

**פרויקט גמר 5 יחידות לימוד בהתמחות
תכנון ותכנות 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.....	תכולת הפרויקט:
6.....	תרשים זרימה של כל קבצי הפיתוח בפרויקט:
7.....	תרשים Use Case :UML Actors Diagram
8.....	תרשים המחלקות: UML Class Diagram
8.....	ראה פירוט על קבצי הפיתוח 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.....	מאגרי התמונות
20.....	מאגר התמונות המלא dataset
20.....	מאגר התמונות המיועדות לחיזוי predictions
21.....	מדריך למפתח
21.....	ראה תרשימים בפרק "מבנה / ארכיטקטורה של הפרויקט"
21.....	הסבר קבצי python בפרויקט:
23.....	הסבר ופירוט על כל קובץ python בפרויקט:
23.....	הקובץ הראשי- menu_tkinter.py
29.....	קובץ התצורה- config.py
31.....	קובץ חילוף התמונות- extractzipfiles.py
33.....	קובץ הגדרת מבנה המודל- themodel.py
36.....	הקובץ האחראי על יצירת ושמירת הגרפים- makegraphs.py
40.....	הקובץ האחראי על אימון ובדיקת 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 ההדפסות- printmessage.py
59.....	מסקנות הרצת המודל
59.....	הסבר מושגים:
61.....	מסקנות מהרצות קודמות של מודלים:

62.....	מסקנות מהרצת המודל הנוכחית:
64.....	ערכים סופיים של המדדים להצלחתו של המודל ומסקנות:
65.....	סיכום אישי / רפלקציה
67.....	ביבליוגרפיה
68.....	נספחים

מבוא

השנה, במסגרת התמחות Deep Learning Computer Vision שנלמדת זו השנה השנייה בבית הספר שלי, נדרשנו לבצע את פרויקט הגמר שלנו בנושא זה. ספר זה סוקר את הפרויקט שלי בנושא, וכולל בין היתר את הרקע לפרויקט, הוראות הרצה, הסברים ותרשימים על תכולתו עבור משתמשים ומפתחים, מילון מושגים, מסקנות מההרצה וסיכום אישי שלי על התהליך כולו.

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

בחיים האמיתיים המידע אינו תמיד שחור-לבן, ישר, אחיד ומסודר. לכן רציתי לעסוק בזיהוי של מידע "מעניין" יותר. במקרה הזה, כל תמונה שונה לגמרי מהשנייה מבחינת הסגנון של הספרה והרקע, ועניין אותי לראות איך תוכנה מבוססת למידה עמוקה תתמודד עם השוני. בנוסף, גיליתי עניין מיוחד בתחום ה-Classification לאורך שנת הלימודים הזו, ורציתי מאוד להגיע לתוצר סופי שיוכל לסווג 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 המיועדים לאפשר למחשב ללמוד מתוך דוגמאות, ופועל במגוון משימות חישוביות בהן התכנות הקלאסי אינו אפשרי. הלמידה העמוקה (Deep Learning), היא תת תחום ב-Machine Learning אשר מתבסס על רשתות נוירונים בעלות מספר רב של שכבות. בתחום זה אנו מתבססים על ההנחה שהמחשב יכול ללמוד וללמד את עצמו בדומה למוח האנושי, ולמעשה מנסים לחקות אותו. מהותו של תחום ה-DL היא ליצור חיקוי ממוחשב של פעולת המוח האנושי. בפרק "מסקנות מהרצת המודל" הכנתי מילון מושגים בסיסיים בתחום. אני ממליצה לכל מי שקורא את הספר הזה לקפוץ למילון על מנת לקבל קצת רקע חיוני בנושא.

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

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

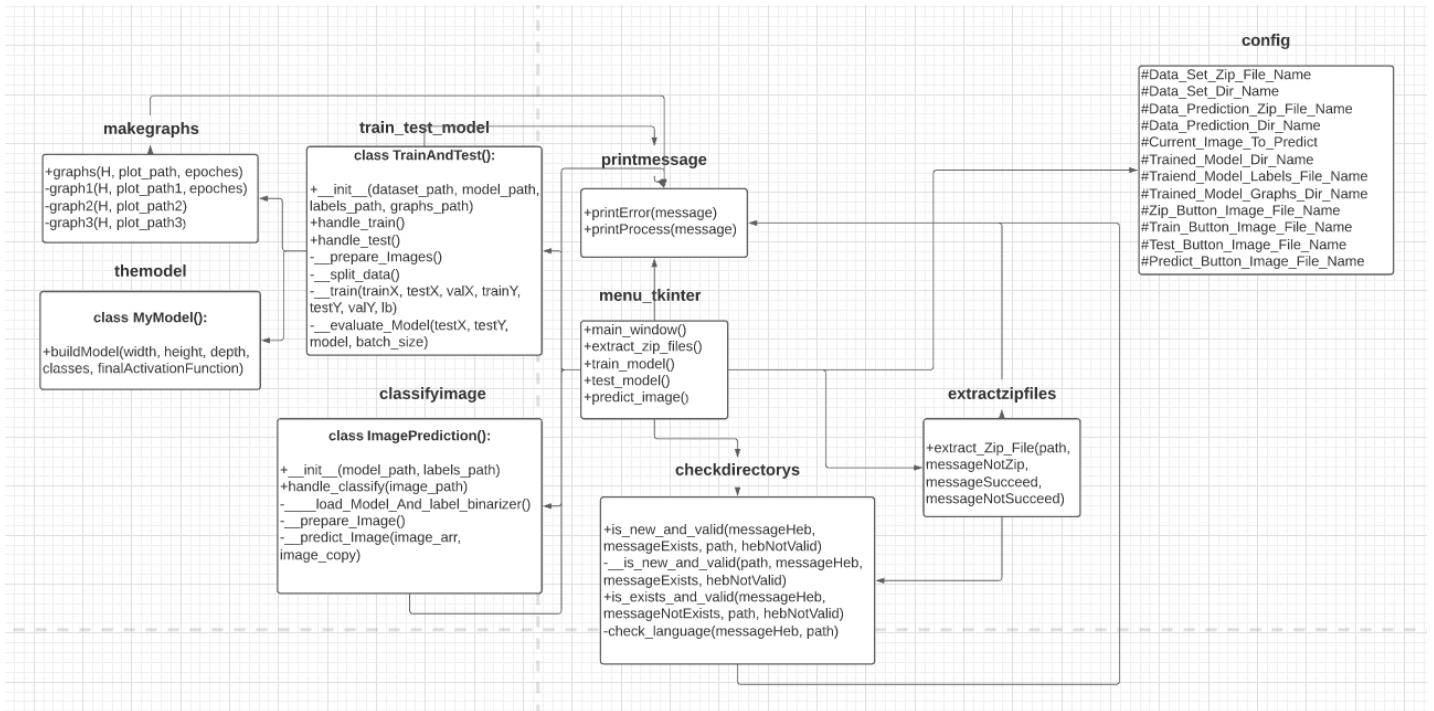
מבנה / ארכיטקטורה של הפרויקט

הפרויקט מכיל מספר קבצים ותיקיות:

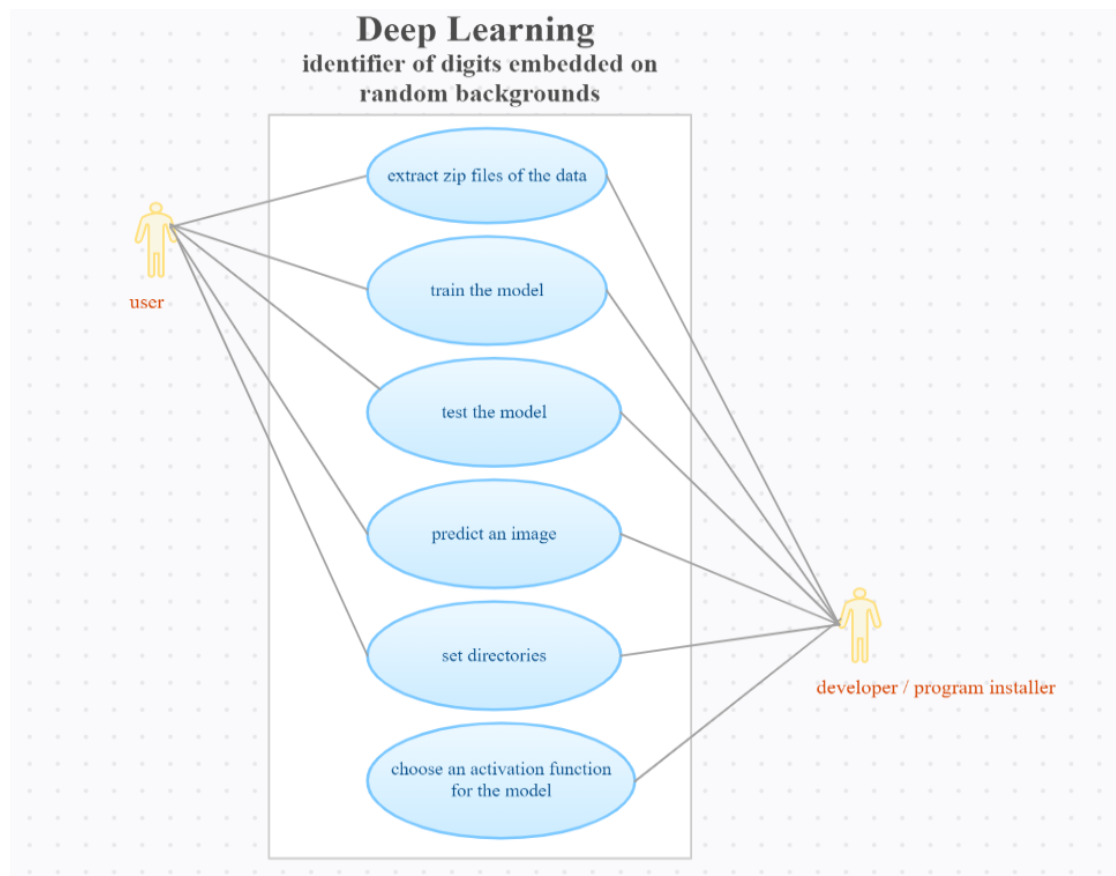
תכולת הפרויקט:

