanDim = 1

        model.add(Conv2D(32, (3, 3), padding="same", input_shape = inputShape))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(MaxPooling2D(pool_size=(3, 3)))
        model.add(Dropout(0.25))
```

```
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(classes))
model.add(Activation(finalActivationFunction))

#return the constructed network architecture
return model
```

הקובץ האחראי על ייצרת ושמירת הגרפים- makegraphs.py

קובץ זה, שמצוון כאשר מתבצע אימון המודל מתוך המתודה (`handle_train()` שבקובץ `train_test_model.py`, אחראי על ייצור הגרפים המતארים את תהליכי האימון, הצגתם לשימוש ושמירתם בתיקייה יייעודית. הקובץ מכיל מתודה ראשית האחראית לכך, אשר מזמןת מספר מטותות, כל אחת האחראית על תמונה/ מערכת גרפים אחרת. בנוסף, מתודה זו מדפיסה הודעה מתאימה עם מקום התמונות של הגרפים במחשב.

תפקיד	מתודה
<p>מתודה האחראית על ייצרת הגרפים המתארים את תהליכי האימון על ידי הצגת השנתנות ערכי <code>accuracy</code> ו-<code>loss</code> לאורך. המתודה מקבלת את היסטוריות אימונו המודול ואת הנתיב לתיקייה בה ישמרו הגרפים ובעזרתו יוצרת תיקייה בה ישמרו כל התמונות ומגדירה את נתיבי כל תמונה. בנוסף המתודה מקבלת את ה-<code>epoches</code>- מספר הפעמים שהתמונה רצוח על המודל. המתודה מזמןת כל אחת מהמתודות היווצרות ושמורות את הגרפים. היא מעבירה לכל אחת מהן כפרמטר את הנתיב הספציפי בו ישמר הגרף שיוצרת אותה מתודה, וכן את היסטוריות אימון המודול שבעזרתה כל מתודה תשרטט את הגרפים. בנוסף, היא מעבירה למתודה <code>graph2(H, plot_path2, epoches)</code> גם את ה-<code>epoches</code>.</p>	<code>graphs(H, plot_path, epoches)</code>
<p>מתודה היוצרת את מערכת הצלרים הראשונה, ועל גביה את הגרפים המציגים את השנתנות ערכי <code>accuracy</code> וה-<code>loss</code> über החומר <code>train</code> והה-<code>validation</code> לאורך כל תהליכי האימון. המתודה מشرطת את הגרפים באמצעות היסטוריות הלימוד שקיבלה כפרמטר <code>H</code> ובעזרת הפרמטר <code>epoches</code> שקיבלה, ושומרת את תמונה תמונה הגרפים בנטייב שקיבלה לשם כך ה-<code>plot_path1</code>. לאחר מכן היא מציגה את התמונה עם הגרפים לשימוש.</p>	<code>graph1(H, plot_path1, epoches)</code>
<p>מתודה היוצרת את מערכת הצלרים השנייה, ועל גביה את הגרפים המતארים את פונקציית <code>loss</code> <code>cross_entropy</code> עבור הה-<code>train</code> והה-<code>validation</code>. פונקציה זו קטנה כל שההסתברות החזויה שספק המודל זרה לתווית האמיתית של התמונה. המתודה מشرطת את הגרפים באמצעות ההיסטוריה הלימוד שקיבלה כפרמטר <code>H</code>, ושומרת את תמונה הגרפים בנטייב שקיבלה לשם כך <code>plot_path2</code>. לאחר מכן היא מציגה את התמונה עם הגרפים לשימוש.</p>	<code>graph2(H, plot_path2)</code>

<p>המетодה הייצרת את מערכת הצירים השנייה, ועל גביה את הגرافים המתאים את מדד classification accuracy עבור החומר validation. ממד זה מסכם את הביצועים של המודל כמספר התחזיות הנכונות חלקו המספר הכלל של התחזיות. הגרף עולה ככל שהחלק של התחזיות הנכונות מתווך סך התחזיות גדול.</p> <p>המетодה משרטת את הגرافים באמצעות היסטוריית הלימוד שקיבלה כפרמטר H, ושומרת את תמונה הגرافים בתיב שקיבלה בשם כר3 plot_path3. לאחר מכן היא מציגה את התמונה עם הגرافים למשתמש.</p>	graph3(H, plot_path3)
---	-----------------------

תדפיו הקיימים:

```

"""
this python file is responsible for making and saving graphs of the training process
"""

import matplotlib
matplotlib.use( 'tkagg' )
import matplotlib.pyplot as plt
import numpy as np
import os

import printmessage

def graphs(H, plot_path, epoches):
    """
    responsible for making, saving and presenting the graphs of the training process
    & printing a message with the location of the plots in the computer.

    param plot_path: the directory in which the graphs will be saved
    param epoches: the number of epochs- The number of times the images ran on the model for
learning purposes
    """

    os.mkdir(plot_path)
    plotpath1 = plot_path + r"\plot1.png"
    plotpath2 = plot_path + r"\plot2.png"
    plotpath3 = plot_path + r"\plot3.png"
    graph1(H, plotpath1, epoches)
    graph2(H, plotpath2)
    graph3(H, plotpath3)
    printmessage.printProcess("Graphs of the training process are in " + plot_path)

def graph1(H, plot_path1, epoches):
    """
    Creates a png image file in which it draws the learning graph of the model, saving it and showing it
at the end of the training.
    (training & validation accuracy & loss)

    param H: the history of the model training
    param plot_path1: the directory in which graphs 1 will be saved
    param epoches: the number of times the images ran on the model for learning purpose
    """

    plt.style.use("ggplot")
    plt.figure()
    N = epoches
    plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
    plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
    plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
    plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")

```

```

plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="upper left")
plt.savefig(plot_path1) #save plot to file
plt.show()

def graph2(H, plot_path2):
    """
    Creates a png image file in which it draws the learning graph of the model, saving it and showing it
    after the user closed the first plot.
    (training & validation cross entropy loss)

    param H: the history of the model training
    param plot_path2: the directory in which graphs 2 will be saved
    """
    plt.style.use("ggplot")
    plt.figure()
    plt.title('Cross Entropy Loss')
    plt.plot(H.history['loss'], color='blue', label='train')
    plt.plot(H.history['val_loss'], color='orange', label='val')
    plt.legend(loc="upper left")
    plt.savefig(plot_path2)
    plt.show()

def graph3(H, plot_path3):
    """
    Creates a png image file in which it draws the learning graph of the model, saving it and showing it
    after the user closed the second plot.
    (training & validation classification accuracy)