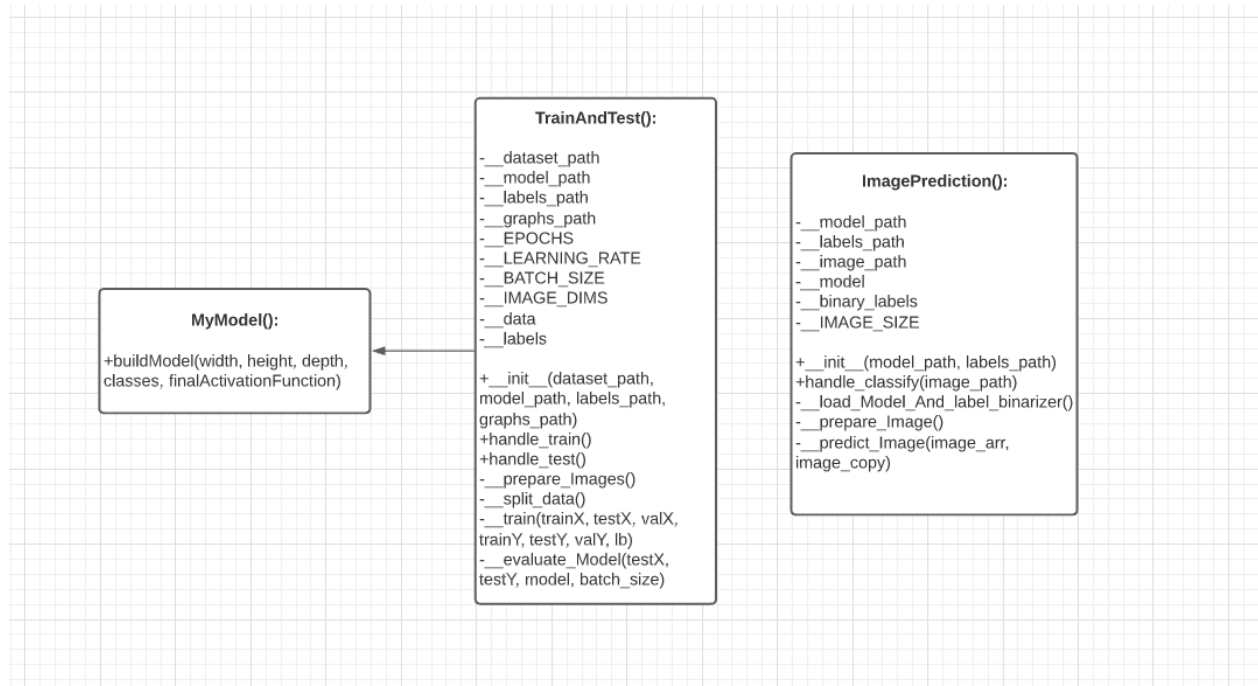
- תיקיית python המכילה את קבצי python של הפרויקט:
 - tkinter_menu- הקובץ הראשי האחראי על ניהול התוכנית והתקשורת עם המשתמש
 - config- קובץ התצורה בו מוגדרים הנתיבים הדרושים בפרויקט
 - train_test_model- הקובץ האחראי על אימון המודל והטסט
 - printmessage- הקובץ האחראי על ההדפסות
 - themodel- הקובץ המכיל את המודל- רשת הנירונים
 - classifyimage- הקובץ האחראי על ביצוע סיווג/ prediction לתמונה
 - extractzipfiles- הקובץ האחראי על חילוץ התמונות מקבצי zip
 - checkdirectorys- הקובץ האחראי על בדיקת הנתיבים המוגדרים בקובץ התצורה
 - makegraphs- הקובץ האחראי על יצירת ושמירת הגרפים המתארים את תהליך האימון
- תיקיית Data המכילה את קבצי zip של מאגרי התמונות בהם נעשה שימוש בפרויקט:
 - קובץ zip המכיל את תמונות datasetn בהן ישתמש המודל לאחר חלוקה עבור: train, validation, test
 - קובץ zip המכיל את תמונות predictions המיועדות לחיזוי (prediction)
- קובץ README המכיל הוראות קצרות להרצת הפרויקט
- תיקיית trained_model בה קבצי המודל המאומן
- קובץ בינארי labels של תוויות התמונות לאחר אימון המודל
- תיקיית graphs בה שלושה גרפים המתארים את תהליך האימון של המודל
- תיקיית buttons המכילה את תמונות הכפתורים שיוצגו למשתמש במסך:
 - תמונה מסוג png של הכפתור הראשון במסך שיוצג למשתמש 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
- תיקיית diagrams המכילה את התרשימים המתארים את הפרויקט:
 - תרשים זרימה של כל קבצי הפיתוח בפרויקט
 - תרשים Use Case
 - תרשים המחלקות
- תרשים model_layers המציג את שכבות המודל

תרשים זרימה של כל קבצי הפיתוח בפרויקט:



תרשים Use Case :UML Actors Diagram



תרשים המחלקות: UML Class Diagramראה פירוט על קבצי הפיתוח 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, אעבור על הדרישות להרצת התוכנית, הוראות ההתקנה והקבצים הנדרשים.

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

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

יש להוריד מספר ספריות קוד בהן הפרויקט משתמש:

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

יש להוריד מחשבון GitHub שלי את קבצי הפרויקט, עליהם ניתן למצוא פירוט בפרק "מבנה / ארכיטקטורה של הפרויקט".

יש לארגן את הקבצים בתוך תיקייה אחת ששם הנתיב לה מורכב מאותיות באנגלית בלבד. על שמות נתיבי התמונות להכיל אותיות באנגלית בלבד.

יש לפתוח את קובץ ה-`python: config.py` באמצעות כל עורך טקסט ולהתאים את הנתיבים המוגדרים שם לאופן בו שמרת אותם במחשב שלך, ועל פי הטבלה בה סיכמתי את הדרישות וההסבר לכל נתיב המופיעה בעוד מספר עמודים.

הפרויקט הורץ על מחשבים בהם מותקנת מערכת ההפעלה Windows 10 בגרסת 64 bit עם הנתונים האלו:

מעבד:	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 x64

מעבד:	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 x64

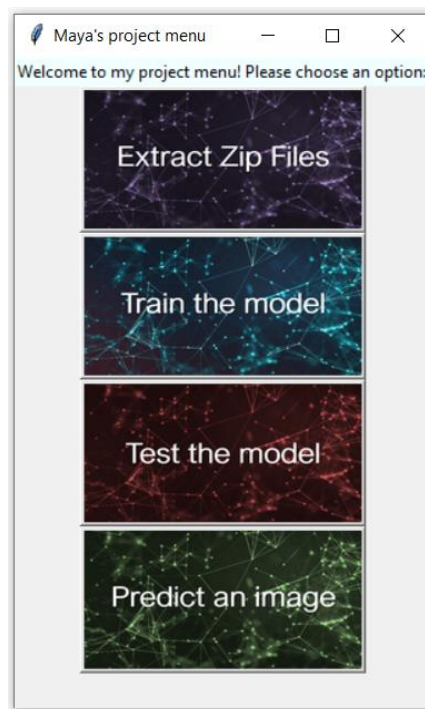
הרצת הסקריפט:

כעת, אסביר כיצד להריץ את התוכנית.

ראשית, יש לפתוח את קובץ הpython: config.py וכאמור לוודא שכל הנתבים המוגדרים בו מתאימים וחוקיים.

לאחר מכן, יש להריץ ב-Command Line את קובץ הpython: menu_tkinter.py. (ניתן לחילופין להריצו ב-Command Line של סביבת העבודה Anaconda על פייתון 3.8.8).

הרצת קובץ זה, המזמן מתודות מקבצי python נוספים בפרויקט, תריץ למעשה את כל הפרויקט. לאחר הרצת התוכנית ייפתח החלון הבא:



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

כפתור 1- חילוץ התמונות מקבצי zip של מאגר התמונות המלא וכן של מאגר התמונות המיועדות לחיזוי, לתיקיות רגילות עם שם זהה (ללא הסיומת zip). זאת, על מנת שהתוכנית תוכל לגשת לתמונות.

כפתור 2- הרצת תהליך האימון למודל (train)

כפתור 3- ביצוע בדיקה למודל (test)

כפתור 4- חיזוי לתמונה ספציפית (prediction). בעת לחיצה על כפתור זה התוכנית מעניקה חיזוי לתמונה שנתיבה מוגדר בקובץ config.py, כלומר מחליטה לאיזו קטגוריה היא שייכת.

יש לשם לב למספר נקודות חשובות, שאי הבנה שלהן עלולה לגרום לך לא להצליח להריץ את הפרויקט כראוי:

בהרצה הראשונה של הפרויקט, יש ללחוץ קודם כל על הכפתור הראשון בתפריט על מנת שהתמונות יחולצו, אחרת התוכנית לא תוכל לגשת אליהן. לאחר מכן בהרצה זו ובהרצות

הבאות אין סיבה ללחוץ עליו, והדבר יגרור הודעת שגיאה, כיוון שהתמונות כבר חולצו. (כמובן שאם ברצונך לבצע את החילוץ ידנית ולשים לב שהנתיבים מתאימים, אין ללחוץ על הכפתור). בכל מקרה, יש לשים לב שהנתיבים המוגדרים למאגרי התמונות קיימים וחוקיים.

אם ברצונך לבצע אימון, עליך לשנות את הנתיבים למודל, קובץ התוויות ותיקיית הגרפים לנתיבים חדשים שעדיין לא קיימים משום שבכל תהליך אימון נוצרים תיקיות וקבצים אלו מחדש בהתאם לתהליך האימון הנוכחי.

אם ברצונך לבצע חיזוי, עליך לשים לב שהנתיבים למודל ולקובץ התוויות הם נתיבים קיימים משום שעבור אפשרות זו יש צורך במודל שמור ובקובץ תוויות 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 באימון כעת כבר קיימים לכן אין בעיה ללחוץ כפתור הטסט מיד לאחר סיוע ביצוע האימון.

אסכם בטבלה את הנתיבים, משמעותם ודרישותיהם:

נתיב	משמעות	דרישה
Data_Set_Zip_File_Name	נתיב לקובץ zip של מאגר התמונות המלא	קובץ zip קיים.
Data_Set_Dir_Name	נתיב לתיקייה המחולצת של מאגר התמונות המלא	הנתיב אליו חולץ קובץ zip של מאגר התמונות המלא- צריך להיות בעל שם זהה ללא הסימט zip.
Data_Prediction_Zip_File_Name	נתיב לקובץ zip של התמונות המיועדות לחיזוי	קובץ zip קיים.
Data_Prediction_Dir_Name	נתיב לתיקייה המחולצת של התמונות המיועדות לחיזוי	הנתיב אליו חולץ קובץ zip של התמונות המיועדות לחיזוי- צריך להיות בעל שם זהה ללא הסימט zip.
Current_Image_To_Predict	שם התמונה המיועדת לחיזוי כולל הסימט	שם קיים של תמונה עם סימט נכונה (jpg או png).
Trained_Model_Dir_Name	נתיב למודל המאומן	אם ברצונך ללחוץ על כפתור 2 (train)- נתיב חוקי וחדש (עדיין לא קיים) אם ברצונך ללחוץ על כפתור 3 (test) או כפתור 4 (predict) ללא ביצוע train בהרצה הנוכחית- נתיב קיים

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

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

אם אחד הנתיבים המוגדרים בקובץ config.py אינו תקין, תודפס למשתמש הודעת שגיאה. למעשה, עבור כל סוג של שגיאה בכל אחד מהנתיבים תודפס הודעה אחרת. הודעות אלו מפרטות באיזה נתיב קיימת הבעיה ומהי הבעיה הספציפית שמנעה מהתוכנית לרוץ כהלכה. בעמוד הבא אתחיל לפרט מה יקרה ומה יוצג למשתמש עבור כל לחיצה על אחד הכפתורים. יש לשים לב כי הדפסות בצבע כחול מספקות אינפורמציה על התהליכים המתרחשים, ואילו הדפסות בצבע אדום מספקות הודעות שגיאה על נתיבים לא תקינים.