    param H: the history of the model training
    param plot_path3: the directory in which graphs 3 will be saved
    """
    plt.style.use("ggplot")
    plt.figure()
    plt.title('Classification Accuracy')
    plt.plot(H.history['accuracy'], color='blue', label='train')
    plt.plot(H.history['val_accuracy'], color='orange', label='val')
    plt.legend(loc="upper left")
    plt.savefig(plot_path3)
    plt.show()

```

כפתרו שני ושלישי**הקובץ האחראי על אימון ובדיקה המודל - train_test_model.py**

קובץ זה אחראי על אימון ובדיקה המודל. בקובץ זה ישנה מחלוקת בשם TrainAndTest שבה בין השאר שתי מתודות פומביות, האחת אחראית על החומר (validation) יחד עם המודל (train), והשנייה אחראית על המודל (test). שתיהן נועזרות במתודות פרטיות על מנת לבצע את תפקידן.

ההחלטה שיש צורך בחלוקת משותם של מетодות רבות בקובץ זה(ZKOKOT) לאותם פרמטרים. זו גמ הסיבה שהחלטתי שהטיפול בחומר וטיפול בtest יתרכחו באותה ישות-חלוקת אחת.

בחלוקת ישן תכונות המכילות נתיבים ומשתנים שונים, מетодה פומבית האחראית על החלטה, מетодה פומבית נוספת האחראית על המודל, test, ושלל מетодות עזר.

למעשה, על מנת לבצע את החומר, בקובץ `py`, `menu_tkinter` מетодה `train_model()` יצרתי אובייקט מסווג המחלוקת, העברתי לבנאיחלוקת את הפרמטרים החדשניים וזמן נתיבים. על מנת לבצע את ה`train`, בקובץ `handle_train()`. על מנת לבצע את ה`test`, בקובץ `menu_tkinter.py` יצרתי אובייקט מסווג המחלוקת, והעבירתי לבנאי `test_model()`. המחלוקת את הפרמטרים החדשניים וזמן נתיבים. על מנת לבצע את המETHODICA ה`handle_test()`.

הסבר לבנייהחלוקת:

תפקיד	ערך	תמונה
מחזיקה את הנתיב למאגר התמונות המלא	Input	<code>__dataset_path</code>
הנתיב בו ישמר המודל (במקרה של טריין) או המודל השמור (במקרה של טסט)	Input	<code>__model_path</code>
הנתיב בו תוויות התמונות ישמרו (במקרה של טריין)	Input	<code>__labels_path</code>
הנתיב בו ישמרו הגרפים (במקרה של טריין)	Input	<code>__graphs_path</code>
מספר הפעמים שירצוז התמונות על גבי המודל לצורך ביצוע האימון/ הלמידה (במקרה של טריין)	50	<code>__EPOCHES</code>
קצב הלמידה- ערך שקובע כמה לשנות את המודל בתגובה לשגיאה המשוערת בכל פעם שימושי המודול מתעדכנים (במקרה של טריין)	1×10^{-3}	<code>__LEARNING_RATE</code>
מספר התמונות שהמודול יעבד במקביל בזמן האימון/ הלמידה (במקרה של טריין)	32	<code>__BATCH_SIZE</code>

ממדים התמונה שהמודל יריץ. (width, height, color)	(50,50,1)	<code>__IMAGE_DIMS</code>
רשימה שתחזיק את מערכימ ה מייצגים את התמונות.	[]	<code>__data</code>
רשימה שתחזיק את התוויות של כל התמונות.	[]	<code>__labels</code>

הסביר שני התכונות האחרונות: הפונקציה `__prepare_Images()` שבה מתבצעת טיעינת התמונות ממירה את התמונות למערכים ושומרת אותן ברשימה `__data`. במקביל פונקציה זו שומרת את הקטגוריה של כל תמונה ברשימה `__labels`.

הסביר המתודות:

תפקיד	המתודה
<p>מתודה פומבית של המחלקה, אשר ניתנת לקרוא לה רק מטור אובייקט מאוחROL של המחלקה (זו הסיבה שיש בנאי עם פרמטרים).</p> <p>מתודה זו דואגת לביצוע האימון בעזרת זימון המתוודות הפרטיות, אף מזמןת את המתודה <code>makegraphs graphs</code> שבקובץ <code>binarize</code> אשר יוצרת ושומרת גרפים מתאימים את תהליך האימון.</p> <p>המתודה מזמןת את המתודה הפרטית <code>__prepare_Images()</code> על מנת להcin את התמונות <code>test</code>. לאחר כך המתודה שומרת את רשיימת התוויות בקובץ <code>binarize</code> שישמש בחלוקת התמונות. בנוסף המתודה יוצרת טרנספורמציה מן הקובץ הבינארי שתשמש לאימון המודל. לאחר מכן המתודה מזמןת את <code>train(self, trainX, testX, valX, valY, lb)</code> היסטורית אימון המודל שהוא מחזירה במשתנה <code>training_history</code>. לאחר מכן היא מעבירה את הנתיב לגרפים, את ההיסטוריה האימון ואת <code>EPOCHES</code> למתודה <code>makegraphs graphs</code> שבקובץ <code>binarize</code>.</p>	<code>handle_train()</code>
<p>מתודה פומבית נוספת של המחלקה. מתודה זו דואגת לביצוע <code>test</code> בעזרת זימון המתוודות הפרטיות.</p> <p>המתודה מזמןת את המתודה הפרטית <code>__prepare_Images()</code> על מנת להcin את התמונות <code>test</code>. לאחר כך המתודה שומרת את רשיימת התוויות בקובץ <code>binarize</code> שישמש בחלוקת התמונות. בנוסף המתודה יוצרת טרנספורמציה מן הקובץ הבינארי שתשמש</p>	<code>handle_test()</code>

<p>לאימון המודל. לאחר מכן מתודה זו מזמנת את המתודה הפרטית (<code>split_data()</code> על מנת לבצע את חלוקת מאגר התמונות. אחר כך היא טעונה את המודל. לסיום היא מזמנת את המתודה הפרטית <code>evaluate_Model(testX, testY, model, batch_size = 32)</code> אשר מבצעת את <code>test</code> עצמו.</p>	
<p>מתודה פרטית האחראית על הכנת התמונות לתהילך האימון. המתודה אחראית למעשה על טיענת התמונות ממאגר התמונות והתאמתן לריצה על גבי המודל. האופן בו תפקיך זה מתבצע: המתודה משנה את גודלי ממדי התמונה לגודל אחיד הקבוע בתוכנה <code>IMAGE_DIMS</code>, ממירה את התמונה לRGB ומשמירה אותו למערכים המחזיקים את הפיקסלים של התמונות. כמו כן היא מקטינה את טווח הפיקסלים של המערכים השמורים ב <code>data</code> מתחום [0,255] לתחום [1, 0] (נורמל). בנוסף מתודה זו מחלצת שם התקייניה שבה כל תמונה נמצאת את הקטגוריה שלה ומתאיםה ביןיה באמצעות שתי רשימות המהוות חלק מתקנות המחלקה: רשיימה אחת מאכלסת את התוויות התמונות <code>data</code> והשנייה את התוויות <code>labels</code> (סוג התמונה- הספרה), עליה פירטתי לעיל. ההוספה לרשימות נעשית ביחד בלולאה ולכן ישנה התאמת במספר האינדקס בין שתי הרשימות. כך ניתן לחבר בין כל תמונה לקטגוריה שלה. המתודה אינה מחייבת ערך מסוים השניי נועשה על תכונות האובייקט של המחלקה.</p>	<code>__prepare_Images()</code>
<p>מתודה פרטית אשר מחלקת את מאגר התמונות ל 3 קטגוריות: 1. תמונות המיועדות ל <code>train</code> (70%) 2. תמונות המיועדות ל <code>test</code> (20%) 3. תמונות המיועדות ל <code>validation</code> (10%) המתודה מחייבת את המידע המחולק. <code>X</code> = <code>trainX, testX, valX</code> = רשימות המרכיבים המיצגים את התמונות <code>Y</code> = <code>trainY, testY, valY</code> = רשימות התוויות של התמונות</p>	<code>__split_data()</code>

<p>מתודת פרטיטית אשר מבצעת את אימון המודל בפועל.</p> <p>המתודה מבצעת רצף פעולות שבסופן אימונם המודל עצמו: ראשית, היא בונה את מוחול התמונות לצורך הגדלת <code>data</code> שתתרחש בהמשך, תוך כדי האימון. זאת על מנת לשפר את ביצועי המודל, שיופעל על מאגר תמונות קטן יחסית. כמו כן, המתודה קובעת אלגוריתם אופטימיזציה אשר ישמש לאופטימיזציה של המודל, קובעת את תצורת המודל, יוצרת טבלה של סיכום המודל, ולבסוף, כאמור, מכשירה את רשת הנירונים, כלומר מבצעת את אימון המודל. לבסוף, המתודה שומרת את המודל המאומן בקובץ ואת קובץ התוויות הבינארי כדי שייהי ניתן להשתמש במליצה הנוכחית גם בהרצאות אחרות של התוכנית ובשbill שהייתה ניתנת לביצוע <code>predict</code> מבל' לבצע את האימון בכל פעם מחדש. לבסוף המתודה מחזירה את היסטוריה אימון המודל, שתשתמש לבניית הגרפים בהמשך.</p>	<code>__train(self, trainX, testX, valX, __trainY, testY, valY, lb)</code>
<p>מתודת פרטיטית זו מבצעת את <code>test</code> בפועל.</p> <p>זהו אחת מהדריכים להעריך את דיוקו של המודל.</p>	<code>__evaluate_Model (testX, testY, themodel, batch_size=32)</code>

תדPIO הקיים:

```
"""
this python file is responsible for the training and for the test
"""

import matplotlib
matplotlib.use("Agg")

#import the necessary packages
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split

from imutils import paths
import numpy as np
import random
import pickle
import cv2
import os

from keras.models import load_model

import themodel
import printmessage
import makegraphs

class TrainAndTest():

    def __init__(self, dataset_path, model_path, labels_path, graphs_path):
        """
        param dataset_path: the data set directory
        param model_path : the directory the user chose to save the trained model OR
        param labels_path: the directory the user chose to save the images labels OR
        param graphs_path: the directory that the user chose to save the graph images in

        Constructs a new 'TrainAndTest' object
        """

        self.__dataset_path = dataset_path
        self.__model_path = model_path
        self.__labels_path = labels_path
        self.__graphs_path = graphs_path

        self.__EPOCHS = 50 #number of epochs
        self.__LEARNING_RATE = 1e-3 #learning rate
        self.__BATCH_SIZE = 32 #batch size
        self.__IMAGE_DIMS = (50, 50, 1) #image dimensions
```

```

self.__data = [] #list of all the images as arrays
self.__labels = [] #list labels of all the images

def handle_test(self):
    """
    a public method that manages the test process
    """

    #preparing the data for testing
    self.__prepare_Images()

    #binarize the labels (a tool for classification)
    lb = LabelBinarizer()

    #Linear transformation
    self.__labels = lb.fit_transform(self.__labels)

    #spliting the date for train, test and validation
    (trainX, testX, valX, trainY, testY, valY) = self.__split_data()

    #loading the model
    model = load_model(self.__model_path)

    self.__evaluate_Model(testX, testY, model, batch_size=32)

def handle_train(self):
    """
    a public method that manages the train section: responsible for:
    the training process, updating the model & label paths,
    creating graphs that describe the learning of the model, and saving them in the plot folder.
    """

    #preparing the data for training
    self.__prepare_Images()

    #binarize the labels (a tool for classification)
    lb = LabelBinarizer()

    #Linear transformation
    self.__labels = lb.fit_transform(self.__labels)

    #spliting the date for train, test and validation
    (trainX, testX, valX, trainY, testY, valY) = self.__split_data()

    #train the model and get the history of the training
    training_history = self.__train(trainX, testX, valX, trainY, testY, valY, lb)

    #making, saving & showing the graphs
    makegraphs.graphs(training_history, self.__graphs_path, self.__EPOCHS)

```

```

def __prepare_Images(self):
    """
    a private method that prepare the images for running on the model:
    Resizes the images to a uniform size I set (50, 50), converts the images from RGB to GrayScale,
    then converts the images to an array,
    reduces its pixel range from [0,255] to [0,1].
    """

    printmessage.printProcess("INFO: Loading images...")

    #making a list of the images paths arranged randomly
    images_paths = sorted(list(paths.list_images(self.__dataset_path)))
    random.seed(42)
    random.shuffle(images_paths)

    #loop over the data set images in order to prepare the data for running on the model
    for image_path in images_paths:
        #this loop:
        #1. loads the image as gray scale
        #2. converts it to numpy array
        #3. stores it in the data list

        image = cv2.imread(image_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (self.__IMAGE_DIMS[1], self.__IMAGE_DIMS[0]))
        image = img_to_array(image)
        self.__data.append(image)

    #extracting the class label from the image path and update the labels list
    label = image_path.split(os.path.sep)[-2]
    self.__labels.append(label)

    self.__data = np.array(self.__data, dtype = "float") / 255.0
    self.__labels = np.array(self.__labels)
    printmessage.printProcess("INFO data matrix: {:.2f}MB".format(self.__data.nbytes / (1024 *
1000.0)))

```



```

def __split_data(self):
    """
    a private method that splits the data for 3 categories: train set (70%), test set (20%), and
    validation set (10%)

    return: the train, validation and test data after the splitting.
    """

    #splitting the data to: train set (80%) and test set (20%)
    (trainX, testX, trainY, testY) = train_test_split(self.__data, self.__labels, test_size=0.2,
random_state=42)

    #splitting the train data to: train set (87.5% of 80% = 70%) and validation set (12.5% of 80% = 10%)

```

```
(trainX, valX, trainY, valY) = train_test_split(trainX, trainY, test_size=0.125, random_state=42)

return(trainX, testX, valX, trainY, testY, valY)

def __evaluate_Model(self, testX, testY, model, batch_size = 32):
    """
    a method that evaluates the model on the test data using `evaluate` - Performs the testing

    param testX: a list of the test images
    param testY: a list of the test images' labels
    param model: the trained model
    param batch_size: the batch size- The number of images that the model will work in parallel
    while learning.
    """

    printmessage.printProcess('\n# Evaluate on test data')
    results = model.evaluate(testX, testY, batch_size=32)
    print('test loss ' + str(results[0]) + ', test acc ' + str(results[1]))


def __train(self, trainX, testX, valX, trainY, testY, valY, lb):
    """
    a private method that performs the actual model training:
    This method saves the list of labels in a binary file that will be used in the model training as well
    as
    in predicting categories of images later.
    This method also creates a transformation from the binary file that will be used to model the
    model.
    This method divides the data using a method from another module.
    In addition, this method saves the weights/ model file and the binary label file so that the
    learning can be used in other runs of the program and so we would not have to run the model
    again for each prediction.

    param trainX: a list of the train images
    param testX: a list of the test images
    param valX: a list of the validation images
    param trainY: a list of the train images' labels
    param testY: a list of the test images' labels
    param valY: a list of the validation images' labels
    param lb: Label Binarizer

    return: the history of model learning.
    """

#constructing the image generator for data augmentation which will occur while running
data_augmentation = ImageDataGenerator(rotation_range=25,
                                         width_shift_range=0.1,
                                         height_shift_range=0.1,
                                         shear_range=0.2,
                                         zoom_range=0.2,
                                         horizontal_flip=True,
                                         fill_mode="nearest")
```

```
#initializing the model
printmessage.printProcess("INFO: Compiling model...")

model = themodel.MyModel.buildModel(width=self.__IMAGE_DIMS[1],
height=self.__IMAGE_DIMS[0],
depth= self.__IMAGE_DIMS[2], classes=len(lb.classes_))

#setting the Adam Optimization Algorithm which will be used for optimizing the model
optimization_algorithm = Adam(lr=self.__LEARNING_RATE, decay=self.__LEARNING_RATE /
self.__EPOCHS)

#configing the model before training: compile the model and define the loss function, the
optimizer and the metrics
model.compile(loss="categorical_crossentropy", optimizer = optimization_algorithm,
metrics=["accuracy"])

#printing a model summary table
model.summary()

#training the network
printmessage.printProcess("INFO: Training neural network...")

history = model.fit(
data_augmentation.flow(trainX, trainY, batch_size=self.__BATCH_SIZE),
validation_data=(valX, valY),
steps_per_epoch = len(trainX) // self.__BATCH_SIZE,
epochs=self.__EPOCHS, verbose=1)

#saving the model
printmessage.printProcess("INFO: Serializing neural network...")
model.save(self.__model_path)