לחיצה על כפתור Extract zip files:

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

במידה שהנתיב המגדיר את קובץ מאגר התמונות המלא או הנתיב המגדיר את קובץ מאגר התמונות המיועדות לחיזוי אינם נתיבי zip, תודפס הודעת שגיאה המציינת זאת:

```
the dataset path is not a zip file
```

במידה שנתיב אינו חוקי והחילוץ לא הצליח מכל סיבה שהיא, תודפס הודעות שגיאה:

```
Could not finish to extract dataset dirs
```

במידה שנתיב חוקי, תודפס הודעה המבשרת על הצלחת החילוץ:

```
Finished to extract predictions dirs
```

כשמסתיים תהליך זה המשתמש חופשי שוב ללחוץ על כפתורים אחרים.

לחיצה על כפתור Train the model:

אם הנתיב למאגר התמונות אינו נתיב חוקי, כלומר אינו נתיב קיים, מחולץ וחוקי, יודפסו הודעות שגיאה המסבירות שנתיב זה אינו חוקי ואת הסיבה לכך. נשים לב כי הימצאות אותיות עבריות בנתיב זה נחשבת לשגיאה. זאת מכיוון שבשלב טעינת תמונה באמצעות opencv ישנה שגיאה כשאר הdirectory 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 הנתיב לתיקייה בה יישמר המודל, לקובץ התוויות או לתיקיית הגרפים (שיווצרו בהרצה), אינו נתיב חוקי, כלומר אינו נתיב חדש וחוקי, יודפסו הודעות שגיאה המצביעות על הנתיב 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 שכל הנתיבים המוגדרים מתאימים:

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

ראשית התוכנית טוענת את התמונות מהdirectory של מאגר התמונות. לאחר מכן יוצגו הודעות אינפורמציה על הגודל בMB של רשימת כל התמונות כמערכים. אחר כך תודפס הודעה על כך שהמודל התקמפל.

```
(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 הכולל מידע אודות: השכבות וסדרן במודל, צורת הפלט של כל שכבה, מספר הפרמטרים (משקלים) בכל שכבה, והמספר הכולל 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 (Batch Normalization)	(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 Normalization)	(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 Normalization)	(None, 16, 16, 64)	256
max_pooling2d_1 (MaxPooling2D)	(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 Normalization)	(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 Normalization)	(None, 8, 8, 128)	512
max_pooling2d_2 (MaxPooling2D)	(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 Normalization)	(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...
```

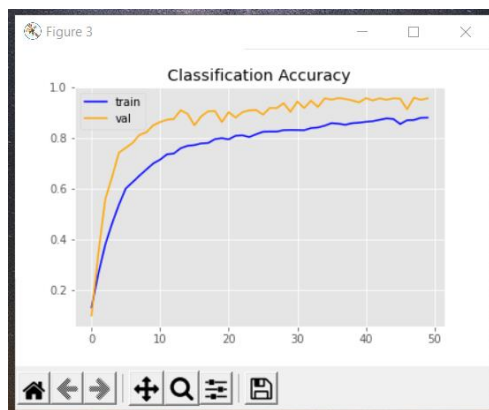
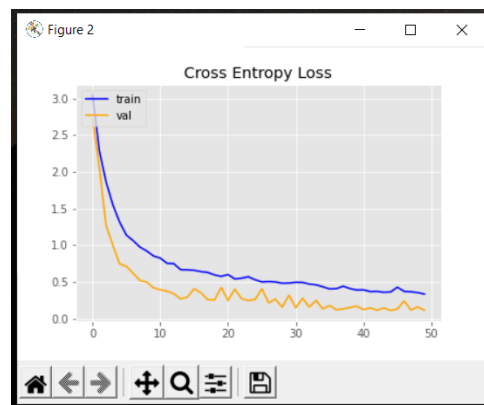
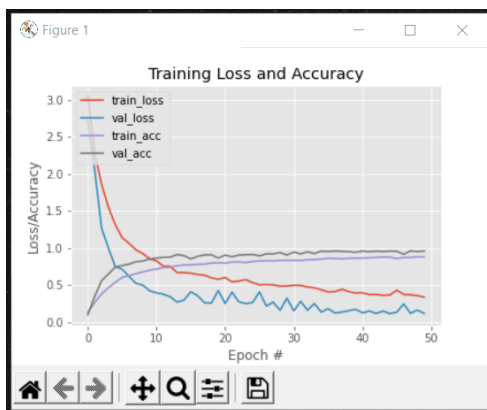
לאחר מכן נראה את האימון מתרחש בפועל כאשר ערכי accuracy וloss של train data ושל validation data משתנים בכל ריצה מחדש, עד ערכים סופיים:

```

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 תהליך האימון ייפתח למשתמש מסך עם גרפים המתארים את תהליך האימון. כאשר יסגור את המסך, ייפתח לו מסך חדש עם גרפים חדשים, וכך גם פעם נוספת (בסך הכל 3 מסכים).



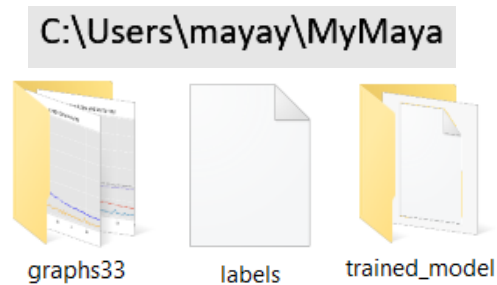
שם הפרויקט: זיהוי ספרות המוטבעות על רקע אקראי

שם התלמידה: מיה יוסף

בסופו של האימון תודפס הודעה למשתמש על הנתיב בו נשמרו הגרפים שהתוכנית זה עתה יצרה, אשר מתארים את תהליך האימון.

Graphs of the training process are in graphs33

המשתמש יכול לבדוק בנתיבי הגרפים, המודל והתוויות, שאכן נוצרו קבצים במקומות אלה:



כשמסתיים תהליך אימון המודל המשתמש חופשי שוב ללחוץ על כפתורים אחרים.
תהליך זה לוקח כ-45 דקות.

לחיצה על כפתור Test the model:

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