#saving the label binarizer
printmessage.printProcess("INFO: Serializing label binarizer...")
file = open(self.__labels_path, "wb")
file.write(pickle.dumps(lb))
file.close()

return history
```

כפתרו רביעי**הקובץ האחראי על ביצוע חיזוי לתמונה- [classifyimage.py](#).**

קובץ זה אחראי על ביצוע תפקידי של הכפטור הרביעי. בהינתן תמונה שהנתיב אליה מוגדר בקובץ `uk.config`, הקובץ חוצה לאיזו קטgorיה היא שייכת (כלומר, מהי הספרה אותה היא מתארת). בקובץ זה ישנה מחלוקת בשם `ImagePrediction` הדורשת לבנייה שלה שני פרמטרים: נתיב למודל השמור, ונתיב לקובץ הבינארי המכיל את תוויות התמונות.

במחלקה זו מתודת `__init__` מקבלת את הנעדרת במתודות פרטיות שבאמצעותן היא מימושת את מטרת הקובץ- ביצוע החיזוי. מתודה זו מקבלת את הנתיב לתמונה אותה רוצים לחזות.

לפני דימון מתודה זו מתוך המודול (`predict_image()` בקובץ `predict_image.py`) שעה יצרתי אובייקט מסווג המחלוקת והעברתי לבנייה המחלוקת את הפרמטרים הדרושים. אחת מתודות המחלוקת טעונה את המודל השמור במחשב. אחרת מרים על גבי את התמונה שהתקבלה כקלט מן המשתמש. מתודה נוספת מוצאת את הקטגוריה אשר לה הסיכון הגבוה ביותר להיות הנכונה, ובעזרת הקובץ הבינארי שנשמר בעת אימון המודל מציגה למשתמש כפלט האם חיזוי המודל נכון ידי השווה בין התו הראשון בתמונה, שהוא למעשה למשה הקטגוריה אליה היא שייכת (הספרה המתאימה) לבין תוצאה החיזוי של המודל.

הסבר בניין המחלוקת:

תפקיד	ערך	תכונה
מכילה את הנתיב בו שומר המודל.	input	<code>__model_path</code>
מכילה את הנתיב בו שומר הקובץ הבינארי המכיל את תוויות התמונות.	input	<code>__labels_path</code>
מכילה את נתיב התמונה אותה רוצים לסוג.	""	<code>__image_path</code>
תכונה שאליה יטען המודל השמור.	None- a default value	<code>__model</code>
תכונה שאליה יקרא הקובץ הבינארי השמור.	None- a default value	<code>__label_binarizer</code>

הסבר המתודות במחלוקת:

תפקיד	מתודה
המתודה הראשית והפומבית היחידה בחלוקת זו אשר אחראית לביצוע החיזוי. ומזמנת את שאר המתודות לשם כך. המתודה מקבלת כפרמטר את הנתיב לתמונה אותה רוצים לסוג לקטגוריה הנכונה (הספרה שהיא מתארת). המתודה מזמנת את המתודה <code>__load_Model_And_Label_Binarizer()</code> על מנת לטען את המודל השמור והקובץ הבינארי של התוויות. לאחר מכן היא	<code>handle_classify(image_path)</code>

<p>מאתחלת את התוכנה <code>image_path</code>. After</p>	
<p>כך היא מזמנת את המתודה <code>prepare_Image</code> שמבצעת את התאמת התמונה המיועדת לחיזוי ליריצה על המודל השמור לתוך התוכנה <code>model</code> וקוראת את הקובץ הבינארי לתוך התוכנה <code>label_binarizer</code>.</p>	<code>load_Model_And_label_binarizer()</code>
<p>מתודה פרטית אשר מבצעת את ההתאמה של התמונה השמורה בנתיב <code>image_path</code> ליריצה על המודל, בדומה לתהילך שבוצע על מאגר התמונות המלא לפני תחילת האימון שלהן. המתודה למעשה משנה את ממד התמונה לגודל אחיד (96,54,1), ממירה את התמונה <code>RGB</code> ל<code>GrayScale</code>, ממירה את התמונה למערך ומnormמל אותה (מקטינה את טווח הפיקסלים שלה מ [0,255] ל [0,1]). המתודה מחזירה את המערך המיציג את התמונה וכן העתק של התמונה כפי שהתקבלה כקלט.</p>	<code>prepare_Image()</code>
<p>מתודה פרטית האחראית על ביצוע החיזוי עצמו. היא מקבלת כקלט את הפלט של המתודה <code>prepare_Image</code> (המיציג את התמונה והעתק של התמונה כפי שהתקבלה כקלט). המתודה מבצעת <code>predict</code> לתמונה על גבי המודל הטוען. בנוסף, היא מדפיסה את העתק התמונה שקיבלה כקלט ועל גביו את תוצאת החיזוי שלה, את הסיכוי שהמודל זיהה נכון את הקטגוריה אליה שייכת התמונה על פי חישובי, והאם הזיהוי היה נכון. זאת על ידי השוואת התו הראשון של שם התמונה, שהוא הספרה שהיא מתארת, לתווית שהמודול זיהה, המציינת את הקטגוריה- הספרה- אליה שייכת התמונה.</p>	<code>predict_Image(image_array, image_copy)</code>

תדPIO הקוד:

```
"""
this python file is responsible for predicting an image
"""

from keras.preprocessing.image import img_to_array
from keras.models import load_model
import numpy as np
import imutils
import pickle
import os
import cv2

import printmessage


class ImagePrediction():

    def __init__(self, model_path, labels_path):
        """
        Constructs a new 'ImagePrediction' object

        param model_path: the directory of the trained model
        param labels_path: the directory of the labels file
        """

        self.__model_path = model_path
        self.__labels_path = labels_path
        self.__image_path"" =
        self.__model = None
        self.__label_binarizer = None
        self.__IMAGE_SIZE(50,50) =


    def handle_classify(self, image_path):
        """
        a public method that maneges the classifition. It is responsible for
        summoning the rest of the methods in the correct order for the final image_copy.

        param image_path: the directory for the image that the user chooses to return
        The method initializes its attribute, image_path in this parameter
        """

        self.__load_Model_And_Label_Binarizer()
        self.__image_path = image_path

        image_array, image_copy = self.__prepare_Image()

        self.__predict_Image(image_array, image_copy)

    def __load_Model_And_Label_Binarizer(self):
```

```
"""
    a private method that loads the saved model i.e. the trained convolutional neural network
    and the prediction_label binarizer.
    The method initializes the class attributes
"""

printmessage.printProcess("INFO: Loading network...")

self.__model = load_model(self.__model_path)
self.__label_binarizer = pickle.loads(open(self.__labels_path, "rb").read())


def __prepare_Image(self):
"""
    a private method that adjusts the saved image in the image's paths to run on the model, just like
    before the training.
    Resizes the image to a uniform size I set (____), converts the image from RGB to GrayScale, then
    converts the image to an array,
    reduces its pixel range from [0,255] to [0,1]. Also returns a copy of the image as received as input.

    return:
.1   the image numpy array (after fitting the image colors, dims, pixels scale... )
.2   copy of the image as gray scale
"""

printmessage.printProcess("INFO: Loading image...")

#   loading the image
image = cv2.imread(self.__image_path)
image_copy = image.copy()

#   pre-process the image for classification
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
image = cv2.resize(image, self.__IMAGE_SIZE)
image = image.astype("float") / 255.0##
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
return image, image_copy


def __predict_Image(self, image_arr, image_copy):
"""
    a private method that is responsible for performing the prediction itself. The method performs
    predict for the image on the loaded model and prints:
.1   Its prediction
.2   The probabilitiesability that a model correctly identified the image according to its calculations
.3   Is the identification correct or not, by comparing the prediction_label of the image that is in
the file name with
    the prediction_label identified by the model (the prediction_label indicates the category to which
the image belongs).
    In addition, the method creates and displays an image of the predictive image on which 1, 2, 3
appear.

    param image_arr: the image for prediction as array
    param image_copy: a copy of the image

```

```
"""

# classify the input image
printmessage.printProcess("INFO: Classifying image...")

probabilities = self.__model.predict(image_arr)[0]
index_most_likely = np.argmax(probabilities)
prediction_label = self.__label_binarizer.classes_[index_most_likely]

# print "correct" if the input image prediction_label is fit to the prediction prediction_label
# else print "incorrect"
filename = os.path.basename(self.__image_path)
is_correct = filename.startswith(str(prediction_label))
is_correct_txt = "correct" if is_correct else "incorrect"

# bulid the prediction_label and draw the prediction_label on the image
prediction_label_for_user = "{}: {:.2f}% ({})".format(prediction_label,
probabilities[index_most_likely] * 100, is_correct_txt)

image_copy = imutils.resize(image_copy, width=400)
text_color = (0, 255, 0) if is_correct else(0,0,255)
cv2.putText(image_copy, prediction_label_for_user, (10, 25), cv2.FONT_HERSHEY_SIMPLEX, ,0.7
text_color, 2)

# show the image_copy image
printmessage.printProcess("INFO: {}".format(prediction_label_for_user))
cv2.imshow("image_copy", image_copy)
cv2.waitKey (0)
```

קבצים נוספים:**הקובץ האחראי על בדיקת תקינות- checkdirectorys.py**

קובץ זה אחראי על בדיקת תקינות הנתיבים שהוגדרו בקובץ `uk.config`, ועל הדפסת הודעות שגיאה למשתמש במידה הצורך. הודעות אלו מפרטות באיזה נתיב קיימת הבעה ומה' למשזה הבעה הספציפית שמנעה מהתוכנית לróż כהלה. הוא מכיל מתודה אחת האחראית על בדיקת נתיבים קיימים, מתודה נוספת נספפת האחראית על בדיקת נתיבים חדשים, ועוד שתי מתודות עזר. במידה שהודפסה הודעה שגיאה, על המשתמש לשנות בהתאם אליה את הגדרת הנתיב שגררה את השגיאה בקובץ `uk.config`.

תפקיד	מתודה
<p>מתודה זו בודקת אם הנתיב <code>path</code> שקיבלה הוא נתיב חוקי ליצירת תיקייה חדשה, שאיןו קיימים עדין.</p> <p>ראשית, היא מזמנת את המתודה <code>is_new_and_valid(path, messageHeb, messageExists, hebNotValid = False)</code></p> <p>אם מדובר בנתיב חדש וחוקי מבחינתי או הימצאות אותיות עבריות בנתיב במידה שאותיות עבריות בנתיב ייחסבו כשגיאה. אם מתודה זו מחזירה שקר, יוחזר שקר.</p> <p>אם מתודה זו מחזירהאמת, המתודה מנסה לפתח תיקייה בנתיב זה. אם היא מצליחה, היא מסירה אותה ומחזירהאמת.</p> <p>אם לא, היא מחזירה שקר.</p> <p><code>hebNotValid</code>- משתנה בוליאני שערכו אמת אם אותיות עבריות בנתיב ייחסבו כשגיאה, וערכו שקר אחרת ואפ' באופן דיפוליטיבי.</p> <p>המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהוא בודקת.</p>	<code>def is_new_and_valid(messageHeb, messageExists, path, hebNotValid = False)</code>
<p>המתודה מחזירה אמת אם שני תנאים (ישנם עוד) לנטיב חוקי מתקיים. ראשית, אם ערכו של <code>hebNotValid</code> הוא אמת, כלומר אסור שהוא אותיות עבריות בנתיב, היא בודקת זאת, ואם ישן היא מחזירה שקר. במידה ואין, או <code>hebNotValid</code> הוא שקר, המתודה בודקת האם הנתיב כבר קיים או חדש. אם הוא קיים, היא מחזירה שקר. אחרת, מחזירה אמת.</p> <p>המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהוא בודקת.</p>	<code>__is_new_and_valid(path, messageHeb, messageExists, hebNotValid = False)</code>

<p>המתודה מחזירה אמת במידה והנתיב path כבר קיים וחוקי, וSKU אחרה. אם hebNotValid הוא SKU המתודה רק בודקת אם הנתיב קיים. אם id hebNotValid הוא אמת היא בודקת בנוסף אם קיימות בו אותיות עבריות.</p> <p>המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהוא בודק.</p>	<code>is_exists_and_valid(messageHeb, messageNotExists, path, hebNotValid = False)</code>
<p>המתודה מחזירה אמת אם בנתיב path קיימות אותיות עבריות וSKU אחרה. המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהוא בודק.</p>	<code>check_language(messageHeb, path)</code>

המתודות הפומביות בקובץ, () | is_exists_and_valid() | is_new_and_valid() | is_is_mozmanot בקובץ yk_menu וכן בקובץ extractzipfiles.py על מנת לבדוק את תקינותם של הנתיבים המוגדרים בקובץ yk.config.py.

תדפיו הקיימים:

```
"""
this python file is responsible for checking the paths
"""

import os
import string
import printmessage

def is_new_and_valid(messageHeb, messageExists, path, hebNotValid = False):
    """
    Checks if the path received is a valid path to create a new folder, that does not exist yet .
    Prints appropriate messages.

    param messageHeb: a message to print in case path contains Hebrew letters
    param messageExists: a message to print if the path already exists
    param path: a directory
    param hebNotValid: True if Hebrew letters in the path are considered an error, False else & by
default

    return: True if the path is new and valid, false else

    """

    if not __is_new_and_valid(path, messageHeb, messageExists, hebNotValid):
        return False

    try:
        # tries to make a folder in the current path .
        # If it succeeded: the path is valid, return true.
        # if it failed: the path is not valid, return false.
        os.mkdir(path) #create
        os.rmdir(path) #remove
        return True

    except :
        if(path != ""):
            return False

def __is_new_and_valid(path, messageHeb, messageExists, hebNotValid = False):
    """
    preforms the actual checking of the path- is it new and valid.
    Prints appropriate messages.

    param path: a directory
    param messageHeb: a message to print in case path contains Hebrew letters
    param messageExists: a message to print if the path already exists
    param hebNotValid: True if Hebrew letters in the path are considered an error, False else & by
default

```

return: True if the path is new and valid, False else.

```
"""
if(hebNotValid):
    if not check_language(messageHeb, path):
        printmessage.printError(messageHeb)
        return False

    if(os.path.exists(path)):
#      check if the path is already exists. If so, returns false.
        printmessage.printError(messageExists)
        return False

    return True

def is_exists_and_valid(messageHeb, messageNotExists, path, hebNotValid = False):
"""
    A static and private method that checks whether the path it is receiving exists and whether the flag
it is receiving is True .
    If so, it also checks the characters that make up the address.
    The method will return True if the address is valid. Otherwise, will return False.

    param messageHeb: a message to print in case path contains Hebrew letters
    param messageNotExists: a message to print if the path does not exist
    param path: a directory
    param hebNotValid: True if Hebrew letters in the path are considered an error, False else & by
default

"""
if(hebNotValid):
    if not check_language(messageHeb, path):
        return

    if not os.path.exists(path):
        printmessage.printError(messageNotExists)

def check_language(messageHeb, path):
"""
    A static and private method that checks if there are hebrew letters in a path

    param messageHeb: a message to print in case path contains Hebrew letters
    param path: a directory

    return: True if path does not contain Hebrew letters, False else
"""

    for character in path:
        if not (character in string.printable) :
            printmessage.printError(messageHeb)
            return False

    return True
```

הקובץ האחראי על הדפסות- printmessage.py

קובץ זה אחראי על ביצוע הדפסות למשתמש, המסוגות לשני סוגים:

1. הדפסות שונות אינפורמציה על המתרחש בזמן ריצה- יודפסו בכתב **כחול**
2. הדפסות של הודעות שגיאה- יודפסו בכתב **אדום**

כל מתודה מקבלת כקלט הודעה להציג למשתמש וקובעת צבע הדפסה שוננה:

תפקיד	מתודה
מקבלת הודעה ומדפיסה אותה בכתב אדום.	printError(message)
מקבלת הודעה ומדפיסה אותה בכתב כחול.	printProcess(message)

2 המתודות הללו מזומנות על ידי מתודות שונות בקבצים ,train_test_model.py ,extractzipfiles.py ,checkdirectorys.py ,makegraphs.py ,classifyimage.py וmenu_tkinter.py ומקבלות הודעות שונות להדפסה המספקות אינפורמציה על התהליכים השונים או על שגיאות.

תדפסו הלו:

```
"""
This python file is responsible for printing messages for the user in colors
"""

from colorama import init, Fore, Style

def printError(message):
    """
    prints a message in red

    param message: a meesage for printing
    """
    init(convert = True)
    print(Fore.RED + message)
    Style.RESET_ALL

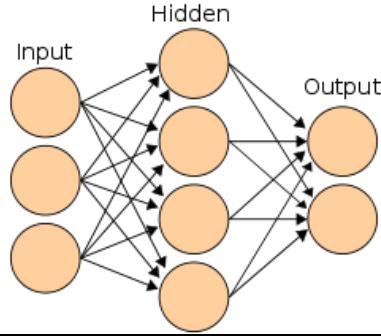
def printProcess(message):
    """
    prints a message in blue

    param message: a meesage for printing
    """
    init(convert = True)
    print(Fore.BLUE + message)
    Style.RESET_ALL
```

מסקנות הרצת המודל

הסבר מושגים:

לפני שאתחל לפרט על מסקנות הרצת המודל, אסביר מספר מושגים בסיסיים:

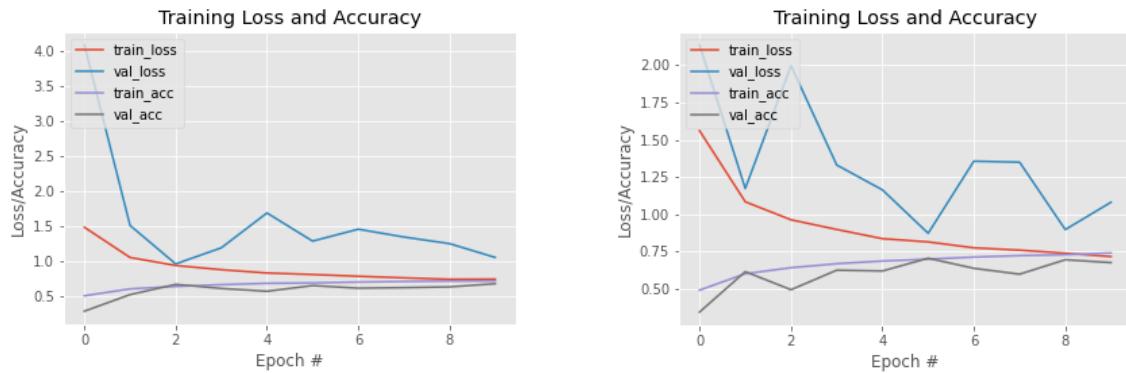
הסבר	מושג
<p>רשת נוירונים מלאכותית שמחקה את החשיבה האנושית. נוירון במוח הוא תא עצב פשוט, חלק מרשת עצבית עצמה. ANN היא רשת עצבית מלאכותית, המוח. ANN שמעתיקה את הפעולות של שכבות תאי העצב בנויאו-קורטיקס, האזורי שתופס את רוב המוח האנושי.</p>	ANN (Artificial Neural Network) 
<p>סוג של רשת נוירונים מלאכותית אשר משמשת לזיהוי ועיבוד תמונה, ומתוכננת במיוחד לעיבוד נתונים פיקסלים. לרשותה אלה מספר גדול של שכבות חביות. השכבות של CNN מורכבות משכבה קולט, שכבה פלט ושכבה נסתרת הכוללת מספר רב של שכבות פיטול, שכבות מאגדים, שכבות מחוברות לחלווטן ושכבות נורמליזציה.</p>	CNN (convolutional neural network)
<p>מערכת הנתונים עליו פועל מודל למדידת המכונה מחולק ל-3 קטגוריות:</p> 	
<p>מדגם הנתונים המשמש להתאמת המודל. מערכת הנתונים שבו אנו משתמשים לאימון המודל (weights - biases). המודל לומד מנתונים אלה. רשת נוירונים).</p>	Train set
<p>מדגם הנתונים המשמש למתן הערכה של התאמת המודל train set תוך כוונון של היפרפרמטרים של המודל.</p>	Validation set

מדגם הנתונים המשמש למבחן הערכה של התאמת המודל הסופי <i>train set</i> . מערך זה מספק את "תקן החבא" המשמש להערכת המודל. משתמשים בו רק ברגע שהמודל מאומן לחloatין.	Test set
מדדיהם להצלחת מודלים של סיווג:	
מתאר את אחוז ההצלחה של המודל בחיזוי הקטגוריות של התמונות. פועל באופן ביןארי - חיזוי נכון או לא נכון. ככל שה <i>accuracy</i> גבוהה יותר, זה אומר שהמודל פועל טוב יותר.	accuracy
מתאר כמה קרובות היו תוצאות חיזוי התמונות, לקטgorיה האמיתית שאליה הן שייכות. לא פועל בערך ביןארי. ככל שה <i>loss</i> נמוך יותר, זה אומר שהמודל פועל טוב יותר.	loss
היפרפרמטרים (משתנים השולטים בתהליכי האימון כלו), אשר לא ניתן לאמוד מຕוך מערך הנתונים):	
מספר הפעמים שריצזו התמונות על גבי המודל לצורך ביצוע הלמידה.	epochs
מספר התמונות שהמודל יעבד במקביל בזמן הלמידה.	batch size
קצב הלמידה- ערך שקבע כמה לשנות את המודל בתגובה לשגיאה המשוערת בכל פעם שימושי המודל מתעדכנים.	learning rate
בעיות שעלוות להתרחשות:	
התאמת יתר- מצב בו המודל מותאם יתר על המידה לאוסף נתונים ספציפי (ה <i>train set</i> במקורה זה) ועל כן מצליח פחות בביצוע תחזיות. התאמת יתר מתרחשת כאשר המודל נקבע על ידי יותר פרמטרים מאשר הנתונים מצדיקים.	Overfitting
תת התאמת- מצב בו המודל פשוט מדי מכדי לייצג כראוי את המבנה הבסיסי של הנתונים, למשל בעקבות מיעוט פרמטרים המגדירים את המודל.	Underfitting

לאחר שעברית על המושגים הבסיסיים, אפרט בעמודים הבאים את תוצאות הרצת המודל הנוכחי אך לפני כן אסביר מעט על התהליכי שעברית עם הרצות קודמות:

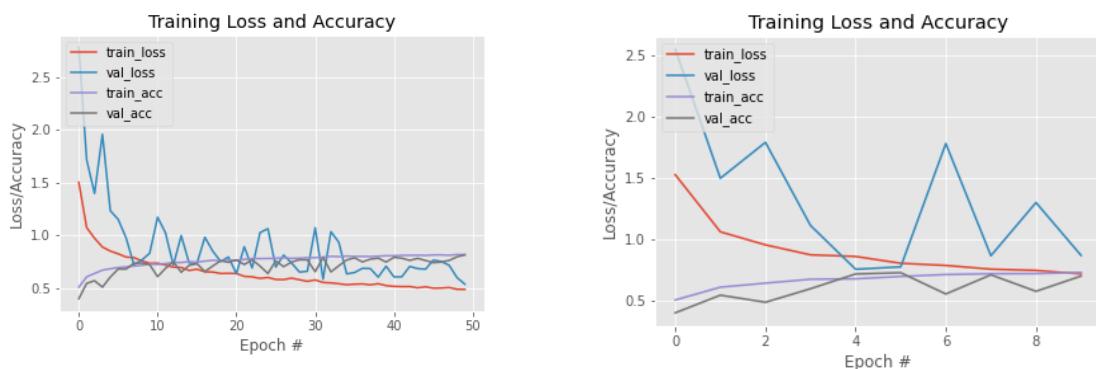
מסקנות מהרצות קודמות של מודלים:

בתחילת דרכי כשניתי להבין איך עובד מודל CNN, והתנסיתי בהרצת המודל הנוכחי, ומודלים אחרים על מאגרי תמונות שונים, קיבלתי ערכים לא מספקים של accuracy וloss, והגרפים שנוצרו נראה כך, למשל:



לאט לאט הבנתי מה הבועית ושיפרתי את ביצועי המודל, והגרפים המתארים את ערכי accuracy וloss נראו טוב יותר. תוך כדי תהליכי העבודה על הפרויקט למדתי להכיר טוב יותר את העבודה עם המודלים. למשל, למדתי שכל שימושים שכבות למודל כך ניתן להימנע ממצבים של תות התאמת. כמו כן, רأיתי איך הגדלת מספר epoch (מספר הפעמים שתתבצע הלמידה מtbody מוחדר) מסייעת מאוד לשיפור ערכי accuracy והloss. לבסוף, הגיעו למסקנה (שאני רוצה עוד לבחון) שמודלים מסוימים מספקים ביצועים טובים יותר מאשר בתמונות שרצו על גביהם ישבו אובייקט הבולט על רקע שונה ממנו.

הגעתי לכל המסקנות הללו מתוך ההתנסות האישית שלי לאור תהליכי העבודה. מימי נמצאים גרפים שמתראים ערכים לא מספיק טובים של accuracy וloss כאשר מספר epoch הגדיל יותר מאשר מספר epoch הנוכחי בפרויקט שלי. משמאלו נמצאים גרפים שמראים כיצד המודל מתתקשה כאשר רצوت על גביו תמונות שלא מתאימות אליו בתצורתן הנוכחית.

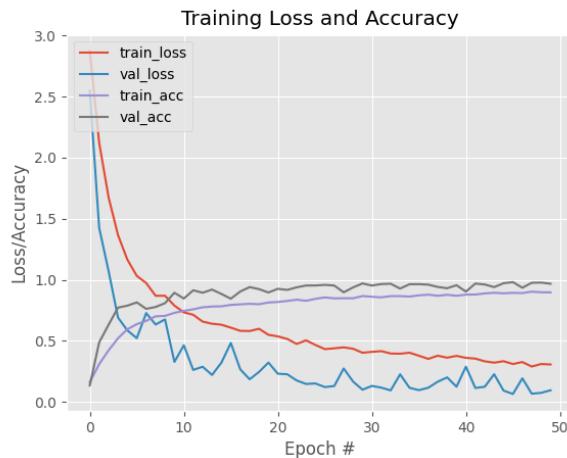


מסקנות מהרצת המודל הנוכחיית:

כעת, לאחר שיישמתי את הלקחים שלמדתי מתחילה העבודה עלuproject, הגעתו ל吒וצאות טובות. אציג את הגרפים שלי וסביר על כל אחד מהם:

מערכת 1:

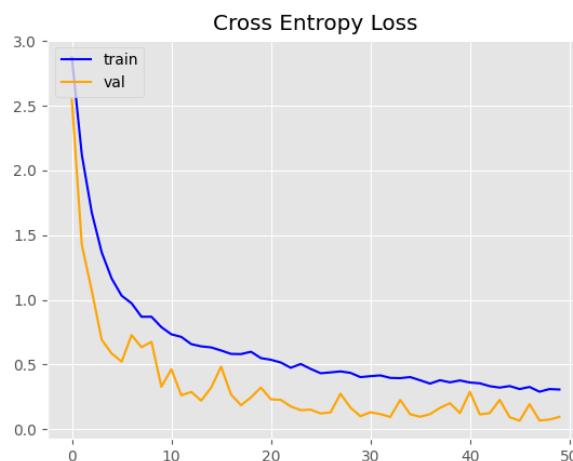
מיכלה 4 גראפים המציגים את השתנות ערכי accuracy והloss עבור החומר והvalidation לאורך תהליכי האימון.



מערכות 2 ו-3, כמו מערכת 1, מציגות את ערכי accuracy והloss, רק בנפרד.
אפרט על הפונקציה שמתאר כל גרפ:

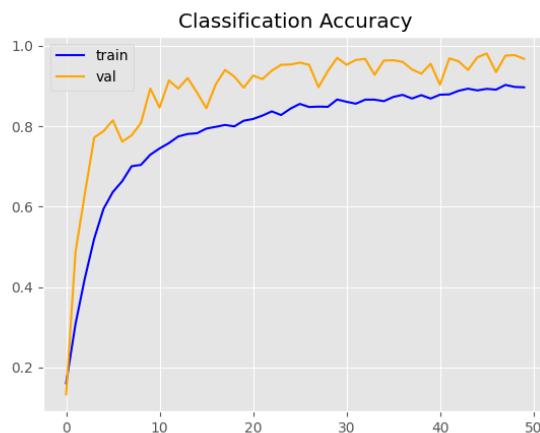
מערכת 2:

מיכלה 2 גראפים המתארים את פונקציית loss cross entropy עבור החומר והvalidation לאורך תהליכי האימון. פונקציה זו קטנה ככל שההסתברות החזויה שסיפק המודל זהה לתווית האמיתית של התמונה.



מערכת 3:

מיכלה 2 גրפים המתארים את ממד **classification accuracy** עבור החומר **train** והחומר **validation** לאחר תהליכי האימון. ממד זה מסכם את הביצועים של המודל כמספר התחזיות הנכונות חלקי המספר הכללי של התחזיות. הגרף עולה ככל שהחלוקת של התחזיות הנכונות מתווך סר התחזיות גדול.



ניתן לראות בכל המערכות את גрафי lossו יורדים עד לערכים קטנים, ואת גрафי accuracy העולים עד לערכים גדולים. הדבר מצביע על כך ששאכן מתחרש למידה טובה של התמונהות והמודול מצליח לספק ביצועים טובים. פונקציית lossו קטנה ככל שההסתברות החזיה שסיפוק המודול זהה לתוויות האמיתיות של התמונה. לעומת, כפי שניתן לראות בגרפים, כל שמתקדמים באימון המודול טוב יותר ומספק יותר פעמים חיזי נכון. גраф accuracy כאמור עולה ככל שהחלוקת של התחזיות הנכונות מתווך סר התחזיות גדול. לעומת, ככל שמתקדמים באימון למודל יש יותר ויותר תחזיות נכונות.

בסוף, אציג את הערכים הסופיים של loss accuracy של train set, validation set ו test set ואסכם את מסקנותי בעמוד הבא:

ערכים סופיים של המודים להצלחתו של המודל ומסקנות:

אלו הם הערכים סופיים של ה-
 :train loss, validation loss, test loss

test	validation	train	
0.9694 (96.94%)	0.970535 (97.05%)	0.8836 (88.36%)	accuracy
0.0867 (8.67%)	0.0959 (9.59%)	0.3343 (33.43%)	loss

ניתן להסיק שהמודל פועל טוב כאשר תוצאות הספורות רצות על גביו, ותהליכי *training* וה*test* כתובים בצורה טובה. עוד אני מסיקה שחלוקת התמונות ל-10%, 20%, 70% בין *train*, *validation*, *test*, *loss* היא חלוקה הסטנדרטית, היא חילוקה טובה. כמו כן ערכי *learning rate* ו-*epoches* מוכחים עצםם כערבים מתאים, וכך גם אלגוריהם האופטימיזציה *Adam* של המודל ניכר כמתאים. נוסף על כך, הגדלת הנתונים תוך כדי ריצה מסייעת גם היא להשגת תוצאות אלה, שהרי ה*dataset* בו אני עובשה שימוש אינו גדול כל כך.

סיכום אישי / רפלקציה

כתיבת הפרויקט הייתה תהליך מעוניין ומעשר, אך יחד עם זאת גם לא קל ומופטל. נתקלתי בקשישים ואתגרים מסווגים שונים במהלך העבודה על הפרויקט.

אמנה כמה מהם. ראשית, מציאת נושא מתאים לקחה לי הרבה זמן ושיניתי את נושא הפרויקט שלי מספר פעמים, גם לאחר שהחלתי לעבוד. כמו כן, בעיות שונות שהיו לי הן הגיעו לערבים לא מספיק טובים של accuracy loss, בעיות עם הורדת ספריות ומודולים שונים, בעיות בהרצה בcmd, אתגרים בلمידה עצמית של מרבית החומר עוד.

יחד עם זאת, ידעת מראש שזה חלק מהתהליך ולאורך כל העבודה השקעתית אתamate' במציאת פתרונות והבנת דברים מסוימים, בין היתר בעזרת מקורות מידע שונים כמו אתרים רבים באינטרנט וחברים, וכן בעזרת חשיבה עצמאית וניסוי וטעייה.

בסוף דבר, אני מרגישה שהפקתי לךים וגהנתי למסקנות רבות שאקה איתי הלאה בנוגע לכך שהיא יכולה להננה בתהליך של מטלה מורכבת.