```
Error! the images path contains hebrew letters
```

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

במידה שהנתיבים תקינים:

בדיקת המודל (test) תתחיל, בתהליך הלוקח כדקה.

ראשית התוכנית טוענת את התמונות מהdirectory של מאגר התמונות. לאחר מכן תודפס הודעות אינפורמציה על הגודל בMB של רשימת כל התמונות כמערכים.

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

אחר כך תודפס ההודעה "evaluate on test data" ויודפסו ערכי test accuracy ו test loss.

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

כשמסתיים תהליך זה המשתמש חופשי שוב ללחוץ על כפתורים אחרים.

לחיצה על כפתור Predict an image:

אם הנתיב לתמונה המיועדת לחיזוי אינו חוקי, כלומר אינו קיים או חוקי, תודפס הודעת שגיאה מתאימה. גם כאן, מכיוון שמדובר בתמונה שיש לטעון עם הספרייה opencv, directory בו שמורה 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 אי תקינות בנתיב למודל או לקובץ התוויות:

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

במידה שהנתיבים תקינים:

ראשית, תודפס הודעה על כך שהמודל השמור נטען בידי התוכנית. לאחר מכן תודפס הודעה נוספת על כך שהתמונה המיועדת לסיווג נטענת גם היא. הודעה שלישית תסביר שתהליך הסיווג מתרחש ברגעים אלה.

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

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



בנוסף, ייפתח מסך חדש עליו התמונה ועל גביה תווית עם תוצאת החיזוי, הסיכוי שהתוצאה נכונה והאם התוצאה נכונה או לא. אם התוצאה נכונה, המידע על גבי התמונה יהיה בצבע ירוק. אם התוצאה לא נכונה, המידע יהיה בצבע אדום.

תהליך זה לוקח כדקה.

מאגרי התמונות

לאחר שמתבצע החילוץ של התמונות מקבצי zip, מאגרי התמונות הם למעשה תיקיות רגילות המכילות תמונות מסוג jpg. אסביר על כל אחד ממאגרי התמונות.

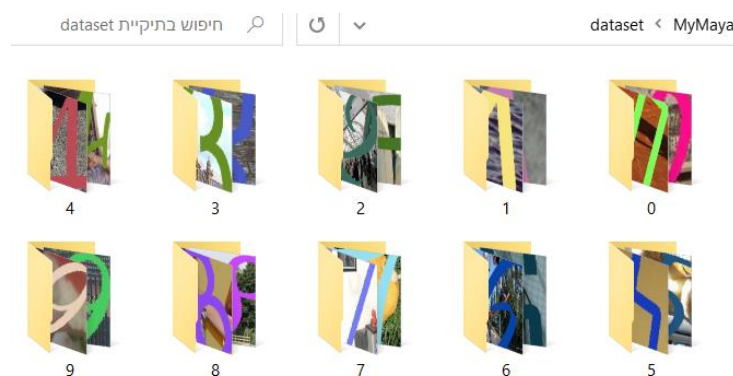
מאגר התמונות המלא dataset

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. בתוך התיקייה ישנן 10 תת תיקיות שבכל אחת מהן כ-1,200 תמונות של ספרה אחת. התמונות הן וריאציות שונות של הספרות: צבע שונה, רקע שונה, זווית שונה, רקע שונה, עובי שונה וכו'. מדובר בספרות שנוצרו באופן מלאכותי בצבעים וגופנים שונים, המסובכות בזוויות שונות ומוטבעות על גבי רקעים אקראיים שונים. או באנגלית: Synthetically generated images of English digits embedded on random backgrounds.

אציין שבמהלך תהליך האימון data מוגדל (מפורט בפרק "מדריך למפתח").



מאגר התמונות המיועדות לחיזוי predictions

תיקיית התמונות השנייה בפרויקט שלי היא תיקייה המכילה כ-70 תמונות המיועדות לביצוע predict לתמונה של ספרה כלשהי על ידי המודל. תיקייה זו מורכבת מתמונות שהעברתי מכל תת תיקייה במאגר התמונות המלא (6 עבור כל ספרה, 54 בסה"כ) ואף מתמונות שיצרתי בעצמי בעזרת תוכנת הציור. כאשר ברצונך לבצע predict לתמונה כלשהי, עליך לבחור באחת מהתמונות בתיקייה זו ולשנות את נתיב התמונה המיועדת לחיזוי בקובץ config.py לנתיב של התמונה שבחרת. יש לשים לב אם סוג התמונה הוא jpg (תמונות רגילות) או png (תמונות שיצרתי בעצמי).



predictions

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

מדריך למפתח

ראה תרשימים בפרק "מבנה / ארכיטקטורה של הפרויקט"

הסבר קבצי python בפרויקט:

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

ראשית, אסביר בקצרה על כל אחד מהקבצים:

קובץ	תפקיד
menu_tkinter.py	הקובץ הראשי של הפרויקט, אשר אחראי על ניהול התוכנית ועל התקשורת עם המשתמש. בקובץ זה מצויה הפונקציה הראשית main_window() המהווה פונקציית שירות עבור המשתמש. כאשר המשתמש מריץ את קובץ פייתון זה, נפתח בפניו חלון עם מספר כפתורים. על מנת לבצע את תפקיד הכפתור 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 שלך (המיקומים בהם המידע הנחוץ כמו מאגר התמונות שמור) ולמטרה שלך (למשל, אם ברצונך ללחוץ על הכפתור של testn עלייך לספק בקובץ זה נתיב למודל שמור, לעומת הכפתור של train שדורש נתיב חדש שמודל מאומן חדש יישמר בו במהלך האימון- הסבר מפורט בפירוט על קובץ זה בהמשך ובמדריך למשתמש).
extractzipfiles.py	קובץ זה אחראי על חילוץ התמונות מקבצי zip של מאגר התמונות המלא ושל מאגר התמונות המיועדות לחיזוי, לתיקיות רגילות עם שם זהה (ללא הסיומת zip).
themodel.py	קובץ זה אחראי על בניית שכבות המודל. המודל הוא למעשה מהות בפרויקט. ביצועים טובים של המודל משמעותם הצלחת מטרת הפרויקט וחיזויים נכונים של התמונות. תהליך למידת התמונות מבוצע על גבי המודל וביצוע testn והחיזוי נעשה בעזרת מודל שמור.

קובץ זה אחראי על ביצוע אימון המודל (למידת התמונות) והtest (בדיקת המודל). קובץ זה הוא למעשה אחד החלקים המרכזיים של הפרויקט.	train_test_model.py
קובץ זה אחראי על ביצוע החיזוי לתמונה כלשהי, כלומר, סיווג התמונה לקטגוריה המתאימה לה.	classifyimage.py
קובץ זה אחראי על בדיקת תקינות הנתיבים שהוגדרו בconfig.py 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

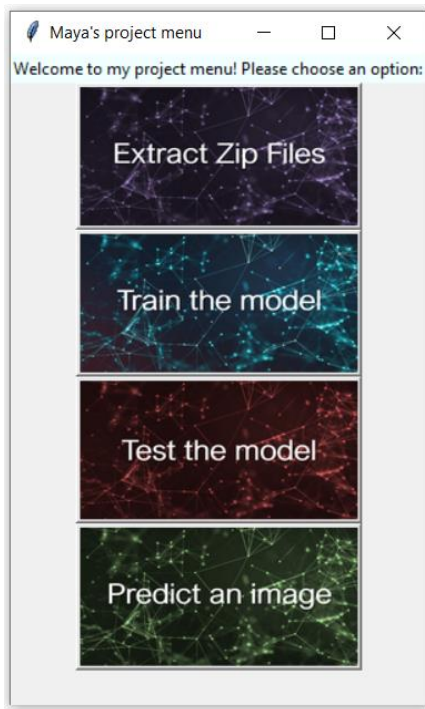
בעמוד הבא אתחיל לפרט על כל אחד מהקבצים.

הסבר ופירוט על כל קובץ python בפרויקט:

הקובץ הראשי- menu tkinter.py

קובץ זה אחראי על ניהול התוכנית ועל התקשורת עם המשתמש. על מנת לבצע את תפקיד הכפתור עליו המשתמש לחץ, קובץ זה מזמן מתודות משאר הקבצים. בקובץ זה מצויה הפונקציה הראשית בשם `main_window()` המהווה 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 מהכפתורים תביא לביצוע המשימה שכתובה על גביו:



1- חילוץ התמונות מקבצי zip של מאגר התמונות המלא וכן של מאגר התמונות המיועדות לחיזוי, לתיקיות רגילות עם שם זהה (ללא הסימנים zip). זאת, על מנת שהתוכנית תוכל לגשת אליהן.

בהרצה הראשונה של הפרויקט, יש ללחוץ קודם כל על הכפתור הראשון בתפריט על מנת שהתמונות יחולצו, אחרת התוכנית לא תוכל לגשת אליהן. לאחר מכן בהרצה זו ובהרצות הבאות אין סיבה ללחוץ עליו, והדבר יגורר הודעת שגיאה, כיוון שהתמונות כבר חולצו.

2- אימון למודל (training)

3- בדיקת המודל (test)

4- ביצוע חיזוי לתמונה ספציפית (prediction)

יש לשם לב למספר נקודות חשובות, שאי הבנה שלהן עלולה לגרום לך לא להצליח להריץ את הפרויקט כראוי:

אם ברצונך לבצע אימון, עליך לשנות את הנתיבים למודל, קובץ התוויות ותיקית הגרפים לנתיבים חדשים משום שבכל תהליך אימון נוצרים תיקיות וקבצים אלו מחדש בהתאם לתהליך האימון הנוכחי.

אם ברצונך לבצע חיזוי, עליך לשים לב שהנתיבים למודל ולקובץ התוויות הם נתיבים קיימים משום שעבור אפשרות זו יש צורך במודל שמור ובקובץ תוויות 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 באימון כעת כבר קיימים לכן אין בעיה ללחוץ כפתור הטסט מיד לאחר סיוע ביצוע האימון.

הסבר המתודות מפורט בעמוד הבא.

המתודה	תפקיד
main_window()	זוהי המתודה הראשית אשר יוצרת את המסך והכפתורים על גביו שהמשתמש בוחר. במתודה זו מוגדרים ומופעלים ארבעת הכפתורים ומזומנות מתוך הכפתורים ארבע הפונקציות האחראיות לביצוע תפקיד כל כפתור.
extract_zip_files()	מתודה זו מגדירה את הנתבים של מאגרי התמונות ומזמנת את המתודה extract_Zip_Files(path, messageNotZip, messageSucceed, messageNotSucceed) datasetn על מנת לחלץ את התמונות של datasetn והתמונות המיועדות לחיזוי מקבצי zip.
train_model()	מתודה זו מגדירה את הנתבים dataset_path (למאגר התמונות המחולצות), model_path (היכן שיישמר המודל המאומן) labels_path (היכן שיישמר קובץ התוויות הבינארי) graphs_path (היכן שיישמרו הגרפים). המתודה בודקת אם הנתבים תקינים בעזרת המתודות is_new_and_valid(messageHeb, messageExists, path, hebNotValid = False) is_exists_and_valid(messageHeb, messageNotExists, path, hebNotValid = False) מהקובץ checkdirectorys.py. בנוסף, היא בודקת האם יש כפילות בנתבים החדשים (למשל, אותו נתיב עבור המודל ועבור הגרפים). אם יש כפילות כזו, היא מדפיסה הודעת שגיאה מתאימה בעזרת המתודה printError(message) שבקובץ printmessage.py. המתודה יוצרת אובייקט חדש בשם training של המחלקה TrainAndTest שבקובץ train_test_model.py ומעבירה את הפרמטרים לבנאי המחלקה. לאחר מכן היא מפעילה את המתודה הפומבית handle_train() על גבי אובייקט זה על מנת לבצע את תהליך האימון.
test_model()	מתודה זו מגדירה את הנתבים dataset_path, model_path ומעבירה אותם כפרמטרים לבנאי המחלקה TrainAndTest שבקובץ train_test_model על גבי אובייקט חדש מסוג המחלקה הזו בשם testing. לאחר מכן היא מפעילה את המתודה הפומבית handle_test() על גבי אובייקט זה על מנת לבצע את test. labels_path, graphs_path מקבלים את הערך "" משום שהם אינם רלוונטיים לtest אלא לתהליך האימון. כמו כן, המתודה בודקת אם הנתבים תקינים בעזרת המתודה is_exists_and_valid(messageHeb, messageNotExists, path, hebNotValid = False) שבקובץ checkdirectorys.py.
predict_image()	מתודה זו מגדירה את הנתבים labels_path, model_path ומעבירה אותם כפרמטרים לבנאי המחלקה ImagePrediction שבקובץ classifyimage על גבי אובייקט חדש מסוג המחלקה הזו. המתודה מגדירה את הנתב image_path לתמונה המיועדת לסיווג 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 image_path כפרמטר על מנת לבצע את prediction. כמו כן, המתודה בודקת אם הנתבים תקינים בעזרת המתודה is_exists_and_valid(messageHeb, messageNotExists, path, hebNotValid = False) שבקובץ checkdirectorys.py.

תדפיס הקוד:

```

"""
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.Traied_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.Traiened_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
    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.Traied_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+"/"+ "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()

```

קובץ התצורה - config.py

בקובץ זה מוגדרים הנתיבים השונים הנמצאים בשימוש בפרויקט:

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

באפשרותך 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 ותיקיית הגרפים לנתיבים חדשים משום שבכל תהליך אימון נוצרים תיקיות וקבצים אלו מחדש בהתאם לתהליך האימון הנוכחי.

אם ברצונך לבצע חיזוי, עליך לשים לב שהנתיבים למודל ולקובץ התוויות הם נתיבים קיימים משום שעבור אפשרות זו יש צורך במודל שמור ובקובץ תוויות 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 באימון כעת כבר קיימים לכן אין בעיה ללחוץ כפתור הטסט מיד לאחר סיוע ביצוע האימון.

יש לשים לב שלפני ההרצה הראשונית צריך ללחוץ פעם אחת (ואחרונה) על כפתור חילוץ קבצי zip על מנת שהתוכנית תוכל לגשת לתמונות.

בקובץ הראשי בפרויקט menu_tkinter.py ישנן פניות לנתיבים המוגדרים בקובץ config.py. השימוש בקובץ תצורה במקום הגדרה של נתיבים בקבצים השונים בפרויקט או בקשת קלט הוא מקצועי, מסודר, ולפעמים גם נוח יותר. במקום לחפש את המקומות בהם יש לבצע שינויים בהגדרות 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.

תדפיס 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"
```

אמשיך לפרט על כל אחד מהקבצים תוך התייחסות לכפתורים המפעילים אותם:**כפתור ראשון****קובץ חילוץ התמונות - extractzipfiles.py**

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

המתודה `extract_Zip_Files()` מזומנת מתוך המתודה `extract_zip_files()` שבקובץ הראשי `menu_tkinter.py`.

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

המתודה	תפקיד
<code>extract_Zip_File(path, messageNotZip, messageSucceed, messageNotSucceed)</code>	<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> המתודה בודקת האם הנתיב שהועבר אליה חדש וחוקי. אם כן, היא מבצעת את החילוץ לתיקייה רגילה עם שם זהה לשם קובץ zip ללא הסיומת zip. כמו כן, המתודה אחראית להדפסת הודעות מתאימות אודות תהליך החילוץ או הודעות שגיאה במקרה הצורך, ונעזרת לשם כך במתודה <code>is_new_and_valid(messageHeb, messageExists, path, hebNotValid = False)</code> מהמחלקה שבקובץ <code>checkdirectorys</code> כאמור. המתודה הזו 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>

תדפיס 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) #####

```


כפתור שני

קובץ הגדרת מבנה המודל - themodel.py

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

מתודה זו מקבלת את ממדי התמונות, מספר הקטגוריות והactivation function שתבצע בשכבה האחרונה של המודל (softmax function מוגדרת כברירת מחדל), ובונה את שכבות המודל בעזרתם. שכבות המודל מהוות למעשה מוח מלאכותי, רשת נוירונים מלאכותית אשר לומדת את התמונות. בכל אחת מהשכבות מספר שונה של נוירונים מלאכותיים, אשר יכולים "לתקשר" עם הנוירונים שבשכבות הסמוכות. זאת, בעזרת משקלים שמהווים חישוב סטטיסטי אשר יניב את אחוז ההצלחה הגבוה ביותר. בעת זימון המתודה buildModel ערכי המשקלים הללו רנדומליים, ובעת אימון המודל הם משתנים בהתאם למערכי התמונות שרצים על גביהם. לבסוף המתודה מחזירה את המודל שבנתה.

המתודה מזומנת בקובץ train_test_model.py על ידי המתודה __train(trainX, testX, valX, trainY, testY, valY, lb) על מנת לאתחל את המודל כדי להשתמש בו בתהליך האימון. לאחר תהליך האימון המתודה שומרת את המודל בנתיב המוגדר בקובץ config.py. בעזרת המודל השמור מתבצעים test והprediction.

השכבה הראשונה במודל היא שכבת הקלט input_shape המגדירה את ממדי התמונה (50,50,1). השכבה האחרונה במודל היא שכבת הפלט, אשר משתמשת בפונקציית softmax במידה ואין קלט אחר. זוהי למעשה שכבת הסיווג. בשכבה שלפניה מוגדר מספר הקטגוריות האפשריות, כלומר מספר נוירוני הפלט. כאשר מריצים תמונה על גבי המודל המאומן היא צפויה להגיע אל אחד מנוירוני הפלט. המתודה למעשה בוחרת את נוירון הפלט המתאים ביותר עבור אותה התמונה, כלומר, בוחרת את החיזוי בעל הסיכוי הגבוה ביותר להיות נכון. ניתן למצוא את תרשימים שכבות המודל בפרק "נספחים".

הסבר על פונקציית האקטיבציה softmax - זוהי פונקציה מתמטית הממירה וקטור מספרים לווקטור הסתברויות, כאשר ההסתברויות של כל ערך פרופורציונליות לסולם היחסי של כל ערך בווקטור. פונקציית softmax משמשת כactivation function בשכבת הפלט של רשתות נוירונים מלאכותיות עבור בעיות multi-class classification בהן יש יותר משתי תוויות (קטגוריות).

בחרתי במודל זה לאחר תהליך ארוך של ניסוי וטעייה, שבסופו הגעתי למסקנה שמודל זה הוא המתאים ביותר לפרויקט שלי, בזכות התוצאות הטובות שהניב ביחס למודלים האחרים בהם התנסיתי.

תדפיס הקוד:

```

"""
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 יצירת ושמירת הגרפים - makegraphs.py

קובץ זה, שמזמין כאשר מתבצע אימון המודל מתוך המתודה `handle_train()` שבקובץ `train_test_model.py`, אחראי על יצירת הגרפים המתארים את תהליך האימון, הצגתם למשתמש ושמירתם בתיקייה ייעודית. הקובץ מכיל מתודה ראשית האחראית לכך, אשר מזמנת מספר מתודות, כל אחת אחראית על תמונה/ מערכת גרפים אחרת. בנוסף, מתודה זו מדפיסה הודעה מתאימה עם מיקום התמונות של הגרפים במחשב.

מתודה	תפקיד
<code>graphs(H, plot_path, epoches)</code>	מתודה האחראית על יצירת הגרפים המתארים את תהליך האימון על ידי הצגת השתנות ערכי <code>loss</code> ו <code>accuracy</code> לאורך. המתודה מקבלת את היסטוריית אימון המודל ואת הנתיב לתיקייה בה יישמרו הגרפים ובעזרתו יוצרת תיקייה בה יישמרו כל התמונות ומגדירה את נתיבי כל תמונה. בנוסף המתודה מקבלת את ה <code>epochs</code> - מספר הפעמים שהתמונות רצות על המודל. המתודה מזמנת כל אחת מהמתודות היוצרות ושומרות את הגרפים. היא מעבירה לכל אחת מהן כפרמטר את הנתיב הספציפי בו יישמר הגרף שיוצרת אותה מתודה, וכן את היסטוריית אימון המודל שבעזרתה כל מתודה תשרטט את הגרפים. בנוסף, היא מעבירה למתודה <code>graph2(H, plot_path2)</code> גם את ה <code>epochs</code> .
<code>graph1(H, plot_path1, epoches)</code>	מתודה היוצרת את מערכת הצירים הראשונה, ועל גביה את הגרפים המציגים את השתנות ערכי <code>accuracy</code> ו <code>loss</code> עבור ה <code>train</code> וה <code>validation</code> לאורך כל תהליך האימון. המתודה משרטטת את הגרפים באמצעות היסטוריית הלימוד שקיבלה כפרמטר <code>H</code> ובעזרת הפרמטר <code>epoches</code> שקיבלה, ושומרת את תמונת תמונת הגרפים בנתיב שקיבלה לשם כך <code>plot_path1</code> . לאחר מכן היא מציגה את התמונה עם הגרפים למשתמש.
<code>graph2(H, plot_path2)</code>	מתודה היוצרת את מערכת הצירים השנייה, ועל גביה את הגרפים המתארים את פונקציית <code>cross entropy loss</code> עבור ה <code>train</code> וה <code>validation</code> . פונקציה זו קטנה ככל שההסתברות החזויה שסיפק המודל זהה לתווית האמיתית של התמונה. המתודה משרטטת את הגרפים באמצעות היסטוריית הלימוד שקיבלה כפרמטר <code>H</code> , ושומרת את תמונת הגרפים בנתיב שקיבלה לשם כך <code>plot_path2</code> . לאחר מכן היא מציגה את התמונה עם הגרפים למשתמש.

<p>מתודה היוצרת את מערכת הצירים השנייה, ועל גביה את הגרפים המתארים את מדד Classification accuracy עבור הtrain והvalidation. מדד זה מסכם את הביצועים של המודל כמספר התחזיות הנכונות חלקי המספר הכולל של התחזיות. הגרף עולה ככל שהחלק של התחזיות הנכונות מתוך סך התחזיות גדל.</p> <p>המתודה משרטטת את הגרפים באמצעות היסטוריית הלימוד שקיבלה כפרמטר H, ושומרת את תמונת הגרפים בנתיב שקיבלה לשם כך plot_path3. לאחר מכן היא מציגה את התמונה עם הגרפים למשתמש.</p>	<p>graph3(H, plot_path3)</p>
--	------------------------------