למשל, למדתי על החשיבות העבודה מוסדרת וחולקה לוגית של חלקים הפרויקט. בנוסף, הידע שלי בשפת Python בה כתבתי את הפרויקט ובנושא deep learningdeep כובן, גדל בצורה משמעותית. מתוך ההתמודדות עם שגיאות רבות והניסיון להבין דברים מורכבים למדתי דרכים לבדוק את מקור הבעיה ולמצוא אותה וכן להבין את משמעותן של שורות קוד מסוימות מהאינטרנט בהן נעדרתי. למדתי כיצד להננה בצורה טובה מבחינות חישוב מיידע באינטרנט ומציאת פתרונות להבעיות שהתעוררו אצלי באטרים שונים. לו הייתה חוזרת בזמן, הייתי מתחילה את הפרויקט מוקדם יותר, וכך נוננת לעצמי יותר מרחב וזמן להתנסויות וטעויות עם פחות לחץ. העבודה הייתה עיליה יותר אם הייתה יודעת את כל מה שאני יודע עכשו, אבל אני שמחה על כל טעויות שעשית, משומם שהיא קירבה אותי להיות כותבת קוד טובה יותר. ככל שהתקדמה בתהליך, הרגשתי שאני הופכת חזקה יותר בתחום וכן הביצועים שלי השתפרו. למדתי למשול שבניסיון למצאו טעות במקום מסוים לא כדאי לבצע את כל הזמן על בדיקה חוזרת ונשנית של החלק הקטן שבו הטעות נמצאת במחשבה הראשונה, אלא להסתכל על התמונה הרחבה יותר, למשל על אופן היזמן של הפעולה שמבצעת את אותו חלק בקוד, הפרמטרים המועברים או המוחזרים וכדומה.

על אף שפעמים רבות במהלך כתיבת הפרויקט חשתי מתוסכלת, מצאתי את התהליך כולל מהנה, מעשר ומלמד. הידעdeep learning הוא נושא מרתק כשלעצמו. לא קשור לנושא הספציפי הזה, אני אוהבת את עולם התוכנות ואהבת לכתוב קוד. מעבר לכך, אני אוהבת ללמוד ולהציג את המטרות שאני מציבה לעצמי. לכן, על אף שהתהליך לא היה קל,בסוף דבר הוא היה חיובי, והקשי בדיעבד הוא גם דבר חיובי.

במהלך הפרויקט יعلתי את עצמי. למשל, ככל שעבר הזמן סייגתי לעצמי יכולת טובה יותר של עבודה על כמה דברים במקביל. הפקתי חכמה יותר ובשלת היכרות טובה יותר עם הנושא. דוגמה קטנה היא פעמים רבות בהן הרצתי את אימון המודל לאחר ששיפרתי בקוד דבר מסוים ורציתי לבחון איך הדבר משפיע על ביצועיו, בזמן כתיבת חלק נוסף בפרויקט בספר הפרויקט, על מנת לא לבזבז זמן. ככל שהתקדמה בתהליך לממדתי להציג לעצמי מטרות יומיות, להעלות על הכתב את הבעיה שלי ואת התכוננים והיעדים שלי כדי לעבוד טוב יותר ולעמוד בלוחות הזמן.

שאלת שהתעוררה אצלי במהלך הפרויקט, ושארצה לתת עליה את הדעת בהמשך, היא עד כמה המודל הנוכחי מתאים ל>Zihui ספרות שאין מtower מאגר המידע בו השימושתי, וכן איך אפשר לשפרו כדי שייתאים לספרות שנכתבו בכתב יד למשל, ולא בצורה סינטטית. כדי

להתחל לערנות על השאלה החזאת יצרתי בעצמי בעצרת תוכנת הצייר ספרות על רקעים שונים שציירתי, והוספתי את התמונות למאגר התמונות המיועדות לחיזוי. לאחר ניסוי וטעייה הבנתי שככל שהספרה גדולה יותר ומגיעה לקצוות התמונה, כמו הספרות בתמונות המקוריות, כך תוצאות החיזוי טובות יותר. בהמשך, אוצר ספרות באופן ידני על גבי דף נייר ולא על מחשב, אבחן את התמודדות המודל עם תמונות כאלו ואשפרו במידת הצורך.

אם אסכם את הדברים העיקריים שלמדתי במהלך העבודה, אדג'ש את הלמידה לעומק של נושא Deep Learning, פתרון בעיית הסיווג בעזרת CNN, שימוש בספריות שונות כמו Tkinter ורחבת הידע בשפת Python. כמו כן למדתי מה כתיבה של ספר הפרויקט על האופן בו מסבירים פרויקט לנמענים שונים ועל סוגים טרשיים שונים כמו UML Use Case.

לסיום, אני שמחה על ההזדמנות למתוח מעט את הгалומות של הדברים שחשבתי שאני מסוגלת להם. אני אסירת תודה לכל מיי אנשי אקרים ברחבי האינטרנט שmagdilim את הידע שאפשר לצבור ממש לאין שיעור, ואני מצפה בקוצר רוח לישם את התובנות והלקחים שלי מהפרויקט הזה בפרויקט הבא שלי.

ביבליוגרפיה

Amos, D. Python GUI Programming With Tkinter. Real Python.

<https://realpython.com/python-gui-tkinter/>

Chollet, F (2016, June 5). Building powerful image classification models using very little data. The Keras Blog.

<https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>

Image data preprocessing. K Keras.

<https://keras.io/api/preprocessing/image/>

Narkar, M (2019, August 6). Image classification with Convolution Neural Networks (CNN)with Keras. Medium

<https://medium.com/@manasnarkar/image-classification-with-convolution-neural-networks-cnn-with-keras-dbd71c05ed2a>

Rosebrock, A (2017, December). Image classification with Keras and deep learning. Pyimagesearch.

<https://www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/>

Shorten, C (2018, October 15). Image Classification Keras Tutorial: Kaggle Dog Breed Challenge. Towards Data Science

<https://towardsdatascience.com/image-classification-python-keras-tutorial-kaggle-challenge-45a6332a58b8>

Stack Overflow.

<https://stackoverflow.com/>

Tarang, S (2017, December 6). About Train, Validation and Test Sets in Machine Learning. Towards data science.

<https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>

נספחים**תרשים שכבות המודל:**