תדפיס הקוד:

```

"""
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, epochs):
    """
    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 epochs: 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, epochs)
    graph2(H, plotpath2)
    graph3(H, plotpath3)
    printmessage.printProcess("Graphs of the training process are in " + plot_path)

def graph1(H, plot_path1, epochs):
    """
    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 epochs: the number of times the images ran on the model for learning purpose
    """
    plt.style.use("ggplot")
    plt.figure()
    N = epochs
    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()

כפתור שני ושלישי

הקובץ האחראי על אימון ובדיקת המודל - train_test_model.py

קובץ זה אחראי על אימון ובדיקת המודל. בקובץ זה ישנה מחלקה בשם TrainAndTest שבה בין השאר שתי מתודות פומביות, אחת אחראית על הtrain (יחד עם validation) והשנייה אחראית על הtest. שתיהן נעזרות במתודות פרטיות על מנת לבצע את תפקידן.

החלטתי שיש צורך במחלקה משום שמתודות רבות בקובץ זה זקוקות לאותם פרמטרים. זו גם הסיבה שהחלטתי שהטיפול בtrain והטיפול בtest יתרחשו באותה ישות - מחלקה אחת.

במחלקה ישנן תכונות המכילות נתיבים ומשתנים שונים, מתודה פומבית אחראית על training, מתודה פומבית נוספת אחראית על test, ושלל מתודות עזר.

למעשה, על מנת לבצע את הtrain, בקובץ menu_tkinter.py במתודה train_model() יצרתי אובייקט מסוג המחלקה, העברתי לבנאי המחלקה את הפרמטרים הדרושים וזימנתי את המתודה הפומבית handle_train(). על מנת לבצע את test, בקובץ menu_tkinter.py במתודה test_model() יצרתי אובייקט מסוג המחלקה, והעברתי לבנאי המחלקה את הפרמטרים הדרושים וזימנתי את המתודה הפומבית handle_test().

הסבר בנאי המחלקה:

תכונה	ערך	תפקיד
__dataset_path	Input	מחזיקה את הנתיב למאגר התמונות המלא
__model_path	Input	הנתיב בו יישמר המודל (במקרה של טריין) או המודל השמור (במקרה של טסט)
__labels_path	Input	הנתיב בו תוויית התמונות יישמרו (במקרה של טריין)
__graphs_path	Input	הנתיב בו יישמרו הגרפים (במקרה של טריין)
__EPOCHES	50	מספר הפעמים שירוצו התמונות על גבי המודל לצורך ביצוע האימון/ הלמידה (במקרה של טריין)
__LEARNING_RATE	1*e - 3	קצב הלמידה - ערך שקובע כמה לשנות את המודל בתגובה לשגיאה המשוערת בכל פעם שמשקלי המודל מתעדכנים (במקרה של טריין)
__BATCH_SIZE	32	מספר התמונות שהמודל יעבד במקביל בזמן האימון/ הלמידה (במקרה של טריין)

ממדי התמונות שהמודל יריץ (width, height, color).	(50,50,1)	__IMAGE_DIMS
רשימה שתחזיק את מערכים המייצגים את התמונות.	[]	__data
רשימה שתחזיק את התוויות של כל התמונות.	[]	__labels

הסבר שתי התכונות האחרונות: הפונקציה `__prepare_Images()` שבה מתבצעת טעינת התמונות ממירה את התמונות למערכים ושומרת אותם ברשימה `__data`. במקביל פונקציה זו שומרת את הקטגוריה של כל תמונה ברשימת `labels`.

הסבר המתודות:

המתודה	תפקיד
<code>handle_train()</code>	<p>מתודה פומבית של המחלקה, אשר ניתן לקרוא לה רק מתוך אובייקט מאותחל של המחלקה (זו הסיבה שיש בנאי עם פרמטרים).</p> <p>מתודה זו דואגת לביצוע האימון בעזרת זימון המתודות הפרטיות, ואף מזמנת את המתודה <code>graphs</code> שבקובץ <code>makegraphs</code> אשר יוצרת ושומרת גרפים מתארים את תהליך האימון.</p> <p>המתודה מזמנת את המתודה הפרטית <code>__prepare_Images()</code> על מנת להכין את התמונות ל-<code>test</code>. אחר כך המתודה שומרת את רשימת התוויות בקובץ בינארי שישמש בחלוקת התמונות. בנוסף המתודה יוצרת טרנספורמציה מן הקובץ הבינארי שתשמש לאימון המודל. לאחר מכן המתודה מזמנת את <code>train(self, trainX, testX, valX, trainY, testY, valY, lb)</code> היסטוריית אימון המודל שהיא מחזירה במשתנה <code>training_history</code>. לאחר מכן היא מעבירה את הנתיב לגרפים, את היסטוריית האימון ואת <code>EPOCHES</code> למתודה <code>graphs</code> שבקובץ <code>makegraphs.py</code>.</p>
<code>handle_test()</code>	<p>מתודה פומבית נוספת של המחלקה. מתודה זו דואגת לביצוע <code>test</code> בעזרת זימון המתודות הפרטיות.</p> <p>המתודה מזמנת את המתודה הפרטית <code>__prepare_Images()</code> על מנת להכין את התמונות ל-<code>test</code>. אחר כך המתודה שומרת את רשימת התוויות בקובץ בינארי שישמש בחלוקת התמונות. בנוסף המתודה יוצרת טרנספורמציה מן הקובץ הבינארי שתשמש</p>

<p>לאימון המודל. לאחר מכן מתודת זו מזמנת את המתודה הפרטית <code>__split_data()</code> על מנת לבצע את חלוקת מאגר התמונות. אחר כך היא טוענת את המודל. לסיום היא מזמנת את המתודה הפרטית <code>evaluate_Model(testX, testY, __model, batch_size = 32)</code> אשר מבצעת את <code>testn</code> עצמו.</p>	
<p>מתודה פרטית האחראית על הכנת התמונות לתהליך האימון. המתודה אחראית למעשה על טעינת התמונות ממאגר התמונות והתאמתן לריצה על גבי המודל. האופן בו תפקיד זה מתבצע:</p> <p>המתודה משנה את גדלי ממדי התמונה לגודל אחיד הקבוע בתכונה <code>__IMAGE_DIMS</code>, ממירה את התמונות מ RGB ל <code>GrayScale</code> וממירה אותן למערכים המחזיקים את הפיקסלים של התמונות. כמו כן היא מקטינה את טווח הפיקסלים של המערכים השמורים ב <code>__data</code> מהתחום <code>[0,255]</code> לתחום <code>[0, 1]</code> (נרמול). בנוסף מתודה זו מחלצת משם התיקיה שבה כל תמונה נמצאת את הקטגוריה שלה ומתאימה ביניהן באמצעות שתי רשימות המהוות חלק מתכונות המחלקה: רשימה אחת מאכלסת את מערכי התמונות <code>__data</code> והשנייה את התוויות <code>__labels</code> (סוג התמונה- הספרה), עליהן פירטתי למעלה. ההוספה לרשימות נעשית ביחד בלולאה ולכן ישנה התאמה במספר האינדקס בין שתי הרשימות. כך ניתן לקשר בין כל תמונה לקטגוריה שלה. המתודה אינה מחזירה ערך משום השינוי נעשה על תכונות האובייקט של המחלקה.</p>	<p><code>__prepare_Images()</code></p>
<p>מתודה פרטית אשר מחלקת את מאגר התמונות ל 3 קטגוריות:</p> <ol style="list-style-type: none"> 1. תמונות המיועדות ל <code>train</code> (70%) 2. תמונות המיועדות ל <code>test</code> (20%) 3. תמונות המיועדות ל <code>validation</code> (10%) <p>המתודה מחזירה את המידע המחולק. <code>trainX, testX, valX</code> = רשימות המערכים המייצגים את התמונות <code>trainY, testY, valY</code> = רשימות התוויות של התמונות</p>	<p><code>__split_data()</code></p>

<p>מתודה פרטית אשר מבצעת את אימון המודל בפועל.</p> <p>המתודה מבצעת רצף פעולות שבסופן אימון המודל עצמו: ראשית, היא בונה את מחולל התמונות לצורך הגדלת datan 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 לפני האימון, יוצרת טבלה של סיכום המודל, ולבסוף, כאמור, מכשירה את רשת הניורונים, כלומר מבצעת את אימון המודל. בנוסף, המתודה שומרת את המודל המאומן בקובץ ואת קובץ התוויות הבינארי כדי שיהיה ניתן להשתמש בלמידה הנוכחית גם בהרצות אחרות של התוכנית ובשביל שיהיה ניתן לבצע predict וtest מבלי לבצע את האימון בכל פעם מחדש. לבסוף המתודה מחזירה את היסטוריית אימון המודל, שתשמש לבניית הגרפים בהמשך.</p>	<pre>__train(self, trainX, testX, valX, __ trainY, testY, valY, lb)</pre>
<p>מתודה פרטית זו מבצעת את הtest בפועל. זוהי אחת מהדרכים להעריך את דיוקו של המודל.</p>	<pre>__evaluate_Model (testX, testY, themodel, batch_size=32)</pre>

תדפיס הקוד:

```

"""
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)

    #splitting the data 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)

    #splitting the data 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
```


כפתור רביעי

הקובץ האחראי על ביצוע חיזוי לתמונה - classifyimage.py

קובץ זה אחראי על ביצוע תפקידו של הכפתור הרביעי- בהינתן תמונה שהנתיב אליה מוגדר בקובץ config.py, הקובץ חוזר לאיזו קטגוריה היא שייכת (כלומר, מהי הספרה אותה היא מתארת). בקובץ זה ישנה מחלקה בשם ImagePrediction הדורשת בבנאי שלה שני פרמטרים: נתיב למודל השמור, ונתיב לקובץ הבינארי המכיל את תוויות התמונות.

במחלקה זו מתודה פומבית אחת הנעזרת במתודות פרטיות שבאמצעותן היא מממשת את מטרת הקובץ- ביצוע החיזוי. מתודה זו מקבלת את הנתיב לתמונה אותה רוצים לחזות.

לפני זימון מתודה זו מתוך המתודה predict_image() בקובץ menu_tkinter.py יצרתי אובייקט מסוג המחלקה והעברתי לבנאי המחלקה את הפרמטרים הדרושים. אחת ממתודות המחלקה טוענת את המודל השמור במחשב. אחרת מריצה על גביו את התמונה שהתקבלה כקלט מן המשתמש. מתודה נוספת מוצאת את הקטגוריה אשר לה הסיכוי הגבוה ביותר להיות הנכונה, ובעזרת הקובץ הבינארי שנשמר בעת אימון המודל מציגה למשתמש כפלט האם חיזוי המודל נכון על ידי השוואה בין התו הראשון בשם התמונה, שהוא למעשה הקטגוריה אליה היא שייכת (הספרה המתאימה) לבין תוצאת החיזוי של המודל.

הסבר בנאי המחלקה:

תכונה	ערך	תפקיד
__model_path	input	מכילה את הנתיב בו שמור המודל.
__labels_path	input	מכילה את הנתיב בו שמור הקובץ הבינארי המכיל את תוויות התמונות.
__image_path	""	מכילה את נתיב התמונה אותה רוצים לסווג.
__model	None- a default value	תכונה שאליה ייטען המודל השמור.
__label_binarizer	None- a default value	תכונה שאליה ייקרא הקובץ הבינארי השמור.

הסבר המתודות במחלקה:

מתודה	תפקיד
handle_classify(image_path)	המתודה הראשית והפומבית היחידה במחלקה זו אשר אחראית לביצוע החיזוי ומזמנת את שאר המתודות לשם כך. המתודה מקבלת כפרמטר את הנתיב לתמונה אותה רוצים לסווג לקטגוריה הנכונה (הספרה שהיא מתארת). המתודה מזמנת את המתודה __load_Model_And_label_binarizer() על מנת לטעון את המודל השמור והקובץ הבינארי של התוויות. לאחר מכן היא

<p>מאתחלת את התכונה <code>image_path</code>. אחר כך היא מזמנת את המתודה <code>__prepare_Image()</code> שמבצעת את התאמת התמונה המיועדת לחיזוי לריצה על המודל ומחזירה את המערך המותאם למודל שלה וכן עותק של התמונה המקורית, ולבסוף מעבירה אותם כפרמטרים למתודה <code>predict_Image(image_array, __image_copy)</code> על מנת לבצע את החיזוי עצמו ולהציג פלט מתאים.</p>	
<p>מתודה פרטית אשר טוענת את המודל השמור לתוך התכונה <code>__model</code> וקוראת את הקובץ הבינארי לתוך התכונה <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> לריצה על המודל, בדומה לתהליך שבוצע על מאגר התמונות המלא לפני תהליך האימון שלהן. המתודה למעשה משנה את ממדי התמונה לגודל אחיד (1, 54, 96), ממירה את התמונה מ RGB ל GrayScale, ממירה את התמונה למערך ומנרמלת אותה (מקטינה את טווח הפיקסלים שלה מ [0,255] ל [0,1]). המתודה מחזירה את המערך המייצג 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 הפרטית <code>__prepare_Image()</code>: המערך המייצג את התמונה והעתק של התמונה כפי שהתקבלה כקלט. המתודה מבצעת <code>predict</code> לתמונה על גבי המודל הטעון. 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 חישוביו, והאם הזיהוי היה נכון. זאת על ידי השוואת התו הראשון של שם התמונה, שהוא הספרה שהיא מתארת, לתווית שהמודל זיהה, המציינת את הקטגוריה- הספרה- אליה שייכת התמונה.</p>	<code>__predict_Image(image_array, image_copy)</code>

תדפיס 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 classification. 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 probabilitiesbility 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 config.py, ועל הדפסת הודעות שגיאה למשתמש במידת הצורך. הודעות אלו מפרטות באיזה נתיב קיימת הבעיה ומה למעשה הבעיה הספציפית שמנעה מהתוכנית לרוץ כהלכה. הוא מכיל מתודה אחת האחראית על בדיקת נתיבים קיימים, מתודה נוספת האחראית על בדיקת נתיבים חדשים, ועוד שתי מתודות עזר. במידה שהודפסה הודעת שגיאה, על המשתמש לשנות בהתאם אליה את הגדרת הנתיב שגררה את השגיאה בקובץ config.py.

תפקיד	מתודה
מתודה זו בודקת אם הנתיב path שקיבלה הוא נתיב חוקי ליצירת תיקייה חדשה, שאינו קיים עדיין. ראשית, היא מזמנת את המתודה is_new_and_valid(path, __messageHeb, messageExists, hebNotValid = False) על מנת לבדוק אם מדובר בנתיב חדש וחוקי מבחינת אי הימצאות אותיות עבריות בנתיב במידה שאותיות עבריות בנתיב ייחשבו כשגיאה. אם מתודה זו מחזירה שקר, יוחזר שקר. אם מתודה זו מחזירה אמת, המתודה מנסה לפתוח תיקייה בנתיב זה. אם היא מצליחה, היא מסירה אותה ומחזירה אמת. אם לא, היא מחזירה שקר. hebNotValid - משתנה בוליאני שערכו אמת אם אותיות עבריות בנתיב ייחשבו כשגיאה, וערכו שקר אחרת ואף באופן דיפולטיבי. המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהיא בודקת.	def is_new_and_valid(messageHeb, messageExists, path, hebNotValid = False)
המתודה מחזירה אמת אם שני תנאים (ישנם עוד) לנתיב חוקי מתקיימים. ראשית, אם ערכו של hebNotValid הוא אמת, כלומר אסור שיהיו אותיות עבריות בנתיב, היא בודקת זאת, ואם ישנן היא מחזירה שקר. במידה ואינן, או שhebNotValid הוא שקר, המתודה בודקת האם הנתיב כבר קיים או חדש. אם הוא קיים, היא מחזירה שקר. אחרת, מחזירה אמת. המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהיא בודקת.	__is_new_and_valid(path, messageHeb, messageExists, hebNotValid = False)

<p>המתודה מחזירה אמת במידה והנתיב path כבר קיים וחוקי, ושקר אחרת. אם hebNotValid הוא שקר המתודה רק בודקת אם הנתיב קיים. אם hebNotValid הוא אמת היא בודקת בנוסף אם קיימות בו אותיות עבריות. המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהיא בודקת.</p>	<p>is_exists_and_valid(messageHeb, messageNotExists, path, hebNotValid = False)</p>
<p>המתודה מחזירה אמת אם בנתיב path קיימות אותיות בעברית ושקר אחרת. המתודה מקבלת הודעות אפשריות להדפסה ומדפיסה את ההודעה המתאימה לפי התנאים שהיא בודקת.</p>	<p>check_language(messageHeb, path)</p>

המתודות הפומביות בקובץ, is_new_and_valid() ו is_exists_and_valid() מזומנות בקובץ menu_tkinter.py וכן בקובץ extractzipfiles.py על מנת לבדוק את תקינותם של הנתיבים המוגדרים בקובץ config.py.

תדפיס הקוד:

```

"""
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 ההדפסות - printmessage.py

קובץ זה אחראי על ביצוע ההדפסות למשתמש, המסווגות לשני סוגים:

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

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

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

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

תדפיס הקוד:

```

"""
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

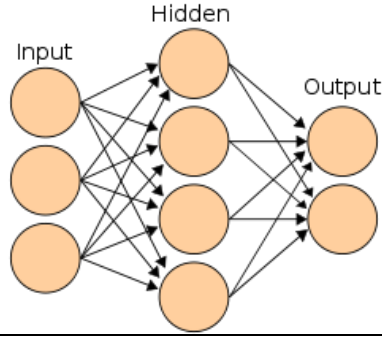
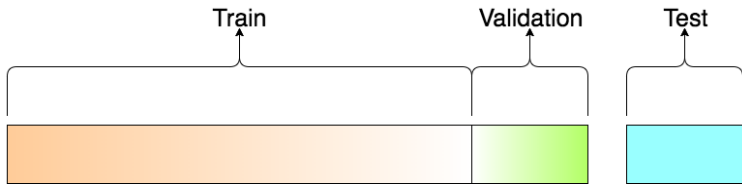
    param message: a meesage for printing
    """
    init(convert = True)
    print(Fore.BLUE + message)
    Style.RESET_ALL

```

מסקנות הרצת המודל

הסבר מושגים:

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

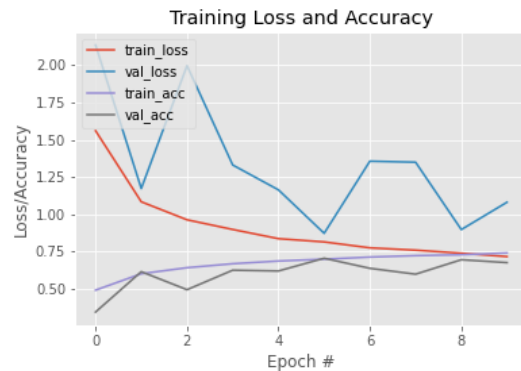
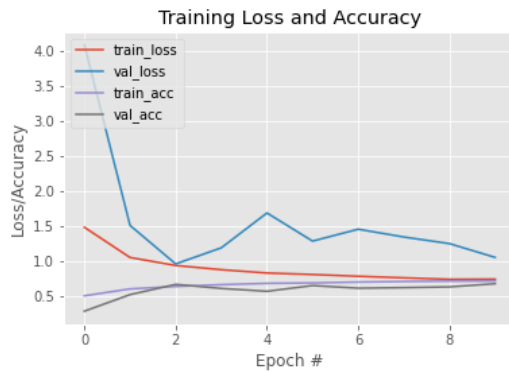
מושג	הסבר
ANN (Artificial Neural Network) 	רשת נוירונים מלאכותית שמחקה את החשיבה האנושית. נוירון במוח הוא תא עצב פשוט, חלק מרשת עצבית עצומה-המוח. ANN היא רשת עצבית מלאכותית, שמעתיקה את הפעילות של שכבות תאי העצב בניאו-קורטקס, האזור שתופס את רוב המוח האנושי.
CNN (convolutional neural network)	סוג של רשת נוירונים מלאכותית אשר משמשת לזיהוי ועיבוד תמונה, ומתוכננת במיוחד לעיבוד נתוני פיקסלים. לרשתות אלה מספר גדול של שכבות חביוות. השכבות של CNN מורכבות משכבת קלט, שכבת פלט ושכבה נסתרת, הכוללת מספר רב של שכבות פיתול, שכבות מאגדים, שכבות מחוברות לחלוטין ושכבות נורמליזציה.
מערך הנתונים עליו פועל מודל למידת המכונה מחולק ל3 קטגוריות: 	
Train set	מדגם הנתונים המשמש להתאמת המודל. מערך הנתונים שבו אנו משתמשים לאימון המודל (weights ו-biases במקרה של רשת נוירונים). המודל לומד מנתונים אלה.
Validation set	מדגם הנתונים המשמש למתן הערכה של התאמת המודל לtrain set תוך כוונן של הייפרפרמטרים של המודל.

Test set	מדגם הנתונים המשמש למתן הערכה של התאמת המודל הסופי לtrain set. מערך זה מספק את "תקן הזהב" המשמש להערכת המודל. משתמשים בו רק ברגע שהמודל מאומן לחלוטין.
מדדים להצלחת מודלים של סיווג:	
accuracy	מתאר את אחוז ההצלחה של המודל בחיזוי הקטגוריות של התמונות. פועל באופן בינארי- חיזוי נכון או לא נכון. ככל שהaccuracy גבוה יותר, זה אומר שהמודל פועל טוב יותר.
loss	מתאר כמה קרובות היו תוצאות חיזוי התמונות, לקטגוריה האמיתית שאליה הן שייכות. לא פועל בערך בינארי. ככל שהloss נמוך יותר, זה אומר שהמודל פועל טוב יותר.
הייפרפרמטרים (משתנים השולטים בתהליך האימון כולו, אשר לא ניתן לאמוד מתוך מערך הנתונים):	
epochs	מספר הפעמים שירוצו התמונות על גבי המודל לצורך ביצוע הלמידה.
batch size	מספר התמונות שהמודל יעבד במקביל בזמן הלמידה.
learning rate	קצב הלמידה- ערך שקובע כמה לשנות את המודל בתגובה לשגיאה המשווערת בכל פעם שמשקלי המודל מתעדכנים.
בעיות שעלולות להתרחש:	
Overfitting	התאמת יתר- מצב בו המודל מותאם יתר על המידה לאוסף נתונים ספציפי (ה train set במקרה זה) ועל כן מצליח פחות בביצוע תחזיות. התאמת יתר מתרחשת כאשר המודל נקבע על ידי יותר פרמטרים מאשר הנתונים מצדיקים.
Underfitting	תת התאמה- מצב בו המודל פשוט מדי מכדי לייצג כראוי את המבנה הבסיסי של הנתונים, למשל בעקבות מיעוט בפרמטרים המגדירים את המודל.

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

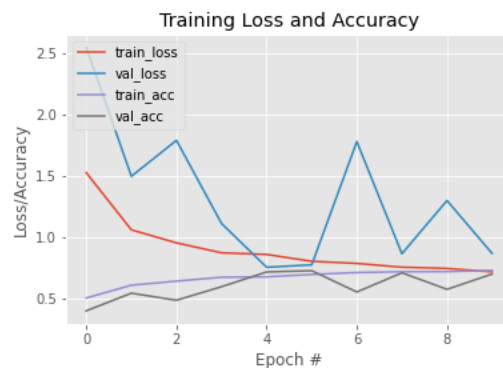
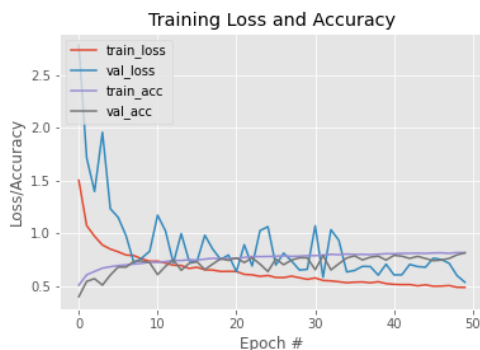
מסקנות מהרצות קודמות של מודלים:

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



לאט לאט הבנתי מה הבעיות ושיפרתי את ביצועי המודל, והגרפים המתארים את ערכי loss ו accuracy נראו טוב יותר. תוך כדי תהליך העבודה על הפרויקט למדתי להכיר טוב יותר את העבודה עם המודלים. למשל, למדתי שכלל שמוסיפים שכבות למודל כך ניתן להימנע ממצבים של תת התאמה. כמו כן, ראיתי איך הגדלת מספר epochs (מספר הפעמים שתהליך הלמידה מתבצע מחדש) מסייעת מאוד לשיפור ערכי loss ו accuracy. בנוסף, הגעתי למסקנה (שאני רוצה עוד לבחון) שמודלים מסוימים מספקים ביצועים טובים יותר כאשר בתמונות שרצות על גביהם ישנו אובייקט הבולט על רקע שונה ממנו.

הגעתי לכל המסקנות הללו מתוך ההתנסות האישית שלי לאורך תהליך העבודה. מימין נמצאים גרפים שמתארים ערכים לא מספיק טובים של loss ו accuracy כאשר מספר epochs נמוך יותר ממספר epochs הנוכחי בפרויקט שלי. משמאל נמצאים גרפים שמראים כיצד המודל מתקשה כאשר רצות על גביו תמונות שלא מתאימות אליו בתצורתו הנוכחית.



מסקנות מהרצת המודל הנוכחית:

כעת, לאחר שיישמתי את הלקחים שלמדתי מתחילת העבודה על הפרויקט, הגעתי לתוצאות טובות. אציג את הגרפים שלי ואסביר על כל אחד מהם:

מערכת 1:

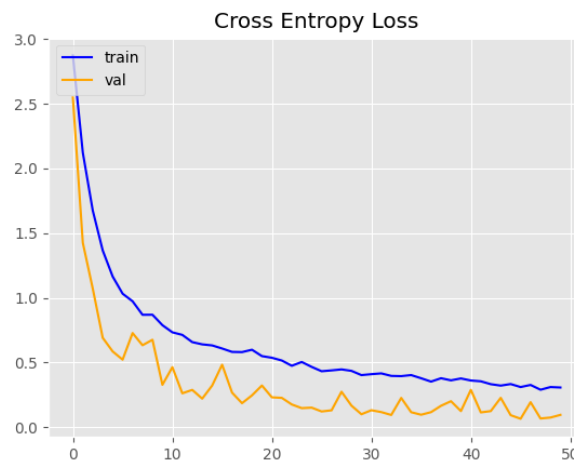
מכילה 4 גרפים המציגים את השתנות ערכי הaccuracy והloss עבור הtrain והvalidation לאורך תהליך האימון.



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

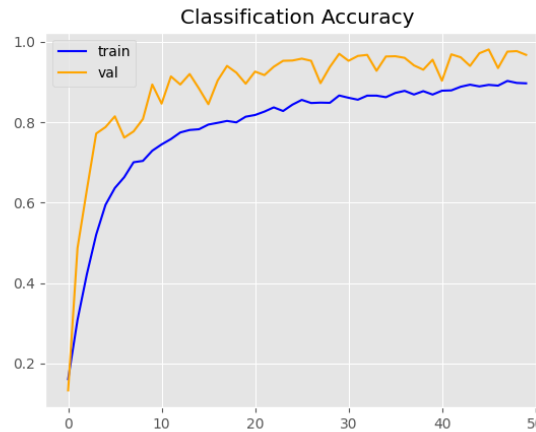
מערכת 2:

מכילה 2 גרפים המתארים את פונקציית cross entropy loss עבור הtrain והvalidation לאורך תהליך האימון. פונקציה זו קטנה ככל שההסתברות החזויה שסיפק המודל זהה לתווית האמיתית של התמונה.



מערכת 3:

מכילה 2 גרפים המתארים את מדד Classification accuracy עבור הtrain והvalidation לאורך תהליך האימון. מדד זה מסכם את הביצועים של המודל כמספר התחזיות הנכונות חלקי המספר הכולל של התחזיות. הגרף עולה ככל שהחלק של התחזיות הנכונות מתוך סך התחזיות גדל.



ניתן לראות בכל המערכות את גרפי loss יורדים עד לערכים קטנים, ואת גרפי accuracy עולים עד לערכים גדולים. הדבר מצביע על כך שאכן מתחרשת למידה טובה של התמונות והמודל מצליח לספק ביצועים טובים. פונקציית loss קטנה ככל שההסתברות החזויה שסיפק המודל זהה לתווית האמיתית של התמונה. כלומר, כפי שניתן לראות בגרפים, כל שמתקדמים באימון המודל טוב יותר ומספק יותר פעמים חיזוי נכון. גרף accuracy כאמור עולה ככל שהחלק של התחזיות הנכונות מתוך סך התחזיות גדל. כלומר, ככל שמתקדמים באימון המודל יש יותר ויותר תחזיות נכונות.

לסיום, אציג את הערכים הסופיים של accuracy והloss עבור הtrain set, validation set וtest set ואסכם את מסקנותי בעמוד הבא:

ערכים סופיים של המדדים להצלחתו של המודל ומסקנות:

אלו הם הערכים סופיים של ה- train accuracy, validation accuracy, test accuracy, 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 תמונות הספרות רצות על גביו, ותהליכי הtraining והtest כתובים בצורה טובה. עוד אני מסיקה שחלוקת התמונות ל70%, 20%, 10% בין הtrain, test, validation, שהיא החלוקה הסטנדרטית, היא חלוקה טובה. כמו כן ערכי הlearning rate וepochs מוכיחים עצמם כערכים מתאימים, וכך גם אלגוריתם האופטימיזציה Adam של המודל ניכר כמתאים. נוסף על כך, הגדלת הנתונים תוך כדי ריצה מסייעת גם היא להשגת תוצאות אלה, שהרי הdataset בו אני עושה שימוש אינו גדול כל כך.

סיכום אישי / רפלקציה

כתיבת הפרויקט הייתה תהליך מעניין ומעשיר, אך יחד עם זאת גם לא קל ומפותל. נתקלתי בקשיים ואתגרים מסוגים 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 שהתחלתי לעבוד. כמו כן, בעיות שונות שהיו לי הן הגעה לערכים לא מספיק טובים של `loss accuracy`, בעיות עם הורדת ספריות ומודולים שונים, בעיות בהרצה `cmd`, אתגרים בלמידה עצמית של מרבית החומר ועוד.

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

בסופו של דבר, אני מרגישה שהפקתי לקחים רבים והגעתי למסקנות רבות שאקח איתי הלאה בנוגע לדרך בה כדאי להתנהל בתהליך של מטלה מורכבת.

למשל, למדתי על החשיבות העבודה מסודרת וחלוקה לוגית של חלקי הפרויקט. בנוסף, הידע שלי בשפת `python` בה כתבתי את הפרויקט ובנושא `deep learning` כמובן, גדל בצורה משמעותית. מתוך ההתמודדות עם שגיאות רבות והניסיון להבין דברים מורכבים למדתי דרכים לבדוד את מקור הבעיה ולמצוא אותה וכן להבין את משמעותן של שורות קוד מסוימות מהאינטרנט בהן נעזרתי. למדתי כיצד להתנהל בצורה טובה מבחינת חיפוש מידע באינטרנט ומציאת פתרונות לבעיות שהתעוררו אצלי באתרים שונים. לו הייתי חוזרת בזמן, הייתי מתחילה את הפרויקט מוקדם יותר, וכך נותנת לעצמי יותר מרחב וזמן להתנסויות וטעויות עם פחות לחץ. העבודה הייתה יעילה יותר אם הייתי יודעת את כל מה שאני יודעת עכשיו, אבל אני שמחה על כל טעות שעשיתי, משום שהיא קירבה אותי להיות כותבת קוד טובה יותר. ככל שהתקדמתי בתהליך, הרגשתי שאני הופכת חזקה יותר בתחום וכך הביצועים שלי השתפרו. למדתי למשל שבניסיון למצוא טעות במקום מסוים לא כדאי לבזבז את כל הזמן על בדיקה חוזרת ונשנית של החלק הקטן שבו הטעות נמצאת במחשבה ראשונה, אלא להסתכל על התמונה הרחבה יותר, למשל על אופן הזימון של הפעולה שמבצעת את אותו חלק בקוד, הפרמטרים המועברים או המוחזרים וכדומה.

על אף שפעמים רבות במהלך כתיבת הפרויקט חשתי מתוסכלת, מצאתי את התהליך כולו מהנה, מעשיר ומלמד. `deep learning` הוא נושא מרתק כשלעצמו. ללא קשר לנושא הספציפי הזה, אני אוהבת את עולם התכנות ואוהבת לכתוב קוד. מעבר לכך, אני אוהבת ללמוד ולהשיג את המטרות שאני מציבה לעצמי. לכן, על אף שהתהליך לא היה קל, בסופו של דבר הוא היה חיובי, והקושי בדיעבד הוא גם דבר חיובי.

במהלך הפרויקט ייעלתי את עצמי. למשל, ככל שעבר הזמן סיגלתי לעצמי יכולת טובה יותר של עבודה על כמה דברים במקביל. הפכתי חכמה יותר ובעלת 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 או בספר הפרויקט, על מנת לא לבזבז זמן. ככל שהתקדמתי בתהליך למדתי להציב לעצמי מטרות יומיות, להעלות על הכתב את הבעיות שלי ואת התכנונים והיעדים שלי כדי לעבוד טוב יותר ולעמוד בלוחות הזמנים.

שאלה שהתעוררה אצלי במהלך הפרויקט, ושארצה לתת עליה את הדעת בהמשך, היא עד כמה המודל הנוכחי מתאים לזיהוי ספרות שאינן מתוך מאגר המידע בו השתמשתי, וכן איך אפשר לשפרו כדי שיתאים לספרות שנכתבו בכתב יד למשל, ולא בצורה סינטטית. כדי

להתחיל לענות על השאלה הזאת יצרתי בעצמי בעזרת 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, אצור 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 ביצוע העבודה, אדגיש את הלמידה לעומק של נושא Deep Learning, פתרון בעיית הסיווג בעזרת CNN, שימוש בספריות שונות כמו Tkinter והרחבת הידע בשפת Python. כמו כן למדתי מהכתיבה של ספר הפרויקט על האופן בו מסבירים פרויקט לנמענים שונים ועל סוגי תרשימים שונים כמו UML Use Case.

לסיכום, אני שמחה על ההזדמנות למתוח מעט את הגבולות של הדברים שחשבתי שאני מסוגלת להם. אני אסירת תודה לכל מיני אנשים אקראיים ברחבי האינטרנט שמגדילים את הידע שאפשר לצבור משם לאין שיעור, ואני מצפה בקוצר רוח ליישם 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:

